Igloo Project Documentation

Release 1.0

Kobalt

Apr 15, 2024

PROJECT INIT

CHAPTER

ONE

INIT PROJECT

In this page, we will follow the complete workflow to properly create a project, starting from nothing to finally be able to run the project on a server. In the following steps, we will call the project **hello-world**.

1.1 Clone the igloo-parent repository

First of all, clone the igloo-parent project :

```
git clone git@github.com:igloo-project/igloo-parent.git
```

1.2 Generate the new project and push it on gitlab

Note: You can find a more detailed documentation of this *part*.

In order to generate the project, we need to build the archetype :

```
cd ~/git/igloo-parent/basic-application
./build-and-push-archetype.sh ../basic-application/ local
```

After that, in /tmp/<generated-hash> folder, we generate the project using maven archetype plugin:

```
cd /tmp/<generated-hash>
mvn archetype:generate -DarchetypeVersion=1.0 -DarchetypeCatalog=local -
↪DartifactId=hello-world -DgroupId=fr.hello.world -Dversion=0.1-SNAPSHOT -Dpackage=fr.
↪hello.world -DarchetypeApplicationNamePrefix="HelloWorld" -
↪DarchetypeSpringAnnotationValuePrefix="helloWorld" -DarchetypeFullApplicationName=
↪"Customer - Hello World" -DarchetypeDatabasePrefix=hello_world -
↪DarchetypeDataDirectory=hello-world
```

The script asks what archetype we want to use. Choose the number corresponding to local, and check the different values we entered previously.

Go to the newly generated project folder and make this modification: add a line specifying Igloo's version in the file *hello-world/pom.xml* between the markers *properties* :

```
<properties>
  <igloo.version>X.X-SNAPSHOT</igloo.version>
</properties>
```

Note: **X.X-SNAPSHOT** must be replaced by the targetted version.

After that, push the project on git by executing the script located in the project folder :

```
/bin/bash init-git.sh hello-world git@gitlab.tools.kobalt.fr:<group>/<project>.git
git push -u origin master
```

After pushing the project on git, we have to delete the created folder and start working with a fresh one.

```
rm -rf /tmp/<generated-hash>/
```

We will make a new clone of the project using Oomph in the next step.

1.3 Create a fresh clone and a properly configured workspace with Oomph

Open an Eclipse (> 4.7 Oxygen, last release preferred) and select a new and clean workspace.

After that, follow the *Oomph documentation* until the window with multiple variables to fill in.

Here are the values to fill :

- Nom du clone git : `hello-world`
- Url du dépôt: git url where project is pushed
- Branche : master
- Répertoire du tomcat : ${user.home}/Documents/apps/apache-tomcat-8.5.23; provide a folder where an Apache Tomcat binary distribution is unpacked.
- Nom du projet maven : hello-world
- Nom de la webapp : hello-world-webapp
- Nom du projet gitlab : hello-world

From here, follow the ending steps from *Oomph documentation*.

1.4 Create and initialize the database

In this part, we will create the database with the proper user and schema, and we will fill it with a script. Before performing the following commands, make sure you have PostgreSQL installed.

To create the database, we execute some commands directly in a terminal:

```
createuser -U postgres -P hello_world
createdb -U postgres -O hello_world hello_world
psql -U postgres hello_world
#Here you are connected to the database as the user postgres
DROP SCHEMA public;
\q
psql -U hello_world hello_world
```

(continues on next page)

(continued from previous page)

```
#Here you are connected to the database as the user hello_world
CREATE SCHEMA hello_world;
```

Note: Use the name of the project for the password (here: hello_world)

After that we have to enable an option which will allow the project to create new entities in the database.

1.5 Create and initialize filesystem

```
sudo mkdir /data/services/basic-application
sudo chown "${USER}." /data/services/basic-application
```

1.6 Launch the webapp

Now we have all the tools properly configurated and ready to run our project.

To do that, we just start Tomcat in Eclipse (if you don't have the server view : **Window -> Show view -> Other -> Server/Servers**).

To access to our project, we can go to http://localhost:8080/basic-application . To access the console, the address is http://localhost:8080/basic-application/console/ .

Note: Until you change it, the login/password for the project and the project's console is admin/admin.

CHAPTER

TWO

ECLIPSE OOMPH

Clone the igloo-oomph-project repository that provides the Oomph setup file.

```
git clone git@github.com:igloo-project/igloo-oomph-project.git
```

- Open Eclipse, use a new workspace.
- Select **File > Import > Oomph > Projects into workspace**.
- Click on the + (green icon, top-right) button :
- Choose the catalog Eclipse Projects
- Browse file : choose the file `git/igloo-oomph-project/org.iglooproject.eclipse.project.setup`
- Check the box **igloo simple project**

Check and complete the window with required variables:

- Nom du clone git : the name of the file which will contain the git clone on your computer
- Nom du projet gitlab : the name of the project as it appears in gitlab or github (correspond to <project> in the url host:igloo-project/<project>.git)
- Branche : the branch which will be cloned
- Répertoire du tomcat : the path to your tomcat folder on your computer
- Nom du projet maven : the artifactId in your pom.xml
- Nom de la webapp : the name of the webapp which will be generated
- Url du dépôt

Click next, then finish. The installation may take a few minutes, and Eclipse might have to restart (keep an eye on the status bar at the bottom right; an blinking icon may require a restart).

In this case, the installation will go back to where it stopped automatically. It will clone the application, install and setup a Tomcat server.

Once the project initialized, a last restart is needed to fully initialize the Tomcat run configuration.

CHAPTER

THREE

FEATURES

3.1 Backend

3.1.1 Configuration management

Basic usage

Quick guide

- Write <property>=... in configuration.properties.
- If you want to override this property by environment on a regular basis (database settings for example), uses a property=environment.<property> definition, and put environment.<property>=... definition in configuration-env-*.properties files.
- If you want to override any property on your development environment, put definition in configuration-user-<username>.properties.
- Use /etc/<applicationName>/configuration.properties for environment sensitive information (password, ...) or not-versionned properties.

Selecting Igloo profile

Profile can be selected with IGLOO_PROFILE environment variable or igloo.profile system property.

Advanced usage

Default log4j configuration

With default bootstrap configuration, a log4j-igloo.properties is provided with some default configurations. This properties can be overriden in your local log4j-*.properties.

Override local configuration path

For configuration overrides, set igloo.config system property to a java resource url (default value: file:/etc/${igloo.applicationName}/configuration.properties).

For log4j configuration overrides, set igloo.log4j system property to a java resource url (default value: classpath:/log4j-extra.properties).

Default value for this properties can be set in configuration-bootstrap.properties.

Manage spring profiles

In configuration-bootstrap.properties, configure your default Spring profiles with igloo.default.spring.profiles.active.

If you want to use custom Spring profiles based on your Igloo profile setting, initialize custom igloo.<profile>.spring.profiles.active=... properties.

Use @TestPropertySource(properties="igloo.profile=xxx") for Igloo profile switching in your unit-test.

Create a custom Igloo profile

You can create custom Igloo profile (example: myProfile) by initializing `igloo.myProfile.(configurationLocations|log4j.configurationLocations|spring.profiles.active)` in `configuration-bootstrap.properties`.

`environment.<property>` pattern

Push your application configuration in `configuration.properties`. If this configuration relies on a Igloo profile switch, a best practice is to use a `<property>=${environment.<property>}` configuration to highlight the fact this property is overriden by environment.

For the definitions of `environment.<property>`:

- Set common overrides in `configuration-env-default.properties`.
- Set qualification, preproduction, production overrides in `configuration-env-deployment.properties`.
- Set specific overrides in `configuration-env-<profile>.properties` (including test profile).

Note: using `<property>=${environment.<property>}` is just a way to keep an exhaustive property definition in `configuration.properties` and to highlight generic and specific configurations. You can override any configuration in `configuration-*.properties` if needed.

Configuration loading logging

`igloo@config` logger allow to visualize loaded configurations. Verbose logs are handled by org.iglooproject.config.bootstrap.* logger names. Missing configurations are logged with a WARN level.

```
log4j.logger.igloo@config=INFO
log4j.logger.org.iglooproject.config.bootstrap=WARN
```

This configuration is included in `log4j-igloo.properties`.

Configuration debugging

Igloo 1.1.27 includes a debugging tool for configuration. It allows to write in a file targetted by `igloo.propertySource.outputFileName` the content of both environment and PropertySourcesPlaceholderConfigurer.

It is intended to be configured using a system property (e.g. `-Digloo.propertySource.outputFileName=/tmp/debug.properties`).

The generated files contains all listable properties and their values.

Not enumerable PopertySource (like JNDI ones) and not resolvable properties are skipped (a comment is included in the generated file for each skipped item).

This file may contain sensitive data (password, ...).

Custom bootstrap files

Loaded bootstrap files can be overriden with `IGLOO_BOOTSTRAP_LOCATIONS` environment variable or `igloo.bootstrapLocations` system property.

By default, these new locations are **added** to the default configuration.

To completely override bootstrap locations, you must use `IGLOO_BOOTSTRAP_OVERRIDE_DEFAULT` or `igloo.bootstrapOverrideDefault`.

Implementation's overview

Configuration loading

From 1.5.0, Igloo uses a early loading for properties:

- Igloo initializer (AbstractExtendedApplicationContextInitializer children classes):
  - loads `configuration-bootstrap*.properties` files. Property switching are based system property settings (`igloo.profile` value).
  - adds IglooPropertySourcesLevelsConfig @Configuration class to order `igloo/component`, `igloo/framework`, `igloo/application`, `igloo/bootstrap` and `igloo/overrides` @PropertySource's names (listed from the lower to the higher precedence).
  - add IglooBootstrapPropertySourcesConfig @Configuration class to manage loading of properties files targetted by `igloo.configurationLocations`.
- Vanilla @PropertySource annotation must be used to perform properties loading. Constants from IglooPropertySourcePriority must be used for the `name` attribute to manage property precedence.
- For a same @PropertySource name, bean discovery order is used to manage precedence. We must not rely on this mechanism to override property (as bean discovery order is hard to predict and control).

From 1.1 to 1.1.27, Igloo used the following mechanisms to handle configuration:

- Two-phase loading: **bootstrap** (early and minimal) and **spring configuration** (late and exhaustive)
- **Spring configuration locations** from `@ConfigurationLocations` annotations (either from your project, or from Spring Java Config beans provided by Igloo)
- **Configuration overriding** by user or by Igloo profile with an added bunch of files configured during bootstrap phase
- **Log4j** configured by user or by Igloo profile during bootstrap phase
- **Spring profiles** by user or by Igloo profile during bootstrap phase

> **Warning:** Igloo profile and Spring profiles are not the same thing. Igloo profile is loaded first, and Spring profiles are computed from `igloo.<profile>.spring.profiles.active`.

configuration-bootstrap* properties

configuration-bootstrap.properties, located in your project-core/src/main/resources may define the following values, for each needed profile:

- igloo.<profile>.configurationLocations: files to add to **override** configurations computed from @ConfigurationLocations files.
- igloo.<profile>.log4j.configurationLocations: files to merge to build log4j configuration.
- igloo.<profile>.spring.profiles.active: Spring profiles to activate. If there is no difference between the profile, you may use igloo.default.spring.profiles.active.
- tests are handled as a plain Igloo profile. Profile activation is done by using @PropertySource("igloo.profile=test") in AbstractTestCase.

Default configuration may be sufficient for simple projects. Just let the provided lines commented out.

Default configurations for bootstrap phase can be viewed here: https://github.com/igloo-project/igloo-parent/blob/dev/igloo/igloo-components/igloo-component-spring-bootstrap-config/src/main/resources/configuration-bootstrap-default.properties

It handles:

- configuration.properties loading
- configuration-env-{default,development}.properties configuration-user-${user.name}.properties loading for development
- configuration-env-{default,deployment,<profile>}.properties,
- /etc/<applicationName>/configuration.properties loading for qualification, preproduction, production profiles; this path can be overriden by igloo.config property.
- Log4j configuration loading with the same patterns
- Spring profiles configuration based on igloo.default.spring.profiles.active and an empty (no profile) default

Basic-application defaults are to activate flyway profile and to use defaults behavior.

Migrating from @ConfigurationLocations to @PropertySource

Reason of the breaking change

With 1.5.0, configuration subsystem is deeply modified to allow further improvements and to use vanilla spring boot mechanisms (autoconfiguration, conditionals, ...).

The old behavior was:

- Bootstrap configuration loads some low-level configuration properties (like igloo.configurationLocations, igloo.profile, log4j configuration).
- Spring bean configuration is fully discovered and computed.
- ApplicationConfigurerBeanFactoryPostProcessor initializes a PropertySourcesPlaceholderConfigurer based on Spring environment, based on the list provided by @ConfigurationLocations annotations (low precedence) and igloo.configurationLocations property (high precedence).
- Spring beans are initialized.

This behavior is problematic:

- It is not possible to control Spring bean discovery by Spring boot autoconfiguration by properties loaded by @ConfigurationLocations or `igloo.configurationLocations`
- @ConfigurationLocations is a custom annotation that cannot interact with @PropertySource
- Switch some properties to `configuration-bootstrap*.properties` (as it is early loaded) is a valid option, but it can lead to difficulties :
  - It is not clear for a developer how a property must be early-loaded or not.
  - If configuration is split, some properties may be wrongly updated between initialization and running phase.

So we choose to remove the old configuration behavior and to write a guide on how to update an Igloo application:

Configuration migration guide

- Generates a file containing your original configuration. Update your application to Igloo 1.4.x and starts it with `-Digloo.propertySource.outputFileName=/tmp/debug.properties.orig`.
- Switch to Igloo 1.5.0.
- Modify your `configuration-bootstrap.properties`
  - Add a property `igloo.applicationName=xxx`; Replace xxx with the name of `@ApplicationDescription`.
  - Add `classpath:/configuration.properties` as first item of all `igloo.<profile>.configurationLocations=`, either this property is commented out or not, to prevent any future error.
- Remove `@ApplicationDescription` annotation
- Remove empty `@ConfigurationLocations`
- For tests config files, remove all `@ConfigurationLocations` and add `@TestPropertySource` directly on your tests classes

```
@TestPropertySource(properties = "igloo.profile=test")
```

- Replace other `@ConfigurationLocations` by a named `@PropertySource`

```
@PropertySource(
   name = IglooPropertySourcePriority.APPLICATION,
   value = "classpath:configuration-init.properties"
)
```

- Start your application with `-Digloo.propertySource.outputFileName=/tmp/debug.properties.new`.
- Checks the diff of the 2 files:
  - Differences on `igloo.build.*` are OK.
  - Differences from Manifest information are OK (date, created-by, ...).
  - Added `classpath:/configuration.properties` in `igloo.configuurationLocations` is normal.
  - Modified `igloo.version` is OK.

Here is an example of a normal diff.

```
--- debug-propertySources.properties.orig     2019-07-23 14:33:01.983824649 +0200
+++ debug-propertySources.properties.new      2019-07-23 14:23:40.451085807 +0200
@@ -1,7 +1,7 @@
#skipped PropertySource: StubPropertySource {name='servletConfigInitParams'} as it is␣
↪not enumerable
#skipped PropertySource: JndiPropertySource {name='jndiProperties'} as it is not␣
↪enumerable
#
-#Tue Jul 23 14:32:59 CEST 2019
+#Tue Jul 23 14:23:20 CEST 2019
autocomplete.limit=20
awt.toolkit=sun.awt.X11.XToolkit
@@ -127,45 +127,45 @@
hibernate.search.default.elasticsearch.host=http\://127.0.0.1\:9220
hibernate.search.default.elasticsearch.index_schema_management_strategy=CREATE
igloo.applicationName=application
-igloo.build.date=1563884774310
-igloo.build.sha=a3664b7820ca11d2b34c157ee8373039ae5bf9a1
+igloo.build.date=1563882188918
+igloo.build.sha=f8af62cbc8ebdafa03b14b843246b724ca0fe720
igloo.build.user.name=lalmeras
igloo.component-spring.Build-Jdk=1.8.0_212
igloo.component-spring.Built-By=lalmeras
-igloo.component-spring.Built-Date=1563884774310
+igloo.component-spring.Built-Date=1563882188918
igloo.component-spring.Created-By=Apache Maven 3.5.4
-igloo.component-spring.Implementation-Build=a3664b7820ca11d2b34c157ee8373039ae5bf9a1
+igloo.component-spring.Implementation-Build=f8af62cbc8ebdafa03b14b843246b724ca0fe720
igloo.component-spring.Implementation-Title=Igloo - Component - Spring
igloo.component-spring.Implementation-Vendor=Kobalt
igloo.component-spring.Implementation-Vendor-Id=org.iglooproject.components
-igloo.component-spring.Implementation-Version=1.2-SNAPSHOT
+igloo.component-spring.Implementation-Version=1.2-PS
igloo.component-spring.Manifest-Version=1.0
igloo.config=file\:/etc/application/configuration.properties
-igloo.configurationLocations=classpath\:/configuration-env-default.properties,classpath\
↪:/configuration-env-development.properties,classpath\:/configuration-user-lalmeras.
↪properties
+igloo.configurationLocations=classpath\:/configuration.properties,classpath\:/
↪configuration-env-default.properties,classpath\:/configuration-env-development.
↪properties,classpath\:/configuration-user-lalmeras.properties
igloo.default.spring.profiles.active=flyway
-igloo.development.configurationLocations=classpath\:/configuration-env-default.
↪properties,classpath\:/configuration-env-development.properties,classpath\:/
↪configuration-user-lalmeras.properties
+igloo.development.configurationLocations=classpath\:/configuration.properties,classpath\
↪:/configuration-env-default.properties,classpath\:/configuration-env-development.
↪properties,classpath\:/configuration-user-lalmeras.properties
igloo.development.log4j.configurationLocations=classpath\:/log4j-igloo.properties,
↪classpath\:/log4j.properties,classpath\:/log4j-env-development.properties,classpath\:/
↪log4j-user-lalmeras.properties
igloo.development.spring.profiles.active=flyway
-igloo.integration.configurationLocations=classpath\:/configuration-env-default.
```

(continues on next page)

(continued from previous page)

```
↪properties,classpath\:/configuration-env-deployment.properties,classpath\:/
↪configuration-env-preproduction.properties,file\:/etc/application/configuration.
↪properties
+igloo.integration.configurationLocations=classpath\:/configuration.properties,classpath\
↪:/configuration-env-default.properties,classpath\:/configuration-env-deployment.
↪properties,classpath\:/configuration-env-preproduction.properties,file\:/etc/
↪application/configuration.properties
igloo.integration.log4j.configurationLocations=classpath\:/log4j-igloo.properties,
↪classpath\:/log4j.properties,classpath\:/log4j-env-deployment.properties,classpath\:/
↪log4j-env-preproduction.properties,classpath\:/log4j-extra.properties
igloo.integration.spring.profiles.active=flyway
igloo.log4j=classpath\:/log4j-extra.properties
igloo.log4j.configurationLocations=classpath\:/log4j-igloo.properties,classpath\:/log4j.
↪properties,classpath\:/log4j-env-development.properties,classpath\:/log4j-user-
↪lalmeras.properties
-igloo.preproduction.configurationLocations=classpath\:/configuration-env-default.
↪properties,classpath\:/configuration-env-deployment.properties,classpath\:/
↪configuration-env-preproduction.properties,file\:/etc/application/configuration.
↪properties
+igloo.preproduction.configurationLocations=classpath\:/configuration.properties,
↪classpath\:/configuration-env-default.properties,classpath\:/configuration-env-
↪deployment.properties,classpath\:/configuration-env-preproduction.properties,file\:/
↪etc/application/configuration.properties
igloo.preproduction.log4j.configurationLocations=classpath\:/log4j-igloo.properties,
↪classpath\:/log4j.properties,classpath\:/log4j-env-deployment.properties,classpath\:/
↪log4j-env-preproduction.properties,classpath\:/log4j-extra.properties
igloo.preproduction.spring.profiles.active=flyway
-igloo.production.configurationLocations=classpath\:/configuration-env-default.
↪properties,classpath\:/configuration-env-deployment.properties,classpath\:/
↪configuration-env-production.properties,file\:/etc/application/configuration.properties
+igloo.production.configurationLocations=classpath\:/configuration.properties,classpath\
↪:/configuration-env-default.properties,classpath\:/configuration-env-deployment.
↪properties,classpath\:/configuration-env-production.properties,file\:/etc/application/
↪configuration.properties
igloo.production.log4j.configurationLocations=classpath\:/log4j-igloo.properties,
↪classpath\:/log4j.properties,classpath\:/log4j-env-deployment.properties,classpath\:/
↪log4j-env-production.properties,classpath\:/log4j-extra.properties
igloo.production.spring.profiles.active=flyway
igloo.profile=development
igloo.profile.default=development
-igloo.qualification.configurationLocations=classpath\:/configuration-env-default.
↪properties,classpath\:/configuration-env-deployment.properties,classpath\:/
↪configuration-env-qualification.properties,file\:/etc/application/configuration.
↪properties
+igloo.qualification.configurationLocations=classpath\:/configuration.properties,
↪classpath\:/configuration-env-default.properties,classpath\:/configuration-env-
↪deployment.properties,classpath\:/configuration-env-qualification.properties,file\:/
↪etc/application/configuration.properties
igloo.qualification.log4j.configurationLocations=classpath\:/log4j-igloo.properties,
↪classpath\:/log4j.properties,classpath\:/log4j-env-deployment.properties,classpath\:/
↪log4j-env-qualification.properties,classpath\:/log4j-extra.properties
igloo.qualification.spring.profiles.active=flyway
```

(continues on next page)

(continued from previous page)

```
-igloo.test.configurationLocations=classpath\:/configuration-env-default.properties,
↪classpath\:/configuration-env-test.properties,classpath\:/configuration-user-lalmeras-
↪test.properties
+igloo.test.configurationLocations=classpath\:/configuration.properties,classpath\:/
↪configuration-env-default.properties,classpath\:/configuration-env-test.properties,
↪classpath\:/configuration-user-lalmeras-test.properties
igloo.test.log4j.configurationLocations=classpath\:/log4j-igloo.properties,classpath\:/
↪log4j.properties,classpath\:/log4j-env-test.properties,classpath\:/log4j-user-lalmeras-
↪test.properties
igloo.test.spring.profiles.active=flyway
-igloo.version=1.2-SNAPSHOT
+igloo.version=1.2-PS
imageMagick.convertBinary.path=/usr/bin/convert
infinispan.enabled=false
infinispan.roles=QueueTaskHolder\#initQueuesFromDatabase
```

3.1.2 Logging configuration

Default configuration

Logging configuration is based on the following components :

- SLF4J API for every logging invocation (Logger and MDC)
- Log4j2 (from X.X.X) for logging backend
- Log4j2 CompositeConfiguration for logging configuration

How dependencies are managed

Logging subsystem is provided by these dependencies :

- igloo-component-core-logging : import slf4j API and slf4j bindings for JCL (commons-logging) and JUL (java.util.logging)
  - JUL binding implies to call `SLF4JBridgeHandler.install()`. This is done in Igloo project using `SLF4JLoggingListener` servlet listener. This listener is added in basic-application-webapp `web.xml`.

    Logger `org.iglooproject.slf4j.jul.bridge.SLF4JLoggingListener`, level INFO, can be used to check if the binding is done.
  - JCL binding is automatic
- igloo-component-core-logging-log4j1 : dependency to use log4j 1.2 (deprecated) backend
- igloo-component-core-logging-log4j2 : dependency to use log4j2 backend

How log4j configuration can be customized

Log4j 1.2 and Log4j2 uses similar behavior. log4j2 must be transformed to log4j when Log4j 1.2 is used.

At early startup stage :

- log4j 1.2 uses log4j.properties file (default configuration)
- log4j2 uses log4j2.properties as configured by `log4j2.component.properties` in `igloo-component-spring-bootstrap-config` module

During Spring early initialization :

- Log4j2 configuration is customized by loading custom files
- It is managed by `AbstractExtendedApplicationContextInitializer` class and uses `log4j2.configurationLocations` property.
- This property is set (default) in `configuration-bootstrap-default.properties`, `configuration-bootstrap.properties` (in your project).
- A custom bootstrap properties path can be specified : *Custom bootstrap files*
- Switch between log4j 1.2 and log4j2 is done base on SLF4J selected backend.
- It is adviced to ensure you include only 1 implementation and 1 SLF4J binding in your classpath to ensure proper backend is used. So check your application dependencies.

With default configuration (take a look at `configuration-bootstrap-default.properties`), you can use :

- `log4j2-user-USERNAME.properties` to manage your personal configuration
- `log4j2-env-development.properties` to manage configuration dedicated to development
- `log4j2-env-deployment.properties` to manage configuration for all deployment targets (qualification, preproduction, production)
- `log4j2-env-{qualification,preproduction,production}.properties` to manage configuration for earch target
- You can use `igloo.log4j2` property to specify a path to an extra configuration. It is commonly used to provides a local on-disk extra configuration. Default path for this value is `classpath:/log4j2-extra.properties`.

Last configuration has the highest precedence. Configuration are loaded with this order :

- General igloo config (log4j2-igloo.properties)
- Project configuration (log4j2.properties)
- Environment configuration
- User configuration
- Extra configuration

Logger `igloo@config` allows to see loaded files.

```
INFO  - igloo@config - Log4j configurations (ordered): classpath:/log4j2-igloo.
↪properties, classpath:/log4j2.properties, classpath:/log4j2-env-development.properties,
↪ classpath:/log4j2-user-lalmeras.properties
```

Other customizations

`org.iglooproject.wicket.servlet.filter.Log4jUrlFilter`, a servlet filter, installed in basic-application-webapp web.xml set current page URL using `MDC` API.

It allows to include URL in logging messages by using `%X{ow-url}` in the configured pattern.

How to keep log4j 1.2

To keep log4j 1.2, you need to add this new dependency to your `application-core/pom.xml`:

```
<!-- Logging backend -->
<dependency>
   <groupId>org.iglooproject.dependencies</groupId>
   <artifactId>igloo-dependency-core-logging-log4j1</artifactId>
   <version>${igloo.version}</version>
   <type>pom</type>
</dependency>
```

Check that your core, init and webapp dependencies only include log4j 1.2.

How to migrate a log4j 1.2 configuration

To migrate to log4j2, you need to add this new dependency to your `application-core/pom.xml`:

```
<!-- Logging backend -->
<dependency>
   <groupId>org.iglooproject.dependencies</groupId>
   <artifactId>igloo-dependency-core-logging-log4j2</artifactId>
   <version>${igloo.version}</version>
   <type>pom</type>
</dependency>
```

Check that your core, init and webapp dependencies only include log4j2.

If you use a custom `configuration-bootstrap.properties`, you need to create the multiple `log4j2.configurationLocations` entries. You can use igloo basic-application-core `configuration-bootstrap.properties` as a model.

You need to check if deployment uses custom bootstrap files (ask your deployment guy) to update it. You also need to update any local log4j configuration file.

Then `log4j-*.properties` must be renamed `log4j2-*.properties`, and configuration rewritten to use log4j2 syntax.

Here are examples:

Listing 1: Main configuration

```
#
# log4j 1.2
#
log4j.appender.Stdout=org.apache.log4j.ConsoleAppender
log4j.appender.Stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.Stdout.layout.ConversionPattern=[%d{ISO8601}][%X{PID}] %-5p - %-26.26c{1}␣
```

(continues on next page)

(continued from previous page)

```
↪- %X{ow-url} - %m\n

log4j.rootLogger=WARN, Stdout

#
# log4j2
#
status=error
dest=err
name=PropertiesConfig

appender.console.type=Console
appender.console.name=STDOUT
appender.console.layout.type=PatternLayout
appender.console.layout.pattern=[%d{ISO8601}] %-5p - %-26.26c{1} - %X{ow-url} - %m\n

rootLogger.level=warn
rootLogger.appenderRef.stdout.ref=STDOUT
```

Listing 2: Logger configuration

```
#
# log4j 1.2
#
log4j.logger.package.Class=DEBUG

#
# log4j2
#
# NAME is an arbitrary string used to link name and level together
logger.NAME.name=package.Class
logger.NAME.level=DEBUG
```

Listing 3: Advanced logger configuration

```
#
# log4j 1.2
#
# synchro is an appender
log4j.logger.package.Class=INFO, synchro
log4j.additivity.package.Class=false

#
# log4j2
#
# NAME is an arbitrary string used to link name and level together
# EITHER is an arbitrary string used to link append configurations together
# REFNAME is an arbitrary string used to link logger and appender together
# [...]
appender.EITHER.name=synchro
# [...]
logger.NAME.name=package.Class
```

(continues on next page)

(continued from previous page)

```
logger.NAME.level=DEBUG
logger.NAME.additivity=false
logger.NAME.appenderRef.REFNAME.ref=SYNCHRO
logger.NAME.SYNCHRO.ref=synchro
# if multiple append must be specified
#logger.NAME.appenderRef.REFNAME.ref=SYNCHRO
#logger.NAME.appenderRef.REFNAMEStdout.ref=STDOUT
#logger.NAME.SYNCHRO.ref=synchro
#logger.NAME.STDOUT.ref=STDOUT
```

Listing 4: Alternate appender configuration

```
#
# log4j 1.2
#
log4j.appender.SYNCHRO=org.apache.log4j.FileAppender
log4j.appender.SYNCHRO.File=${catalina.base}/logs/synchro.log
log4j.appender.SYNCHRO.layout=org.apache.log4j.PatternLayout
log4j.appender.SYNCHRO.layout.ConversionPattern=[%d{ISO8601}] %-5p - %-26.26c{1} - %X{ow-
↪url} - %m\n

#
# log4j2
#
# sys: prefix is used to use system properties
# env: prefix may be used for environment variables
appender.synchro.type=File
appender.synchro.name=SYNCHRO
appender.synchro.fileName=${sys:catalina.base}/logs/synchro.log
appender.synchro.layout.type=PatternLayout
appender.synchro.layout.pattern=[%d{ISO8601}] %-5p - %-26.26c{1} - %X{ow-url} - %m\n
```

Listing 5: Multiple rootLogger appender configuration

```
#
# log4j 1.2
#
log4j.rootLogger=WARN, Stdout, FILE

log4j.appender.FILE=org.apache.log4j.FileAppender
log4j.appender.FILE.File=${catalina.base}/logs/synchro.log
log4j.appender.FILE.layout=org.apache.log4j.PatternLayout
log4j.appender.FILE.layout.ConversionPattern=[%d{ISO8601}] %-5p - %-26.26c{1} - %X{ow-
↪url} - %m\n

#
# log4j2
#
# NAME is an arbitrary string used to link name
# EITHER is an arbitrary string used to link append configurations together
rootLogger.appenderRef.EITHER.ref=File
rootLogger.appenderRef.stdout.ref=STDOUT
```

(continues on next page)

(continued from previous page)

```
appender.NAME.type=File
appender.NAME.name=FILE
appender.NAME.fileName=${sys:catalina.base}/logs/synchro.log
appender.NAME.layout.type=PatternLayout
appender.NAME.layout.pattern=[%d{ISO8601}] %-5p - %-26.26c{1} - %X{ow-url} - %m\n
```

3.1.3 PropertyService

About

The PropertyService gets and sets the values for your application's properties in a typesafe manner. These might be stored in `*.properties` files (being "immutable" properties) or in database ("mutable" properties).

Architecture

Properties are referenced to using their *id*, which may have one of two types:

- `ImmutablePropertyId` for immutable properties that are linked to configuration.properties.
- `MutablePropertyId` for mutable properties that are stored in database in table Parameter as String. Beware that the other columns of this table are deprecated.

Here are the basic principles:

- Properties are first registered using their ID during the application initialization. The registration involves telling to the `PropertyService` how to convert the property value to and from its string representation, which is done by providing a `Converter` (a Guava type), or (for immutable properties that don't need to be converter back to string) a `Function`.
- During the execution of the application, the property values may be retrived using the service's `get` method, or altered using the `set` method.

Using it in your application

Declare the IDs

You must first declare constants for your property ids. You'd better have two sets of property ids: one for you core (non-UI) project and the other for you webapp (UI) project.

The keys passed as parameters to `AbstractPropertyIds`'s methods are the one used to retrieve the values, either in the `configuration.properties` file or in database.

```
public final class YourAppCorePropertyIds extends AbstractPropertyIds {

        public static final ImmutablePropertyId<Integer> CORE_IMMUTABLE_INTEGER_PROPERTY␣
↪= immutable("core.immutable.integer.property");
        public static final ImmutablePropertyId<MyType> CORE_IMMUTABLE_MYTYPE_PROPERTY =␣
↪immutable("core.immutable.mytype.property");

        public static final MutablePropertyId<String> CORE_MUTABLE_STRING_PROPERTY =␣
↪mutable("core.mutable.string.property");
```

(continues on next page)

(continued from previous page)

```
}
```

```
public final class YourAppWebappPropertyIds {

        public static final ImmutablePropertyId<Long> WEBAPP_IMMUTABLE_LONG_PROPERTY =␣
↪immutable("webapp.immutable.long.property");
        public static final ImmutablePropertyId<MyType> WEBAPP_IMMUTABLE_MYTYPE_PROPERTY␣
↪= immutable("webapp.immutable.mytype.property");

        public static final MutablePropertyId<String> WEBAPP_MUTABLE_STRING_PROPERTY =␣
↪mutable("webapp.mutable.string.property");

}
```

Register the properties

In your core module set up a JavaConfig as follows.

Beware that `AbstractApplicationPropertyConfig` also declares the property service, which is a singleton. Thus it must only be used once (in your core module).

```
@Configuration
public class YourAppCoreApplicationPropertyConfig extends␣
↪AbstractApplicationPropertyConfig {

        @Override
        public IMutablePropertyDao mutablePropertyDao() {
                return new ParameterDaoImpl();
        }

        @Override
        public void register(IPropertyRegistry registry) {
                // register core properties here
        }

}
```

In your webapp module set up a JavaConfig as follows:

```
@Configuration
public class YourAppWebappApplicationPropertyRegisterConfig extends␣
↪AbstractApplicationPropertyRegistryConfig {

        @Override
        public void register(IPropertyRegistry registry) {
                // register webapp properties here
        }

}
```

`IPropertyRegistry` provides a bunch of methods to register properties. See details below.

Access the properties

Anywhere an `IPropertyService` is available (it can be injected), do the following:

```
@Autowired
private IPropertyService propertyService;

// Get a value
propertyService.get(YourAppCorePropertyIds.CORE_IMMUTABLE_MYTYPE_PROPERTY);

// Set a value (only for mutable properties)
propertyService.set(YourAppCorePropertyIds.CORE_MUTABLE_STRING_PROPERTY, "NewValue");
```

Details about registration

Important note

It is strongly recommended to define a default value for collection properties in order to always get a collection even if the value is `null`.

Examples

Basic immutable property

```
public static final ImmutablePropertyId<String> MAINTENANCE_URL = immutable("maintenance.
↪url");
```

```
propertyService.registerString(MAINTENANCE_URL);
```

Immutable property with custom converter/function

```
public static final ImmutablePropertyId<Date> DATE_PICKER_RANGE_MAX_DATE = immutable(
↪"datePicker.range.max.yearsFromNow");
```

```
propertyService.registerImmutable(DATE_PICKER_RANGE_MAX_DATE, new Function<String, Date>
↪() {
        @Override
        public Date apply(String input) {
                Integer years = Ints.stringConverter().convert(input);
                if (years == null) {
                        return null;
                }
                return DateUtils.truncate(
                                DateUtils.addYears(new Date(), years),
                                Calendar.DAY_OF_MONTH
                );
        }
});
```

Mutable property with dynamic key with default value

```
public static final MutablePropertyIdTemplate<Boolean> DATA_UPGRADE_DONE_TEMPLATE =␣
↪mutableTemplate("dataUpgrade.%1s");
public static final MutablePropertyId<Boolean> dataUpgrade(IDataUpgrade dataUpgrade) {
        return DATA_UPGRADE_DONE_TEMPLATE.create(dataUpgrade.getName());
}
```

```
propertyService.registerBoolean(DATA_UPGRADE_DONE_TEMPLATE, false);
```

Property - enum

```
public static final ImmutablePropertyId<Environment> ENVIRONMENT = immutable("environment
↪");
```

```
propertyService.registerEnum(ENVIRONMENT, Environment.class, Environment.production);
```

Property - collection

```
public static final ImmutablePropertyId<List<MediaType>> FICHIER_PIECE_JOINTE_MEDIA_
↪TYPES = immutable("fichier.pieceJointe.mediaTypes");
```

```
propertyService.register(FICHIER_PIECE_JOINTE_MEDIA_TYPES, new StringCollectionConverter
↪<>(Enums.stringConverter(MediaType.class), Suppliers2.<MediaType>arrayList()));
```

3.1.4 E-mail notifications

Features

Email notification system provides the following feature:

- Builder design pattern for sending notifications (with method chaining, #from(), #sender(), #to(), #subject(), #textBody(), ...). See `NotificationBuilder` class and `INotificationBuilderState*` interfaces.
- Freemarker templates for text notifications (deprecated).
- Wicket-based templates for HTML notifications.
- CSS style inlining to ensure a large client compatibility for reading generated HTML messages.

- Support Wicket generation both inside and outside request's context (Wicket-based HTML emails can be generated during request, scheduled jobs or tasks).
- Notification grouping or splitting based on recipient-based preférences (split one #send() call to one message for french recipients and another for english recipients).
- Special behaviors for test environments (email filtering).

NotificationBuilder

Send a simple notification

```
INotificationBuilderBaseState builder = NotificationBuilder.create().
↪init(applicationContext);
builder.from("from@example.com").to("to@example.com")
  .subject("subject").textBody("content").send();
```

Send a wicket-based HTML notification

To send a wicket-based HTML notification, you'll need to provide a *NotificationContentDescriptorFactoryImpl*. basic-application provides an example named `BasicApplicationNotificationContentDescriptorFactoryImpl`.

This object provides for each of your notification a `INotificationContentDescriptor` with the methods to render subject, text body and html body. Inhering `AbstractWicketNotificationDescriptor` allows to delegate html body generation to a wicket component.

Common behavior

- A method in `NotitificationServiceImpl`
- Build a `NotificationBuilder` instance
- Setup recipients (to, cc), reply-to, ...
- `INotificationRecipient` can be used for user to email / locale / profile information
- Wicket context :
  - If mail is sent outside of wicket context (task, scheduled job), wicket context is determined from `IWicketContextProvider` bean. `wicket.backgroundThreadContextBuilder.*` properties must be setup
  - Else current wicket context is reused
  - Wicket context lookup is performed in `AbstractOfflinePanelRendererServiceImpl` (try-with-resources with `ITearDownHandle`)
- Rendering is done with a `runAsSystem` wrapper (as the sender context is rarely useful to manager component visibility) (done in `AbstractWicketRendererServiceImpl`)
- Rendering locale (in wicket context) is done consistently with recipient's declared locale (see `INotificationRecipient#getLocale` method). If your application allow multi-locale settings, ensure this is the expected behavior
- If multiple locales are needed to honor recipient's locales, there is one generation and one sending by targetted locale

Warnings

- A System security context is used for email rendering. Do not rely on dynamic component visibility if you use default sending options
- Pay special attention to wicket context if you use multiple Wicket applications. Wicket application affects Page to URL translation (page may have different URL, or no bookmarkable URL in some application) and absolute URL translation (hostname can be customized for each application)
- If you use multiple wicket applications, pay attention that email sending is done with the expected application (example: extranet/intranet splitted application may trigger send from intranet to extranet users)

Offline renderding and URL

For offline or overriden rendering, page to URL translation is done with hostname, scheme and port extracted from `wicket.<APPLICATION_NAME>.backgroundThreadContextBuilder.url.(servletPath|scheme|port|serverName)`.

Default values are extracted from `wicket.backgroundThreadContextBuilder.url.(servletPath|scheme|port|serverName)`.

Offline rendering are done by task or scheduled jobs.

Overriden rendering is needed if we want to render the page with an different wicket application.

`APPLICATION_NAME` is the wicket application name (`WebApplication.getName()`). It defaults to wicket application servlet filter name, but it can be overriden in constructor :

```
        public ExtranetFront() {
                super();
                this.setName("extranet");
        }
```

Force mail rendering application context

To force email rendering within a given application (to ensure that absolute URLs are valid):

- override your `I*NotificationContentDescriptorFactory#context(Locale locale)` method to provide your own wicket application.

```
@Override
  protected IExecutionContext context(Locale locale) {
          return getWicketContextProvider().context(extranetFront, locale);
  }
```

- setup your wicket application name (used by configuration keys):

```
  public ExtranetFront() {
    super();
    this.setName("extranet");
  }
```

- add to your configuration application needed hostname, port, scheme and servletPath configurations

```
wicket.extranet.backgroundThreadContextBuilder.url.scheme=https
wicket.extranet.backgroundThreadContextBuilder.url.serverName=domain.com
wicket.extranet.backgroundThreadContextBuilder.url.serverPort=443
# servletPath may need to be set if root path is not used
```

Configuration

NotificationBuilder behavior is driven by the following properties:

locale.default
: As each recipient is associated with a preferred locale, `locale.default` is used to associate a locale for recipient when raw-email-based methods are used to provide To, Cc, Bcc. *INotificationRecipient*-based methods can be used to force explicit locale setting.

notification.mail.from
: Fallback for From: address if not set explicitly. Both `local-part@domain` and `Personal <local-part@domain>` formats are allowed.

notification.mail.sender.behavior (enum)
: **EXPLICIT** | FALLBACK_TO_CONFIGURATION | FALLBACK_TO_FROM

  Control *NotificationBuilder* behavior when sender is not explicitly set.

  - FALLBACK_TO_CONFIGURATION: use `notification.mail.sender` to configure sender when it is not set explicitly.
  - FALLBACK_TO_FROM: use current notification From: field to set sender.
  - EXPLICIT: do nothing

notification.mail.sender
: Address used when `notification.mail.sender.behavior=true`. Both `local-part@domain` and `Personal <local-part@domain>` formats are allowed.

notification.mail.subjectPrefix
: String used to prefix all sent emails. This is usefull to explicitly differentiate environments (testing, development). A space is automatically added to this prefix.

Note: `[Dev]` prefix is automatically added when `configurationType=development`

---

notification.mail.disabledRecipientFallback
: Address used when a `INotificationRecipient#isNotificationEnabled() == false` or does not provide an email address.

notification.mail.recipientsFiltered (boolean)
: Use email filtering. All notifications are redirected to addresses listed by `notification.test.emails`. A block is added at the start of the notification to list the original recipients (To, Cc, Bcc) of the mail.

  This setting is made for testing purpose.

Note: `configurationType=development` enforce recipient filtering, regardless of this setting.

---

notification.test.emails (boolean)
: Space separated list of emails receiving all notifications when `notification.mail.recipientsFiltered=true`.

wicket.backgroundThreadContextBuilder.url.(servletPath|scheme|port|serverName)
servletPath (root path), scheme (http or https), port (80 or 443) and serverName to use to generate absolute page URLs. This is default value used when application values are not provided (see next entry). This is needed for all wicket HTML rendering outside of a wicket request. (if a Wicket context is available, then these values are extracted from HTTP request).

wicket.APPLICATION_NAME.backgroundThreadContextBuilder.url.(servletPath|scheme|port|serverName)
see previous entry. This values allow to override setting by wicket application name. Example : it allows to handle extranet and intranet in separated applications, each with their own domain name.

3.1.5 Autoprefixer

Just like Bootstrap, Igloo uses Autoprefixer to automatically add vendor prefixes to some CSS properties at build time. Doing so saves us time and code by allowing us to write key parts of our CSS.

Autoprefixer is enabled by default in our scss build process in deployment mode, but it's disabled in development mode. This behavior is defined by the property `autoprefixer.enabled`.

Use `autoprefixer.enabled=true` to enable Autoprefixer in a local environment.

3.1.6 HistoryLog & Audit

Principles

The history log machinery is designed to track the action performed on our business entities.

It can also be used to track the differences but this feature should be used with caution as it can be quite time consuming to configure and has a negative impact on performance.

Logs are performed upon commit (just before the commit), in order to:

- Remove duplicate logs
- Ensure that all edits on the entities are over, so that a diff can safely be computed

Warning: Please note that when performing a batch processing, you should take care of calling `ITransactionSynchronizationTaskManagerService.beforeClear()` before flushing the indexes and clearing the Hibernate session, so that the logs can be safely flushed too.

Using it in your application

Setting up

The basic application provide a basic (but functional) template for setting up the HistoryLog. See package `org.iglooproject.basicapp.core.business.history` for more information.

The minimal set of classes to define includes:

- The concrete enum representing `HistoryEvents` (create, update, sign-in, ...)
- The concrete class for your `HistoryLog`s, with (optionally) custom fields.
- Your `HistoryLogDao` and `HistoryLogService`, which may simply extend the provided abstract classes
- Your own implementation of the bean that will store additional informations about log objects: `HistoryLogAdditionalInformationBean`. This bean serves as a way to pass additional parameters to the `HistoryLogService`: secondary objects (for actions performed on more than one object), contextual information (to enable later search on logs with filters on objects *linked* to the main object at the

time of the action), … **It's typically not recommended to randomly use the object1…4 fields (see `AbstractHistoryLogAdditionalInformationBean` for more information).**

You may also want to define a `HistoryLogSearchQuery` for querying you history. An example is provided in the basic application.

Recording simple logs

```
historyLogService.log(HistoryEventType.SIGN_IN, user,␣
↪HistoryLogAdditionalInformationBean.of(user));
```

Recording diff-enabled logs

Linking a full diff of the main object to the log:

```
historyLogService.logWithDifferences(HistoryEventType.UPDATE, person,␣
↪HistoryLogAdditionalInformationBean.of(person), userDifferenceService);
```

The code above will add to the log every change on every relevant fields of the given entity. This is typically what we do upon updates.

Sometimes, though, you only want to record changes on a subset of the entity fields. This is typically what's required upon create/delete (where we know that almost all fields will change), but it may be done upon updates, too.

Here's how to link a minimal diff of the main object to the log:

```
historyLogService.logWithDifferences(HistoryEventType.UPDATE, person,
                                     HistoryLogAdditionalInformationBean.of(person),
                                     userDifferenceService.getMinimalDifferenceGenerator(),
                                     userDifferenceService);
```

Optionnally, you may pass to those methods one or several `IDifferenceHandler<T>`, which are ways for you to inspect a diff, and do something based on that information. This is typically used to update a date on the entity when some field changed.

For more information about setting up a difference service, see *DifferenceService*.

Displaying the logs on the user interface

Basic stuff

The basic application contains everything needed to display the audit. See `UserHistoryLogPanel`.

Renderers/Converters

The history log machinery uses the renderers/converters infrastructure so you need to define either converters (in your `YourAppApplication.newConverterLocator()` or renderers (in `YourAppWebappConfig.rendererService()`).

Resource keys for difference display

When the label of a field needs to be displayed, several resource keys are tried, in this order:

- `history.difference<entity resource key>.<property.path>`
- `business<entity resource key>.<property path>`
- `history.difference.common.<property.path>`
- `business.<property path>`

The first key that actually exists in you properties is used. This behavior is defined in the basic application class `HistoryDifferencePathRenderer`.

The resource key for each entity is defined in `AbstractHistoryRenderer`.

3.1.7 Difference Service (TODO)

Principles

Igloo provides a way for applications to compute a diff between two objects, i.e. of recursively computing field-by-field differences between a "working" version and a "reference" version of an object.

The diff infrastructure is powered by Java-object-diff. Igloo itself provides:

- basic setup of java-object-diff
- abstract bases for your diff services, with clearly identified points of configuration
- interfaces to integrate your diff services to other parts of your application (mainly *HistoryLog & Audit*)
- a way for you to perform a diff between an object in your Hibernate session and the version currently in your database

Using it in your application

Setting up

TODO: this part is still evolving. See https://trello.com/c/gUTpyoMr in particular.

Performing a diff between two objects

```
// Obtain your IDifferenceGenerator
// You may also use differenceService.getMinimalDifferenceGenerator(), depending on your␣
↪use case
IDifferenceGenerator generator = differenceService.getMainDifferenceGenerator();
// Perform the  diff
Difference<T> diff = generator.diff(workingObject, referenceObject);
```

Performing a diff between your (modified) object and the version in your DB

```
// Obtain your IDifferenceGenerator
// You may also use differenceService.getMinimalDifferenceGenerator(), depending on your␣
↪use case
IDifferenceFromReferenceGenerator generator = differenceService.
↪getMainDifferenceGenerator();
// Perform the  diff
Difference<T> diff = generator.diffFromReference(workingObject);
```

Using it in the history log machinery

See *HistoryLog & Audit*.

3.1.8 Task Executor

About

Igloo comes with a task service allowing to persist and execute background tasks.

Configuration

The task service can manage multiple queues (typically, a queue for short tasks and a queue for long tasks).

```
@Configuration
public class YourAppTaskManagementConfig extends AbstractTaskManagementConfig {

        @Override
        @Bean
        public Collection<? extends IQueueId> queueIds() {
                return EnumUtils.getEnumList(YourAppTaskQueueId.class);
        }

}
```

The task manager can be configured with the following properties:

- `task.startMode`: auto or manual: define if the task manager is started automatically at application startup
- `task.queues.config.<queueId>.threads`: number of execution threads for the given queue (by default 1)

- `task.stop.timeout`: timeout of a task: the task is stopped if its execution lasts longer than the timeout

Queueing a task

A task extends `AbstractTask`. It's going to be serialized as JSON (using Jackson) in order to be persisted.

A task has a name (for identification by a human beand a type.

You can define the queue used by implementing the `selectQueue()` method. It allows you to choose a different queue depending on the parameters passed (e.g. selecting a different queue based on the task type, or based on how many items you have to export).

It's quite easy to submit a new task:

```
YourTask task = new YourTask(user, parameters...);
queuedTaskHolderService.submit(task);
```

Inside a task

TODO GSM

Console

The administration console contains an administration UI for the task manager.

3.1.9 External Link Checker

About

Igloo has a service called the external link checker. It's used to check that external links stays valid.

It's currently used in production to validate several 100k links.

It is based on the HTTPClient library. It first tries a HEAD request to limit the bandwidth usage, it tries a GET request if the HEAD request fails.

If a link is unreachable, it is marked as `OFFLINE`. After several checks being offline (to avoid transient errors), the link is marked as `DEAD_LINK`.

Model

The link must be wrapped in an `ExternalLinkWrapper`:

```
@OneToOne(fetch = FetchType.LAZY, cascade = { CascadeType.ALL }, orphanRemoval = true)
private ExternalLinkWrapper externalLinkWrapper;
```

The status of the link is available in the `ExternalLinkWrapper`. A link is considered invalid if it's in the `DEAD_LINK` status.

Usage

Maven

Starting with Igloo 0.11, ExternalLinkChecker has its own Maven module:

```
<dependency>
    <groupId>org.iglooproject.components</groupId>
    <artifactId>igloo-component-jpa-externallinkchecker</artifactId>
    <version>${project.version}</version>
</dependency>
```

Configuration

The configuration is as follows:

```
externalLinkChecker.timeout=30000
externalLinkChecker.userAgent=Your user agent
externalLinkChecker.batchSize=400
externalLinkChecker.retryAttemptsNumber=4
externalLinkChecker.maxRedirects=5
externalLinkChecker.cronExpression=0 0/15 3-6,18-23 * * ?
```

The cron expression must be used to configure a scheduled task which launches `externalLinkCheckerService.checkBatch()`.

You can ignore links by adding regexps using the `externalLinkCheckerService.addIgnorePattern(pattern)` method.

In `YourAppCoreCommonConfig`, you have to:

- add `JpaExternalLinkCheckerBusinessPackage` to the `basePackageClasses` of `@ComponentScan`

In `YourAppCoreCommonJpaConfig`, you have to:

- add `JpaExternalLinkCheckerBusinessPackage` to the package scanned for entities in `applicationJpaPackageScanProvider()`
- declare an Hibernate interceptor:

```
        @Bean
        public Interceptor hibernateInterceptor() {
                return new ChainedInterceptor()
                                .add(new ExternalLinkWrapperInterceptor());
        }
```

3.1.10 Table import

About

Igloo comes with an optional feature that allows you to import data from table structured documents. Today, it means that you can import data from documents under either CSV or Excel format.

A classical case which can be treated with this feature is the following:

1. Iterate over rows in a sheet.
2. For each row, get data from some columns and check its consistency.
3. Save each row as an element of a table in your database (or in a collection, etc.).
4. At the end of the treatment, commit the transaction if no error occurs or rollback all changes if so.

The main benefit of it is to trace all errors / warnings in a unique run. It avoids a boring couple of "run - error on line 84 - run again - error on line 123 - etc.".

Include sub-module

To be able to use the classes we will mention later in this documentation, you need to include the Maven sub-module containing them.

```
<dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-imports</artifactId>
        <version>${igloo.version}</version>
</dependency>
```

Interfaces and implementation

The `org.iglooproject.imports.table.common` module is currently divided in 3 parts:

- `common`: all interfaces and abstract class common to all implementation ;
- `apache.poi`: implementation to read Excel format using the Apache POI API ;
- `opencsv`: implementation to read CSV format using Opencsv.

Using it in your application

Notes about the sample coming:

1. We will use the `apache.poi` implementation. However, it should be really close of it to deal with a CSV file.
2. We will have a main class `CarDataUpgrade` containing all other classes. If you project architecture needs to do something different, you are obviously free to make separate classes.

Declare you columns

First of all, you need to declare how your data file is structured. Each column defines :

- its position (0 based) ;
- its type.

```
private static final class CarSheetColumnSet extends ApachePoiImportColumnSet {
        private final Column<Long> id = withIndex(0).asLong().build();
        private final Column<String> brand = withIndex(3).asString().build();
        private final Column<String> referenceName = withIndex(4).asString().build();
        private final Column<Date> firstSellingDate = withIndex(5).asDate().build();
}
```

In the definition above, we can see that columns at positions 1, 2 and 3 are ignored. They may contains informative data, or whatever else, and don't concern the data import. We just ignore them in our structure declaration.

Iterate over sheets

After that, you will need to use an `IExcelImportFileScanner` to iterate over sheets in the Excel file. Our sample contains only one sheet, but you could have a more complex workbook and you could deal with it making different cases inside the `visitSheet()` method.

```
@Autowired
private ITransactionScopeIndependantRunnerService␣
↪transactionScopeIndependantRunnerService;

[...]

private final class CarExcelFileImporter {
        private final ApachePoiImportFileScanner scanner = new␣
↪ApachePoiImportFileScanner();

        private final CarSheetColumnSet carSheetColumnSet = new CarSheetColumnSet();

        public void doImportExcelFile(InputStream stream, String fileName) throws␣
↪TableImportException {
                scanner.scan(stream, fileName, SheetSelection.ALL, new␣
↪IExcelImportFileVisitor<Workbook, Sheet, Row, Cell, CellReference>() {
                        @Override
                        public void visitSheet(final ITableImportNavigator<Sheet, Row,␣
↪Cell, CellReference> navigator, Workbook workbook, final Sheet sheet)
                                        throws TableImportException {
                                transactionScopeIndependantRunnerService.run(false, new␣
↪Callable<Void>() {
                                        @Override
                                        public Void call() throws Exception {
                                                ITableImportEventHandler eventHandler =␣
↪new LoggerTableImportEventHandler(LOGGER);
                                                importCarSheet(navigator, sheet,␣
↪eventHandler);
                                                eventHandler.checkNoErrorOccurred();
```

(continues on next page)

(continued from previous page)

```
                                                return null;
                                        }
                                });
                        }
                });
        }

        [...]
}
```

Transaction

Please note that we use an `ITransactionScopeIndependantRunnerService` to be sure that all database actions are performed in a unique transaction. It allows us to log all potential errors and rollback all changes only at the end of the Excel sheet.

Event handler

The `ITableImportEventHandler` allow the import process we build to log some messages about the treated data. We can initialize it with a `TableImportNonFatalErrorHandling` mode:

- `THROW_ON_CHECK` (default) will throw an exception when the `checkNoErrorOccurred()` is called ;
- `THROW_IMMEDIATELY` will throw an exception when the event is handle ; following rows are not treated.

Iterates overs rows

Now that we are in a sheet, we can iterate over its rows. We can do it simply like shown below.

```
private void importCarSheet(ITableImportNavigator<Sheet, Row, Cell, CellReference>␣
↪navigator,
                Sheet sheet, ITableImportEventHandler eventHandler) throws␣
↪TableImportContentException, TableImportMappingException {
        CarSheetColumnSet.TableContext sheetContext = carSheetColumnSet.map(sheet,␣
↪navigator, eventHandler);

        for (CarSheetColumnSet.RowContext rowContext : Iterables.skip(sheetContext, 1)) {
                CarSheetColumnSet.CellContext<Long> idCell = rowContext.
↪cell(carSheetColumnSet.id);
                CarSheetColumnSet.CellContext<String> brandCell = rowContext.
↪cell(carSheetColumnSet.brand);
                CarSheetColumnSet.CellContext<String> referenceNameCell = rowContext.
↪cell(carSheetColumnSet.referenceName);
                CarSheetColumnSet.CellContext<Date> firstSellingDateCell = rowContext.
↪cell(carSheetColumnSet.firstSellingDate);

                Long idFromXls = idCell.getMandatory("Car id is mandatory.");
                String brandFromXls = brandCell.getMandatory("Brand is mandatory");
                String referenceNameFromXls = referenceNameCell.get();
                Date firstSellingDateFromXls = firstSellingDateCell.get();

                Car car = carService.getById();
                if (car != null) {
                        // The id cannot be found, the car will not be updated
```

(continues on next page)

(continued from previous page)

```
                        idCell.error("Car {} not found.", idFromXls);
                        continue;
                }

                if (firstSellingDateFromXls != null && firstSellingDateFromXls.after(new␣
↪Date())) {
                        // The first selling date should be wrong, but it's a secondary␣
↪information,
                        // the car will be updated with this information
                        firstSellingDateCell.warn("Car {} - The first selling date ({})␣
↪is in the future.", firstSellingDateFromXls);
                }

                car.setBrand(brandFromXls);
                car.setReferenceName(brandFromXls);
                car.setFirstSellingDate(firstSellingDateFromXls);

                try {
                        carService.update(car);
                } catch (ServiceException | SecurityServiceException e) {
                        LOGGER.error("An error occured while updating a car.", e);
                        rowContext.error("An error occured while updating a car.");
                }
        }
}
```

Getting values

You can handle some basic behavior while getting values:

- treat result with some functions at column description like `withDefault()`, `extract()` or `capitalize()`
- raise an error in case of missing value with `getMandatory()` instead of a simple `get()`

Error location

Please note that using cell or row context to record logs will produce messages with precise location details (i.e.:

```
(at TableImportLocation[fileName=my-pretty-cars.xlsx,tableName=cars,rowIndex (1-
↪based)=123,cellAddress=F123])
```

Eating the data file

Obviously, we need to give the file to our data import mechanic. Here we get this file from the project's resources, but we also could get it from the file system, from a user input, etc.

```
public class CarDataUpgrade implements IDataUpgrade {

        private static final Logger LOGGER = LoggerFactory.getLogger(CarDataUpgrade.
↪class);

        private static final String FILE_PATH = "/dataupgrade/my-pretty-cars.xlsx";
```

(continues on next page)

(continued from previous page)

```
        [...]

        public void perform() throws TableImportException {
                InputStream inputStream = null;
                try {
                        LOGGER.info("Car import...");
                        inputStream = CarDataUpgrade.class.getResourceAsStream(FILE_
↪PATH);
                        new CarExcelFileImporter().doImportExcelFile(inputStream,␣
↪FilenameUtils.getName(FILE_PATH));
                        LOGGER.info("Car import completed.");
                } finally {
                        IOUtils.closeQuietly(inputStream);
                }
        }
}
```

3.2 UI

3.2.1 Webjars

Webjars integration relies on:

- **webjars-locator-core** (https://github.com/webjars/webjars-locator-core): in default configuration, we strip commons-{codec,compress} and jackson dependencies, as they are used only web wbejars extractor that we do not use.
- **wicket-webjars** (https://github.com/l0rdn1kk0n/wicket-webjars)

You can refer to these projects' documentation for advanced use-cases.

Add a webjar dependency

`jar` dependency type is implied.

```
<dependency>
  <groupId>org.webjars.npm</groupId>
  <artifactId>popper.js</artifactId>
  <version>...</version>
</dependency>
```

Construct a resource reference

Use either `WebjarsJavaScriptResourceReference` or `WebjarsCssResourceReference`. Check webjar content to determine the path to use. The path format is: `<webjar name>[/<version>]/<resource path>`. `current` can be used as a magic placeholder to use the only available webjars without specifying an explicit version.

```
public final class PopperJavaScriptResourceReference extends␣
↪WebjarsJavaScriptResourceReference {

      private static final long serialVersionUID = 1762476460042247594L;

      private static final PopperJavaScriptResourceReference INSTANCE = new␣
↪PopperJavaScriptResourceReference();

      private PopperJavaScriptResourceReference() {
              super("popper.js/current/dist/umd/popper.js");
      }

      public static PopperJavaScriptResourceReference get() {
              return INSTANCE;
      }

}
```

Note: to search available webjars: https://www.webjars.org/. Be aware that webjars can contain only final artifacts (scss compiled or transpiled javascript) or source artifacts (scss or unprocessed javascript). Choose the right resource for your usage.

Sass @import

Webjars resource import with bootstrap is possible with the following url pattern:

`@import "webjars://<webjar name>[/<version>]/<resource path>";`

As an example:

`@import "webjars://bootstrap:4.0.0/scss/variables";` (explicit version)

`@import "webjars://bootstrap:current/scss/variables";` (wicket-webjars *current* magic version)

`@import "webjars://bootstrap/scss/variables";` (no version)

Note: webjar import handling is done with `org.iglooproject.wicket.more.css.scss.service.JSassClassPathImporter`. **current magic version** support is added here to mimic wicket-webjars behavior.

3.2.2 UI Bootstrap

We use Bootstrap 3 as our main source for UI components.

Core

Application

Stylesheets

The basic application comes with 2 stylesheets:

- `styles.less`: the stylesheet for most of the UI
- `service.less`: the stylesheet used for all the login process (sign in, forgot password…)

Variables

You can set the value of the variables by editing the file `<yourapp>/web/common/template/styles/variables.less`.

3.2.3 UI Font Awesome

We use Font Awesome as our main source for icons.

Icon reference: http://fortawesome.github.io/Font-Awesome/icons/

Examples: http://fortawesome.github.io/Font-Awesome/examples/

Please be especially aware of the existence of `fa-fw` to align properly the icons in lists.

3.2.4 UI Plugins (JS, CSS) (TODO)

TODO list plugins and explain (briefly) why and when to use them.

CHAPTER

FOUR

DATABASE

4.1 Flyway model update

Igloo reuses Spring-boot Flyway integration to manage database migrations.

4.1.1 Properties configuration

Basic properties

Before using Flyway's features, you need to configure the following properties:

```
# the schema that Flyway will use
spring.flyway.schemas=X
spring.flyway.default-schema=X
# the table that Flyway will use, by convention `flyway_schema_version`
spring.flyway.table=flyway_schema_version
```

Migration properties

Database model migration can be used to perform :

- schema initialization only
- schema initialization and data import

First use-case is used for integration tests, as data are generally bootstraped by test code (and not by migration scripts).

Second use-case is used for application bootstrap in early realease stages.

Default configurations provided by Igloo use the following paths:

- `db/migration/common/*` : contains schema initialization scripts
- `db/migration/init/*` : contains data initialization scripts

When you want to include both `common/` and `init/` scripts, this property must be modified :

```
# `true`  : import files from `common/` and `init/`
# `false` : only import files from `common/` (default behavior)
migration.init.enabled=true
```

Advanced usage: Spring boot configuration

Spring-boot flyway configuration properties are available here: https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#appendix.application-properties.data-migration

common/init behavior is controlled by an Igloo configuration that manages `flyway.spring.locations` based on `migration.init.enabled` value. If you want to use this mechanism, you must not redefine `flyway.spring.locations`.

If you want to customize `flyway.spring.locations`, it is advised to not use `db.migration` package (keep this classpath empty).

If you want to customize `flyway.spring.locations` and use both SQL and Java migrations, keep in mind that Java migrations you want to be Spring-injection-enabled must not be located in `flyway.spring.locations`; Spring-injection-enabled migrations must be Spring managed beans (declared with `@Bean` or `@Component` annotation).

You may restrict file discovery to sql files by using a pattern in your configuration: `path/**/*.sql`.

4.1.2 Create a Flyway data upgrade

You can write your data upgrades either in SQL or Java. Here we have chosen to :

- put your upgrades .sql in the folder `src/main/resources/db/migration/common/` or `src/main/resources/db/migration/init/`
- put your upgrades .java in the package `db.migration.common` or `db.migration.init` and annotate it with `@Component`
- follow Flyway naming convention: `V<major>_<minor>__<name>.sql/java`. Follow your project's convention to choose major/minor version (it may not match application version).

If you want to specify multiple locations, you have to separate them with commas.

Use SQL scripts to perform schema and data migration that can be performed reliably with SQL scripts.

If you need to use Hibernate API to perform data migrations, you need to create a DataUpgradeRecord entry so that it can be launched once flyway migration and hibernate startup are performed.

4.1.3 Create a SQL formatted data upgrade

Just write your SQL script and place it in the configured migration locations. You can use basic templating with values defined in configuration properties by using Flyway placeholders mechanism.

4.1.4 Create a Java formatted data upgrade

When Flyway migration are applied, Hibernate is not started. You can use Java code that do not use Hibernate API to access data (plain JDBC, JdbcTemplate, ...).

If you want to use Hibernate or other components initialized after flyway, you have to create a `DataUpgradeRecord` row in database. This table is looked up once the Spring context is fully initialized to trigger a DataUpgrade. This DataUpgrade can use Hibernate APIs.

You can use `V1_2__InitDataFromExcel` as an example, and customize the targetted DataUpgrade:

```
@Override
protected Class<? extends IDataUpgrade> getDataUpgradeClass() {
  return DataUpgrade_InitDataFromExcel.class;
}
```

4.1.5 Create an Igloo data upgrade

An Igloo data upgrade is a java class which implements `IDataUpgrade`.

Migration operations are performed by the `perform()` function.

Each Igloo data upgrade need to be registered in `DataUpgradeManagerImpl` bean:

```
@Override
public List<IDataUpgrade> listDataUpgrades() {
  return ImmutableList.<IDataUpgrade>of(
      new DataUpgrade_InitDataFromExcel()
  );
}
```

4.1.6 Igloo 5.1.x Flyway migration

From Igloo 5.1.X (with Flyway >`9.9.0` version), Igloo custom flyway integration is replaced by spring-boot implementation. It allows to use the `spring.flyway.*` properties to control Flyway behavior.

The migration steps are needed to adapt an existing application so that:

- SQL migration are present in the expected locations
- Java migration are now Spring registered bean
- Properties are renamed to the spring-boot equivalents

Property files

1. In `configuration-env-*.properties`, delete `environment.flyway.locations.*` properties:

```
# DELETE
environment.flyway.locations.withInit=org/iglooproject/basicapp/core/config/migration/
↪common,org/iglooproject/basicapp/core/config/migration/init,db/migration/common/
environment.flyway.locations.withoutInit=org/iglooproject/basicapp/core/config/migration/
↪common,db/migration/common/
environment.flyway.locations=
```

2. In `configuration.properties`, you must make the following changes :

```
-flyway.locations=${environment.flyway.locations}

-flyway.schemas=${db.user}
-flyway.table=flyway_schema_version
+spring.flyway.schemas=${db.user}
+spring.flyway.table=flyway_schema_version
```

(continues on next page)

(continued from previous page)

```
+spring.flyway.default-schema=${db.user}
```

3. Check and modify in **all** `.properties` files (choose appropriate case):

```
-environment.flyway.locations=${environment.flyway.locations.withInit}
+migration.init.enabled=true
```

```
-environment.flyway.locations=${environment.flyway.locations.withoutInit}
+migration.init.enabled=false
```

4. Disable flyway migration when using `hibernate.hbm2ddl.auto=create|update` in `configuration*.properties`:

```
spring.flyway.enabled=false
```

Migration files

1. Modify `AbstractDataUpgradeMigration` class according to this commit.
2. Modify your `JAVA` script classes by adding the `@Component` annotation (otherwise they will not be applied). An example here.
3. Move files :
   - From `<project>.core.config.migration.init` to `db.migration.init`
   - From `<project>.core.config.migration.commom` to `db.migration.common`

4.2 SQL script generation

4.2.1 Model - Database comparisons

Note: `*SqlUpdateScriptMain.java` is named at project generation's time: for instance `MyProjectSqlUpdateScriptMain.java`.

The class `<Project>SqlUpdateScriptMain.java` generates the differences between your java model and your database model to write the result as an update sql script. It can also generate an sql script for the creation of your whole database.

To launch this script, make sure you are in the `basic-application/basic-application-core` directory, then execute

```
mvn exec:java -Dexec.mainClass="org.iglooproject.basicapp.core.cli.<Project>
↪ApplicationSqlUpdateScriptMain" -Dexec.args="update stdout"
```

Available args are:

- *[update|create]*: compute create (from scratch) or update (from current situation) SQL script.
- *[stdout|filename]*: output script in console or in provided filename.
- By default, this command outputs update script on stdout

4.2.2 Igloo < 4.0.0

```
mvn exec:java -Dexec.mainClass="org.iglooproject.basicapp.init.<Project>
↪ApplicationSqlUpdateScriptMain" -Dexec.args="arg0 arg1"
```

You need to provide two arguments :

- `arg0` is the mode of the script, you have the choice between `create` for generating your database's creation script, and `update` for generating the update of your database.
- `arg1` is the path to the file which will contain the result script.

4.2.3 SqlUpdateScriptMain not available

If you project does not supply `*SqlUpdateScriptMain`, you can copy and rename the file from package `org.iglooproject.basicapp.core.cli` in you project and adapt them.

4.3 Use a JDNI datasource

A JNDI datasource can be used in place of the application managed pool.

To use a JNDI datasource:

- Application configuration:
  - set `db.datasourceProvider=JNDI` in `configuration.properties`
  - set `db.jndiName=java:comp/env/jdbc/name` in `configuration.properties`; `jdbc/name` is to be replaced consistently with the name of the JNDI resource.
- Container configuration (Tomcat):
  - inside server.xml > `<Context>` element, or in a dedicated or in a `Catalina/localhost/webapp.xml`, add the resource definition. Adapt values to your application.

```
<Context>
      <Resource name="jdbc/name" auth="Container" type="javax.sql.DataSource"
              username="username" password="password"
              driverClassName="org.postgresql.Driver"
              url="jdbc:postgresql://localhost:5432/database"
              maxTotal="30" maxIdle="5" />
</Context>
```

In an Eclipse/WTP environment, you may modify `server.xml` file in the path targetted by `Configuration path:`. Beware that Eclipse overwrite this file when you modify server.xml-related configurations.

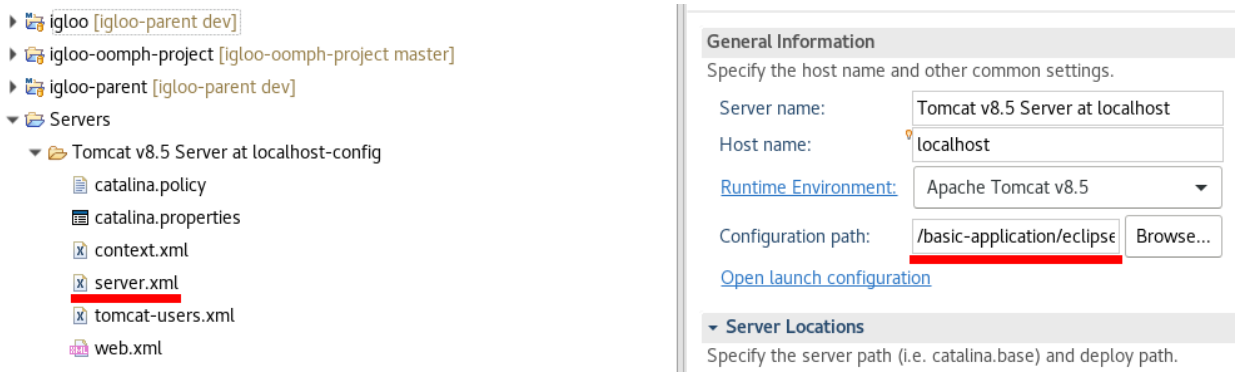

Fig. 1: Configuration path configuration and server.xml file

CHAPTER

FIVE

ASSERTION

5.1 Check non null

It is recommended to use `Objects.requireNonNull` (with `Objects` being the one of Java 8):

```
Objects.requireNonNull(executionResult, "executionResult must not be null");
```

5.2 More advanced conditions

As for more advanced conditions, it is recommended to use Guava's `Preconditions`.

5.3 In Wicket code

In Wicket code, you can use `Args.notNull`, `Args.notEmpty`, `Args.isTrue`, `Args.isFalse`. Be careful to use the `Args` class from Wicket.

CHAPTER

SIX

PREDICATE (TODO)

CHAPTER

SEVEN

RENDERER (TODO)

CHAPTER

EIGHT

TOOLS

8.1 Maven Archetype (TODO)

8.2 Code processors

8.2.1 About

Projects based on Igloo, and Igloo itself, use automatically generated code to manipulate data metamodels.

At the moment, there are two metamodels in an application:

- The bindgen metamodel, which allows us to manipulate objects representing bean properties, and use it to access said properties in java code.
- The QueryDSL metamodel, which allows us to manipulate objects representing entity properties, and use it to build JPA queries.

8.2.2 Troubleshooting

Bindgen's bindings code won't compile

In some cases, bindgen will generate code that won't compile. You may ignore code generation for selected attributes by adding `skipAttribute.your.package.YourClass.yourAttribute` to `bindgen.properties`.

There are known issues with bindgen code generation of some classes in Igloo. You may use the following lines to work around these issues. If those are not up-to-date, check out the basic application's bindgen.properties

```
skipAttribute.org.iglooproject.jpa.business.generic.model.GenericEntityReference.
↪entityClass=true
skipAttribute.org.iglooproject.commons.util.fieldpath.FieldPath.root=true
```

"Cannot find symbol"

If you spot errors like this in your maven build:

```
[ERROR] diagnostic: /data/home/ANONYMIZED/Documents/ANONYMIZED/livraison/tmp/ANONYMIZED-
↪core/src/main/java/com/ANONYMIZED/core/business/document/dao/FileDaoImpl.java:17:␣
↪error: cannot find symbol
import com.ANONYMIZED.core.business.document.model.QDocument;
                                                  ^
  symbol:   class QDocument
  location: package com.ANONYMIZED.core.business.document.model
```

... then just ignore these errors. You are in one of these two situations:

- You generated the bindings for the very first time. In that case, the errors are false positives, and if no other error occurred, the bindings should be generated anyhow.
- Another error occurred during the generation of bindings. In that case, you should check out the very last error, which should be different and is the real cause of your generation failure.

8.3 Maven

8.3.1 JDK level validation

JDK level validation using Animal sniffer is enabled by default from Igloo 0.12 on.

If the default JDK version (1.7 at the time of this writing) does not suit you, you should:

- change the value of the `jdk.version` property to whatever suits you
- and change the value of the `jdk.signature.artifactId` property to match one of the artifacts found here: http://search.maven.org/#search|ga|1|g%3A"org.codehaus.mojo.signature"

If this is somehow impossible and you want to disable these checks completely, you should disable the Animal Sniffer execution with id `check-java-version`.

8.3.2 Deploy to several servers using the maven-deploy-plugin

```
<build>
        <plugins>
                <plugin>
                        <groupId>org.codehaus.mojo</groupId>
                        <artifactId>wagon-maven-plugin</artifactId>
                        <executions>
                                <execution>
                                        <id>upload-war-to-front1</id>
                                        <phase>deploy</phase>
                                        <goals>
                                                <goal>upload-single</goal>
                                        </goals>
                                        <configuration>
                                                <fromFile>${project.build.directory}/$
↪{project.artifactId}-${project.version}-${assembly.environment}.tar.gz</fromFile>
```

(continues on next page)

(continued from previous page)

```
                                        <url>${front1-deployment-url}</url>
                                    </configuration>
                                </execution>
                                <execution>
                                    <id>upload-war-to-front2</id>
                                    <phase>deploy</phase>
                                    <goals>
                                        <goal>upload-single</goal>
                                    </goals>
                                    <configuration>
                                        <fromFile>${project.build.directory}/$
↪{project.artifactId}-${project.version}-${assembly.environment}.tar.gz</fromFile>
                                        <url>${front2-deployment-url}</url>
                                    </configuration>
                                </execution>
                            </executions>
                        </plugin>
                    </plugins>
                </build>
```

8.4 Owasp Dependency-Check - Versions maven plugin

Since version 1.24 you have some reporting tools for maven dependencies and plugins.

8.4.1 Owasp Dependency-Check

Owasp Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

Its maven profile is defined in both pom.xml and igloo/igloo-parents/igloo-parent-maven-configuration-common/pom.xml.

To run the dependency-check, use the option `owasp.enabled`. For example you can launch :

```
mvn -Dowasp.enabled=true -U -DskipTests clean install site:site -DnvdApiKey=XXXXXXXXX
```

NVD API Key can be created on https://nvd.nist.gov/developers/request-an-api-key.

After running your command, you can read the report in *project-name/target/site/dependency-check-report.html*.

The dependency check may sometimes identify irrelevants vulnerabilities. To suppress them, copy the xml generated in the report, and add it in both files owasp-suppressions.xml.

If targetted project pom does not inherit igloo-maven, like spring-boot-starter modules, you can invoke dependency checker with the alternate command:

8.4.2 Versions maven plugin

The versions maven plugin is used when you want to manage the versions of artifacts in a project's POM. We use it exclusively for its reporting goals.

Its maven profile is defined in both pom.xml and igloo/igloo-parents/igloo-parent-maven-configuration-common/pom.xml.

To run the versions maven plugin, use the option `update-report.enabled`. For example you can launch :

```
mvn -Dupdate-report.enabled=true -U -DskipTests clean install site:site && mvn -Dupdate-
↪report.enabled=true site:stage
```

When the command has ended (it can take a while), you can find the reports in *project-name/target/site/index.html*.

8.5 Igloo Spring Boot

Since version 1.24, basic-application java configuration is mostly managed with a custom spring-boot auto configuration annotation.

8.5.1 Spring-boot autoconfiguration

If you want to know more about how to declare your own spring-boot auto configuration annotation and how it works, you can read about it here.

The short explanation is, it works with a package of `@Configuration` classes, each with `@Conditional` annotations to constrain when the auto configuration should apply.

8.5.2 @EnableIglooAutoConfiguration

Our custom spring-boot class is named `@EnableIglooAutoConfiguration`. All the classes gathered under this annotation are located in the package org.igloo.spring.autoconfigure.

The class which contains `@EnableIglooAutoConfiguration` in the basic-application configuration is BasicApplicationCoreCommonConfig.

The spring-boot auto configuration is triggered after the declarative java configuration, which means that all the @Conditional parameters will be checked after you have manually declared what you want.

You can enable auto configuration report by adding `debug=true` in the application.properties file. This will display all the positive and negative matches made by spring-boot when verifying your conditional parameters.

8.5.3 Details on specific auto configurations

We have declared under the annotation `@EnableIglooAutoConfiguration` a lot of beans mandatory for the basic-application, but we have left some beans that you have to declare manually. We have also declared some beans with naive implementations in order to make it easy to begin a new application, but it is not safe to keep them and they must be override.

For instance, in the Property auto configuration we have declared a basic implementation of a IPropertyRegistryConfig bean, but it will not register any properties. You have to declare your own in order to make it works properly.

In the security auto configuration, you have several things to override. First, the application needs the property `security.runAsKey`, then you also need to make your own beans `ICorePermissionEvaluator` and `ISecurityUserService`. Those provided by default are unusable in a real application.

In the TaskManagement auto configuration, you have to provide your own bean `Collection<? extends IQueueId>`.

Finally, in the Wicket auto configuration, you need to declare your specific `WebApplication` bean.

8.5.4 Check autoconfiguration behavior

- Add dedicated spring context initializer : `org.springframework.boot.autoconfigure.logging.ConditionEvaluationReportLoggingListener`
- Configure `org.springframework.boot.autoconfigure` logger to `DEBUG`

For a web spring context, add `org.springframework.boot.autoconfigure.logging.ConditionEvaluationReportLoggingListener` to `contextInitializerClasses` configuration:

```
        <context-param>
                <param-name>contextInitializerClasses</param-name>
                <param-value>org.iglooproject.config.bootstrap.spring.
↪ExtendedApplicationContextInitializer,org.springframework.boot.autoconfigure.logging.
↪ConditionEvaluationReportLoggingListener</param-value>
        </context-param>
```

For a log4j2 configuration, add a logger configuration:

```
logger.autoconfig.name=org.springframework.boot.autoconfigure
logger.autoconfig.level=DEBUG
```

CHAPTER

NINE

SECURITY

9.1 Securing accesses

This page explains how to ensure that parts of your application (pages, buttons, resources, but also Spring services) are only accessible to entitled users.

9.1.1 Principles

Here are some basic principles. These are not really formal security science, but are just intended to provide readers with enough understanding of Igloo's security to get started.

Security model

Here are the concepts used in Igloo's security model :

- A *user* is a user of your application.
- A *user group* is a business-level category of users.
- An *object*, or *resource*, is the thing whose access is to be secured. It's generally a business domain object (such as a "customer", or a "deal", and so on).
- A *role* is a functional-level category of users. One or many roles may be attributed to a user or to a user group.
- A *global permission* is an approval of a mode of access to a class of objects, an authorization which is not tied to a an object in particular (such as "write to customer contact details"). One or many global permissions may be attributed to a role.
- An *object permission* is an approval of a mode of access to an object in particular, an authorization which **is** tied to a an object in particular (such as "write to the contact details of customer Initech, Inc"). An object permission is never attributed, it is computed in a fully qualified context: given a user, an object and an object permission, the security system will compute the answer to the question "does the user have this permission on this object?".

You may notice that, depending on your point of view, some concepts seem to have the same purpose: either role and global permission or role and user group. The developers are aware of this issue and it will be addressed in a future version of Igloo.

Architecture

Igloo's security layer is powered by Spring Security, and most concepts detailed here come directly from Spring Security.

Here are the main components of a secured application based on Igloo:

- The `ISecurityService` is an API, the main entry point for security-related queries ("does this user have this permission on this object") and operations (user authentication, authentication invalidation, ...).
- The `PermissionEvaluator` is a SPI, the way for the developer to programmatically define code that will determine whether a user has a given permission on a given object. This code is not just a mapping, as it may use business object's properties in order to answer queries: for instance "is the given user this customer's account manager?". **Think of permissions evaluators as a way to extract business information for security purposes**. The permission evaluator is generally implemented through a subclass of `AbstractCorePermissionEvaluator` which delegates its calls to various `IGenericPermissionEvaluator`, one for each type of object.
- The `UserDetailsService` is a SPI, the way for the developer to programmatically define code that will determine whether a user has a given role or global permission. It's generally a subclass of `CoreJpaUserDetailsServiceImpl`. **Think of the user details service as a way to extract user role and permission attributions.** This code consists, most of the time, in:
- extracting role and global permission attributions from the database
- expanding the results to extensive lists with the help of a role hierarchy and a permission hierarchy
- and optionally inferring hard-coded global permissions based on on attributed roles

9.1.2 Defining your own permissions

Defining role constants

Create a class named <YourApplication>AuthorityConstants, extending CoreAuthorityConstants, and put it in <Your main package>.core.security.model. In this class, add one string constant for each role, making sure that each constant has a unique value:

```
public final class MyApplicationAuthorityConstants extends CoreAuthorityConstants {

        public static final String ROLE_INTRANET_USER = "ROLE_INTRANET_USER";
        public static final String ROLE_EXTRANET_USER = "ROLE_EXTRANET_USER";

}
```

Defining permission constants

Create a class named <YourApplication>PermissionConstants, extending CorePermissionConstants, and put it in <Your main package>.core.security.model. In this class, add one string constant for each global permission or object permission, making sure that each constant has a unique value:

```
public final class MyApplicationPermissionConstants extends CorePermissionConstants {
        public static final String CUSTOMER_WRITE_CONTACT_DETAILS = "CUSTOMER_WRITE_
↪CONTACT_DETAILS";
}
```

Then, create another class named <YourApplication>Permission, extending NamedPermission, and with the following implementation:

```
public final class MyApplicationPermission extends NamedPermission {

        private static final long serialVersionUID = 8541973919257428300L;

        public static final Collection<MyApplicationPermission> ALL;
        static {
                ImmutableSet.Builder<SIPermission> builder = ImmutableSet.builder();
                Field[] fields = MyApplicationPermissionConstants.class.getFields();
                for (Field field : fields) {
                        try {
                                Object fieldValue = field.get(null);
                                if (fieldValue instanceof String) {
                                        builder.add(new␣
↪MyApplicationPermission((String)fieldValue));
                                }
                        } catch (IllegalArgumentException|IllegalAccessException␣
↪ignored) { // NOSONAR
                        }
                }
                ALL = builder.build();
        }

        private MyApplicationPermission(String name) {
                super(name);
        }

}
```

And finally, override permissionFactory in your security configuration class, which extends `AbstractJpaSecurityConfig`. Here is an implementation example:

```
@Configuration
public class MyApplicationCoreSecurityConfig extends AbstractJpaSecuritySecuredConfig {

        /** ... other stuff ... */

        @Override
        public PermissionFactory permissionFactory() {
                return new NamedPermissionFactory(MyApplicationPermission.ALL);
        }

        /** ... other stuff ... */
}
```

Defining role and permission hierarchies

This is done by overriding `roleHierarchyAsString` and `permissionHierarchyAsString` in your security configuration class, which extends `AbstractJpaSecurityConfig`. Here is an implementation example:

```
import static my.application.core.security.model.MyApplicationAuthorityConstants.*;
import static my.application.core.security.model.MyApplicationPermissionConstants.*;

@Configuration
public class MyApplicationCoreSecurityConfig extends AbstractJpaSecuritySecuredConfig {

	/** ... other stuff ... */

	@Override
	public String roleHierarchyAsString() {
		return defaultRoleHierarchyAsString() + hierarchyAsStringFromMap(
				ImmutableMultimap.<String, String>builder()
				.putAll(
						ROLE_ADMIN,
						ROLE_INTRANET_USER,
						ROLE_TECHNICAL_ADMIN
				)
				.putAll(
						ROLE_INTRANET_USER,
						ROLE_AUTHENTICATED
				)
				.putAll(
						ROLE_EXTRANET_USER,
						ROLE_AUTHENTICATED
				)
				.putAll(
						ROLE_SYSTEM,
						ROLE_ADMIN,
						ROLE_MAIN_USER,
						ROLE_EXTRANET_USER
				)
				.build()
		);
	}

	@Override
	public String permissionHierarchyAsString() {
		return defaultPermissionHierarchyAsString() + hierarchyAsStringFromMap(
				ImmutableMultimap.<String, String>builder()
				.put(CUSTOMER_WRITE, CUSTOMER_READ)
				.build()
		);
	}

	/** ... other stuff ... */
}
```

Defining permission evaluators

Then, go to your permission evaluator. You may find a reference to this class in your configuration class that extends `AbstractJpaSecurityConfig`, in the `permissionEvaluator` method.

In this permission evaluator, you will have to dispatch security queries to various permission evaluators, one for each object type. This will look like this:

```
public class MyApplicationPermissionEvaluator extends AbstractCorePermissionEvaluator
↪<User> {

        @Autowired
        private ICustomerPermissionEvaluator customerFormationPermissionEvaluator;

        @Autowired
        private IDealPermissionEvaluator dealPermissionEvaluator;

        @Autowired
        private IInvoicePermissionEvaluator invoicePermissionEvaluator;

        public MyApplicationPermissionEvaluator() {
                // nothing to do
        }

        @Override
        protected boolean hasPermission(User user, Object targetDomainObject, Permission␣
↪permission) {
                if (targetDomainObject != null) {
                        targetDomainObject = HibernateUtils.unwrap(targetDomainObject); /
↪/ NOSONAR
                }

                if (user != null) {
                        user = HibernateUtils.unwrap(user); // NOSONAR
                }

                if (targetDomainObject instanceof Customer) {
                        return customerPermissionEvaluator.hasPermission(user,␣
↪(Customer) targetDomainObject, permission);
                } else if (targetDomainObject instanceof Deal) {
                        return dealPermissionEvaluator.hasPermission(user, (Deal)␣
↪targetDomainObject, permission);
                } else if (targetDomainObject instanceof Invoice) {
                        return invoicePermissionEvaluator.hasPermission(user, (Invoice)␣
↪targetDomainObject, permission);
                }


                return false;
        }
}
```

For each type-bound permission evaluator, you will define an interface (which extends `IGenericPermissionEvaluator`) and an implementation. Here is an example of implementation (you are, of course, totally free of which permissions you will or will not handle):

```
@Service
public class CustomerPermissionEvaluatorImpl extends␣
↪AbstractMyApplicationGenericPermissionEvaluator<Customer>
                implements ICustomerPermissionEvaluator {

        @Autowired
        private IUserService userService;

        @Autowired
        private IParticipationPermissionEvaluator participationPermissionEvaluator;

        @Override
        public boolean hasPermission(User user, Customer customer, Permission␣
↪permission) {
                if (is(permission, READ)) {
                        return hasPermission(user, CUSTOMER_READ);
                } else if (is(permission, CREATE)) {
                        return hasPermission(user, CUSTOMER_CREATE);
                } else if (is(permission, WRITE)) {
                        return user.equals(customer.getAccountManager());
                }
                return false;
        }
}
```

9.1.3 Restricting accesses

Service layer

General configuration

In order to enable security checks upon method calls, you will need to make sure that your security configuration class does not extend `AbstractJpaSecuritySecuredConfig` directly, but its subclass, `AbstractJpaSecuritySecuredConfig`.

Service access

You will need to add annotations on your services' methods. For instance: s

```
public interface ICustomerService extends IGenericEntityService<Long, Customer> {

        @Override
        @PreAuthorize(value = MyAppSecurityExpressionConstants.CREATE)
        void create(@PermissionObject Customer entity) throws ServiceException,␣
↪SecurityServiceException;

}
```

`@PreAuthorize` will perform a security check before executing the method. Other, more exotic annotations exist in package `org.springframework.security.access.prepost`.

It's better to define your security expressions in a separate constants class, such as `MyAppSecurityExpressionConstants` in this example. This class will look something like that:

```
import static org.iglooproject.commons.util.security.PermissionObject.DEFAULT_PERMISSION_
↪OBJECT_NAME;

public final class SISecurityExpressionConstants {

        public static final String READ = "hasPermission(#" + DEFAULT_PERMISSION_OBJECT_
↪NAME + ", '" + SIPermissionConstants.READ + "')";
        public static final String CREATE = "hasPermission(#" + DEFAULT_PERMISSION_
↪OBJECT_NAME + ", '" + SIPermissionConstants.CREATE + "')";
        public static final String WRITE = "hasPermission(#" + DEFAULT_PERMISSION_OBJECT_
↪NAME + ", '" + SIPermissionConstants.WRITE + "')";

}
```

Note that annotating the method's main parameter with `@PermissionObject` and using `PermissionObject.DEFAULT_PERMISSION_OBJECT_NAME` in your security expressions will ensure that changing the name of this parameter will not break your security expressions.

UI layer

General configuration

The general web application security configuration is generally located in a class named `<YourApp>WebappSecurityConfig`.

This class generally refers to an XML file (`security-web-context.xml`) whose content defines:

- security-related beans
- required roles for each page (or set of pages, by using regular expressions)
- login workflow (login page, login failure page, login success page)
- denied access behavior
- session restrictions (such as a maximum number of simultaneous sessions)

The official documentation about the format of this file may be found there: http://docs.spring.io/spring-security/site/docs/4.0.x/reference/html/ns-config.html

Page and resource access

Simple, coarse-grained configuration

You may define, for a given page or resource, which roles or global permissions are required in order to access it.

This is simply done by adding the `@AuthorizeInstantiation` (for roles) or `@AuthorizeInstantiationIfPermission` (for global permissions) on the page's class. For resources, you must use `@AuthorizeResource` instead, and you may not rely on permissions (only roles).

Be aware that, while more annotations are available (`@AuthorizeAction` and `@AuthorizeActionIfPermission` in particular), their use is discouraged because they add restrictions which cannot be checked until the very last moment.

This prevents in particular from disabling links to inaccessible pages (because, when rendering the link, the page is not yet instantiated and thus we can't check action permissions on this page).

Advanced, fine-grained configuration

Most of the time, you will use link descriptors (see *UI-Links*) in order to provide access to pages or resources.

Link descriptors allow to define arbitrary access restrictions (based on model objects, or on anything you want), and this includes in particular authorization restrictions.

For instance, this allows to make a link accessible only when the users has the "READ" permission on the parameter:

```
public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor, MyObject>␣
↪MAPPER =
                new LinkDescriptorBuilder()
                .page(MyObjectPage.class)
                .model(MyObject.class)
                                .permission(CorePermissionConstants.READ)
                                .map(CommonParameters.ID).mandatory()
                .build();
```

You may also enforce checks on global permissions, by calling the `.permission` method before defining any parameter:

```
public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor, MyObject>␣
↪MAPPER =
                new LinkDescriptorBuilder()
                .page(MyObjectPage.class)
                .permission(MyApplicationPermissionConstants.ACCESS_MY_OBJECT_PAGE)
                .model(MyObject.class)
                                .map(CommonParameters.ID).mandatory()
                .build();
```

Please note that all of this also applies to resource link descriptors.

With this configuration, checks will be performed upon link rendering and upon page/resource instantiation:

- when links are rendered, they will be automatically disabled or hidden if the user misses some roles or permissions
- when the a page or resource is instantiated, it will use the link descriptor to extract parameters, which will trigger an exception and abort the page instantiation if the user misses some roles or permissions.

Buttons/links access

When using links created from a link descriptor, if this link descriptor has been properly configured as explained above, the link will automatically be disabled whenever the user hasn't the required permissions.

For other links (external links for instance) or for buttons, ajax links, and so on, you may hide or disable these components using enclosure behaviors:

```
final IModel<T> model = /*...*/;
add(
                new AjaxLink("link", model) {
                        /* ... */
                }
```

(continues on next page)

(continued from previous page)

```
                                .add(
                                                new EnclosureBehavior().condition(
                                                                Condition.
↪permission(model, MyApplicationPermissionConstants.MY_PERMISSION)
                                                )
                                )
);
```

This will trigger server-side hiding, which will prevent users to trigger the server-side code even if they can guess and call the URL for each button: Wicket refuses to execute code on components that were hidden on the server side.

Popups/modals access

For modals which require initialization before showing them, you should add an enclosure behavior on the opening link:

```
MyModal editPopup = new MyModal("popup");
add(editPopup);

// The following code is potentially executed multiple times, for different models
final IModel<T> itemModel = /*...*/;
add(
                new BlankLink("edit")
                                .add(
                                        new AjaxModalOpenBehavior(editPopup, MouseEvent.
↪CLICK) {
                                                private static final long␣
↪serialVersionUID = 1L;

                                                @Override
                                                protected void onShow(AjaxRequestTarget␣
↪target) {
                                                        editPopup.init(itemModel.
↪getObject());
                                                }
                                        }
                                )
                                .add(
                                                new EnclosureBehavior().condition(
                                                                Condition.
↪permission(model, MyApplicationPermissionConstants.MY_PERMISSION)
                                                )
                                )
);
```

This ensures that the modal will be initially visible, but unusable (because it's not initialized), and that it will be "openable" if and only if at least one button is visible.

For modals whose content is fully determined by their main model, and which do not require initialization upon showing them, it is recommended to apply an enclosure behavior on the modal itself:

```
IModel<T> model = /*...*/;
MyModal editPopup = new MyModal("popup", model);
```

(continues on next page)

(continued from previous page)

```
add(
                editPopup
                                .add(
                                                new EnclosureBehavior().condition(
                                                                Condition.
↪permission(model, SIPermissionConstants.WRITE)
                                                )
                                )
                new BlankLink("edit")
                                .add(new AjaxModalOpenBehavior(editPopup, MouseEvent.
↪CLICK))
);
```

This ensures that when the use has no access to the modal, even if the client tries to execute a manually-crafted ajax call to open the modal, the modal will be hidden on the server-side and wicket will thus trigger an error.

CHAPTER

TEN

BACKEND

10.1 Querying

This page explains how to query data using Igloo.

10.1.1 How to expose queries to the web application

Through an `ISearchQuery`

Use case

`ISearchQuery` should be used when providing a search form to users. It makes it easy to define a search query with numerous search criteria that are independent from each other, the ability to sort the result, and the ability to retrieve the paginated result or the result count.

`ISearchQuery` is also commonly used to implement "autocomplete" queries, i.e. the queries behind Select2 select boxes.

Description

`ISearchQuery` is an interface that provides read access to the data while hiding the implementation details, as does a DAO. But on contrary of a DAO:

- Each interface extending `ISearchQuery` provides one method for each search criterion, which will be kept in memory until data retrieval (`list`, `count`). Note that this "keeping in memory" might not be done explicitly by implementors, but just by starting the query building with the query framework under the hood.
- For this reason, an `ISearchQuery` instance is stateful and can only be used for **one** query (i.e. search criteria may be added, but not removed nor cleared).
- `ISearchQuery`s are expected to be used directly from the UI layer

Here is an example of `ISearchQuery`-extending interface:

```
public interface IPersonSearchQuery extends ISearchQuery<Person, PersonSort> {

        IPersonSearchQuery quickSearch(String filter);

        IPersonSearchQuery lastName(String lastName);

        IPersonSearchQuery firstName(String firstName);
```

(continues on next page)

(continued from previous page)

```
        IPersonSearchQuery company(Company company);
}
```

And here is an example of use (from inside a Wicket DataProvider):

```
return createSearchQuery(IPersonSearchQuery.class) // Some Spring magic (beanFactory.
↪getBean(...))
                .lastName(lastNameModel.getObject())
                .firstName(firstNameModel.getObject())
                .company(companyModel.getObject())
                .sort(sortModel.getObject())
                .list(offset, limit)
```

Through a service

Use case

Queries should be exposed to the web application through services (a simple method in a service) when:

- counting the results is not necessary (within a service, that would involve providing two service methods with the same parameters and implementing them, which is a bit of a pain)
- **and**
- paging is not necessary
- **or** criteria are strongly interrelated (they can't be implemented by separate criteria in the underlying query)
- **or** the query is really too simple to justify the overhead of creating an interface and an implementation class

Note that in any of these case, `ISearchQuery` *could* still be used. It's just a matter of guessing whether using `ISearchQuery` would help in implementing your query or not (spoiler: it probably does).

Also, know that in the case of complex queries (reporting for instance), a `ISearchQuery` is still (and maybe more) relevant, since you may just store arguments passed to criteria methods in attributes and use those attributes later when asked for the results. This brings the advantage of consistency with little implementation cost.

Exposing sort selection

Whatever the solution you choose among the two above, you may have to provide clients a way to tune the sorting of retrieved data.

In Igloo, this is generally done by adding a parameter to your query that is a `Map<S, SortOrder>` with `S extends ISort<F>` and with `F` being implementation-dependent. `ISort` will allow the implementor to convert the business-level sort definitions into an internal list of fields on which to sort.

`ISorts` are simple business wrappers. Each `ISort` instance by provides a list of sort "fields" (whose type is implementation-dependent).

The implementations are generally an enum type:

```
public enum PersonSort implements ISort<SortField> {

        SCORE {
```

(continues on next page)

(continued from previous page)

```
                @Override
                public List<SortField> getSortFields(SortOrder sortOrder) {
                        return GenericEntitySort.SCORE.getSortFields(sortOrder);
                }
                @Override
                public SortOrder getDefaultOrder() {
                        return GenericEntitySort.SCORE.getDefaultOrder();
                }
        },
        ID {
                @Override
                public List<SortField> getSortFields(SortOrder sortOrder) {
                        return GenericEntitySort.ID.getSortFields(sortOrder);
                }
                @Override
                public SortOrder getDefaultOrder() {
                        return GenericEntitySort.ID.getDefaultOrder();
                }
        },
        LAST_NAME {
                @Override
                public List<SortField> getSortFields(SortOrder sortOrder) {
                        return ImmutableList.of(
                                        SortUtils.luceneSortField(
                                                        this, sortOrder, SortField.Type.
↪STRING,
                                                        Ressortissant.LAST_NAME_SORT
                                        )
                        );
                }
                @Override
                public SortOrder getDefaultOrder() {
                        return SortOrder.ASC;
                }
        },
        FIRST_NAME {
                @Override
                public List<SortField> getSortFields(SortOrder sortOrder) {
                        return ImmutableList.of(
                                        SortUtils.luceneSortField(
                                                        this, sortOrder, SortField.Type.
↪STRING,
                                                        Ressortissant.FIRST_NAME_SORT
                                        )
                        );
                }
                @Override
                public SortOrder getDefaultOrder() {
                        return SortOrder.ASC;
                }
        },
        FULL_NAME {
```

(continues on next page)

(continued from previous page)

```
                @Override
                public List<SortField> getSortFields(SortOrder sortOrder) {
                        return ImmutableList.of(
                                        SortUtils.luceneSortField(
                                                        this, sortOrder, SortField.Type.
↪STRING,
                                                        Ressortissant.LAST_NAME_SORT
                                        ),
                                        SortUtils.luceneSortField(
                                                        this, sortOrder, SortField.Type.
↪STRING,
                                                        Ressortissant.FIRST_NAME_SORT
                                        )
                        );
                }
                @Override
                public SortOrder getDefaultOrder() {
                        return SortOrder.ASC;
                }
        };

        @Override
        public abstract List<SortField> getSortFields(SortOrder sortOrder);

        @Override
        public abstract SortOrder getDefaultOrder();

}
```

Note that, on the UI side, an utility exists to easily manage a sort selection: `CompositeSortModel`. See *UI-Models* for more information.

10.1.2 How to implement queries

Search queries (`ISearchQuery`)

You may always use your own implementation. But in most cases, extending one of the two provided abstract classes is the way to go.

WARNING: always think to add `@Scope("prototype")` to your implementation, else you will experience very disturbing concurrent modification issues.

AbstractHibernateSearchSearchQuery

`AbstractHibernateSearchSearchQuery` provides sensible protected methods that allow you to stack criteria on each call of a criterion method. For convenience, most of those utility methods have no effect when given `null` parameters. This allow clients to skip null-checks entirely and to call your criteria methods regardless of whether or not the users provided a value for each parameter.

Some full implementations already exist in Igloo (most notably for `org.iglooproject.jpa.more.business.generic.query.ISimpleGenericListItemSearchQuery<T, S>`).

The following assumes that Lucene field have already been defined on your entities. If not, see *Hibernate Search & Lucene*.

Simple match

```
	@Override
	public IPersonSearchQuery company(Company company) {
		must(matchIfGiven(Person.COMPANY /* Lucene field name for field "company
↪" */, company));
		return this;
	}
```

Presence of a single item in a collection field

```
	@Override
	public IPersonSearchQuery company(Company company) {
		must(beIncludedIfGiven(Person.COMPANIES /* Lucene field name for field
↪"companies" */, company));
		return this;
	}
```

Presence of at least one item from a set in a collection field

```
	@Override
	public IPersonSearchQuery company(Set<Company> companies) {
		must(matchOneIfGiven(Person.COMPANY /* Lucene field name for field
↪"company" */, companies));
		return this;
	}
```

Presence of all items from a set in a collection field

```
        @Override
        public IPersonSearchQuery companies(Set<Company> companies) {
                must(matchAllIfGiven(Person.COMPANIES /* Lucene field name for field
↪"companies" */, companies));
                return this;
        }
```

Range query

```
        @Override
        public IPersonSearchQuery modificationDate(Date dateMin, Date dateMax) {
                must(matchRange(
                                Person.MODIFICATION_DATE,
                                dateMin,
                                dateMax
                ));

                return this;
        }
```

"OR" operator

WARNING: If you're ORing multiple criterion, the default mechanisms of not applying null criteria may not be enough. You'd better wrap your code in a `if` checking for the presence of arguments.

```
        @Override
        public IPersonSearchQuery modificationDate(Date dateMin, Date dateMax) {
                if (dateMin != null || dateMax != null) { // BEWARE!
                        must(
                                any( // = "OR"
                                        matchRange(
                                                Person.MODIFICATION_DATE,
                                                dateMin,
                                                dateMax
                                        ),
                                        matchNull(Person.MODIFICATION_DATE)
                                )
                        );
                }

                return this;
        }
```

"AND" operator

If you don't have to nest the "AND" in another "OR", you may simply leverage the fact that criteria are ANDed by default:

```
	@Override
	public IPersonSearchQuery noDateInfo() {
		// Implicit "AND"
		must(matchNull(Person.MODIFICATION_DATE));
		must(matchNull(Person.CREATION_DATE));

		return this;
	}
```

Otherwise:

```
	@Override
	public IPersonSearchQuery modificationDate(Date dateMin, Date dateMax) {
		if (dateMin != null || dateMax != null) {
			must(
				any( // = "OR"
					matchRange(
						Person.MODIFICATION_DATE,
						dateMin,
						dateMax
					),
					all( // = "AND"
						matchNull(Person.MODIFICATION_DATE),
						matchNull(Person.CREATION_DATE),
					)
				)
			);
		}

		return this;
	}
```

Other criteria

Many more utility methods are provided in `org.iglooproject.jpa.more.business.search.query.AbstractHibernateSearchSearchQuery<T, S>`. If what you're looking for wasn't above, check out the code.

Overriding utility methods or extending them

If you feel the need to extend this class with additional utility methods, or to override existing utility methods, know that you may do this simply by overriding `org.iglooproject.jpa.more.config.spring.AbstractJpaMoreJpaConfig.hibernateSearchLuceneQueryFactory()` to return your own query factory.

```
@Override
public IMyHibernateSearchLuceneQueryFactory hibernateSearchLuceneQueryFactory() {
        return new MyHibernateSearchLuceneQueryFactoryImpl();
}
```

Then in any search query implementation, the utility methods will be those defined in your own query factory. You may access additional methods with this snippet of code:

```
@Override
protected IMyHibernateSearchLuceneQueryFactory getFactory() {
        return (IMyHibernateSearchLuceneQueryFactory) super.getFactory();
}

@Override
public IMySearchQuery label(String label) throws SearchException {
        must(getFactory().myAdditionalUtilityMethod(
                        /* ... */
        ));
        return this;
}
```

`AbstractJpaSearchQuery`

TODO

Lower-level solutions (service and DAO methods)

JPA querying

TODO QueryDSL-JPA

Native SQL querying

TODO QueryDSL-SQL, Hibernate native SQL

QueryDSL tips

Generating maps and tables

In order to generate a map, use this syntax:

```
return new JPAQuery<>(getEntityManager())
                .from(QUser.user)
                .groupBy(QUser.user.gender)
                .orderBy(QUser.user.gender.asc())
                .transform(GroupBy2.transformer(GroupBy.sortedMap(QUser.user.gender,␣
↪QUser.user.count().intValue())));
```

If you need a `com.google.common.collect.Table<R, C, V>` instead of a `Map`, you may use `GroupBy2.table` or `GroupBy2.sortedTable` instead of `GroupBy.sortedMap`.

If the keys in database are too precise, and you want to perform another aggregation on the Java side (for instance turning day-precise dates into weeks), you may use the following syntax:

```
return new JPAQuery<>(getEntityManager())
                .from(QUser.user)
                .groupBy(QUser.user.gender, QUser.user.creationDate)
                .orderBy(QUser.user.gender.asc(), QUser.user.creationDate.asc())
                .transform(GroupBy2.transformer(GroupBy2.table(
                                QUser.user.gender,
                                new MappingProjection<Date>(Date.class, QUser.user.
↪creationDate) {
                                        private static final long serialVersionUID = 1L;
                                        @Override
                                        protected Date map(Tuple row) {
                                                return DateDiscreteDomain.weeks().
↪alignPrevious(row.get(0, Date.class));
                                        }
                                },
                                /**
                                  * We sum twice: once in the SQL query (for each date)␣
↪and once in Java (for each week).
                                  * We could have summed only in Java, but it would be␣
↪sub-optimal if
                                  * many user are created each day.
                                  * The even better solution would have been to group by␣
↪week in the SQL query,
                                  * but unfortunately it's not easy to do with JPQL.
                                  */
                                GroupBy.sum(QUser.user.count().intValue())
                )));
```

Lucene (Hibernate Search) querying

TODO Hibernate Search DSL

TODO QueryDSL-HibernateSearch?

10.2 Hibernate mappings (TODO)

TODO: some advice about when to use:

- `Enums` or `GenericListItems`
- `@ElementCollection`
- `AbstractHibernateMapBugWorkaroundValueHolder`
- `AbstractMaterializedPrimitiveValue`
- ...

10.3 Hibernate Interceptors

You can declare Spring managed Hibernate interceptors by adding an `hibernateInterceptor()` method in `YourAppCoreCommonJpaConfig`:

```
        @Bean
        public Interceptor hibernateInterceptor() {
                return new ChainedInterceptor()
                                .add(new YourInterceptor());
        }
```

The `ChainedInterceptor` is a class we provide to be able to chain multiple Hibernate interceptors.

For an example of implementation, see:

- https://github.com/igloo-project/igloo-parent/blob/dev/igloo/igloo-components/igloo-component-jpa-externallinkchecker/src/main/java/org/iglooproject/jpa/externallinkchecker/business/interceptor/ExternalLinkWrapperInterceptor.java
- https://github.com/igloo-project/igloo-parent/blob/dev/igloo/igloo-components/igloo-component-jpa-externallinkchecker/src/main/java/org/iglooproject/jpa/externallinkchecker/business/model/ExternalLinkWrapper.java#L178

10.4 Hibernate Search & Lucene (TODO)

TODO:

- Base
- Explain how we should deal with @IndexedEmbedded/@ContainedIn
- Explain how we should deal with sorts

10.4.1 Sorting

Sorting behavior depends on the data type:

- to sort by `id`, you have to use the field `GenericEntity.ID_SORT`
- to sort by string, you should define an additional field with the `TEXT_SORT` analyzer:

```
        @Column
        @Fields({
                @Field(analyzer = @Analyzer(definition = HibernateSearchAnalyzer.TEXT_
↪STEMMING)),
                @Field(name = EN_SORT, analyzer = @Analyzer(definition =␣
↪HibernateSearchAnalyzer.TEXT_SORT))
        })
        @SortableField(forField = EN_SORT)
        private String en;
```

- to sort by date, you don't need an additional field *BUT* you need to sort using `SortField.Type.LONG` (starting from 0.11)

Note that, you need to add a `@SortableField(forField = "fieldName")` annotation for each field used for sorting.

10.5 Cronjobs

Cronjobs tasks can be defined in the `YourAppCoreSchedulingConfig` class.

We use the `@Scheduled` annotation to define the cronjob expression: `@Scheduled(cron = "...")`.

The syntax for the cron expression respects the following rules: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/scheduling/support/CronSequenceGenerator.html .

CHAPTER

ELEVEN

UI

11.1 UI Links

This page explains various ways of creating bookmarkable links to pages or resources using Wicket and Igloo.

11.1.1 What is a bookmarkable link?

We'll define bookmarkable links as any non-ajax link that points directly to a HTML page or to a user download (XLS file, JPEG image, ...).

This excludes in particular the subclasses of `org.apache.wicket.markup.html.link.AbstractLink` that implement event handlers (`onClick` or `onSubmit` methods): they may *redirect* to a HTML page or to a user download, but they don't point to it *directly*. We'll name those *action* links.

The main differences between bookmarkable links and action links are that:

- on the user side, the action link will render as a URL tied to the page from where the link originates, whereas bookmarkable links will renders as a URL tied to the target.
- on the server side, bookmarkable link are a bit lighter to execute than event handlers implementing a redirection (since they require one request instead of two).

For more information on action links, see *UI User Actions*.

11.1.2 Link descriptor

Link descriptors are an addition from Igloo. They offer several advantages over traditional Wicket linking:

- They enforce type-safety: users provide business object models only, and they do not need to perform manual conversion to strings each and every time they create a link.
- They enforce consistency: link generation *and* parameter extraction is done by the same object, which only has to be defined once.
- They provide dynamic link generation. When you add an `AbstractLink` generated by a link descriptor to your page, you are guaranteed that if an underlying model has its value updated, the link will also be updated the next time it is rendered.

Link descriptors are in fact two things: link *generators* and link *parameter extractors*.

As link generators, they allow to generate an URL, or even a Wicket `AbstractLink` tied to predefined models and that will automatically render with an up-to-date URL with each page render.

As link parameters extractors, they allow to take the content of Wicket `PageParameters`, convert it to actual objects (not just primitive types) and store it in predefined models.

Those "predefined models" are in fact models that were *mapped* to HTTP query parameters. See below for details about link descriptor mappers.

Interfaces

The terms "link descriptor" refer to several interfaces:

- `ILinkGenerator` (see above)
- `ILinkParameterExtractor` (see above)
- `ILinkDescriptor` (an extension of both `ILinkGenerator` and `ILinkParameterExtractor`)
- `IPageLinkGenerator`, an extension of `ILinkGenerator` that provides some features relevant only to pages
- `IImageResourceLinkGenerator`, an extension of `ILinkGenerator` that provides some features relevant only to resources providing image files
- `IPageLinkParametersExtractor`, an extension of `ILinkParameterExtractor` that provides some features relevant only to pages

Examples of use

The following examples are about *using* an already-created link descriptor. For information about *creating* a link descriptor, see the section about link descriptor builders below.

URL generation

```
ILinkGenerator linkGenerator = /* ... */;
String relativeOrAbsoluteUrl = linkGenerator.url();
String absoluteUrl = linkGenerator.fullUrl();
```

Wicket link generation

Note: links generated using this method are automatically disabled (no `href`) when they render and their parameters fail validation. You may hide them instead by calling `hideIfInvalid` as below.

```
// Inside a Wicket component's constructor
ILinkGenerator linkGenerator = /* ... */;
add(
                linkGenerator.link("linkWicketId")
                                .hideIfInvalid()
                                .add(new TargetBlankBehavior())
);
```

Redirection

```
IPageLinkGenerator linkGenerator = /* ... */;
throw linkGenerator.newRestartResponseException();
```

<img> markup

```
// Inside a Wicket component's constructor
IImageResourceLinkGenerator linkGenerator = /* ... */;
add(
                linkGenerator.image("linkWicketId")
                                .hideIfInvalid()
                                .add(new TargetBlankBehavior())
);
IPageLinkGenerator linkGenerator = /* ... */;
```

Validity check

Important note: validity check is normally unnecessary, as it will be performed automatically and an exception will be thrown if the link is invalid. Generally, this is want you want, because an invalid link simply should not have been used (Wicket links obtained through `ILinkGenerator#link(String)`, for instance, are automatically disabled when invalid, so the user cannot click them).

If invalid links are a possibility that you want to handle as part of your business code, though, you may use code similar to the following snippet. **This should be exceptional**: if you're doing this extensively in your code, you probably missed something.

```
ILinkGenerator linkGenerator = /* ... */;
if (linkGenerator.isAccessible()) {
        throw linkGenerator.newRestartResponseException();
} else {
        // Fallback code
}
```

11.1.3 Link descriptor mappers

Link descriptor mappers are factories that take models as arguments and map them to previously incomplete link definitions in order to create a link descriptor.

They are primarily useful to separate the definition of links (list of parameters types on the Java, mapping of those Java parameters to HTTP query parameters, validations, ...) from the actual parameter definition. The link descriptor mapper will then represent the incomplete link definition that only lacks parameter models in order to provide a full link descriptor.

Examples of use

Note: the result of a link descriptor mappers' `map` method is a link descriptor that may be used in each and every way described above in the "Link descriptor" section. We only provide one such example here to avoid unnecessary repetitions.

Wicket link generation

```
// Inside a Wicket component's constructor
IModel<User> userModel = /* ... */;
IOneParameterLinkDescriptorMapper<IPageLinkGenerator, User> mapper = /* ... */;
add(
                mapper.map(userModel).link("linkWicketId")
                                .hideIfInvalid()
                                .add(new TargetBlankBehavior())
);
```

Data table link declaration

See *UI-Displaying Collections* for some context about `DataTableBuilder`.

```
IOneParameterLinkDescriptorMapper<IPageLinkGenerator, User> mapper = /* ... */;
CoreDataTablePanel<?, ?> results =
                DataTableBuilder.start(dataProvider, dataProvider.getSortModel())
                .addLabelColumn(new ResourceModel("business.customer.lastName"),␣
↪Bindings.customer().lastName())
                                .withLink(mapper) // <= USE THE MAPPER HERE
                                .withSort(CustomerSort.LASTNAME, SortIconStyle.ALPHABET,␣
↪CycleMode.NONE_DEFAULT_REVERSE)
                                .withClass("text text-sm")
                /** Add some more columns... */
                .build("results");
```

11.1.4 Link descriptor builder

The link descriptor builder allows to build link descriptors or link descriptor mappers. It provides methods to define the mappable Java-side parameters, the mappings between those parameters and HTTP query parameters, the validations around those parameters and the target of the link.

Examples of use

The following sections provide some examples of use. This is not an exhaustive reference, so if those examples do not match exactly your need, you may start from the closest one and use the builder's Javadoc to find what you're looking for.

Simple link descriptor

This is especially useful for pages with no parameters (home page, lists, ...).

```
@AuthorizeInstantiationIfPermission(permissions = {MyPermissionConstants.READ_CUSTOMER})
public class CustomerListPage extends MainTemplate {

        public static final IPageLinkDescriptor linkDescriptor() {
                return LinkDescriptorBuilder.start()
                                .page(CustomerListPage.class);
        }

        public CustomerListPage(PageParameters parameters) {
                super(parameters);
                /* ... */
        }
}
```

Link descriptor mapper

```
public class CustomerDescriptionPage extends MainTemplate {

        public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor,␣
↪Customer> MAPPER =
                        LinkDescriptorBuilder.start()
                        .model(Customer.class).map(CommonParameters.ID).mandatory()
                        .permission(MyPermissionConstants.READ)
                        .page(CustomerDescriptionPage.class);

        public CustomerDescriptionPage(PageParameters parameters) {
                super(parameters);

                IModel<Customer> customerModel = new GenericEntityModel<Long, Customer>
↪();

                MAPPER.map(customerModel)
                                .extractSafely(
                                                parameters,
                                                CustomerListPage.linkDescriptor(),
                                                getString("common.error.unexpected")
                                );

                /* ... */
        }
}
```

Link descriptor mapper with multiple parameters

```
public class CustomerDescriptionPage extends MainTemplate {

        public static final ITwoParameterLinkDescriptorMapper<IPageLinkDescriptor,␣
↪Customer, String> MAPPER_TAB =
                        LinkDescriptorBuilder.start()
                        .model(Customer.class)
                        .model(String.class)
                        .pickFirst().map(CommonParameters.ID).mandatory()
                        .pickSecond().map("tab").optional()
                        .pickFirst().permission(MyPermissionConstants.READ)
                        .page(CustomerDescriptionPage.class);

        public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor,␣
↪Customer> MAPPER =
                        MAPPER_TAB.ignoreParameter2();

        public CustomerDescriptionPage(PageParameters parameters) {
                super(parameters);

                IModel<Customer> customerModel = new GenericEntityModel<Long, Customer>
↪();
                IModel<String> selectedTabNameModel = new Model<>();

                MAPPER_TAB.map(customerModel, selectedTabNameModel)
                                .extractSafely(
                                                parameters,
                                                CustomerListPage.linkDescriptor(),
                                                getString("common.error.unexpected")
                                );

                /* ... */
        }
}
```

Link descriptor mapper with a collection parameter

```
public class MyPage extends MainTemplate {

        public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor, List
↪<Customer>> MAPPER =
                        LinkDescriptorBuilder.start()
                        .<List<Customer>>model(List.class).mapCollection("list",␣
↪Customer.class).mandatory()
                        .permission(MyPermissionConstants.READ)
                        .page(CustomerDescriptionPage.class);

        public MyPage(PageParameters parameters) {
                super(parameters);
```

(continues on next page)

(continued from previous page)

```
                IModel<Customer> customerListModel = CollectionCopyModel.custom(
                                Suppliers2.<Customer>arrayListAsList(),␣
↪GenericEntityModel.<Customer>factory()
                );

                MAPPER.map(customerListModel)
                                .extractSafely(
                                                parameters,
                                                CustomerListPage.linkDescriptor(),
                                                getString("common.error.unexpected")
                                );

                /* ... */
        }
}
```

Link descriptor mapper with custom validation condition

```
public class CustomerDescriptionPage extends MainTemplate {

        public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor,␣
↪Customer> MAPPER =
                        LinkDescriptorBuilder.start()
                        .model(Customer.class).map(CommonParameters.ID).mandatory()
                        .permission(MyPermissionConstants.READ)
                        .validator(DetachableFactories.forUnit(
                                        new AbstractDetachableFactory<IModel<Customer>,␣
↪Condition>() {
                                                private static final long␣
↪serialVersionUID = 1L;
                                                @Override
                                                public Condition create(IModel<Customer>␣
↪parameter) {
                                                        return new␣
↪MyCondition(parameter);
                                                }
                                        }
                        ))
                        .page(CustomerDescriptionPage.class);

        public CustomerDescriptionPage(PageParameters parameters) {
                super(parameters);

                IModel<Customer> customerModel = new GenericEntityModel<Long, Customer>
↪();

                MAPPER.map(customerModel)
                                .extractSafely(
                                                parameters,
                                                CustomerListPage.linkDescriptor(),
```

(continues on next page)

(continued from previous page)

```
                                                    getString("common.error.unexpected")
                                );

                /* ... */
        }
}
```

Other examples

See Igloo's tests, in particular the test methods in `org.iglooproject.test.wicket.more.link.descriptor.AbstractAnyTargetTestLinkDescriptor` and `org.iglooproject.test.wicket.more.link.descriptor.AbstractAnyTargetTestLinkDescriptorMapper`.

11.1.5 Other links

EmailLink

`EmailLink`s is a `mailto:` link that automatically defines its body as the email it points to.

```
IModel<String> emailModel = /* ... */
add(new EmailLink("email", emailModel));
```

11.2 UI Redirecting

This page explains various methods for redirecting from one page to another in you web application.

Please note that we're talking about redirection as part of a server-side process, such as a form that redirects to a different page based on the user input. If you just want a link in your HTML page, please see *UI-Links*.

11.2.1 Redirecting as part of the authentication/authorization process

Basics

When some page is accessed, but the current user has no right to access it (either because the user is not authenticated or he hasn't got the proper authorization), Igloo throws a `org.springframework.security.access.AccessDeniedException`, which is caught by Spring Security's servlet filter. Spring Security then handles this exception with whatever behavior you configured; by default in Igloo, it's a redirection to "/access-denied/", on which the `AccessDeniedPage` is mapped.

`AccessDeniedException`s are thrown:

- When Spring Security detects an unauthenticated access (access to a page without authorization while being unauthenticated).
- When a Wicket's `AuthorizationException` is caught by the `CoreDefaultExceptionMapper`.

Customizing the general behavior

If you want to customize the behavior when an access is denied, you should either:

- change Spring Security's configuration to customize the access denied handler
- map your own page to "/access/denied/"

Customizing the behavior for specific pages

If you want to customize the behavior for specific pages, you may do so:

- at the Wicket level, by declaring your own exception mapper by overriding `org.apache.wicket.Application.getExceptionMapperProvider()` and defining a specific behavior when an `AuthorizationException` is caught. Be aware that this will not cover cases when an access is denied by Spring Security, though, only cases when an access is denied by Wicket itself (due to an annotation on a page, for instance).

  This can be done this way (for instance) in the `map` method of a class extending `CoreDefaultExceptionMapper`:

```
        try {
                if (
                                Exceptions.findCause(e, AccessDeniedException.
↪class) != null
                        ||        Exceptions.findCause(e, AuthorizationException.
↪class) != null
                ) {
                        IPageRequestHandler handler = PageRequestHandlerTracker.
↪getFirstHandler(RequestCycle.get());
                        Class<? extends IRequestablePage> pageClass = handler.
↪getPageClass();
                        PageParameters pageParameters = handler.
↪getPageParameters();
                        Component component = null;
                        if (handler instanceof IComponentRequestHandler &&␣
↪((IComponentRequestHandler) handler).getComponent() instanceof Component) {
                                component = (Component)␣
↪((IComponentRequestHandler) handler).getComponent();
                        }
                        if (
                                        IMyPageWhoseAccessMayBeDenied.class.
↪isAssignableFrom(pageClass)
                                ||        (component != null &&␣
↪IMyPageWithAPopupWhoseAccessMayBeDenied.class.isAssignableFrom(pageClass))
                        ) {
                                Session.get().error(rendererService.localize(
↪"access.denied.customMessage", Session.get().getLocale()));
                                PageParameter parameters = /* ... */;
                                return new RenderPageRequestHandler(new␣
↪PageProvider(MyRedirectPage.class, ));
                        }
        } catch (RuntimeException e2) {
                if (LOGGER.isDebugEnabled()) {
                        LOGGER.error("An error occurred while handling a previous␣
```

(continues on next page)

(continued from previous page)

```
↪error: " + e2.getMessage(), e2);
                }

                // We were already handling an exception! give up
                LOGGER.error("unexpected exception when handling another␣
↪exception: " + e.getMessage(), e);
                return new ErrorCodeRequestHandler(500);
        }

        return super.map(e);
```

- or at the Spring Security level by defining your own access denied handler in Spring Security's configuration

11.2.2 Redirecting to the current page

Full refresh (keeping the same page instance)

If you're in a non-Ajax component, know that handling the request will automatically trigger a full-page refresh. No need for you to do anything.

In the context of an Ajax component you may use this snippet in order to fully refresh the current page:

```
target.add(getPage());
```

Or if you want to completely abort your currently executing code, you may throw an exception:

```
throw new RestartResponseException(getPage());
```

Redirecting to another instance of the same page

In some cases, you will want to redirect to another instance of the same page with the same parameters. This is mostly used when a fatal error occurs.

```
throw new RestartResponseException(getPage().getClass(), getPage().getPageParameters());
```

11.2.3 For any other redirection (most cases)

Redirection is mainly done through exceptions. These come in various flavors, depending on your redirection target.

Please note that `IPageLinkGenerators` (see *UI-Links*) offer methods for easily generating the exception of your choice. This is the recommended way of redirecting.

Here are the main exception types:

- `RestartResponseException` when you simply want to redirect to another page in your Wicket application.
- `RestartResponseAtInterceptPageException` when you want to redirect to another page which will later trigger another redirection to the current page (mainly used for sign-in pages).
- `RedirectToUrlException` when you want to redirect to an external URL (outside of your Wicket application).

You may also encounter the following patterns in Wicket components or pages. These should be avoided, as they only throw an exception but they do not make it clear, neither to you nor to the compiler. Thus you may end up with dead code after your `redirect` call.

```
// AVOID THIS
redirect(MyPage.class);
```

```
// AVOID THIS
redirectToInterceptPage(MyPage.class);
```

Adding an anchor

If you want to point to an anchor on the target page, then you must use a `RedirectToUrlException`. This feature is built in the `IPageLinkGenerator`.

11.3 UI IE 8 Support

A reasonable, transparent IE8 support can be achieved in a Wicket application the following way.

Obviously, you'll still run into slowdowns, bugs and limitations due to IE8 being IE8. But the most visible issues will be gone.

11.3.1 1. Add IE8-specific CSS

Put these files beside your `StylesLessCssResourceReference.java`:

`IE8AndLesserLessCssResourceReference.java`:

```
public final class IE8AndLesserLessCssResourceReference extends LessCssResourceReference
↪{

        private static final long serialVersionUID = 4656765761895221782L;

        private static final IE8AndLesserLessCssResourceReference INSTANCE = new␣
↪IE8AndLesserLessCssResourceReference();

        private IE8AndLesserLessCssResourceReference() {
                super(IE8AndLesserLessCssResourceReference.class, "ie8-and-lesser.less");
        }

        public static IE8AndLesserLessCssResourceReference get() {
                return INSTANCE;
        }

}
```

`ie8-and-lesser.less`:

```
@import "@{scope-core-bs3}bootstrap/variables.less";
@import "@{scope-core-bs3}bootstrap/override/variables.less";
@import "variables.less";
```

(continues on next page)

(continued from previous page)

```
.placeholder { // Added by the "placeholder" jquery polyfill plugin. See jquery.
↪placeholder.js
        color: @input-color-placeholder;
}

.btn-icon-only.btn-placeholder, .btn-placeholder {
        visibility: hidden;
}

/* Put any IE8-specific CSS here */
```

11.3.2 2. Add global listeners

Add this in your Wicket Application's `init` method:

```
IELegacySupport.init(this);
```

With `IELegacySupport.java` being:

```
public final class IELegacySupport {

        private IELegacySupport() {
        }

        public static final String IE_LEGACY_CONDITION = "lte IE 8";

        public static final String IE_UNSUPPORTED_CONDITION = "lt IE 8";

        public static final String IE_8_CONDITION = "IE 8";

        public static final int IE_LEGACY_VERSION = 8;

        public static void init(WebApplication webApplication) {
                webApplication.getComponentInstantiationListeners().add(new␣
↪InstantiationListener());
                webApplication.getAjaxRequestTargetListeners().add(new␣
↪AjaxRequestTargetListener());
        }

        private static class IELegacyHeaderItemsContributorBehavior extends Behavior {

                private static final long serialVersionUID = -1441191136903604013L;

                private final Iterable<HeaderItem> headerItems;

                public IELegacyHeaderItemsContributorBehavior(HeaderItem ...␣
↪headerItems) {
                        this(ImmutableList.copyOf(headerItems));
                }
```

(continues on next page)

(continued from previous page)

```
                public IELegacyHeaderItemsContributorBehavior(Iterable<HeaderItem>␣
↪headerItems) {
                        super();
                        this.headerItems = headerItems;
                }

                @Override
                public void renderHead(Component component, IHeaderResponse response) {
                        WebClientInfo clientInfo = (WebClientInfo) Session.get().
↪getClientInfo();
                        ClientProperties properties = clientInfo.getProperties();
                        if (properties.isBrowserInternetExplorer() && properties.
↪getBrowserVersionMajor() <= IE_LEGACY_VERSION) {
                                for(HeaderItem headerItem : headerItems) {
                                        response.render(headerItem);
                                }
                        }
                }
        }

        private static class InstantiationListener implements␣
↪IComponentInstantiationListener {
                @Override
                public void onInstantiation(Component component) {
                        if (component instanceof Page) {
                                Page page = (Page) component;

                                // Support for the placeholder text of input fields in␣
↪IE8 and lesser
                                page.add(new PlaceholderPolyfillBehavior());

                                page.add(new IELegacyHeaderItemsContributorBehavior(
                                                // Support for media queries in IE8 and␣
↪lesser
                                                JavaScriptHeaderItem.
↪forReference(RespondJavaScriptResourceReference.get()),
                                                // IE8 and lesser specific CSS
                                                CssHeaderItem.
↪forReference(IE8AndLesserLessCssResourceReference.get())
                                ));
                        }
                }
        }

        private static class AjaxRequestTargetListener extends AjaxRequestTarget.
↪AbstractListener {
                @Override
                public void updateAjaxAttributes(AbstractDefaultAjaxBehavior behavior,␣
↪AjaxRequestAttributes attributes) {
                        WebClientInfo clientInfo = (WebClientInfo) Session.get().
↪getClientInfo();
                        ClientProperties properties = clientInfo.getProperties();
```

(continues on next page)

(continued from previous page)

```
                        if (properties.isBrowserInternetExplorer() && properties.
↪getBrowserVersionMajor() <= IE_LEGACY_VERSION) {
                                attributes.getAjaxCallListeners().add(
                                                new AjaxCallListener().onBefore(
                                                                // Prevents placeholder␣
↪text from being submitted
                                                                PlaceholderPolyfillBehavior.
↪disable().render()
                                                )
                                );
                        }
                }

                @Override
                public void onBeforeRespond(Map<String, Component> map,␣
↪AjaxRequestTarget target) {
                        // Refresh the placeholder text (for instance when rendering a␣
↪popup)
                        target.appendJavaScript(PlaceholderPolyfillBehavior.statement().
↪render());
                }

                @Override
                public void onAfterRespond(Map<String, Component> map,␣
↪IJavaScriptResponse response) {
                        // Nothing to do
                }
        }

}
```

11.4 UI Models (TODO)

This page explains the Wicket concept of models and details various types of models that you might use in your applications.

11.4.1 What is an `IModel`?

TODO

Special cases

Collections

For details about how to display collections, and some tips about how to choose the correct interface for accessing you collection data, see *UI-Displaying Collections*.

`IDataProvider`, `ISequenceProvider`

Both `IDataProvider` and `ISequenceProvider` are interfaces designed to provide access to large datasets, with built-in paging.

They

They mainly differ in the way they wrap elements into models. `IDataProvider` exposes a `Iterator<T> iterator(long, long)` method and a `IModel<T> model(T)` method that must be called by clients. This works, but causes trouble when `IDataProvider` implementor wants to always return the same model for the same element (for example because the model carries more mutable information than just a reference to the element).

`ISequenceProvider` solves the issue by directly returning a `Iterator<IModel<T>> iterator(long, long)`. This gives more flexibility to implementors and changes next to nothing (save the interface) for clients.

Please note that in most cases, you should not have to implement an `ISequenceProvider` yourself: simply implementing an `IDataProvider` should do the job. This interface is available for very specific features, such as those implemented in `CollectionCopyModel`.

`ICollectionModel`, `IMapModel`

Those interfaces are implemented by models that:

- provide access to a collection or a map
- implement `ISequenceProvider` so as to always return the same model to wrap the same collection/map element
- optionally, provide write operations (`add`/`put`, `clear`, ...)

Those implementations are noteworthy:

- `CollectionCopyModel`
- `CollectionMapModel`
- `ReadOnlyCollectionModel`
- `ReadOnlyMapModel`

See below on this page for details.

11.4.2 Main use cases

`GenericEntityModel`

TODO

`BindingModel`

TODO

`IBindableModel` et al.

The use case: inter-dependent form components

When designing complex forms, often we have to update some parts of the form whenever a given field changes, even before the form was submitted. This may happen for instance:

- Because some field's proposed values depend on another field's value
- Because of some read-only panel must dynamically display detailed information about a selected value

In any case, you've got what we will call "source" form components (the ones whose value another component depends on) and "target" components (the ones which depend on another form component's value).

That kind of feature is generally achieved by adding an Ajax behavior on the source component that will update the underlying model whenever a client-side change occurs, and refresh the target components.

Why caches are needed

Most of the time, the underlying model is a `BindingModel`, and its root model is a `GenericEntityModel`, which means that the updated value may not be correctly saved for the next requests:

- If the root object (the one we extract the property value from) is an unpersisted entity, then the updated value will be serialized with this unpersisted entity, which might not be a good idea (for instance if the value is itself an entity).
- If the root object is a persisted entity, then on the next request, the root will be loaded from the database and will thus have its properties reset.

In the particular case where each "target" component depends on only one "source" component, and no other Ajax refreshes are performed, then it's fine, because we won't need the updated value again.

But let's say one "target" component depends on multiple "source" components, say `C` depends on `A` and `B`. We've got the following code:

```
// DON'T DO THIS, IT WON'T WORK AS EXPECTED

IModel<MyEntity> rootModel = /* ... */;
IModel<MyEntity2> propertyAModel = BindingModel.of(rootModel, Bindings.myEntity().
↪propertyA());
IModel<MyEntity3> propertyBModel = BindingModel.of(rootModel, Bindings.myEntity().
↪propertyB());

Form<?> form = new Form<>("form");
```

(continues on next page)

(continued from previous page)

```
final MyDependingComponent depending = new MyDependingComponent("depending",␣
↪propertyAModel, propertyBModel);
form.add(
                new MyEntity2DropDownChoice("propertyA", propertyAModel)
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                // THIS MAY FAIL, because B's value may␣
↪not be up-to-date
                                                target.add(depending);
                                        }
                                }),
                new MyEntity3DropDownChoice("propertyB", propertyBModel, String.class),
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                // THIS MAY FAIL, because A's value may␣
↪not be up-to-date
                                                target.add(depending);
                                        }
                                }),
                depending
);
```

Then the following scenario may fail:

- `A` is modified by the user
- The Ajax behavior updates `A`'s model and refreshes `C`, which will use `A`'s *updated* value and `B`'s *initial* value. So far so good.
- `B` is modified by the user
- The Ajax behavior updates `B`'s model and refreshes `C`. **`C` will use a wrong value for `A`:**
  - If `A`'s model is a `BindingModel` for an entity property whose root model is a `GenericEntityModel` holding an *unpersisted* entity, **`C` will use a serialized entity for `A`'s value, which may throw `LazyInitializationException`s whenever we try to access its properties**.
  - If `A`'s model is a `BindingModel` for an entity property whose root model is a `GenericEntityModel` holding a *persisted* entity, **`C` will use `A`'s initial value**.

How `IBindableModel`s solve the problem

Enter `IBindableModel`. The idea is to wrap the root model in a `IBindableModel`, and then only use this model's methods to access property models, which will have their values cached transparently.

So instead of the snippet of code above, we will do this:

```
IModel<MyEntity> rootModel = /* ... */;
IBindableModel<MyEntity> bindableRootModel = BindableModel.of(rootModel);
IModel<MyEntity2> propertyAModel = bindableRootModel.bindWithCache(Bindings.myEntity().
↪propertyA(), new GenericEntityModel<Long, MyEntity2>());
IModel<MyEntity3> propertyBModel = bindableRootModel.bindWithCache(Bindings.myEntity().
↪propertyB(), new GenericEntityModel<Long, MyEntity2>());
```

(continues on next page)

(continued from previous page)

```
Form<?> form = new CacheWritingForm<>("form", bindableRootModel); // Necessary so the␣
↪caches are written to the object when submitting
final MyDependingComponent depending = new MyDependingComponent("depending",␣
↪propertyAModel, propertyBModel);
form.add(
                new MyEntity2DropDownChoice("propertyA", propertyAModel)
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                target.add(depending);
                                        }
                                }),
                new MyEntity3DropDownChoice("propertyB", propertyBModel, String.class),
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                target.add(depending);
                                        }
                                }),
                depending
);
```

That way, the values used by `depending` are those in `propertyAModel` and `propertyBModel`'s caches, and those are always clean and up-to-date.

If you must make `MyDependingComponent` use a `MyEntity` model instead of a `MyEntity2` model and a `MyEntity3` model, and only use these properties indirectly (for instance because you must call a service wich accept a `MyEntity` parameter), then you can make use of the `IBindingModel#writeAll()` method, which forces the writing of caches to the underlying entity:

```
IModel<MyEntity> rootModel = /* ... */;
final IBindableModel<MyEntity> bindableRootModel = BindableModel.of(rootModel);
IModel<MyEntity2> propertyAModel = bindableRootModel.bindWithCache(Bindings.myEntity().
↪propertyA(), new GenericEntityModel<Long, MyEntity2>());
IModel<MyEntity3> propertyBModel = bindableRootModel.bindWithCache(Bindings.myEntity().
↪propertyB(), new GenericEntityModel<Long, MyEntity2>());

Form<?> form = new CacheWritingForm<>("form", bindableRootModel); // Necessary so the␣
↪caches are written to the object when submitting

// MyDependingComponent depends on a IModel<MyEntity>, and only indirectly uses␣
↪propertyA and propertyB
final MyDependingComponent depending = new MyDependingComponent("depending",␣
↪bindableRootModel);
form.add(
                new MyEntity2DropDownChoice("propertyA", propertyAModel)
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                bindableRootModel.writeAll();
                                                target.add(depending);
                                        }
```

(continues on next page)

(continued from previous page)

```
                                }),
                new MyEntity3DropDownChoice("propertyB", propertyBModel, String.class),
                                .add(new AjaxFormComponentUpdatingBehavior() {
                                        protected void onUpdate(AjaxRequestTarget␣
↪target) {
                                                bindableRootModel.writeAll();
                                                target.add(depending);
                                        }
                                }),
                depending
);
```

Do's and don'ts

Declare your caches

Caches must be declared explicitly:

- call `IBindableModel.bindCollectionWithCache()` on any collection property whose elements may have their properties written to.
- call `IBindableModel.bindMapWithCache()` on any map property whose keys or values may have their properties written to.
- call `IBindableModel#bindWithCache(<binding>, <cache model>)` on any property which both read from (by depending components) and written to (by a `FormComponent` for instance).

Don't mix `BindingModel`s with `IBindableModel`s

Using a `BindingModel` with a `IBindableModel` as root model will result in bypassing the `IBindableModel`'s cache (if any). You may then witness some strange behaviors due to your `BindableModel` returning a stale value.

Thus, if you use `IBindableModel`, stick with it. If you must pass a model to a child component, check that this child component doesn't use `BindingModel`s.

If you really must use a component that uses `BindingModel`s internally, you can, but only if it's read-only (i.e. it doesn't wraps `FormComponent`s). **Keep in mind**, though, that you must explicitly write the caches to your business objects whenever you do an Ajax refresh.

Write caches to your business objects before using them

Caches are not written magically to your business objects. Thus:

- When your form is being submitted and after it has written the submitted values to your models (to your caches, actually), you must ensure that the caches are actually written to the actual properties so that the root object is fully updated.

  Luckily for you, `CacheWritingForm` does exactly that. Just use it as your root form and, if all of your `IBindableModel`s are children of your root model, then they will all be updated upon submit.

- Whenever you do handle events (`Link`s, `AjaxLink`s, ajax behaviors, ...), if any treatment bypasses the `IBindableModel` and reads directly from the business objects (e.g. `bindableRootModel.getObject().getPropertyA()` instead of `bindableRootModel.bind(Bindings.myEntity().propertyA()).getObject()`), then you should call `IBindableModel#writeAll()` beforehand.

Read caches from your business objects when you modify objects directly

Caches are not updated magically when you bypass the `IBindableModel` and write to the properties directly (e.g. `bindableRootModel.getObject().setPropertyA(<something>)` instead of `bindableRootModel.bind(Bindings.myEntity().propertyA()).setObject(<something>)`).

If you have to do such things, make sure that you call `IBindableModel#readAllUnder()` afterwards.

`CollectionCopyModel` and `MapCopyModel`

`CollectionCopyModel` and `MapCopyModel` are simply put, models to store your collections or maps directly in your page, with no persistence.

They provide two main features:

- they always copy the collection/map when `setObject` is called (hence the name). So even if some one calls `setObject` with an immutable collection as a parameter, the collection returned by `getObject` will still be mutable.
- upon detach, they do not reference the collection or its elements directly, but they wrap the elements in models so that each element is detached correctly. This is especially useful when handling collections of `GenericEntity`, that should never be serialized with the page.

Both models require two things when they're created: a mean of instantiating a new empty collection/map (a `Supplier`), and a mean of instantiating the model that will wrap an element (a `Function<T, IModel<T>>`).

Here are some examples:

```
ICollectionModel<?, Set<MyEntity>> myEntitySetModel =
                CollectionCopyModel.custom(Suppliers2.<MyEntity>hashSetAsSet(),␣
↪GenericEntityModel.<MyEntity>factory())
```

```
ICollectionModel<?, SortedSet<MyEnum>> myEnumSortedSetModel =
                CollectionCopyModel.serializable(Suppliers2.
↪treeSetAsSortedSet(MyEnumComparator.get()));
```

```
IMapModel<?, ?, Map<MyEntity, String>> myEntityToStringModel = MapCopyModel.custom(
                Suppliers2.<MyEntity, String>hashMapAsMap(),
                GenericEntityModel.<MyEntity>factory(),
                Models.<String>serializableModelFactory()
);
```

AbstractReadOnlyModel

TODO

LoadableDetachableModel and LoadableDetachableDataProvider

`LoadableDetachableModel` and `LoadableDetachableDataProvider` are two abstract classes that provide a caching feature, so that the data they give access to is "loaded" on the first access, cached, and then returned as it was retrieved on the first access on every subsequent call, until `detach` is called.

This avoids repeated calls to the database during a single request/response cycle.

Caching

As said above, `LoadableDetachableDataProvider` and `LoadableDetachableModel` will only execute the underlying query once per HTTP request, even if their data-access methods (`count`, `iterator`, `getObject`) are called multiple times. This is, in most cases, what you want.

However, in some very particular cases, you may have to first access the data source (`LoadableDetachableDataProvider` or `LoadableDetachableModel`), then change the underlying data (through a service call), then render the page. **Be warned** that in this case, the rendered data will be the data loaded before your change. If it's not what you want, then you should "refresh" the `LoadableDetachableXXX` explicitly by calling `detach()`.

Modifying data

As usual, modifying the entities retrieved from the `LoadableDetachableDataProvider` or `LoadableDetachableModel` won't alter the database: you need to make service calls in order for these changes to be persisted.

Another thing that might be obvious: be aware that calls to `LoadableDetachableModel.setObject()` will, by default, only change the model value for the current request/response cycle. This is normal, because only you know what to do in order to persist this change.

If you want to persist your changes in database, then you should provide a method in your service layers that will do the work.

If you just want a cache that spans multiple requests, then `CollectionCopyModel` or `MapCopyModel` might be what you're looking for. See further down this page for more information.

11.4.3 Utilities

CompositeSortModel

TODO

StreamModel

`StreamModel` is made to manage Wicket models wrapping `Iterable`. A `StreamModel` is a read-only `IModel<Iterable<T>>`.

Use `StreamModel<T> mySteamModel = StreamModel.of(IModel<? extends Iterable<T>> model)` to get started. From there, you can :

- Use it as a classic Wicket model : `mySteamModel.getObject()`.
- Concatenate multiple models : `mySteamModel.concat(IModel<? extends Iterable<? extends T>> firstModel, IModel<? extends Iterable<? extends T>>... otherModels)`.
- Transform (map) elements of the collection : `mySteamModel.map(Function<T, S> function)`.
- Get a `IModel` which provides the elements in a specific collection type: `mySteamModel.collect(Supplier<? extends C> supplier)`
- Combine all of the above : `mySteamModel.concat(IModel<? extends Iterable<? extends T>> firstModel, IModel<? extends Iterable<? extends T>>... otherModels).map(Function<T, S> function).collect(Supplier<? extends C> supplier)`

WorkingCopyModel, CollectionWorkingCopyModel and MapWorkingCopyModel

These model wrap two other models: a reference model and a "copy" model. They delegate read and write access to the copy model, while providing additional methods to *write* from the copy to the reference and *read* from the reference to the copy.

These models should not be used directly as a more high-level feature is available with the `BindableModel` described above.

11.4.4 Troubleshooting

Sometimes, you've got models that are not detached properly, but you simply don't know which ones. You just know that, on the next rendering of you page, everything explodes with a `org.hibernate.LazyInitializationException`. In that case, you've got to dig up a bit, and this chapter aims at helping you doing just that.

Built-in logs

`GenericEntityModel` and `AbstractThreadSafeLoadableDetachableModel` (plus its subclasses) provide built-in logging when attached values are suspiciously serialized.

They show:

- The currently attached value at WARN level
- A stacktrace of the latest attach operation on this model at DEBUG level (don't use this level in production environment: it involves aggressive stacktrace recording)

Breakpoints

If the above logs are not enough (and they should), you may still use breakpoints.

Just put your breakpoints inside the `if` in `GenericEntityModel#writeObject` or `AbstractThreadSafeLoadableDetachableModel#writeObject`. In the stack will appear several `writeObject0` methods: inspect those and the `arg0` parameter to determine the chain of objects that lead to the incorrect serialization of your model. You will then probably have to fix one of this object by adding a missing `detach` somewhere.

11.5 UI Displaying Collections

This page explains how to display collections using Igloo. Several methods are provided, ordered from easiest to the hardest: use the first that fits, so as to avoid unnecessary complexity.

11.5.1 Data sources

Choosing your implementation

There are several types of objects you may use to build an object that will make up the interface between your service/data layers and your view.

When getting data from an entity

If you want to retrieve the data directly from an entity attribute (`myEntity.getMyCollection()`) you may use a `BindingModel`. See *UI-Models* for more information.

When getting data from a service or `IQuery`

If you're not familiar with data querying in Igloo, you probably should read *Querying* before going on.

Special case: `ISearchQuery`

If your query is an `ISearchQuery` (`AbstractHibernateSearchSearchQuery` or `AbstractJpaSearchQuery`), you may simply extend `AbstractSearchQueryDataProvider`. The typical implementation will:

- define some `IModel` attributes and getters, for the search parameters. These models will be used in a form, so that the user may alter them.
- implement `getSearchQuery` by:
  - calling `createSearchQuery()` with the query interface type (`IMyQuery.class`) as a parameter ;
  - and then calling methods on the resulting query in order to set the search parameters.

Other cases (service method call or non-search `IQuery`)

- If your query uses paging (with an offset and a limit), you'd better define a `IDataProvider`. A good place to start is `LoadableDetachableDataProvider`, which you should try extending. Also, see *UI-Models* for more information on `LoadableDetachableDataProvider` and its caveats.
- If your query has no paging feature, you may simply define your own `IModel<WhateverCollectionType<T>>`. A good place to start is `LoadableDetachableModel`, which you should try extending. Also, see *UI-Models* for more information on `LoadableDetachableModel` and its caveats.

11.5.2 Renderers

Renderers offer a way to build a non-HTML (plain text) string from a given `Collection<T>` or `IModel<? extends Collection<T>>`.

When to use it

You should use renderers when:

- Your expected output has a very simple structure
- Your expected output does not require HTML (be warned that this excludes any kind of line break, since this would require a
 or <p>)
- Requirements are very unlikely to change in the future to require HTML inside the output

If you need more complex output, go for the *DataTableBuilder* or *RefreshingViews*.

Examples

Building the renderer

```
Renderer<Iterable<? extends MyItem>> collectionRenderer = Renderer.fromJoiner(
                Joiners.Functions.onComma(),
                MyItemRenderer.get()
);
```

Using the renderer

In a label

```
IModel<Set<MyItem>> myModel = /* ... */;
add(new CoreLabel("id", collectionRenderer.asModel(myModel))); // Will display "item1,␣
↪item2, item3"
```

In a StringResourceModel

*.properties:

```
my.resource.key=List value: {0}
```

Java:

```
IModel<Set<MyItem>> myModel = /* ... */;
IModel<String> stringModel = new StringResourceModel("my.resource.key")
                .setParameters(collectionRenderer.asModel(myModel));
add(new CoreLabel("id", stringModel))); // Will display "List value: item1, item2, item3"
```

In an error message

Just use `Component.getString()` as follows:

```
IModel<Set<MyItem>> myModel = /* ... */;
component.error(component.getString("my.resource.key", collectionRenderer.
↪asModel(myModel)); // Will display "List value: item1, item2, item3"
```

And define your properties as follows:

```
my.resource.key=List value: ${}
```

11.5.3 DataTableBuilder

The DataTableBuilder offers the simplest way to build a HTML table, quick & clean.

When to use it?

In order to use DataTableBuilers, the component you want to build must meet the following requirements:

- The expected output must be a HTML table
- The data source must be some kind of collection of elements (a `IDataProvider`, a `IModel<? extends Collection<?>>` or a `ISequenceProvider`)
- There must be one row in the table's body for each element in the data model (paging aside)
- There must be a pre-defined, static maximum number of columns. Some columns may get hidden dynamically. For instance, you can't have one column for each element of an `IModel<? extends Collection<?>>` if this model's content may change between ajax refreshes.

If all of the above seems fine to you, then go ahead with the `DataTableBuilder`. Otherwise, you may still use *RefreshingViews*.

Overview

The general pattern for building a data table is as follows:

- create a builder through one of the static `start` methods
- add a column though one of the `.add*Column` methods, defining in particular the data to be displayed (with a binding or a `Function`)
- customize the column though the various methods allowing to add CSS classes on cells, to add a link for each row, to add a sort-switching link in the header, and so on
- repeat the same operations for each column
- optionally, call `.decorate` in order to create a table with an upper title and pagers, or `.bootstrapPanel` to the the same in a Bootstrap panel
- call `.build("wicketId")` in order to retrieve the resulting component.

Here is a (simple) example of use of `DataTableBuilder`:

```
DecoratedCoreDataTablePanel<?, ?> results =
                DataTableBuilder.start(dataProvider, dataProvider.getSortModel())
                .addLabelColumn(new ResourceModel("business.customer.lastName"),␣
↪Bindings.customer().lastName())
                                .withLink(CustomerDescriptionPage.MAPPER)
                                .showPlaceholder()
                                .withSort(CustomerSort.LASTNAME, SortIconStyle.ALPHABET,␣
↪CycleMode.NONE_DEFAULT_REVERSE)
                                .withClass("text text-sm")
                .addLabelColumn(new ResourceModel("business.customer.firstName"),␣
↪Bindings.customer().firstName())
                                .withLink(CustomerDescriptionPage.MAPPER)
                                .showPlaceholder()
                                .withSort(CustomerSort.FIRSTNAME, SortIconStyle.ALPHABET,
↪ CycleMode.NONE_DEFAULT_REVERSE)
                                .withClass("text text-sm")
                .addLabelColumn(new ResourceModel("business.customer.birthdate.short"),␣
↪Bindings.customer().birthdate(), DatePattern.REALLY_SHORT_DATE)
                                .showPlaceholder()
                                .withSort(CustomerSort.BIRTHDATE, SortIconStyle.DEFAULT,␣
↪CycleMode.NONE_DEFAULT_REVERSE)
                                .withClass("date date-xs")
                                .withClass(ResponsiveHidden.XS_AND_LESS)
                .addBootstrapLabelColumn(new ResourceModel("business.customer.status"),␣
↪Bindings.customer().status(), CustomerStatusRenderer.get())
                                .withClass("statut statut-md")
                                .withClass(ResponsiveHidden.XS_AND_LESS)
                .addLabelColumn(new ResourceModel("business.customer.sector.short"),␣
↪Bindings.customer().sector())
                                .showPlaceholder()
                                .withClass("code code-sm")
                                .withClass(ResponsiveHidden.XS_AND_LESS)
                .decorate()
                                .count("customer.list.result.count")
                                .ajaxPagers()
                .build("results");
```

Data source

You may provide either a `ISequenceProvider` or a `IDataProvider` to the `start` method as a data source. The resulting data table will contain exactly one row for each element provided by your data source.

Supported column types

Here is a list of the built-in column types:

- Label columns (`addLabelColumn`), which display a simple textual label derived from the underlying value (through the use of a *Renderer*). Optionally, the label may be wrapped in a link, or have a side link (a link on a side button) appended.
- Bootstrap label columns (`addBootstrapLabelColumn`), which display a textual label with a background color and prepended icon that all depend on the underlying value. Optionally, the label may be wrapped in a link, or have a side link (a link on a side button) appended.
- Bootstrap badge columns (`addBootstrapBadgeColumn`), which display a badge with a background color and an icon that depend on the underlying value. Optionally, the label may be wrapped in a link, or have a side link (a link on a side button) appended.
- Action columns (`addActionColumn`), which display one or more buttons, each button being either:
- A link to a bookmarkable page
- An action link: a link which will trigger execution of arbitrary code (with or without a confirmation popup)
- An **ajax** action link (with or without a confirmation popup)

If none of the above suits your needs, keep in mind that you may simply use the `org.iglooproject.wicket.more.markup.repeater.table.builder.DataTableBuilder.addColumn(ICoreColumn<T, S>)` method and pass your own column implementation as a parameter. Most of the time, you will simply have to extend `org.iglooproject.wicket.more.markup.repeater.table.column.AbstractCoreColumn<T, S extends ISort<?>>` and implement `populateItem(Item<ICellPopulator<T>>, String, IModel<T>)` so as to add a `Fragment` defined in your own component.

Adding components around the table

Super headers

You may add arbitrary rows above or below the data table by calling `addTopToolbar` or `addBottomToolbar` and then adding components, optionally attributing a colspan to each of them. This is great in particular if you want to add headers that span multiple columns above your column headers.

Simple title and pager

You may create a "decorated" table, with a top title and pagers, by calling `decorate` after having defined your columns. You may then define the title (optionally making it dependent on the result count, by calling `count`), add top and/or bottom pagers (`.pagers`, `.ajaxPagers`, ...), or even add arbitrary add-ins (`.addIn`).

Bootstrap panel

You may create a "decorated" table as above, but with Bootstrap styling, wrapped in a Bootstrap panel. Just call `bootstrapPanel` instead of `decorate`, and proceed the same as with `decorate`.

11.5.4 RefreshingViews

Compared to the `DataTableBuilder`, the `RefreshingViews` are a lower-level way of displaying collections.

When to use it?

Whenever you can't use the `DataTableBuilder`:

- You don't want a HTML table, but just some repeating `div`s or `li`s (or any other markup, really)
- You want a HTML table, but it's too complex and can't be built using the `DataTableBuilder`. For example you may need multiple `<tr>` for each element in your collection, or you may need to repeat columns instead of rows.

Overview

`RefreshingViews` are generally used this way:

Panel's HTML:

```
...

<div wicket:id="item">
        <span wicket:id="content1" />
        <wicket:container wicket:id="content2" />
</div>
...
```

Panel's Java:

```
add(new SubclassOfRefreshingView<MyItem>("item", dataProvider) {
        @Override
        public void populateItem(final Item<MyItem> item) {
                item.add(new Label("content1", new ResourceModel("some.resource.key"));
                item.add(new SomePanel("content2", item.getModel());
        }
})
```

Which view to use?

It depends on you data source:

- for a `IDataProvider`, use a `SequenceView` (or wicket's `DataView`, which for now should do the same job, but may not gain as much features as SequenceView in the future).
- for a `ISequenceProvider`, use a `org.iglooproject.wicket.more.markup.repeater.sequence.SequenceView`
- for a `ICollectionModel<T, ?>`, use a `org.iglooproject.wicket.more.markup.repeater.collection.CollectionView`.

- for a `IMapModel<K, V, ?>`, use a `org.iglooproject.wicket.more.markup.repeater.map.MapView`.
- for a `IModel<? extends Collection<T>>`, use a `org.iglooproject.wicket.more.markup.repeater.collection.CollectionView`. You will need to provide a factory for the collection item models. (see *below*).
- for a `IModel<? extends Map<K, V>>`, use a `org.iglooproject.wicket.more.markup.repeater.map.MapView`. You will need to provide a factory for the map key models (see *below*).

Please note that all of the above provide a paging mechanism, but it will only be efficient if your data source is a properly implemented `IDataProvider` or `ISequenceProvider`. Otherwise, the whole data set will be loaded, and then reduced to the current page's data.

Item models

The `RefreshingView`s need to obtain a reference to the collection's item model, in order to handle manipulations of this item (for instance, a click on a button in a table row mapped to an item).

The way you will implement the "item model factory" will depend on your data source:

- With Wicket's built-in `IDataProvider`, the `IDataProvider` itself will provide the item model through its `model(T)` method. This method is the "item model factory".
- With Igloo's `ISequenceProvider` the provided items are already wrapped in models: the `iterator(long, long)` method returns an `Iterator<? extends IModel<T>>`. The `ISequenceProvider` itself is the "item model factory". This also applies to `ICollectionModel` and `IMapModel`.
- With an `IModel<? extends Collection<T>>` (be it a `LoadableDetachableModel`, a `BindableModel`, or anything else), nothing in the data source itself allows to build item models. That's why most views defined above require you to provide a `Function<? super T, ? extends IModel<T>>` that will serve as an "item model factory".

When a `Function<? super T, ? extends IModel<T>>` is required, you may:

- Use `GenericEntityModel.factory()` if your items are `GenericEntity`s
- Use `Models.serializableModelFactory()` if your items are serializable (`Integer`, `String`, enums, ...)
- Define your own function if none of the above suits your needs. Take care to make the `Function` also implement `Serializable`, since it will be serialized with the page after the response.

11.5.5 For more advanced needs

Wicket also offers various types of built-in `RepeatingView`s, but the above should encompass most common needs. Only use these views as a fallback if the above clearly won't do.

`ListView` and `IndexedItemListView`

The main advantage of `ListView` is that you don't need to define a specific model for the collection items: they are (by default) mapped by their index to the collection model.

Be warned, though, that with this mechanism, you may run into issues if you implement user operations on the element's models (like opening a modal, or removing an element) and if your underlying collection's content changes between the initial page rendering and the user request: the operation may end up being executed on the wrong item (because the index may not point the the same collection element anymore).

`ListView` and `IndexedItemListView` should be used very rarely, and only if you really know what you're doing.

Adding or removing items using Ajax without refreshing the whole collection view

Wicket, by default, only allows to add or remove items using Ajax to a collection view by refreshing the whole view.

If, for some reason, you don't want to refresh pre-existing, unremoved items, you may use `org.iglooproject.wicket.more.ajax.AjaxListeners.refreshNewAndRemovedItems(IRefreshableOnDemandRepeater)`:

```
AjaxListeners.add(
                target,
                AjaxListeners.refreshNewAndRemovedItems(repeater)
);
```

There are some constraints, though:

- `repeater` must implement `IRefreshableOnDemandRepeater` (that's the case for most view provided in Igloo)
- both the repeater's parent and the repeater's items must have `setOutputMarkupId` set to `true`
- newly added items will be added at the end of the repeater's parent (the Wicket parent). If there is some HTML between the repeater and the end of the parent, you'll probably want to wrap your repeater in an `WebMarkupContainer`.
- only some classes that implement `IRefreshableOnDemandRepeater` allow to detect removed elements, so only these will see their removed items removed from the HTML. `RepeatingView` and its subclasses, in particular, will not have their removed elements removed from the HTML.

11.6 UI User Actions (TODO)

TODO:

- request type (Ajax/non ajax)
- button appearance (bootstrap button with or without label, non-bootstrap button, ...)
- button position (in a form, outside a form, in a table, ...)
- data (submits a form or not)
- redirections (or not)

11.6.1 HTML Markup

The following guidelines should be followed every time you add a button to a page:

- if a click on your button launches javascript code, then use <button type="button">. This goes even if your button is outside of a form. This includes buttons triggering an ajax call (`AjaxLink`, `AjaxSubmitLink`, `AjaxButton`) as well as buttons triggering your own javascript.
- if a click on your button submits a form without using ajax, then use <button type="submit">.
- if a click on your button simply redirects to a bookmarkable page, then it is an actual HTTP link and you should use <a>.

The reason for these guidelines is that, while Wicket does support binding ajax calls to <a> (or even to arbitrary markup), Wicket does not, however, prevent the default event handler for this markup to execute when a user clicks. Unfortunately, that means that when a user click on a <a> with an ajax call bound to the `click` event, then first the ajax call will be performed, then the default action... Which is, for most browser, a scroll to the top of the page. Which probably isn't what you want.

11.7 UI Forms (TODO)

TODO:

- IndependentNestedForm
- Available form components
- How to deal with collections:
- when only adding/removing is required (no element editing)
- when only element editing is required (no add/remove)
- when adding/removing is required **and** element editing is also required
- ...

11.7.1 `AutoLabelResolver` and form component / label order

When using `wicket:for` attribut on a `<label/>`, Wicket retrieves related form component and call `setOutputMarkupId(true)` to create a link between the label and the form component. However this will not work if the form component has already been rendered without markup id. This is the case when the form component comes first in html, e.g.:

```
<div class="custom-control custom-switch">
  <input type="checkbox" class="custom-control-input" id="customSwitch1">
  <label class="custom-control-label" for="customSwitch1">Toggle this switch element</
↪label>
</div>
```

Workaround: explicit call to `setOutputMarkupId(true)` on form component.

11.8 UI Placeholder and Enclosure

This pages explains how to write Enclosure and Placeholder. Enclosure and Placeholder are component which are displayed or hidden (at page-generation level) based on various input variables.

11.8.1 Conditions and component's visibility

Igloo provides `Condition` base class which provides easy-to-use transformation of a Condition (that resolves to true or false result on an *applies* method's call) as an enclosure or placeholder behavior.

Once your `Condition` created, you can generate the following behaviors :

- thenShow(): behavior that sets `visibilityAllowed` property equal to condition's result (enclosure-like behavior)
- thenHide(): behavior that sets `visibilityAllowed` property equal to negated condition's result (placeholder-like behavior)
- thenShowInternal(): behavior that sets `visible` property equal to condition's result
- thenHideInternal(): behavior that sets `visible` property equal to negated condition's result

By convention, `visibilityAllowed` setting (thenHide(), thenShow()) must be preferred for external impact on component visibility (permission, inter-component dependencies driven by application behaviors).

On the contrary, `visible` setting is used for internal behaviors. For example, for a table + pager widget, to control the pager's visibility in relation to result's page number, as this behavior is driven by an internal state of the component.

To circumvent visibility settings when `visible` and `visibilityAllowed` properties conflict, use of an intermediate component may be a solution.

11.8.2 Component's *enabled* property

`thenEnable()` and `thenDisable()` methods are provided to allow `enabled` property control, based on the same mechanism than visibility control.

11.8.3 Deprecated patterns

Before `Condition`, the following components and behaviors were provided:

- `PlaceholderBehavior`, `EnclosureBehavior`: sets `visibilityAllowed` property (by default) or an alternate property when provided. This behavior can be replaced by a Condition method call:
  - `.add(new PlaceholderBehavior().component(component))`: `.add(Condition.componentVisible(component).thenHide())`
  - `.add(new EnclosureBehavior().component(component))`: `.add(Condition.componentVisible(component).thenShow())`
  - `.add(new PlaceholderBehavior(ComponentBooleanProperty.VISIBLE).component(component))`: `.add(Condition.componentVisible(component).thenHideInternal())`
  - `.add(new EnclosureBehavior(ComponentBooleanProperty.VISIBLE).component(component))`: `.add(Condition.componentVisible(component).thenShowInternal())`

When other method than `component()` is used on `...Behavior` object, others `Condition`'s methods or subclasses can be used to provides the right behavior (permission(), anyPermission(), role(), isTrue(), isFalse(), predicate(), ...)

11.9 UI Charts and plots

This page explains how to generate data charts using Igloo.

11.9.1 JQPlot

About

JQplot is "a versatile and expandable JQuery plotting plugin" (see the official site). Igloo uses WQPlot in order to ensure low-level integration of JQPlot into wicket, plus a specific maven module (`org.iglooproject.components:igloo-component-wicket-more-jqplot`) in order to provide easy-to-use, high-level integration. Please note that WQPlot was originally a wicketstuff project, which seems no longer maintained.

High-level usage

Architecture

You will encounter four types of components when you'll set up a chart:

- The data provider, which implements `IJQPlotDataProvider`. Its role is extracting data from your service layer (or an `IModel`, or anything you want) and provide it in a standardized form.
- The data adapter, which implements `IJQPlotDataAdapter`. Its role is to take data from a data provider and transform it in the WQPlot types (`BaseSeries`, `NumberSeries`, and so on) that can be used by the panel.
- The panel itself, which extends `JQPlotPanel`. It's a Wicket component that uses the data adapter and WQPlot in order to generate the actual chart.
- The configurers, which implement `IJQPlotConfigurer`. Their role is to customize various JQPlot options.

Using it in your application

Setup global configuration

Add a dependency in your pom to `org.iglooproject.components:igloo-component-wicket-more-jqplot`.

You will also need to import some Spring configuration: add an `@Import` annotation to your webapp in order to import `org.iglooproject.wicket.more.jqplot.config.JQPlotJavaConfig`.

You may optionally, instead of importing `JQPlotJavaConfig` directly, define you own configuration that extend `JQPlotJavaConfig`. In this case, you will be able to override the default options passed to JQPlot's plots.

Setup your chart

First, setup your data source. Generally it will be a service that returns:

- A `java.util.Map<K,V>` if there's only one data series.
- A `com.google.common.collect.Table<S,K,V>` if there are multiple data series.

See *Generating maps and tables* for more information about how to easily generate a `Map` or `Table` from a QueryDSL query in your DAO.

Then, pick your data provider:

- `org.iglooproject.wicket.more.jqplot.data.provider.JQPlotMapDataProvider<S, K, V>` if you've got a service that returns a `Map`. Then you will wrap the service call in a `LoadableDetachableModel`, and pass the model as a constructor parameter to the data provider.
- `org.iglooproject.wicket.more.jqplot.data.provider.JQPlotTableDataProvider<S, K, V>` if you've got a service that returns a `Table`.
- Or, if for some reason the above won't do, you may implement your own.

Then, pick your chart:

- For bar charts, use `org.iglooproject.wicket.more.jqplot.component.JQPlotBarsPanel<S, K, V>`
- For line charts, use `org.iglooproject.wicket.more.jqplot.component.JQPlotLinesPanel<S, K, V>`
- For pie charts, use `org.iglooproject.wicket.more.jqplot.component.JQPlotPiePanel<K, V>`

- For stacked bar charts, use `org.iglooproject.wicket.more.jqplot.component.JQPlotStackedBarsPanel<S, K, V extends Number & Comparable<V>>`
- For stacked lines charts, use `org.iglooproject.wicket.more.jqplot.component.JQPlotStackedLinesPanel<S, K, V>`

Then, pick your data adapter (it will be the bridge between your panel and your data provider):

- If your data references keys from a discrete domain (i.e. with a small, finite number of values in the observed range), use`org.iglooproject.wicket.more.jqplot.data.adapter.JQPlotDiscreteKeysDataAdapter<S, K, V>`. Please keep in mind that if you want your X axis to have a linear scale, you should either ensure that all keys are represented in your data, or provide to the data adapter a model containing all of the expected keys (so that the adapter can generate a linear axis).
- If, on the other hand, your data references keys from a continuous domain (i.e. with a high or infinite number of values in the observed range), use one of these:
- `org.iglooproject.wicket.more.jqplot.data.adapter.JQPlotContinuousDateKeysDataAdapter<S, K, V>`
- `org.iglooproject.wicket.more.jqplot.data.adapter.JQPlotContinuousNumberKeysDataAdapter<S, K extends Number, V extends Number>`
- Or, if for some reason the above won't do, you may implement your own.

Note: pie charts are a special case: you won't need a data adapter, just a data provider.

And finally, put all of this together: you've got a basic chart. Some examples are provided in the "Statistics" page of Igloo's wicket showcase (`org.iglooproject.showcase:wicket-showcase`).

To go further:

- You may add one or several configurers to the chart in order to customize JQPlot options: either pick a configurer from `org.iglooproject.wicket.more.jqplot.config.JQPlotConfigurers` or use your own implementations.
- You may wrap your data adapter in order to customize the data passed to JQPlot: see `org.iglooproject.wicket.more.jqplot.data.adapter.JQPlotDataAdapters`. You may for instance add a percentage in the tooltip shown when hovering over a stacked bar.

Low-level usage

If `org.igloo-project.components:igloo-component-wicket-more-jqplot` is not flexible enough for your needs, you may still use raw wqplot. Some examples are provided in the "Statistics" page of Igloo's wicket showcase (`org.iglooproject.showcase:wicket-showcase`).

Keep in mind that you'll need to explicitly update jqplot options whenever the data changes (the X axis ticks, for instance, if you declared them explicitly).

Troubleshooting

The plots are not drawn

Please be aware that the supporting markup of your plots must be initially visible (i.e. not `display: none`) if you want the plots to be drawn automatically. If, for instance, your plots are located in an initially inactive (hidden) tab, then they will not be drawn automatically.

To address the specific case of plots in an initially inactive Bootstrap tab, you may use the following snippet:

```
tabContainer.add(new JQPlotReplotBehavior("shown.bs.tab"));
```

This will ensure that plots are "replotted" each time the user switches tabs.

11.9.2 Other plotting libraries

No other plotting library is integrated into Igloo at the moment.

11.10 Scss/Sass processing

11.10.1 Scss/Sass processing

Igloo provides tools to perform Scss/Sass processing:

- `de.larsgrefer.sass:sass-embedded-host` dependency is used for scss processing. It is a wrapper for dart-sass implementation: https://sass-lang.com/dart-sass
- Igloo customization allows to use classpath lookup for scss resources: old-style (and deprecated) `$(scope-name)/...` URLs and `META-INF/resources/webjars/...` webjars URLs.
- `CachedScssServiceImpl` can be used to wrap `ScssServiceImpl` and perform scss cache handling (ehcache). It performs automatic expiration check to allow auto-refresh in development mode.
- `autoprefixer.enabled` configuration allows to perform autoprefixer processing after scss/sass processing; it can be disabled to save processing time.
- Inherit `ScssResourceReference` to implement your wicket stylesheet `ResourceReference`

11.10.2 Build-time generation

`ScssServiceImpl` allows to use build-time generated asset to speed-up application startup or response time.

`scss.static.enabled` configuration enables the usage of build-time static resources. If the static resource is absent, runtime processing is performed. If the static resource is present, it never expires. If `scss.static.enabled=false` then static resource is ignored.

It expects static resources to be available in `igloo-static-scss/` classpath, named from a md5 hexed hash of `package.ClassNameResourceReference:filename.scss` string, with a `.css` extension.

You can use `ScssMain` java main program with `exec-maven-plugin` to generate the expected files.

- Add `plugin` markup to your webapp `pom.xml`. Here is an example, you need to customize scopes and target Scss files.

```
<plugin>
    <!-- generate SCSS at build-time; greatly speed-up application startup -->
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>exec-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>generate-scss</id>
            <goals>
                <goal>java</goal>
            </goals>
```

(continues on next page)

(continued from previous page)

```
            <phase>prepare-package</phase>
            <configuration>
                <mainClass>org.iglooproject.sass.cli.ScssMain</mainClass>
                <arguments>
                    <argument>--generation-path</argument>
                    <argument>${project.build.outputDirectory}</argument>
                    <argument>--scopes</argument>
                    <argument>core-bs5:igloo.bootstrap5.markup.html.template.css.
↪bootstrap.CoreBootstrap5CssScope</argument>
                    <argument>--scopes</argument>
                    <argument>core-console:igloo.console.template.style.
↪CoreConsoleCssScope</argument>
                    <argument>--scopes</argument>
                    <argument>core-fa:igloo.fontawesome.CoreFontAwesomeCssScope</
↪argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.application.application.applicationadvanced.
↪StylesScssResourceReference:styles.scss</argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.application.application.applicationbasic.
↪StylesScssResourceReference:styles.scss</argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.application.console.console.ConsoleScssResourceReference:console.scss
↪</argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.notification.NotificationScssResourceReference:notification.scss</
↪argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.application.console.consoleaccess.
↪ConsoleAccessScssResourceReference:console-access.scss</argument>
                    <argument>org.iglooproject.basicapp.web.application.common.template.
↪resources.styles.application.application.applicationaccess.
↪ApplicationAccessScssResourceReference:application-access.scss</argument>
                </arguments>
            </configuration>
        </execution>
    </executions>
</plugin>
```

- Add `picocli` dependancy to your webapp `pom.xml` :

```
<!-- Only for packaging scss at build-time -->
<dependency>
  <groupId>info.picocli</groupId>
  <artifactId>picocli</artifactId>
  <optional>true</optional>
</dependency>
```

- Launch `mvn clean install -DskipTests` and check that scss files are correctly generated

```
[INFO] --- exec-maven-plugin:3.1.0:java (generate-scss) @ basic-application-webapp ---
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
```

(continues on next page)

(continued from previous page)

```
↪styles.application.application.applicationadvanced.StylesScssResourceReference:styles.
↪scss -> /home/cbaffertbuivan/git/igloo-parent/basic-application/basic-application-
↪webapp/target/classes/igloo-static-scss/
↪c9aa9b01f1b948961aa50745ba2bdb12a45ca1f07fc9116c598807c004c9224a.css (12140 ms.)
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
↪styles.application.application.applicationbasic.StylesScssResourceReference:styles.
↪scss -> /home/cbaffertbuivan/git/igloo-parent/basic-application/basic-application-
↪webapp/target/classes/igloo-static-scss/
↪05f150ec91c975a1ccc21ce410e0ea88e28c880a5eb74c5d932d8794a68a4b81.css (6770 ms.)
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
↪styles.application.console.console.ConsoleScssResourceReference:console.scss -> /home/
↪cbaffertbuivan/git/igloo-parent/basic-application/basic-application-webapp/target/
↪classes/igloo-static-scss/
↪f67eca564bdf215846e66fc79e903268ca9ce3079ebb25bdecbefcb562ca7a52.css (5441 ms.)
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
↪styles.notification.NotificationScssResourceReference:notification.scss -> /home/
↪cbaffertbuivan/git/igloo-parent/basic-application/basic-application-webapp/target/
↪classes/igloo-static-scss/
↪f9eaa28a16432525b6c155f9a48f8d53ad007f883d083aadcecb7f6aa02f91d9.css (1745 ms.)
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
↪styles.application.console.consoleaccess.ConsoleAccessScssResourceReference:console-
↪access.scss -> /home/cbaffertbuivan/git/igloo-parent/basic-application/basic-
↪application-webapp/target/classes/igloo-static-scss/
↪e11f85f307ccff1495e99fc939ce52558ee7843282f80c3af1d6346d490aea6c.css (4665 ms.)
Scss processing: org.iglooproject.basicapp.web.application.common.template.resources.
↪styles.application.application.applicationaccess.
↪ApplicationAccessScssResourceReference:application-access.scss -> /home/cbaffertbuivan/
↪git/igloo-parent/basic-application/basic-application-webapp/target/classes/igloo-
↪static-scss/f36f9ed59aea49965889043f1e368168d0165e3b4e60f213b76746af9abdabd2.css (4388␣
↪ms.)
Scss generation time: 35455 ms.
```

11.10.3 JSass / Dart-sass migration

JSass to dart-sass implies some scss changes:

- `webjars://` urls need to be rewritten:
  - `webjars://bootstrap:current/` -> `META-INF/resources/webjars/bootstrap/`
  - `webjars://bootstrap5-override/` -> `META-INF/resources/webjars/bootstrap5-override/`
- `$(scope-NAME)` must be followed by a `/`: `$(scope-core-fa)scss/core` -> `$(scope-core-fa)/scss/core` (previously, it was optional)
- `@import`-ed files must be `.scss` files
  - check that you have no `.css` file in project-webapp directory (except files in `errors/` folder)
  - if you have some `.css` files, check if you want to be included by scss processing (then proceed to rename) or if they are included / managed by browser (then `.css` extension can be kept)
- Launch your app :
  - Check that css files are correctly loaded

  - Visually check style of your app

Note: `jimportdiff` can handle tedious rewrite tasks (cf. *jimportdiff*).

Use jimportdiff dart-scss :

- scss: rename webjars URLs :
  - `webjars://*:current/` -> `META-INF/resources/webjars/*/`
  - `webjars://*` -> `META-INF/resources/webjars/*`
- scss: rename `$(scope-*)scss/core` -> `$(scope-*)/scss/core`

```
cd jimportdiff
pipenv --rm
pipenv install
pipenv shell
cd ../PROJECT
../jimport/jimportdiff dart-scss
```

CHAPTER

TWELVE

TESTS

12.1 Wicket Tests

The test of Wicket elements and the construction of Wicket pages can be done via `WicketTester`. It provides utilitary methods to assert certain facts on components. We also provide additionnal methods with a `CoreWicketTester` and a `WicketMoreWicketTester`.

12.1.1 Initialization

To start creating tests for your application you should reproduce the organization of Basic Application's webapp module by adding a `src/test/java`` and ``src/test/resource` folder. Make sure to update the `pom.xml` of your webapp module and also from your core module (the plugin `maven-jar-plugin` is mandatory).

You should have a local WicketTester, `[Project]WicketTester` and an abstract test case class. See `BasicApplicationWicketTester` and `AbstractBasicApplicationWebappTestCase`.

12.1.2 Usage

Data initialization

All data must be initialized before each test and cleaned afterwards. These two steps must be implemented in your abstract test case. Two mandatory elements :

- During the initialization phase, you must also initialize a new tester (see `AbstractBasicApplicationWebappTestCase#setUp()`).
- After the data being cleaned you must call `emptyIndexes()` method. It ensures your indexes to be well cleaned even after an abrupt stop of your tests.

Setup of a test

At the beginning of your test you have to load the page you want to test.

WicketTester bypasses Spring security, so every pages protected only in your `security-web-content.xml` won't be protected in your tests.

However, pages protected by a `@AuthorizeInstantiation` are well protected and only the authorized user will be able to access these pages. So you must authenticate yourself with the right user to access those pages. You should copy the pattern present in the Basic Application and use the `authenticateUser()` method.

Once you are authenticated, you can launch the page using `WicketTester#startPage(Class)` or `WicketTester#executeUrl(String)`.

From now it is up to you and your test scenario.

Assertions

All assertions methods check the behavior of a wicket element. You can either provide the element itself or the **path** to access it.

The **path** correspond to the element's `wicketId`. From a page, only the direct child are accessible. If you want to test a label inside a panel added to a page, the path to the label from the page is `panelwicketId:labelWicketId`.

Common

We have multiple methods provided directly by WicketTester itself to test if the elements are visible, invisible, enabled, disabled, etc.

`CoreWicketTester` - It provides utilitary methods similar to those from WicketTester. `WicketMoreWicketTester` - It provides methods to test wicket more components.

- `assertEnabled` : the basic method only checks that the element is enabled, this behavior is override in `CoreWicketTester` to check also its visibility.
- `assertDisabled` : the basic method only checks that the element is disabled, this behavior is override in `CoreWicketTester` to check also its visibility.
- `assertUsability` : this method comes directly from `BaseWicketTester` (`WicketTester` super class). Contrary to assertion methods provided from `WicketTester` it can only be used with a component and doesn't support the path access. To ensure consistency between all methods we provide these methods. You won't have the use of this method knowing that it provides the same behavior than the `assertEnabled` methods.

Note: For the basic methods (visible, enabled, disabled, usability) a other endpoint that also check the component's type is provided. The assertion `assertComponent` is used to do so.

For example : `assertVisible("labelWicketId", CoreLabel.class)` It ensures that the component corresponing to the wicketId "labelWicketId" is visible and of type CoreLabel.

- `assertRenderedPage` : to check the page we are currently
- `assertFeedbackMessages` : to assert the content of feedback messages

Navbar

To test your navbar you can reproduce the pattern in `HomePageTestCase#navbarUserAdmin()`. Calling the `navbar(int)` you have to provide the number of expected tabs including the submenu.

To adapt this pattern to your application you only have to modify the NavbarItem by adding or removing tabs.

Forms

In order to fill form and submit it you have to create a `FormTester` from `Form` component. Here is an example :

```
FormTester form = tester.newFormTester("content:form");

form.setValue(form.getForm().get("username"), "usernameExample");
form.setValue(form.getForm().get("password"), "passwordExample");

form.submit();
```

Depending on the type of field several method are provided

- `FormTester#setValue` : for text field
- `FormTester#select(String, int)` : for select field
- `FormTester#setFile` : for file upload field

For the submission either you provide the component to use or you let the newFormTester determine it himself. It will only work if the submit button is included in the Form Component, which generally is not the case in modal.

Test Html pages

It is possible to directly control the construction of the html pages via a `TagTester` object.

Navigation

To navigate between pages you can simulate a mouse click with the method `WicketTester#clickLink(Component)`

Limits

WicketTester does not interprete Ajax callback or JavaScript callback so every component such as `UserAjaxDropDownSingleChoice` cannot be tested.

CHAPTER

THIRTEEN

PROCESSOR

13.1 Purpose

Some tools need code generation. `maven-processor-plugin` allows to configure code generation steps in maven building process. It is also possible to use `maven-compiler-plugin` for this usage : one downside of `maven-compiler-plugin` is that processing errors are harder to detect and understand, so we stick with maven-processor-plugin for the moment.

13.2 How maven-processor-plugin is managed by Igloo

maven-processor-plugin configuration is managed by igloo-maven plugins-processor.pom.xml module. It is included in general purpose igloo-maven plugins-all/pom.xml module.

maven-compiler-plugin is configured with `-proc:none` argument, to ensure that it does not handle any code generation.

Key configurations for *org.bsc.maven:maven-processor-plugin* are:

- `<processors>${igloo.processors}</processors>`. *igloo.processors* must be a list of comma-`,` separated list of processor names. If *processors* configuration is not used, any processor on the classpath may be invoked; we prefer an explicit configuration.
- `releaseVersion` ensure that processor that may be affected by java version works with the right and expected java version.
- `-encoding ...` forces UTF-8 encoding. For java generation, it is the only sensible configuration. It may be problematic to generate non-ascii properties files.

Some processors need to be configured with system properties:

- `<querydsl.generatedAnnotationClass>javax.annotation.processing.Generated</querydsl.generatedAnnotationClass>` ensures that querydsl generation uses Java 9+ `Generated` annotation.

Code generation is split in two maven *execution*: main and test sources, respectively bound to *generate-sources* and *generate-test-sources* phases.

plugins-processor provides *igloo.processor.** properties to ease processor configuration (no need to know full processor name, just remember bindgen, querydsl, immutables, ...).

13.3 IDE integration

13.3.1 Eclipse

Eclipse m2e + JBoss *Maven integration for Eclipse JDT APT Core* (org.jboss.tools.maven.apt.core) allow to handle code generation. It needs to override *Preferences* > *Maven* > *Annotation Processing* configuration and choose *Experimental: Delegate annotation processing to maven plugins*.

This configuration is handled by oomph igloo and project file.

13.3.2 IntelliJ

IntelliJ natively supports maven processor configuration. Please note that it expects processor to be on the project classpath (Eclipse and Maven allow to declare processor dependency in the plugin classpath).

Igloo >= 4.0.0 is changed to use a configuration pattern to ease IntelliJ configuration.

13.4 How to configure a project

13.4.1 Igloo >= 4.0.0

Add the following configuration to your project:

```
<project>
    <properties>
        <!-- adapt processor list to your use case -->
        <igloo.processors>${igloo.processor.querydsl},${igloo.processor.bindgen}</igloo.
↪processors>
    </properties>
    <dependencies>
        <!-- check processor configuration below to identify needed dependencies-->
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.bsc.maven</groupId>
                <artifactId>maven-processor-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
    <dependencyManagement>
        <dependencies>
            <!-- provides common processors versions -->
            <dependency>
                <groupId>org.iglooproject</groupId>
                <artifactId>dependencies-commons</artifactId>
                <version>${igloo-maven.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
```

(continues on next page)

(continued from previous page)

```
    </dependencyManagement>
</project>
```

If you do not use processor, just drop these configurations.

13.4.2 Igloo < 4.0.0

For Igloo < 4.0.0, adding the following plugin definition enable immutables/querydsl/bindgen generation.

```
<project>
    <build>
        <plugins>
            <plugin>
                <groupId>org.bsc.maven</groupId>
                <artifactId>maven-processor-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

For Maven/Eclipse build, processors are included in plugin dependencies, so code generation can be handled without project classpath pollution.

For IntelliJ build, it is needed either to add processor dependencies in project (use a provided scope if dependency is not needed at runtime) or to reconfigure maven-processor-plugin to remove missing processors (*configuration>processors* node).

13.4.3 Igloo < 3.3.0

Same as Igloo < 4.0.0, but immutables generation is missing.

13.5 Igloo 4 migration guide

This quick guide may handle common cases: apply this this changes **only** when `org.bsc.maven:maven-processor-plugin` is part of module's plugins list. Generally, it is needed both on core and webapp modules.

```
<project>
    <properties>
        <!-- pick the appropriate config -->
        <!-- core-style processor list -->
        <igloo.processors>${igloo.processor.querydsl},${igloo.processor.bindgen}</igloo.
↪processors>
        <!-- webapp-style processor list -->
        <igloo.processors>${igloo.processor.bindgen}</igloo.processors>
    </properties>
    <dependencies>
        <!-- ensure processor dependencies are present when processor is listed -->
```

(continues on next page)

(continued from previous page)

```
        <!-- bindgen dependencies -->
        <dependency>
            <groupId>org.bindgen</groupId>
            <artifactId>bindgen</artifactId>
        </dependency>
        <dependency>
            <groupId>org.iglooproject.components</groupId>
            <artifactId>bindgen-functional</artifactId>
            <version>${igloo-commons.version}</version>
        </dependency>

        <!-- querydsl dependencies -->
        <dependency>
            <groupId>com.querydsl</groupId>
            <artifactId>querydsl-apt</artifactId>
            <classifier>jpa</classifier>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>com.querydsl</groupId>
            <artifactId>querydsl-jpa</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.bsc.maven</groupId>
                <artifactId>maven-processor-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
    <dependencyManagement>
        <dependencies>
            <!-- ensure this dependencyManagement is present -->
            <dependency>
                <groupId>org.iglooproject</groupId>
                <artifactId>dependencies-bindgen</artifactId>
                <version>${igloo-maven.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.iglooproject</groupId>
                <artifactId>dependencies-hibernate</artifactId>
                <version>${igloo-maven.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
</project>
```

For uncommon usecase, querydsl or bindgen processing may be removed from above configuration.

13.6 Processors

Note: This part provides only processor-specific configuration. Check the above configuration to enable code generation.

13.6.1 Bindgen

Bindgen is responsible of **Binding* and **BindingPath* generation. Processor name is available with `${igloo.processor.bindgen}` property. Bindgen jar provides both generation and runtime code, so dependency must be installed with *compile* scope (default scope).

Bindgen can be configured with a `bindgen.properties` file (located at the module's root path) to customize:

- generation scope and exclusions
- custom parent class for generated bindings

Here's a sample file:

```
# keep java.util and java.lang in the list, so that we have IntegerBinding, LongBinding␣
↪and not GenericObjectBindingPath for fields
# adapt first package scope
scope=org.iglooproject,java.util,java.lang
skipBindKeywork=true
# customize binding parent so that it implements SerializableFunction2
bindingPathSuperClass=org.iglooproject.commons.util.binding.AbstractCoreBinding
```

All Igloo code and project code use the superclass customization to ease wicket and binding interactions.

pom.xml configuration must be adapted from this sample:

```
<project>
    <properties>
        <igloo.processors>${igloo.processor.bindgen}</igloo.processors>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.bindgen</groupId>
            <artifactId>bindgen</artifactId>
        </dependency>
        <!-- for custom superclass -->
        <dependency>
            <groupId>org.iglooproject.components</groupId>
            <artifactId>bindgen-functional</artifactId>
            <version>${igloo-commons.version}</version>
        </dependency>
    </dependencies>
</project>
```

It is used by igloo-parent igloo/igloo-components/igloo-component-jpa/pom.xml and **-core* module in projects.

13.6.2 QueryDSL

QueryDSL is responsible for *QEntity* generation.

pom.xml configuration must be adapted from this sample:

```
<project>
    <properties>
        <igloo.processors>${igloo.processor.querydsl}</igloo.processors>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.querydsl</groupId>
            <artifactId>querydsl-apt</artifactId>
            <classifier>jpa</classifier>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>com.querysql</groupId>
            <artifactId>querydsl-jpa</artifactId>
        </dependency>
    </dependencies>
</project>
```

It is used by igloo-parent igloo/igloo-components/igloo-component-jpa/pom.xml and **-core* module in projects.

13.6.3 Immutables

Immutables allows to generate builders for immutables POJO.

```
<project>
    <properties>
        <igloo.processors>${igloo.processor.immutables}</igloo.processors>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.immutables</groupId>
            <artifactId>value</artifactId>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

It is used by igloo-parent igloo/igloo-components/igloo-component-jpa-more/pom.xml to handle `TaskManagement`/`ImmutableTaskManagement` builder.

13.6.4 spring-context-indexer

This processor generates a `META-INF/spring.components` that is an index of spring annotation in the current module. It allows Spring to manage classpath scanning more efficiently (only indexes are parsed, and not the full classpath).

It is a all-or-nothing mechanism. If a *spring.components* file exists, then no classpath scanning is done. Spring can be forced to perform classpath scanning (and ignore index files) with system property `spring.index.ignore=true`.

```
<project>
    <properties>
        <igloo.processors>${igloo.processor.spring}</igloo.processors>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-indexer</artifactId>
            <scope>optional</scope>
        </dependency>
    </dependencies>
</project>
```

Warning: This processor is not yet used as it triggers build discrepancies for Maven and Eclipse:

- This plugin uses `classes/` output folder (as generated file is not a source to be compiled but a file to include in jar)
- Eclipse incorrectly manage this generated file. It is supposed that as it is not generated by Eclipse, it can remove this file during project cleaning/building
- Eclipse correctly manage this generated file if output folder is set `generated-sources/apt` (`<outputClassDirectory>${project.build.directory}/generated-sources/apt</outputClassDirectory>`)
- But with this setting, Maven does not include the generated file in jar file (as this folder is a source file, and is only meant to contain .java files to be compiled)

13.6.5 Spring boot autoconfiguration support

To allow spring boot to perform autoconfiguration, it is needed to generate a `spring-autoconfigure-metadata.properties` that describes applied spring boot autoconfiguration annotations (AutoConfigureAfter, AutoConfigureBefore, ConditionalOn)

CHAPTER

FOURTEEN

DEPENDENCIES MANAGEMENT

14.1 Igloo 4.0.0 dependency migration guide

This guides provides easy to apply dependency replacement for common use-cases. Just substitute removed dependencies with the provided blocks, then clean any duplicates.

14.1.1 Check javax.annotation dependency

Use dependency hierarchy to ensure that `javax.annotation:javax.annotation-api` dependency is added to your core and webapp project. If not, add this dependency declaration:

```
<dependencies>
    <dependency>
        <groupId>javax.annotation</groupId>
        <artifactId>javax.annotation-api</artifactId>
        <scope>provided</scope>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-commons</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Without this dependency, `@Resource` annotations are not handled correctly and Spring managed fields may be unexpectedly null.

14.1.2 igloo-component-*

With Igloo 4.0.0, Maven complains that version is missing for *igloo-component-** artifacts. It is now mandatory to declare *igloo-component-** version. Just add <version>${igloo.version}</version> to dependency declaration.

Examples: `igloo-component-wicket-more-test`, `igloo-component-jpa-more-test`, ...

14.1.3 org.iglooproject.components:igloo-components

Remove this dependency when encountered.

```
<dependency>
    <groupId>org.iglooproject.components</groupId>
    <artifactId>igloo-components</artifactId>
    <version>${igloo.version}</version>
    <type>pom</type>
</dependency>
```

14.1.4 igloo-component-wicket-bootstrap4

Add this dependency into your webapp project.

```
<dependencies>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-wicket-bootstrap4</artifactId>
        <version>${igloo.version}</version>
    </dependency>
</dependencies>
```

14.1.5 igloo-component-config-test

Add `igloo-component-config-test` to your core project. Despite its name, it provides mandatory default configuration.

```
<dependencies>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-config-test</artifactId>
        <scope>runtime</scope>
        <version>${igloo.version}</version>
    </dependency>
</dependencies>
```

14.1.6 igloo-package-core-spring-jpa-app

Replace `igloo-package-core-spring-jpa-app` by the following configuration.

```
<dependencies>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-spring</artifactId>
        <version>${igloo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-jpa-more</artifactId>
        <version>${igloo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-jpa</artifactId>
        <version>${igloo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-jpa-security</artifactId>
        <version>${igloo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-acl</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-beans</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
    </dependency>
```

(continues on next page)

(continued from previous page)

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-expression</artifactId>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-spring</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-hibernate</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

14.1.7 igloo-dependency-test

Replace `igloo-dependency-test` by the following configuration.

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <scope>test</scope>
    </dependency>
```

(continues on next page)

(continued from previous page)

```
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-core</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.assertj</groupId>
        <artifactId>assertj-guava</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-testing</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

You can drop h2 database driver for project that uses only PostgreSQL.

14.1.8 igloo-dependency-core-logging-log4j2

Replace `igloo-dependency-core-logging-log4j2` by the following configuration.

```
<dependencies>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <scope>runtime</scope>
    </dependency>
</dependencies>
<dependencyManagement>
```

(continues on next page)

(continued from previous page)

```
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-logging</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

14.1.9 igloo-package-web-wicket-app

Replace `igloo-package-web-wicket-app` by the following configuration.

```
<dependencies>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-wicket-more</artifactId>
        <version>${igloo.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-jul-to-slf4j</artifactId>
        <version>${igloo.version}</version>
    </dependency>
</dependencies>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-wicket</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

14.1.10 igloo-package-web-spring-security

Replace `igloo-package-web-spring-security` by the following configuration.

```
<dependencies>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-web-security</artifactId>
        <version>${igloo.version}</version>
    </dependency>
</dependencies>
```

14.1.11 Missing Igloo classes

With *igloo-commons* split, some classes can be missing, so it will be needed to add when adequate:

```
<dependencies>
    <!-- pick only needed dependencies -->
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-validator</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>bindgen-functional</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-batch-api</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-bean-api</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-collections</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-context</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-lang</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-security-api</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
    <dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-context</artifactId>
        <version>${igloo-commons.version}</version>
    </dependency>
</dependencies>
```

14.1.12 Missing version for non-igloo dependency

Check if the version definition is available in one of this module:

- igloo-maven dependencies-bindgen/pom.xml
- igloo-maven dependencies-commons/pom.xml
- igloo-maven dependencies-hibernate-search/pom.xml
- igloo-maven dependencies-hibernate/pom.xml
- igloo-maven dependencies-html/pom.xml
- igloo-maven dependencies-jersey2/pom.xml
- igloo-maven dependencies-logging/pom.xml
- igloo-maven dependencies-quality-annotations/pom.xml
- igloo-maven dependencies-spring/pom.xml
- igloo-maven dependencies-testing/pom.xml
- igloo-maven dependencies-tools/pom.xml
- igloo-maven dependencies-wicket/pom.xml

If the version is available, then update add this dependency in your pom.xml:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.iglooproject</groupId>
            <artifactId>dependencies-REPLACEME</artifactId>
            <version>${igloo-maven.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

14.1.13 Check dependency scopes

In your `pom.xml` (all modules), check that:

- All `-test` artifacts are with `<scope>test</scope>` (except `igloo-component-config-test`)
- Following artifacts are `<scope>provided</scope>`: `javax.annotation-api`, `javax.servlet-api`

14.1.14 Check maven build

Launch a command-line build `mvn clean install` and pay attention to any warning on duplicate or broken dependency configuration. Remove or fix issues.

14.2 Compare old and new dependencies

Once you project can be built, checkout your old code in a separate folder and compare your dependency list:

```
# adapt basic-application-webapp with your webapp module name
cd old-folder
mvn clean install -DskipTests
mvn dependency:list --batch-mode -pl :basic-application-webapp "-DoutputFile=$PWD/deps.
↪txt" -Dsort=true -DincludeScope=runtime -DincludeTypes=jar
cut -d: -f 1-3 $PWD/deps.txt > $PWD/deps2.txt

cd new-folder
mvn clean install -DskipTests
mvn dependency:list --batch-mode -pl :basic-application-webapp "-DoutputFile=$PWD/deps.
↪txt" -Dsort=true -DincludeScope=runtime -DincludeTypes=jar
cut -d: -f 1-3 $PWD/deps.txt > $PWD/deps2.txt

vimdiff old-folder/deps2.txt new-folder/deps2.txt
```

Normal difference are:

- test dependencies are correctly removed from dependencies (this test only check runtime dependencies) (junit, mockito, hamcrest, ...)
- `org.iglooproject.components:igloo-\*` are added (new module from commons split)
- dropped `org.iglooproject.components:igloo-component-commons` is removed
- useless `net.java.truecommons:truecommons-key-swing` is removed
- useless `org.springframework:spring-jcl` is removed

14.2.1 Other issues

Just ask for help!

CHAPTER

FIFTEEN

JAVASCRIPT & NPM

15.1 Purpose

We need to use npm build process for some client-side javascript components. This is needed to customize third-party libraries (bootstrap, ...), as it is needed to work on source javascript file, and not with generated/transpiled resources.

This use-case may be addressed with:

- third-party pure-javascript/npm repositories that provides npm packages we can embed in java application with webjars. This solution implies to create new repositories, manage their build/publish processes, and add steps for development environment setup;
- adapt mvn build to include npm building phases. It allows to keep an unified development/build/publish process.

For the time being we use the second solution. First solution may be used for bigger javascript development, to ease third-party contribution or to share individual javascript components.

15.2 Architecture

This description use `bootstrap5-override` as an example.

The purpose of `bootstrap5-override` is to package *bootstrap* resources into a webjar and to customize some resources.

Migrating from *bootstrap 4* to *bootstrap 5*, it is prefered to switch from a process where we can override published resources to a model where we have to override bootstrap sources.

It allows us to prepare our customization base on es2015+ sources, so we can write simpler source code (module import, native class inheritance instead of prototype customization, ...)

15.3 How to update Javascript resources in development environment

Eclipse :

- change source in `src/main/js`
- trigger a clean on your module
- resource(s) in `src/main/generated-js` must be updated, and dependants projects must reflect the change. Running tomcat must refresh the resource after a page reload

15.4 Project structure

Main purpose of this setup is to process javascript sources to distribution-ready assets :

- `src/main/js/*.js` : development files
- `src/main/generated-js/js` : files generated from the development files

This setup also allows to bind other npm tools such as live server for demonstration or unit tests.

This setup is based on a NPM + rollup build workflow.

15.4.1 package.json

- **scripts** : alias allowing to execute several commands or other aliases at once

15.4.2 rollup.config

- **input** : input development file
- **output** :
  - file: processed output file
  - format :
    - `UMD` (Universal Module Definition) : format managed on the browser side
    - `ESM` (ES Modules) : managed in node
  - generatedCode : standard to use -> how I should transform the JS
  - globals : specification of import aliases
- **external** : external library available, avoid importing everything
- **plugins** : the plugins to use
  - `babel` : allows to transpile code -> transform code from one language to another, in our case, only JS but in different versions
  - `resolve` : allows to find missing dependencies

15.5 Usage

Example of use in igloo with `bootstrap5-override`.

To use the commands, go to `igloo-parent/igloo/igloo-webjars/bootstrap5-override/`. It becomes possible to use the commands seen in the `package.json` file.
To see the list of available commands, just run: `npm run-script`. We note that these are aliases that allow to call several commands or other aliases. The most commonly used are :

- `npm run-script js` : generates files from development files -> transpiling, management of missing modules and code restructuring
- `npm run-script start` : starts the hugo server

`igloojs` also includes unit testing configuration.

15.6 Maven integration

See *NPM & Maven*.

CHAPTER

SIXTEEN

NPM & MAVEN

`igloojs` and `bootstrap5-override` uses npm packaging to process source to distribution-ready browser-side javascripts.

Both integrates with maven by using `frontend-maven-plugin`. Configuration is splitted between node/npm/rollup configuration (transform resources from `src/main/js` to `src/main/generated-js`) and maven configuration (trigger build and package `src/main/generated-js` as a webjar).

If you need to use the same mechanism for another module, see igloo-maven plugins-npm/README.md.

CHAPTER

SEVENTEEN

RELEASING

17.1 Releasing igloo

Note: Releasing can be performed only for `igloo-parent` if `igloo-maven` / `igloo-commons` are untouched. Just set IGLOO_XXX_VERSION accordingly and start procedure at `igloo-parent` step.

```
IGLOO_MAVEN_VERSION=xxx
IGLOO_COMMONS_VERSION=xxx

###############
# igloo-maven #
###############

mvn jgitflow:release-start
mvn -DskipTests -DnoDeploy jgitflow:release-finish
git push origin main dev v$IGLOO_MAVEN_VERSION

#################
# igloo-commons #
#################

# update parent
mvn versions:update-parent -pl . -DparentVersion=$IGLOO_MAVEN_VERSION -DskipResolution -
↪DgenerateBackupPoms=false
# update igloo.igloo-maven.version property
mvn versions:set-property -Dproperty=igloo.igloo-maven.version -DnewVersion=$IGLOO_MAVEN_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
# check changes with git diff
git add -A
git commit
# perform jgitflow release
mvn jgitflow:release-start
mvn -DskipTests -DnoDeploy jgitflow:release-finish
git push origin main dev v$IGLOO_COMMONS_VERSION

################
# igloo-parent #
################
```

(continues on next page)

(continued from previous page)

```
# update parents
mvn versions:update-parent -pl .,:igloo-parent-maven-configuration-common -
↪DparentVersion=$IGLOO_MAVEN_VERSION -DskipResolution -DgenerateBackupPoms=false
# update igloo-maven.version, igloo-commons.version
mvn versions:set-property -Dproperty=igloo-maven.version -DnewVersion=$IGLOO_MAVEN_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
mvn versions:set-property -Dproperty=igloo-commons.version -DnewVersion=$IGLOO_COMMONS_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
# check changes with git diff
git add -A
git commit
# perform jgitflow release
mvn jgitflow:release-start
mvn -DskipTests -DnoDeploy jgitflow:release-finish
git push origin master dev vX.X.X

########################################
# Switch back to snapshot dependencies #
########################################

IGLOO_MAVEN_VERSION=xxx-SNAPSHOT
IGLOO_COMMONS_VERSION=xxx-SNAPSHOT

#################
# igloo-commons #
#################

git checkout dev
# update parent
mvn versions:update-parent -pl . -DparentVersion=$IGLOO_MAVEN_VERSION -DskipResolution -
↪DgenerateBackupPoms=false
# update igloo.igloo-maven.version property
mvn versions:set-property -Dproperty=igloo.igloo-maven.version -DnewVersion=$IGLOO_MAVEN_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
git commit -a -m "switch back to SNAPSHOT"
git push

################
# igloo-parent #
################

git checkout dev
# update parents
mvn versions:update-parent -pl .,:igloo-parent-maven-configuration-common -
↪DparentVersion=$IGLOO_MAVEN_VERSION -DskipResolution -DgenerateBackupPoms=false
# update igloo-maven.version, igloo-commons.version
mvn versions:set-property -Dproperty=igloo-maven.version -DnewVersion=$IGLOO_MAVEN_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
mvn versions:set-property -Dproperty=igloo-commons.version -DnewVersion=$IGLOO_COMMONS_
↪VERSION -DprocessAllModules=true -DgenerateBackupPoms=false
git commit -a -m "switch back to SNAPSHOT"
git push
```

Commands listed above allow for each Igloo sub-project to update igloo dependencies version, perform commit, and push to repository. Release is performed by CI/CD.

17.2 Releasing org.iglooproject.webjars:boostrap4

Repository https://github.com/igloo-project/bootstrap4 is a lightweight wrapper to repackage `org.webjars.npm:bootstrap` under a specific groupId/artifactId. This is needed to allow embedding of different bootstrap version in the same igloo project (needed if console is bootstrap 5 and application is still bootstrap 4).

If a new bootstrap version is needed, change project version and push modification on main branch. main branch is automatically published.

17.3 Updating an igloo-based project

Once `igloo-parent` is released:

- `pom.xml`: parent version (igloo-parent-core-project)
- `pom.xml`: `igloo.version` property

17.4 Release a backport

If you need to release a hotfix on a previously released version (adapt version numbers):

```
# Create a branch from the desired root version
git checkout -b ft-4.4-deploy v4.4.0
# Apply a new SNAPSHOT version
mvn versions:set versions:commit -DnewVersion=4.4.2-SNAPSHOT -DprocessAllModules=true
git commit -a -m "Prepare 4.4.2 backport release"
# Backport and commit your fixes (manually, cherry-pick, ...)
# you can publish this version if needed by pushing your branch to CI
[...]

# When your branch is OK, update your version
mvn versions:set versions:commit -DnewVersion=4.4.2 -DprocessAllModules=true
git commit -a -m "Release 4.4.2 backport"
git tag v4.4.2
git push origin ft-4.4-deploy
git push refs/tags/v4.4.2

# CI build for ft-4.4-deploy will publish your new release
# Branch ft-4.4-deploy can be removed once published (we only keep tag).
```

CHAPTER

EIGHTEEN

MORE ABOUT MAVEN ARCHETYPE

Note: This page give precisions about Maven archetype deployment and project generation. Quick tutorial with step by step usage is also available *here*.

To initialize a new project based on the basic-application, you have to follow several steps. Each steps are detailed one by one in the following sections.

18.1 Build the archetype (optional)

Note: If archetype is available on Nexus repository, there is no need to build and publish it.

In the folder `igloo-parent/basic-application`

Listing 1: Install the archetype locally

```
./build-and-push-archetype.sh ../basic-application/ local
```

Listing 2: Install the archetype on our repository

```
./build-and-push-archetype.sh ../basic-application/ snapshot
```

18.2 Generate a new project

Place yourself in a new folder or somewhere like /tmp/. This command will generate a new folder where you are containing your new project.

Listing 3: Using you local repository

```
mvn archetype:generate -DarchetypeVersion=X.X -DarchetypeCatalog=local -DartifactId=your-
↪artifact-id -DgroupId=your.group.id -Dversion=0.1-SNAPSHOT -Dpackage=com.your.package -
↪DarchetypeApplicationNamePrefix="YourApplication" -
↪DarchetypeSpringAnnotationValuePrefix="yourApplication" -DarchetypeFullApplicationName=
↪"Customer - Your application" -DarchetypeDatabasePrefix=c_database_prefix -
↪DarchetypeDataDirectory=your-data-directory
```

Listing 4: Using the snapshot repository

```
mvn archetype:generate -DarchetypeCatalog=https://nexus.tools.kobalt.fr/repository/igloo-
↪snapshots/ -DartifactId=your-artifact-id -DgroupId=your.group.id -Dversion=0.1-
↪SNAPSHOT -Dpackage=com.your.package -DarchetypeApplicationNamePrefix="YourApplication"␣
↪-DarchetypeSpringAnnotationValuePrefix="yourApplication" -
↪DarchetypeFullApplicationName="Customer - Your application" -
↪DarchetypeDatabasePrefix=c_database_prefix -DarchetypeDataDirectory=your-data-directory
```

Listing 5: Using the release repository

```
mvn archetype:generate -DarchetypeCatalog=https://nexus.tools.kobalt.fr/repository/igloo-
↪releases/ -DartifactId=your-artifact-id -DgroupId=your.group.id -Dversion=0.1-SNAPSHOT␣
↪-Dpackage=com.your.package -DarchetypeApplicationNamePrefix="YourApplication" -
↪DarchetypeSpringAnnotationValuePrefix="yourApplication" -DarchetypeFullApplicationName=
↪"Customer - Your application" -DarchetypeDatabasePrefix=c_database_prefix -
↪DarchetypeDataDirectory=your-data-directory
```

18.3 Push the new project

Go in the newly generate folder containing your project and push it on gitlab :

```
/bin/bash init-git.sh <project_name> <git_url>
git push -u origin master
```

Note: please check issue|ci management and scm urls in root `pom.xml`

Warning: After having push your project, delete the project folder and initialize a new one directly from gitlab before starting your work.

CHAPTER

NINETEEN

JIMPORTDIFF

Long-term Igloo maintenance implies to split some code by modules, to ease testing, code and repository management. It implies to rename some java packages, so that some Igloo release are not backward compatible.

To ease Igloo project code rewriting on migration, `jimportdiff` is written to generate Igloo release report (what class/package/module is created in a new release).

It works by comparing two Igloo source tree to track deletion and code moves.

Tool is located here: https://github.com/igloo-project/jimportdiff.

Release notes provide further information when `jimportdiff` is appropriate.

19.1 Generate a release report

cf https://igloo-doc.readthedocs.io/en/latest/maven/jimportdiff.html

```
pipenv run ./jimportdiff rewrite --migration igloo5 igloo-4.4.1-5.0.0.json ../target-
↪project
```

CHAPTER

TWENTY

IGLOO LOGGING (LOG4J2 JMX CONFIGURATION)

`org.iglooproject.components:log4j2-jmx-helper` and `org.iglooproject.components:jul-helper` allow to override log4j2 configuration at runtime (including not predefined loggers and JUL loggers, contrary to log4j2 native JMX implementation).

Source code is hosted at https://github.com/igloo-project/igloo-logging/.

This modules do not rely on igloo framework and can be used independently of igloo.

20.1 Prerequisites

Your projet must manage/include the following dependencies:

- `javax.servlet:javax.servlet-api`: javax package, only servlet listener API is used
- `jakarta.servlet:jakarta.servlet-api`: jakarta package, only servlet listener API is used
- `org.slf4j:jul-to-slf4j`: 1.7.x or 2.0.x
- `org.slf4j:slf4j-api`: 1.7.x or 2.0.x
- `org.apache.logging.log4j:log4j-core`: 2.17+ version

Project uses only mainline API from these dependencies and may adapt to any recent version.

Jakarta or Javax servlet API is used based on the listener classes you use.

20.2 Installation

Ensure you use Igloo maven repository: https://nexus.tools.kobalt.fr/repository/igloo-releases/.

Add `org.iglooproject.components:log4j2-jmx-helper` and `org.iglooproject.components:jul-helper` to your project's dependencies.

Remove `org.iglooproject.components:igloo-component-jul-to-slf4j` as `jul-helper` handles sl4j to JUL binding.

```
                <dependency>
                        <groupId>org.iglooproject.components</groupId>
                        <artifactId>jul-helper</artifactId>
                        <version>1.0.1</version>
                        <scope>runtime</scope>
                </dependency>
                <dependency>
```

(continues on next page)

(continued from previous page)

```
                        <groupId>org.iglooproject.components</groupId>
                        <artifactId>log4j2-jmx-helper</artifactId>
                        <version>1.0.1</version>
                        <scope>runtime</scope>
                </dependency>

                <!-- check dependency tree for the following dependencies -->
                <dependency>
                        <groupId>org.slf4j</groupId>
                        <artifactId>jul-to-slf4j</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.apache.logging.log4j</groupId>
                        <artifactId>log4j-core</artifactId>
                </dependency>
```

For a war project, use `web.xml` listeners to trigger JMX helper loading (you must replace `org.iglooproject.slf4j.jul.bridge.SLF4JLoggingListener` if present):

```
        <!-- javax API (Tomcat < 10) -->
        <listener>
                <listener-class>igloo.julhelper.servlet.JulLoggingListener</listener-
↪class>
        </listener>
         <listener>
                <listener-class>igloo.log4j2jmx.servlet.Log4j2LoggingManagerListener</
↪listener-class>
        </listener>

        <!-- jakarta API (Tomcat >= 10) -->
        <listener>
                <listener-class>igloo.julhelper.servlet.JakartaJulLoggingListener</
↪listener-class>
        </listener>
         <listener>
                <listener-class>igloo.log4j2jmx.servlet.
↪JakartaLog4j2LoggingManagerListener</listener-class>
        </listener>
```

```
        // java API
        @Override
        public void onStartup(ServletContext servletContext) throws ServletException {
                // ...
                servletContext.addListener(JulLoggingListener.class);
                servletContext.addListener(Log4j2LoggingManagerListener.class);
        }

        // jakarta API
        @Override
        public void onStartup(ServletContext servletContext) throws ServletException {
                // ...
                servletContext.addListener(JakartaJulLoggingListener.class);
```

(continues on next page)

(continued from previous page)

```
            servletContext.addListener(JakartaLog4j2LoggingManagerListener.class);
        }
```

Else you need to invoke `JulLoggingManagerMBean.registerMBean()` and `Log4j2LoggingManagerMBean.registerMBean()`.

Jul helper must be initialized before Log4j2 helper.

20.3 Usage

Your MBeans tree must show a new `igloo` subtree with two `type=LoggingManager` MBeans:

- JulLoggingManager
- Log4j2LoggingManager

`JulLoggingManager` is controlled by `Log4j2LoggingManager`. It is not needed to interact directly with it.

Use an MBean browser (visualvm, jmxterm) to override your configurations at runtime. Any modification performed is lost at application restart.

Here are examples for jmxterm:

```
# MBean selection
bean igloo:type=LoggingManager,name=Log4j2LoggingManager
# MBean operations/attributes
info
# Override a logger
run setLevel "org.apache.wicket" "debug"
# Loading a web page may trigger debug logs
run reset
# Logger level must be restore to its previous state
```

20.4 JUL implementation

Specific code is needed to allow JUL reconfiguration at runtime, because `jul-to-slf4j` performs log redirection at root level (default behavior in `SLF4JBridgeHandler.install()`).

It implies to configure JUL not to block any log record, so that they can fo to root logger, so that `SLF4JBridgeHandler` can redirect it to log4j2. It implies a performance hit as not wanted log record need to be processed.

A common caveat is that JUL must be configured for maximum verbosity. Else high verbosity log records are discarded before `jul-to-slf4j` can redirect it.

JUL helper allow to automatically add and remove `SLF4JBridgeHandler` for the provided logger name when log4j2 is reconfigured to address this issues.

jullKnownLoggers setting (can be configured at runtime with JMX, or at startup-time with `JulLoggingListener` `julKnownLoggersResourcePath` servlet parameter) is used to limit `SLF4JBridgeHandler` initialization.

Default configuration is located in `jul-helper/well-known-jul-loggers.txt` classpath resource. By default, `SLF4JBridgeHandler` initialization is performed only for logger (and sub loggers):

```
org.glassfish.jersey
com.google.common
```

20.5 Project release

Use jgitflow procedure.

20.6 Changelog

20.6.1 1.1.0 (28.08.2023)

Support both `javax.servlet` and `jakarta.servlet` API. You must use appropriate listener variant for you application:

- unchanged for javax: `JulLoggingListener` and `Log4j2LoggingManagerListener`
- jakarta: `JakartaJulLoggingListener` and `JakartaLog4j2LoggingManagerListener`

20.6.2 1.0.2 (13.02.2023)

- Fix error messages when servlet is stopped due to servlet listener not registering installed mbeanObjectName at startup

20.6.3 1.0.1 (10.02.2023)

- Fix a deployment issue

20.6.4 1.0.0 (10.02.2023)

- Initial release
- Use 1.0.1 that fixes a deployment issue

CHAPTER

TWENTYONE

RELEASES 5.X

CHAPTER

TWENTYTWO

5.18.0 (TBD)

CHAPTER

TWENTYTHREE

5.17.0 (2024-04-15)

23.1 Bugfix

- Fix scss environment section.
- BS5: fix table row disabled.

23.2 Dependencies

- jackson/-annotations/-core/-databind: 2.16.0 -> 2.16.2
- jackson-dataformat-xml: 2.16.0 -> 2.16.2
- jackson-jaxrs-json-provider / jackson-jaxrs-xml-provider: 2.16.0 -> 2.16.2
- jackson-module-jaxb-annotations: 2.16.0 -> 2.16.2
- guava: 32.1.3jre -> 33.0.0jre
- querydsl-jpa / querydsl-apt / querydsl-core: 5.0.0 -> 5.1.0
- jsass: 5.10.5 -> 5.11.0
- junit-jupiter-api: 5.10.1 -> 5.10.2
- junit-platform-suite-engine: 1.10.1 -> 1.10.2
- log4j-core / log4j-slf4j-impl / log4j-api: 2.22.0 -> 2.23.1
- assertj-core / assertj-guava: 3.24.2 -> 3.25.3
- jersey-container-grizzly2-servlet: 2.40 -> 2.41
- jersey-client / jersey-server: 2.40 -> 2.41
- jersey-spring5: 2.40 -> 2.41
- jersey-media-multipart: 2.40 -> 2.41
- jersey-test-framework-provider-grizzly2: 2.40 -> 2.41
- jsoup: 1.17.1 -> 1.17.2
- mockito/-core/-junit-jupiter: 5.8.0 -> 5.11.0
- postgresql: 42.7.1 -> 42.7.2
- jcl-over-slf4j / jul-to-slf4j / slf4j-api: 2.0.9 -> 2.0.12
- spring-*: 5.3.31 -> 5.3.32

- spring-security-*: 5.8.8 -> 5.8.10
- webjars-locator-core: 0.55 -> 0.58
- value: 2.10.0 -> 2.10.1
- micrometer-core: 1.12.0 -> 1.12.4
- error_prone_annotations: 2.23.0 -> 2.26.0
- byte-buddy: 1.14.10 -> 1.14.12
- commons-compress: 1.25.0 -> 1.26.1
- commons-codec: 1.16.0 -> 1.16.1
- openpdf/openpdf-fonts-extra/pdf-swing/pdf-toolbox: 1.3.34 -> 1.4.1
- maven-compiler-plugin: 3.11.0 -> 3.12.1
- maven-failsafe-plugin: 3.2.2 -> 3.2.5
- maven-surefire-plugin: 3.2.2 -> 3.2.5
- maven-site-plugin: 4.0.0-M12 -> 4.0.0-M13
- maven-assembly-plugin: 3.6.0 -> 3.7.0
- spotless-maven-plugin: 2.41.1 -> 2.43.0
- dependency-check-maven: 9.0.4 -> 9.0.9
- exec-maven-plugin: 3.1.1 -> 3.2.0
- flatten-maven-plugin: 1.5.0 -> 1.6.0

CHAPTER

TWENTYFOUR

5.15.1 (2024-02-20)

24.1 Bugfix

- BS5: fix body bg color.

CHAPTER

TWENTYFIVE

5.15.0 (2024-02-05)

25.1 Breaking changes

- Remove Log4j 1.x / Reload4j support
- Remove ehcache2 support

25.2 Enhancement

- Add `properties.hidden` configuration to hide confidential values in Properties page in admin console.
- Add Wicket listener form clear input.

25.3 Bugfix

- Storage: add missing transactionnal read only on `getFichierById(Long)`.
- Fix `BootstrapBadge` markup html comments.
- Select2 : fix `allowClear` setting if not required.
- BasicApp: fix `ReferenceData` drop down search bean interface.
- Fix Wicket converter locator for enums anonymous class.

25.4 Dependencies

- **Font Awesome 6.3.0 -> 6.5.1**
- jquery-ui: 1.12.1 -> 1.13.2
- select2: 4.0.10 -> 4.0.13
- jackson: 2.15.2 -> 2.16.0
- guava: 32.1.2jre -> 32.1.3jre
- h2: 2.2.220 -> 2.2.224
- HikariCP: 5.0.1 -> 5.1.0
- commons-io: 2.13.0 -> 2.15.1

- junit-jupiter-api: 5.10.0 -> 5.10.1
- junit-platform-suite-engine: 1.10.0 -> 1.10.1
- log4j-core: 2.20.0 -> 2.22.0
- poi: 5.2.3 -> 5.2.5
- wicketstuff-select2: 9.13.0 -> 9.16.0
- flyway-core: 9.22.2 -> 9.22.3
- jsoup: 1.16.1 -> 1.17.1
- mockito: 5.5.0 -> 5.8.0
- passay: 1.6.3 -> 1.6.4
- postgresql: 42.6.0 -> 42.7.1
- slf4j-api: 2.20.0 -> 2.22.0
- spring: 5.3.30 -> 5.3.31
- spring-security: 5.8.7 -> 5.8.8
- webjars-locator-core: 0.53 -> 0.55
- value: 2.9.3 -> 2.10.0
- micrometer-core: 1.11.4 -> 1.12.0
- error_prone_annotations: 2.22.0 -> 2.23.0
- spring-boot: 2.7.16 -> 2.7.18
- byte-buddy: 1.14.8 -> 1.14.10
- commons-compress: 1.24.0 -> 1.25.0
- commons-text: 1.10.0 -> 1.11.0
- commons-lang3: 3.13.0 -> 3.14.0
- commons-validator: 1.7 -> 1.8.0
- openpdf: 1.3.30 -> 1.3.34
- maven-clean-plugin: 3.3.1 -> 3.3.2
- maven-failsafe-plugin: 3.1.2 -> 3.2.2
- maven-surefire-plugin: 3.1.2 -> 3.2.2
- maven-dependency-plugin: 3.6.0 -> 3.6.1
- maven-javadoc-plugin: 3.6.0 -> 3.6.3
- maven-project-info-reports-plugin: 3.4.5 -> 3.5.0
- maven-site-plugin: 4.0.0-M9 -> 4.0.0-M12
- maven-processor-plugin: 5.0jdk8 -> 5.0
- jacoco-maven-plugin: 0.8.10 -> 0.8.11
- dependency-check-maven: 8.4.0 -> 9.0.4
- versions-maven-plugin: 2.16.1 -> 2.16.2
- exec-maven-plugin: 3.1.0 -> 3.1.1

- frontend-maven-plugin: 1.14.0 -> 1.15.0

CHAPTER

TWENTYSIX

5.14.0 (2023-11-24)

26.1 Bugfix

- Important fix - issue 84: Broken browser cache support for scss. Since version 5.10.0, timestamp information included in scss filename is broken. Timestamp is constant, so it defeats cache refresh mechanism. Issue only appears if build-time scss is used. This release fixes this problem.

  If you use any of the workarounds given on issue, you should remove it.

- ClipboardBehavior fix: text configuration is correctly set
- `GenericEntity.*COLLATOR`: fix foreign language collator configuration
- Workaround focustrap for bootstrap Modal: we disable focustrap that is buggy when select2 or complex javascript widget are used inside modal. No action is needed for this fix.

  Issue still exists for offcanvas.

  Further work is to be done to enable both focustrap and complex javascript components on both modal and offcanvas components.

- DataTableBuilder: use `button` markup for action instead of `a`. It allows not to focus empty anchor (top of the page) when clicked.

26.2 Enhancement

- `DataTableBuilder`: update side link icon.

CHAPTER

TWENTYSEVEN

5.13.0 (2023-09-22)

27.1 Bugfix

- `DataTableBuilder`: fix side link icon position.
- Update `BootstrapColor` values (add `dark` and `light` - remove `todo`)

If you encounter `package com.google.errorprone.annotations does not exist` at compilation time, it's related to code generation change from immutables library. Add `com.google.errorprone:error_prone_annotations` to your project's dependencies.

27.2 Dependencies

- spring: 5.3.28 -> 5.3.30
- spring-boot: 2.7.14 -> 2.7.16
- spring-security: 5.8.3 -> 5.8.7
- guava: 32.1.1-jre -> 32.1.2-jre
- caffeine: 3.1.6 -> 3.1.8
- ph-css: 7.0.0 -> 7.0.1
- byte-buddy: 1.14.5 -> 1.14.8
- commons-compress: 1.23.0 -> 1.24.0
- commons-lang3: 3.12.0 -> 3.13.0
- micrometer: 1.11.2 -> 1.11.4
- picocli: 4.7.4 -> 4.7.5
- flyway-core: 9.20.1 -> 9.22.2
- h2: 2.2.220 -> 2.2.224
- slf4j: 2.0.7 -> 2.0.9
- junit: 5.9.3 -> 5.10.0
- mockito: 5.4.0 -> 5.5.0
- errorprone: 2.20.0 -> 2.22.0

Maven plugins:

- frontend-maven-plugin: 1.13.4 -> 1.14.0
- maven-antrun-plugin: 3.1.0
- maven-javadoc-plugin: 3.5.0 -> 3.6.0
- maven-enforcer-plugin: 3.3.0 -> 3.4.1
- owasp-maven-plugin: 8.3.1 -> 8.4.0
- versions-maven-plugin: 2.16.0 -> 2.16.1
- maven-processor-plugin: 5.0-rc3 -> 5.0-jdk8

CHAPTER

TWENTYEIGHT

5.12.1 (2023-08-28)

Warning: 5.12.0: flyway + spring-boot is broken, version is not released ; use 5.12.1 instead.

28.1 Enhancement

- `JpaSearchQuery`: add `innerJoin` for `EntityPath`.
- Backport 6.x: allow AbstractNotificationContentDescriptorFactory to override wicket application context (see *E-mail notifications*)

28.2 Dependencies

- spring: 5.3.27 -> 5.3.28
- spring-boot: 2.7.12 -> 2.7.14
- spring-security: 5.8.2 -> 5.8.3
- jersey2: 2.39.1 -> 2.40
- guava: 32.0.0-jre -> 32.1.1-jre
- commons-codec: 1.15 -> 1.16.0
- commons-io: 2.11.0 -> 2.13.0
- micrometer: 1.11.0 -> 1.11.2
- picocli: 4.7.3 -> 4.7.4
- jboss-logging-annotations: 3.5.0.Final -> 3.5.3.Final
- flyway-core: 9.19.1 -> 9.20.1
- h2: 2.1.214 -> 2.2.220
- webjars-locator-core: 0.52 -> 0.53
- popper2: 2.11.7 -> 2.11.8
- mockito: 4.3.1 -> 5.4.0
- logunit: 1.1.3 -> 2.0.0
- errorprone: 2.19.1 -> 2.20.0

Maven plugins:

- maven-surefire-plugin: 3.1.0 -> 3.1.2
- maven-failsafe-plugin: 3.1.0 -> 3.1.2
- maven-war-plugin: 3.3.2 -> 3.4.0
- buildnumber-maven-plugin: 3.1.0 -> 3.2.0
- frontend-maven-plugin: 1.12.1 -> 1.13.4
- maven-clean-plugin: 3.2.0 -> 3.3.1
- maven-site-plugin: 4.0.0-M8 -> 4.0.0-M9
- maven-project-info-reports-plugin: 3.4.4 -> 3.4.5
- owasp-maven-plugin: 8.2.1 -> 8.3.1
- versions-maven-plugin: 2.15.0 -> 2.16.0

CHAPTER

TWENTYNINE

5.11.0 (2023-08-08)

29.1 Bugfix

- BS4 / BS5: fix datepicker z-index.
- Task: fix markup detail page.
- DataTableBuilder: fix action btn placeholder.
- issue 82: @ManifestPropertySource on @Configuration inner class fails

29.2 Enhancement

- Storage: add update filename service method
- Storage: disable Fichier caching
- Consistency on `IConverter` ressource keys error.
- `GenericEntity`: rename Hibernate Search field `ID_SORT` to `ID`.

CHAPTER

THIRTY

5.10.1 (2023-06-05)

30.1 Bugfix

- issue 81: Loading SCSS from jar file triggers NPE

CHAPTER

THIRTYONE

5.10.0 (2023-06-02)

31.1 Breaking change

- dropped TrueVFS support: historically used by Igloo
  - **needed change**: you need to remove `openTFileRegistryFilter` `filter` and `filter-mapping` from your servlet container configuration (`web.xml` or java configuration).
  - to read excel files in `AbstractImportDataServiceImpl`; these feature now uses resource loading mechanisms, and TrueVFS removal has no effect.
  - to unwrap file from archives in `SimpleFileStoreImpl`; if you do not rely on implicit archive-related behavior and FileStore, TrueVFS removal has no effect.
  - if you rely on TrueVFS for custom usage, you may:
    * replace TrueVFS with classic file APIs
    * keep your TrueVFS code, and perform the needed modification related to the `openTFileRegistryFilter` servlet filter removal.

CHAPTER

THIRTYTWO

5.9.0 (2023-06-02)

32.1 Dependencies

- jackson: 2.14.2 -> 2.15.2
- wicket-webjars: 3.0.6 -> 3.0.7
- junit-jupiter-api: 5.9.2 -> 5.9.3
- junit-platform-suite-engine: 1.9.2 -> 1.9.3
- wicket: 9.12.0 -> 9.13.0
- wicketstuff-select2: 9.12.0 -> 9.13.0
- flyway-core: 9.16.1 -> 9.19.1
- jsoup: 1.15.4 -> 1.16.1
- mockito: 5.2.0 -> 5.3.1
- spring: 5.3.26 -> 5.3.27
- spring-security: 5.8.2 -> 5.8.3
- micrometer-core: 1.10.5 -> 1.11.0
- error_prone_annotations: 2.18.0 -> 2.19.1
- spring-boot: 2.7.10 -> 2.7.12
- byte-buddy: 1.14.3 -> 1.14.5
- picocli: 4.7.1 -> 4.7.3
- caffeine: 3.1.5 -> 3.1.6
- guava: 31.1-jre -> 32.0.0->jre

Maven plugins:

- maven-dependency-plugin: 3.5.0 -> 3.6.0
- maven-assembly-plugin: 3.5.0 -> 3.6.0
- maven-source-plugin: 3.2.1 -> 3.3.0
- maven-failsafe-plugin: 3.0.0 -> 3.1.0
- maven-surefire-plugin: 3.0.0 -> 3.1.0
- maven-project-info-reports-plugin: 3.4.2 -> 3.4.4
- maven-site-plugin: 4.0.0-M6 -> 4.0.0-M8

- jacoco-maven-plugin: 0.8.9 -> 0.8.10
- maven-enforcer-plugin: 3.2.1 -> 3.3.0
- flatten-maven-plugin: 1.4.1 -> 1.5.0
- buildnumber-maven-plugin: 3.0.0 -> 3.1.0

CHAPTER

THIRTYTHREE

5.8.0 (2023-06-02)

33.1 Bugfix

- BS5: fix modal sizing override.

CHAPTER

THIRTYFOUR

5.7.1 (2023-05-25)

34.1 Bugfix

- AbstractExcelTableExport : fix addTextCell method. Check length of text, if text length is greater than maxTextLength then text is susbtring.

CHAPTER

THIRTYFIVE

5.7.0 (2023-05-10)

35.1 Bugfix

- Remove target behavior on unbind in collapse behavior.
- BS5: fix scss `.btn-form-control-check`.

35.2 Enhancement

- BS5: fix sass slash div depercated.
- BS5: add collapse toggle css visibility utilities.

CHAPTER

THIRTYSIX

5.6.0 (2023-04-19)

36.1 Bugfix

- issue 79: Storage: split on StorageUnit size is broken
- Lookup BS4/BS5: fix web socket broadcast case.

36.2 Enhancement

- Bootstrap 5: rename `.heading-section-white` to `.heading-section-light` for consistency.
- Storage: added `StorageService#isCreatedBy`
- maven-site-plugin: added XML format for dependencies reports

36.3 Breaking change

- JUnit4 : dropped support ; `AbstractJunit4TestCase` is removed

36.4 Dependencies

- hibernate: 5.6.14 -> 5.6.15
- hibernate-search: 5.11.11 -> 5.11.12
- spring: 5.2.25 -> 5.2.26
- spring-security: 5.8.1 -> 5.8.2
- flywaydb: 9.14.1 -> 9.16.1
- spring-boot: 2.7.8 -> 2.7.10
- jersey2: 2.38 -> 2.39.1
- slf4j: 2.0.6 -> 2.0.7
- log4j2: 2.19.0 -> 2.20.0
- commons-fileupload: 1.4 -> 1.5
- postgresql: 45.2.3 -> 42.6.0

- commons-compress: 1.22 -> 1.23.0
- byte-buddy: 1.12.23 -> 1.24.3
- popper2: 2.11.6 -> 2.11.7
- reload4j: 1.2.24 -> 1.2.25
- passay: 1.6.2 -> 1.6.3
- micrometer: 1.10.3 -> 1.10.5
- mockito: 5.1.1 -> 5.2.0
- jsoup: 1.15.3 -> 1.15.4

Maven plugins:

- owasp-maven-plugin: 8.0.2 -> 8.1.2
- versions-maven-plugin: 2.14.2 -> 2.15.0
- maven-resources-plugin: 3.3.0 -> 3.3.1
- jacoco-maven-plugin: 0.8.8 -> 0.8.9
- maven-compiler-plugin: 3.10.1 -> 3.11.0
- maven-surefire-plugin: 3.0.0-M8 -> 3.0.0
- maven-assembly-plugin: 3.4.2 -> 3.5.0
- maven-javadoc-plugin: 3.4.1 -> 3.5.0
- maven-failsafe-plugin: 3.0.0-M8 -> 3.0.0
- maven-deploy-plugin: 3.0.0 -> 3.1.1
- maven-site-plugin: 4.0.0-M4 -> 4.0.0-M6
- maven-install-plugin: 3.1.0 -> 3.1.1
- flatten-maven-plugin: 1.3.0 -> 1.4.1

CHAPTER

THIRTYSEVEN

5.5.2 (2023-03-31)

37.1 Bugfix

- Fix console configuration for custom configuration.
- Fix BS5 popover and tooltip html content visibility.

CHAPTER

THIRTYEIGHT

5.5.1 (2023-03-21)

38.1 Bugfix

- Fix dart-css `@import` triggered by bootstrap 4 basic-application (*has no known prefix* error).

CHAPTER

THIRTYNINE

5.4.1 (2023-03-21)

39.1 Bugfix

- Fix dart-css `@import` triggered by bootstrap 4 basic-application (*has no known prefix* error).

CHAPTER

FORTY

5.5.0 (2023-03-15)

40.1 Enhancement

- cache : new reference implementation for caching is caffeine

40.2 Bugfix

- BS5: rollback HTML emails to BS4.
- BS5: remove BS4 webjar dependency from `wicket-bootstrap5`.

40.3 Breaking changes

- Spring cache must switch to caffeine (mandatory), Hibernate cache can switch to Caffeine (recommended). See caffeine-migration.

CHAPTER

FORTYONE

5.4.0 (2023-03-13)

41.1 Enhancement

- *Scss/Sass processing*
  - jsass/libsass replaced by dart-sass
  - scss generated at build-time (optional)

41.2 Breaking changes

- scss must be adapted for dart-sass processing (mandatory). See *JSass / Dart-sass migration*.
- webapp `pom.xml` must be adapted for build-time generation (optional). See *Build-time generation*.

CHAPTER

FORTYTWO

5.3.1 (2023-03-13)

42.1 Bugfix

- BS5 - Tab : fix selector tabRenderHead.

CHAPTER

FORTYTHREE

5.3.0 (2023-02-24)

43.1 Bugfix

- BS5: fix feedback dismiss padding.

43.2 Enhancements

- Add WIP modal for indexation process.

43.3 Dependencies

- Bootstrap 5 5.1.3 -> 5.2.3
- Font Awesome 6.1.2 -> 6.3.0
- junit5-suite 1.9.1 -> 1.9.2

CHAPTER

FORTYFOUR

5.2.0 (2023-02-15)

44.1 Enhancements

- BasicApp: use postgresql instead of h2 for unit tests.
- BasicApp: clean db port in properties files.
- BasicApp: webapp now uses *Igloo logging (log4j2 JMX configuration)* JMX helpers

44.2 Dependencies

- jackson 2.14.1 -> 2.14.2
- ph-css 6.5.0 -> 7.0.0
- junit-jupiter-api 5.9.1 -> 5.9.2
- assertj 3.5.0 -> 3.24.2
- flyway 9.11.0 -> 9.14.1
- freemarker 2.3.31 -> 2.3.32
- jersey 2.37 -> 2.38
- mockito 4.9.0 -> 5.1.1
- postgresql 42.5.1 -> 42.5.3
- jul-to-slf4j / slf4j-api 2.0.5 -> 2.0.6
- spring 5.3.24 -> 5.3.25
- spring-security 5.8.0 -> 5.8.1
- immutables value 2.9.2 -> 2.9.3
- micrometer-core 1.10.2 -> 1.10.3
- error_prone_annotations 2.16 -> 2.18.0
- spring-boot 2.7.6 -> 2.7.8
- byte-buddy 1.12.19 -> 1.12.23
- picocli 4.7.0 -> 4.7.1

Maven plugins:

- maven-failsafe-plugin 3.0.0-M7 -> 3.0.0-M8

- maven-surefire-plugin 3.0.0-M7 -> 3.0.0-M8
- maven-dependency-plugin 3.4.0 -> 3.5.0
- maven-project-info-reports-plugin 3.4.1 -> 3.4.2
- maven-install-plugin 3.0.1 -> 3.1.0
- dependency-check-maven 7.4.1 -> 8.0.2
- maven-enforcer-plugin 3.1.0 -> 3.2.1
- versions-maven-plugin 2.13.0 -> 2.14.2

CHAPTER

FORTYFIVE

5.1.2 (2023-02-13)

45.1 Bugfix

- BS5: fix datepicker `z-index` on inputs with errors.

CHAPTER

FORTYSIX

5.1.1 (2023-02-09)

46.1 Bugfix

- Fix content css class utility.

CHAPTER

FORTYSEVEN

5.1.0 (2023-02-07)

47.1 Enhancements

- Replace igloo custom flyway integration with spring-boot flyway integration.

47.2 Breaking changes

- Spring-boot Flyway integration needs some configuration and code migration on your projects. Please use *Igloo 5.1.x Flyway migration* to update your project.

CHAPTER

FORTYEIGHT

5.0.5 (2023-02-06)

48.1 Bugfix

- Remove overflow hidden on table cells.
- BasicApp: multiple BS5 fixes.

48.2 Enhancements

- BasicApp: minor code consistency.

CHAPTER

FORTYNINE

5.0.4 (2023-01-10)

49.1 Bugfix

- Popover: hide the whole popover panel if the content component is not visibile.

CHAPTER

FIFTY

5.0.3 (2023-01-03)

50.1 Bugfix

- issue 74: AbstractOfflinePanelRendererServiceImpl is broken with bootstrap component

50.2 API Changes

- `AbstractSimpleWicketNotificationDescriptor`: a `getComponentClass()` is needed to provide return type information of `getComponent()` method.
- `AbstractOfflinePanelRendererServiceImpl` `renderComponent` and `renderPage` methods are modified to use `IOfflineComponentProvider` in place of `SerializableSupplier2<Component>`: a helper method `IOfflineComponentProvider.fromSupplier(...)` can be used to easily rewrite your method calls.
- `IBootstrap4Component` and `IBootstrap5Component` are added to mark root component used in notification emails (so that subcomponent can determine bootstrap version).

50.3 Dependencies

- jackson 2.13.4 -> 2.14.1
- byte-buddy 1.12.18 -> 1.12.19
- flywaydb 9.7.0 -> 9.8.3
- postgresql 42.5.0 -> 42.5.1
- spring 5.3.23 -> 5.3.24
- spring-boot 2.7.5 -> 2.7.6
- spring-security 5.7.3 -> 5.8.0
- micrometer 1.9.5 -> 1.10.2

Other dependencies

- mockito 4.8.1 -> 4.9.0
- maven-dependency-plugin 3.3.0 -> 3.4.0
- owasp-maven-plugin 7.3.0 -> 7.4.1

5.0.2 (2022-12-19)

51.1 Bugfix

- issue 72: Fix ignored log4j2 configuration override

CHAPTER

FIFTYTWO

5.0.1 (2022-12-16)

52.1 Bugfix

- issue 73: Fix default Tooltip configuration

CHAPTER

FIFTYTHREE

5.0.0 (2022-12-08)

53.1 Enhancements

- Joiners: add slash with space before and after

53.2 Feature

- Igloo support both bootstrap 4 and bootstrap 5. Existing Igloo UI components support both bootstrap 4 and bootstrap 5 with minimal changes. But any custom project markups may need to be rewritten to accomodate bootstrap behavior changes.
- Check migration guide (below) for practical advice. A tool `jimportdiff` can handle tedious rewrite tasks.

53.3 Removed

- `WicketBootstrapComponentsModule` deleted ; replaced by `IBootstrapProvider`. Does not affect final projects.
- `AbstractPopoverLinkContentPanel`: unused
- `BootstrapTooltipDocumentBehavior` replaced by `BootstrapTooltipBehavior`
- `AlertJavascriptResourceReference`: unused
- `ILongRunningPage`, `AdaptativeTimeoutRequestCycleSettings`: unused
- `CommentOutModifier`: unused
- `AbstractBootstrapWebappConfig`, `WicketBootstrapServiceConfig`: unused
- `IBootstrap*Module`, `Bootstrap*Module`: unused by projects
- `AbstractBootstrapTooltipBehavior`: unused
- `BootstrapModalStatement`: unused by projects
- `BootstrapConfirmStatement`: unused by projects
- `BootstrapTab`: unused by projects
- `BootstrapButtonBehavior`: unused
- `BootstrapAlertBehavior`: unused
- `BootstrapScrollSpyBehavior`: unused

- *ConfirmButton: unused (use ConfirmLink instead)

53.4 Behavior changes

- ActionRenderers (from basic-application): replace all BootstrapRenderer by IBootstrapRenderer (except BootstrapRenderer.constant).
- Popover :
  - customClass become a raw string (no longer a string list)
  - default placement change from `auto` to `right`
- Alert :
  - Igloo alert components are renamed feedback (as this components are not linked with bootstrap alert component)
  - AlertJavascriptResourceReference -> FeedbackJavaScriptResourceReference
  - `$.fn.alert.*` -> `$.fn.feedback.*`
  - Not used by projects, usage is generally hidden and handled by Igloo correctly.
- Sidebar
  - bootstrap 4 : a custom js is used
  - bootstrap 5 : offcanvas is used and activated with data-attributes : `bs-data-toggle` and `bs-data-dismiss`.
- Confirm : this is a custom Igloo component based on bootstrap modal implementation
  - confirmation event is renamed `confirm.bs.confirm`
  - cancellation event is renamed `cancel.bs.confirm`
  - data attributes are renamed (if you use Igloo behavior to generate markup, no change are needed, check that you have no occurence of `modal-confirm`):
    - `modal-confirm-text` -> `bs-text`
    - `modal-confirm-title` -> `bs-title`
    - `modal-confirm-yes-label` -> `bs-yes-label`
    - `modal-confirm-no-label` -> `bs-no-label`
    - `modal-confirm-yes-icon` -> `bs-yes-icon`
    - `modal-confirm-no-icon` -> `bs-no-icon`
    - `modal-confirm-yes-button` -> `bs-yes-button`
    - `modal-confirm-no-button` -> `bs-no-button`
    - `modal-confirm-text-no-escape` -> `bs-no-escape`
    - `modal-confirm-css-class-names` -> `bs-css-class-names`
- Webjars
  - `webjars/` prefixes are not needed in webjars urls, and can trigger unexpected behavior in webjar lookup
  - Check your webjar ResourceReference for the issue:

```
grep -r --include "*ResourceReference.java" webjars/
```

- OpenTFileRegistryFilter
  - moved to `org.iglooproject.truevfs.filter.OpenTFileRegistryFilter`
  - add `org.iglooproject.components:igloo-component-truevfs` dependency if needed
  - update references to `OpenTFileRegistryFilter` (web.xml)

```
<filter>
  <filter-name>openTFileRegistryFilter</filter-name>
  <filter-class>org.iglooproject.wicket.servlet.filter.OpenTFileRegistryFilter</
↪filter-class>
</filter>
```

```
<filter>
  <filter-name>openTFileRegistryFilter</filter-name>
  <filter-class>org.iglooproject.truevfs.filter.OpenTFileRegistryFilter</filter-
↪class>
</filter>
```

53.5 Other major renames

- igloo-component-wicket
  - `org.iglooproject.wicket.markup.html.model` -> `igloo.wicket.model`
  - `org.iglooproject.wicket.markup.html.basic` -> `igloo.wicket.component`
- igloo-component-wicket-bootstrap4 / console
  - module `igloo-component-wicket-bootstrap5`, package `igloo.console`
- `org.iglooproject.wicket.bootstrap4.markup.html.template.js.select2` -> module `igloo-component-wicket-select2/igloo.select2`
- `org.iglooproject.wicket.bootstrap4.markup.html.template.js.jqueryui` -> module `igloo-component-wicket-jqueryui/igloo.jqueryui`
- `org.iglooproject.wicket.bootstrap4.markup.html.template.css.jqueryui` -> module `igloo-component-wicket-jqueryui/igloo.jqueryui`
- `org.iglooproject.wicket.bootstrap4.markup.html.template.css.fontawesome` -> module `igloo-component-wicket-fontawesome/igloo.fontawesome`

53.6 Bootstrap 5 JS resources

All boostrap 5 resources are packed inside a bundle. There is no separated resources for each component. Only Igloo custom components are delivered separatly.

53.7 Migration guide

These instructions may help to migrate a bootstrap 4 project easily. These instructions **do not migrate from bootstrap 4 to bootstrap 5**. Migrated project stay with bootstrap 4.

53.7.1 General procedure

- As usual: switch igloo-maven, igloo-commons and igloo dependencies and parent pom version
- find and update resources extending `Webjars*ResourceReference` and remove `webjars/` prefix in resource reference's name
- Use jimportdiff rewrite (see below):
  - rewrite classes (boostrap 4 priority allowed)
  - scss: rename `webjars://bootstrap-override:` -> `webjars://bootstrap4-override:`
  - scss: rename `webjars://bootstrap:` -> `webjars://bootstrap4:`
  - `web.xml` Log4JUrlFilter and OpenTFileRegistryFilter renames
  - some advices
- check/add `igloo-component-truevfs` dependency
- check/add `igloo-component-wicket` dependency
- check/add `igloo-component-wicket-console` dependency
- find and update base Template to implement `IBootstrap4Page` (check your page inheritance tree)
- manual fix of tooltip behavior
- manual fix of popover behavior
- manual fix of `IBootstrapRenderer`: replace `BootstrapRenderer` by `IBootstrapRenderer` so that interface is correctly implemented
- add table-cell-widths in `_bootstrap-variables.scss`
- update console css:
  - `rm -rf <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/common/template/resources/styles/console/*`
  - `cp -ar ~/git/igloo-parent/basic-application/basic-application-webapp/src/main/java/org/iglooproject/basicapp/web/application/common/template/resources/styles/application/console/* <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/common/template/resources/styles/console/` : 13 modified files
  - `git checkout HEAD $( find <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/common/template/resources/styles/console/ -name 'Console*ResourceReference.java' )` : rollback java file modifications, 2 modified files
- update feedback panel:
  - `cp ~/git/igloo-parent/basic-application/basic-application-webapp/src/main/java/org/iglooproject/basicapp/web/application/console/common/component/ConsoleAccessEnvironmentPanel.java <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/console/common/component/ConsoleAccessEnvironmentPanel.java`

  - cp ~/git/igloo-parent/basic-application/basic-application-webapp/src/main/java/org/iglooproject/basicapp/web/application/console/common/component/ConsoleEnvironmentPanel.java <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/console/common/component/ConsoleEnvironmentPanel.java
  - cp ~/git/igloo-parent/basic-application/basic-application-webapp/src/main/java/org/iglooproject/basicapp/web/application/console/common/component/ConsoleEnvironmentPanel.html <PROJECT_WEBAPP>/src/main/java/<BASE_PACKAGE>/web/application/console/common/component/ConsoleEnvironmentPanel.html
  - fix package declaration / package imports / *APPLICATION*Session call

- adapt custom console pages

53.7.2 jimportdiff

cf *jimportdiff*

```
pipenv run ./jimportdiff rewrite --migration igloo5 igloo-4.4.1-5.0.0.json ../target-
↪project
```

CHAPTER

FIFTYFOUR

RELEASES 4.X

CHAPTER

FIFTYFIVE

4.4.2 (2022-12-19)

55.1 Bugfix

- issue 72: Fix ignored log4j2 configuration override

CHAPTER

FIFTYSIX

4.3.1 (2022-12-19)

56.1 Bugfix

- issue 72: Fix ignored log4j2 configuration override

CHAPTER

FIFTYSEVEN

4.4.1 (2022-11-14)

57.1 Bugfixes

- BasicApp: style improvements https://github.com/igloo-project/igloo-parent/commit/2dada38310da2f15371cd3055b5bf4a582de9be9
- BasicApp: `UserGroup` popup - fix label padding https://github.com/igloo-project/igloo-parent/commit/c3d4dc26cf707b9f21888e45951472b35d41e04c
- Fix form invalid decorator for non ajax forms.
- Update of `log4j-slf4j-impl` to `log4j-slf4j2-impl` following the `slf4j` upgrade (> 2.x.x needs this)

57.2 Enhancements

- BasicApp: remove old sheet in `reference_data.xlsx` https://github.com/igloo-project/igloo-parent/commit/b5a94338fae024f383c97fb59cd4bcf39c2a6ca9

57.3 Dependencies

- jackson 2.13.3 -> 2.13.4
- log4j 1.8.2 -> 1.9.0
- flyway 9.0.4 -> 9.7.0
- hibernate 5.6.10.Final -> 5.6.12.Final
- hibernate-search-orm 5.11.10.Final -> 5.11.11.Final
- jsoup 1.15.2 -> 1.15.3
- mockito 4.6.1 -> 4.8.1
- passay 1.6.1 -> 1.6.2
- postgresql 42.4.0 -> 42.5.0
- spring 5.3.21 -> 5.3.23
- spring-security 5.7.2 -> 5.7.3
- immutables value 2.9.0 -> 2.9.2
- popperjs 2.11.5 -> 2.11.6

- micrometer 1.9.2 -> 1.9.5
- ehcache 3.10.0 -> 3.10.2
- error-prone-annotations 2.14.0 -> 2.16
- spring-boot 2.7.2 -> 2.7.5
- byte-buddy 1.12.12 -> 1.12.18
- jsass 5.10.4 -> 5.10.5
- junit-jupiter 5.9.0 -> 5.9.1
- junit-platform 1.9.0 -> 1.9.1
- poi 5.2.2 -> 5.2.3
- wicket 9.11.0 -> 9.12.0
- wicketstuff-select2 9.11.0 -> 9.12.0
- jersey 2.36 -> 2.37
- slf4j 1.7.36 -> 2.0.3
- picocli 4.6.3 -> 4.7.0
- commons-compress 1.21 -> 1.22
- commons-text 1.9 -> 1.10.0

Maven plugins:

- maven-javadoc-plugin 3.4.0 -> 3.4.1
- maven-project-info-reports-plugin 3.4.0 -> 3.4.1
- maven-site-plugin 3.12.0 -> 4.0.0-M3
- dependency-check-maven 7.1.1 -> 7.3.0
- flatten-maven-plugin 1.2.7 -> 1.3.0
- maven-jar-plugin 3.2.2 -> 3.3.0
- versions-maven-plugin 2.11.0 -> 2.13.0
- exec-maven-plugin 2.11.0 -> 3.1.0

CHAPTER

FIFTYEIGHT

4.4.0 (DO NOT USE THIS VERSION)

CHAPTER

FIFTYNINE

4.3.0 (2022-08-26)

59.1 Enhancements

- Import data: both default xlsx files optional.
- Utility allowing to transform an email list into a list of notification recipients
- Utility to retrieve media type from a workbook
- JPA test case : refactor reference data cleaning

59.2 Dependencies

- Font Awesome 6.0.0 -> 6.1.2
- Update Select2 Bootstrap 4 theme
- reload4j 1.2.20 -> 1.2.22
- jackson 2.13.2 -> 2.13.3
- h2 2.1.212 -> 2.1.214
- wicket-webjars -> 3.0.6
- assertj-core 3.22.0 -> 3.23.1
- assertj-guava 3.4.0 -> 3.5.0
- flyway-core 8.5.10 -> 9.0.4
- jersey 2.35 -> 2.36
- jboss-logging-annotations 2.2.0.Final -> 2.2.1.Final
- mockito 4.5.1 -> 4.6.1
- postgresql 42.3.5 -> 42.4.0
- spring 5.3.20 -> 5.3.21
- spring-security 5.6.2 -> 5.7.2
- webjars-locator-core 0.50 -> 0.52
- popperjs 2.10.2 -> 2.11.5
- micrometer 1.9.0 -> 1.9.2
- spring-boot 2.6.7 -> 2.7.2

- junit-jupiter-api 5.8.2 -> 5.9.0
- log4j 2.17.2 -> 2.18.0
- wicket 9.10.0 -> 9.11.0
- hibernate 5.6.9.Final -> 5.6.10.Final
- jsoup 1.15.1 -> 1.15.2
- wicketstuff-select2 9.10.0 -> 9.11.0
- select2-bootstrap-5-theme 1.2.0 -> 1.3.0
- byte-buddy 1.12.10 -> 1.12.12

Maven plugins:

- maven-failsafe-plugin 3.0.0-M6 -> 3.0.0-M7
- maven-surefire-plugin 3.0.0-M6 -> 3.0.0-M7
- maven-javadoc-plugin 3.3.2 -> 3.4.0
- maven-site-plugin 3.10.0 -> 3.12.0
- maven-toolchains-plugin 3.0.0 -> 3.1.0
- dependency-check-maven 7.1.0 -> 7.1.1
- maven-enforcer-plugin 3.0.0 -> 3.1.0
- maven-project-info-reports-plugin 3.3.0 -> 3.4.0
- maven-deploy-plugin 3.0.0-M2 -> 3.0.0
- maven-install-plugin 3.0.0-M1 -> 3.0.1
- maven-resources-plugin 3.2.0 -> 3.3.0
- maven-assembly-plugin 3.3.0 -> 3.4.2
- exec-maven-plugin 3.0.0 -> 3.1.0

CHAPTER

SIXTY

4.2.0 (2022-06-27)

60.1 Bugfixes

- Exclude Junit 4 from `igloo-component-wicket-bootstrap4` : Make sure that your tests still run
- issue 65: reload4j - Log4j1 Compatibility (when reload4j, log4j 1.x fork, is used)
- issue 66: StorageConsistencyCheck#checkType not set (storage check performance improvement)

60.2 Breaking changes

60.2.1 Hibernate : Changes to the DDL type for CLOB

Check that you have no entity field annotated `@Lob` or typed `java.sql.Clob`.

`StringClobType` is dropped as it uses Clob:

- Search and replace all StringClobType occurences: for `@Type(type = "org.iglooproject.jpa.hibernate.usertype.StringClobType")`, replace by `@Type(type = "text")`.
- With SQL Update script in create-mode, generate a SQL creation file before and after migration and check there is no unexpected change.
- Check that SQL Update script in update-mode, check that proposed updates are expected.
- Igloo entities (QueuedTaskHolder, ...) are fixed in Igloo.

https://github.com/hibernate/hibernate-orm/blob/5.6/migration-guide.adoc#changes-to-the-ddl-type-for-clob-in-postgresql81dialect-a

60.3 Dependencies

- wicket 9.7.0 -> 9.10.0
- wicketstuff select2 9.7.0 -> 9.10.0
- hibernate 5.4.33.Final -> 5.6.9.Final
- spring 5.3.16 -> 5.3.20
- spring security 2.6.4 -> 2.6.7
- jackson 2.13.1 -> 2.13.2
- reload4j 1.2.19 -> 1.2.20

- guava 31.0.1-jre -> 31.1-jre
- h2 2.1.210 -> 2.1.212
- log4j2 2.17.1 -> 2.17.2
- poi 5.2.0 -> 5.2.2
- flyway 8.5.1 -> 8.5.10
- jboss logging 3.4.1.Final -> 3.5.0.Final
- jsoup 1.14.3 -> 1.15.1
- mockito 4.3.1 -> 4.5.1
- postgresql 42.3.3 -> 42.3.5
- immutables value 2.8.2 -> 2.9.0
- webjars npm clipboard 2.0.8 -> 2.0.11
- micrometer 1.8.4 -> 1.9.0
- ehcache 3.9.6 -> 3.10.0
- errorprone 2.11.0 -> 2.14.0

Maven plugins:

- maven-antrun-plugin 3.0.0 -> 3.1.0
- maven-clean-plugin 3.1.0 -> 3.2.0
- maven-compiler-plugin 3.10.0 -> 3.10.1
- maven-failsafe-plugin 3.0.0-M5 -> 3.0.0-M6
- maven-surefire-plugin 3.0.0-M5 -> 3.0.0-M6
- maven-javadoc-plugin 3.3.2 -> 3.4.0
- maven-project-info-reports-plugin 3.2.1 -> 3.3.0
- maven-site-plugin 3.9.1 -> 4.0.0-M1
- maven-processor-plugin 5.0-rc3 -> 5.0-rc3
- jacoco-maven-plugin 0.8.7 -> 0.8.8
- dependency-check-maven 5.3.2 -> 7.1.0
- versions-maven-plugin 2.9.0 -> 2.11.0

CHAPTER

SIXTYONE

4.1.0 (2022-05-27)

61.1 Updates

- Wicket 9.7.0 -> 9.10.0
- Font Awesome 5.15.4 -> 6.0.0

61.2 Bugfixes

- Fix locale for localize with positional parameters in `RendererServiceImpl`.
- `processor.test.skip` is restored; when set to true on processing-enabled module, it allows to skip test source processing.

61.3 Enhancements

- BasicApp: remove home link in 503 error page.
- BasicApp: use external link and LDM for app links in emails.
- BasicApp: move breadcrumb html into page templates.
- BasicApp: announcement content multilines.
- BasicApp: enhancements on sign in checks and feedback messages.

61.4 Breaking changes

- Tests migrated to junit 5:
  - If you want to keep your project's tests Junit 4 - based:
    - Replace occurences of `org.iglooproject.test.jpa.junit.AbstractTestCase` by `org.iglooproject.test.jpa.junit.AbstractJunit4TestCase`.
    - If you use `AbstractWicketTestCase`, copy it from igloo-parent v4.0.0/igloo/igloo-components/igloo-component-wicket-test/src/main/java/org/iglooproject/test/wicket/core/AbstractWicketTestCase.java to create it in your project and changes its parent to `AbstractJunit4TestCase`.
    - You may exclude `org.junit.jupiter:junit-jupiter-api` from your dependencies to keep your test classpath clean.

- If you want to migrate to Junit 5
  - Check that `junit:junit` artifact is excluded from your project and fix errors.
  - Common changes are annotation rewrites : `@Before/@BeforeEach`, `@After/@AfterEach`, `@Rule` must be reworked to use `@ExtendWith`.
  - `basic-application` git history can be checked to find migration examples
- In both situations, ensure to check test number is not changed before and after run (check by-module final test result).

CHAPTER

SIXTYTWO

4.0.0 (2022-05-16)

4.0.0 brings module and repository refactoring. The purpose is to split Igloo codebase in smaller module and to reduce interdependencies.

Big picture changes are the following :

- Maven plugin and dependency managements are moved to igloo-maven
- Low level and/or common utilities are moved to igloo-commons
- *igloo-dependencies-** and *igloo-packages-** modules are removed; this dependency sets were hard to maintain
- *dependencies-** from igloo-maven replaces the later, but it is a *dependencyManagement* modules
- Utilities modules are added:
  - *igloo-hibernate*: Hibernate utilities
  - *igloo-jpa-model*: Parent class for entities, custom types to handle interface backed by enum
  - *jpa-test*: Junit 5 extension for JPA initialization
  - *igloo-monitoring-page*: Wicket utilities to expose adhoc or micrometer data as a web page and enable perfdata-like monitoring
  - *storage-{model,api,impl,micrometer,integration}*: Filesystem-backed storage system
  - *maven-processor-plugin* configuration is modified

62.1 Features

Storage engine is added to replace Filestore utilities. Storage is a more integrated solution, that provides a built-in relational and transactional model, wicket integration and jobs management (cleaning, consistency checks).

Filestore implementation is still available.

A migration guide to move Filestore to Storage engine will be provided.

62.2 Breaking changes

- Add `<masterBranchName>master</masterBranchName>` in your root `pom.xml` if your release branch is called `master` (new default is `main`).
- `maven-processor-plugin` configuration: see *Igloo 4 migration guide*
- *igloo-dependencies-** and *igloo-packages-** removal and other dependency management changes: dependencies now use `dependencyManagement` mechanism and rely on more explicit configuration. See and follow *Igloo 4.0.0 dependency migration guide*.
- Drop `wagon-maven-plugin` (or reconfigure it locally if you want to keep this plugin for delivery)
- Replace `LocaleUtils.initCollator(locale)` occurences by `new SerializableCollator(locale).nullsFirst()`
- Spring security : Java config replace `security-web-context.xml`. To migrate your project check `reference commit for BasicApp <https://gitlab.tools.kobalt.fr/igloo-project/igloo-parent/-/commit/b6ace4c0e6506e37d500d98aa027bf456fea18ce>_` and `commit <https://github.com/igloo-project/igloo-parent/commit/d6a564fd7eb59005ec8b3349bf627704dab88bc6>_`. `security-web-context.xml` is still valid, but it is strongly adviced to plan this change to ease future updates.
- SqlUpdateScript updates:
  - add `info.picocli:picocli` dependency with `<optional>true</optional>`.
  - copy and rename igloo-parent basic-application/basic-application-core/src/main/java/org/iglooproject/basicapp/core/cli/Basic and igloo-parent basic-application/basic-application-core/src/main/java/org/iglooproject/basicapp/core/cli/AbstractBasicApp into your project. Replace `BasicApplicationCoreHeadlessConfig` by your own spring configuration (check existing file).
  - check that « Run as » can be launched.
  - check *Model - Database comparisons* for usage.

The above instructions may be sufficient to migrate a project. If you encounter missing classes or definition, here is a summay of other breaking changes to help to identify new modules or codes to import:

- Use `${igloo-commons.version}` property to define version for the following *artifactId*:
  - bindgen-functional
  - igloo-batch-api
  - igloo-bean-api
  - igloo-collections
  - igloo-component-commons-io
  - igloo-component-functional
  - igloo-component-truevfs
  - igloo-context
  - igloo-lang
  - igloo-security-api
  - igloo-validator
- Moved resources:

  - `PredefinedIdSequenceGenerator` (igloo-component-jpa): package change, check and replace old package name `org.iglooproject.jpa.hibernate.dialect.PredefinedIdSequenceGenerator` by new `org.iglooproject.jpa.hibernate.dialect.PredefinedIdSequenceGenerator`
  - test resources `EntityManagerFactoryHelper` and `PersistenceUnitDescriptorAdapter` are now exposed by `igloo-hibernate` module
  - extension `EntityManagerFactoryExtension`
  - `PostgreSQLIntervalFunction`, `PostgreSQLRegexpOperatorFunction`, `MetadataRegistryIntegrator` are moved (package/module) from igloo-component-jpa to `igloo-hibernate`
  - `org.iglooproject.jpa.hibernate.model.naming` package is kept but moved to `igloo-hibernate`

- *igloo-component-commons* is removed and split to igloo-batch-api, igloo-bean-api, igloo-collections, igloo-context, igloo-lang, igloo-security-api and igloo-validator (igloo-commons repository).
- Dependencies' versions for `hibernate-validator`, `aspectj` and `byte-buddy` are removed as we do not use it directly and there is no transitive dependency conflicts for this dependencies. If this dependencies are listed in you project, you can remove them; they are either useless or already added by transitivity.

CHAPTER

SIXTYTHREE

OLDER RELEASES

63.1 3.5.2 (2022-05-02)

63.1.1 Bugfixes

- Fix model generation in `CacheDataProvider`.

63.2 3.5.1 (2022-04-11)

63.2.1 Bugfixes

- *hibernate.xml_mapping_enabled* wrongly set (https://github.com/igloo-project/igloo-parent/issues/60)

63.3 3.4.1 (2022-04-11)

63.3.1 Bugfixes

- *hibernate.xml_mapping_enabled* wrongly set (https://github.com/igloo-project/igloo-parent/issues/60)

63.4 3.5.0 (2022-03-09)

63.4.1 Updates

- wicket 8.13.0 -> 9.7.0

63.5 3.1.1 (2022-03-08)

63.5.1 Bugfixes

- Fix `TFileRegistry` static create methods.

63.6 3.4.0 (2022-02-24)

63.6.1 Breaking changes

Hibernate type discovery

Since hibernate 5.4.30, it's not possible to use a `@MappedSuperclass` unused entity (if no `@Entity` inherits from it, then it's ignored). It triggers failure on `TypeDefinitions.java` that is used to declare `@TypeDef` custom types definitions.

To migrate your project, it's necessary to take all the types used with `@TypeDef` annotation (usually located in `TypeDefinitions.java`) and you need to transfer them to `<Project>CoreCommonJpaConfig.java`. Check commit and commit for new-style type declaration: declare a `TypeContributor` bean with your custom types so that Igloo can detect them and configure Hibernate accordingly.

`TypeDefinitions.java` file can be deleted.

Hibernate XML mapping (configuration and dependencies)

Default Hibernate configuration is modified to set `hibernate.xml_mapping_enabled=false` and dom4j and jaxb are excluded from hibernate dependencies. If you use XML entity mappings, you must override property and restore dependencies.

63.6.2 Updates

- hibernate 5.4.29 -> 5.4.33
- postgresql driver: 42.3.1 -> 42.3.3
- spring: 5.3.15 -> 5.3.16
- spring-boot: 2.6.2 -> 2.6.4
- spring-security: 5.6.1 -> 5.6.2
- hibernate-search: 5.11.8 -> 5.11.10
- commons-lang3: 3.11 -> 3.12
- slf4j: 1.7.33 -> 1.7.36
- reload4j: 1.2.18.0 -> 1.2.19
- poi: 5.1.0 -> 5.2.0
- ph-css: 6.4.3 -> 6.5.0
- wicket: 8.13.0 -> 8.14.0
- webjars-locator: 0.49 -> 0.50

- flywaydb 8: 8.4.1 -> 8.5.1
- junit 5: 5.8.0 -> 5.8.2
- mockito: 4.1.0 -> 4.3.1
- errorprone: 2.10.0 -> 2.11.0
- rhino: 1.7.11 -> 1.7.14

Maven plugins:

- maven-jar-plugin: 3.2.0 -> 3.2.2
- maven-javadoc-plugin: 3.3.1 -> 3.3.2
- buildnumber-maven-plugin: 1.4 -> 3.0.0

63.6.3 Fixes

- Restore TFileRegistry static methods (removed with 3.0.0) for TrueVFS registry management.

63.7 3.3.0 (2022-01-31)

63.7.1 Bugfix / Workaround

web.xml configuration

`basic-application` is modified to add `request-character-encoding` and `response-character-encoding` configurations in your *web.xml*, initiated to UTF-8.

You may need to update your own project accordingly. See basic-application example: https://github.com/igloo-project/igloo-parent/commit/118281bf9f2b844c7224c07037b489ddd53986a0

@SpringBean workaround

This release addresses an issue with wicket and `@SpringBean` annotation. When we want to inject a collection of beans looked up by type, if there is **one only spring bean** that implements the collection type, it is used, whereas the expected behavior is to build a collection of all the beans matching the expected generic type.

If there is no collection bean, or more than one collection bean, bean lookup is correctly done.

The workaround is to create two beans implementing `Collection<Object>` so that `@SpringBean` trigger the correct behavior. This workaround is triggered by `IglooApplicationConfigAutoConfiguration`.

No action is needed to migrate a project.

@PropertySource, CompositePropertySourceFactory and encoding

CompositePropertySourceFactory does not honor `PropertySource#encoding`. This release fixes this issue, and all Igloo managed `@PropertySource` are configured to use `UTF-8` encoding.

No action is needed to migrate a project. If you use your own `@PropertySource` annotations, you may update encoding attribute.

maven-processor-plugin

`skipSourcesUnchanged` removed from configuraiton. If true, it is needed to fully remove `target/generated-sources/apt` to trigger a binding generation. As Eclipse does not remove whole target folder on clean, it implied manual actions to refresh generated bindings.

63.7.2 Updates

- Spring-security 5.4.10 -> 5.6.1

XML context security files are removed from Igloo and replaced by javaconfig. As XML xsd references spring-security version, it eases spring-security version switch as it no longer complains about xsd version mismatch.

Igloo XML context security file is replaced by an equivalent javaconfig configuration.

If your project contains any `*security-context.xml` file, **you need to update spring-security version from 5.4 to 5.6 in XSD declarations**.

- TaskManagement uses a spring-like Configurer pattern for queueids discovery. It is done to get rid of Collection beans. BasicApplication is modified to use this new pattern, but existing Collection<IQueueId> bean continue to get honored (with a warning at startup), so no change is needed on projects.

  See BasicApplicationCoreTaskManagementConfig for basic-application example **to migrate your queueid definition accordingly.**

- h2 CVE-2022-23221: 2.0.206 -> 2.1.210 (h2 is only used for testing purposes).

  No action is needed to migrate a project.

63.8 3.2.1 (2020-01-26)

63.8.1 Bugfixes

- Fix a bug with Flyway compatibility layer introduced in 3.2.0; without this fix, Flyway 7.x cannot be used.

63.9 3.2.0 (2022-01-17)

63.9.1 Breaking changes

- Flyway is updated (7.x -> 8.x) and Flyway 8.x is not compatible with PostgreSQL 9.6. If you use PostgreSQL 9.6, a compatibility layer is provided to continue use Flyway 7.x:
  - In your core module, add a dependency exclusion on `org.iglooproject.components:igloo-component-flyway-8` and manually add `org.iglooproject.components:igloo-component-flyway-7`

  - Check in your dependencies that flyway-core artifact has version `7.15.0`
  - If this is not the case, add the needed exclusion to ensure that flywaydb version is handled by `igloo-component-flyway-7` artifact
  - Change import *org.iglooproject.jpa.migration.IglooMigration* to *org.iglooproject.flyway.IglooMigration*
- Deprecated log4j 1.2.x dependency is replaced by `reload4j` (https://reload4j.qos.ch/). It is a drop-in replacement (no action needed to use this new dependency, except if you manually import log4j 1.2). You may migrate to log4j 2 as log4j 1.2 is deprecated and reload4j is just an emergency solution.
- JSASS triggers a warning. You may check that styles are correct. If you encounter any issues you can downgrade jsass by setting property `<igloo.jsass.version>5.10.3</igloo.jsass.version>` (see https://gitlab.com/jsass/jsass/-/issues/95).

```
[ERROR] DEPRECATION WARNING on line 1 of classpath:/org/iglooproject/basicapp/web/
↪application/common/template/resources/styles/application/advanced/styles.scss/
↪JSASS_CUSTOM.scss:
[ERROR] !global assignments won't be able to declare new variables in future␣
↪versions.
[ERROR] Consider adding `$jsass-void: null` at the top level.
```

- A lot of dependency updates. Please check the Dependencies entry below.

63.9.2 Dependencies

- Spring 5.2.9 -> 5.3.15
- Spring security 5.4.1 -> 5.4.10
- Spring boot 2.3.4 -> 2.6.2
- Hibernate 5.4.21 -> 5.4.29
- Hibernate Search 5.11.5 -> 5.11.8
- Wicket 8.12.0 -> 8.13.0
- Wicket Wiquery (include JQuery UI 1.12.1 -> 1.13.0) 8.1.1 -> 8.2.0
- FlywayDB 7.15.0 -> 8.4.1 (Flyway 7.x can still be used, see breaking changes)
- HikariCP 3.4.5 -> 5.0.1
- Spring LDAP 2.3.3 -> 2.3.5
- Guava 29.0-jre -> 31.0.1-jre
- PostgreSQL JDBC Driver 42.2.14 -> 42.3.1
- POI 4.1.2 -> 5.1.0
- Jackson 2 2.11.3 -> 2.13.1
- Javax/Jakarta mail 1.6.6 -> 1.6.7
- SLF4J 1.7.30 -> 1.7.33
- Log4j2 2.17.0 -> 2.17.1
- Apache HTTPComponents Core 4.4.13 -> 4.4.15
- AspectJ 1.9.6 -> 1.9.7
- Byte-buddy 1.10.10 -> 1.12.6

- JBoss Logging 3.4.1 -> 3.4.3
- JBoss Logging Annotations -> 2.2.0
- JDK Serializable functions 1.8.6 -> 1.9.0
- Freemarker 2.3.30 -> 2.3.31
- BouncyCastle jdk15on 1.68 -> 1.70
- Flying Saucer 9.1.20 -> 9.1.22
- JSoup 1.14.2 -> 1.14.3
- Pretty-time 4.0.6 -> 5.0.2
- PH-CCS 6.2.3 -> 6.4.2
- Validation API 1.1.0 -> 2.0.1
- Webjars locator 0.46 -> 0.48
- JSASS 5.10.3 -> 5.10.4
- Passay 1.6.0 -> 1.6.1
- Junit 5 5.7.0 -> 5.8.0
- H2 1.4.200 -> 2.0.206
- Mockito 3.5.15 -> 4.1.0
- AssertJ 3.17.2 -> 3.22.0

This release breaks Flyway compatibility layer : Flyway 7.x cannot be used. Please use 3.2.1.

63.10 3.1.0 (2021-12-23)

63.10.1 Breaking changes

- Hibernate Search initialization now authorizes hibernate-search, lucene or elasticsearch dependencies to be removed if not used.

  Add lucene integration to your project:

```
<dependency>
        <groupId>org.iglooproject.components</groupId>
        <artifactId>igloo-component-hibernate-configurator-lucene</artifactId>
        <version>${igloo.version}</version>
</dependency>
```

- `wicket-webjars` 2.0.20 update. Resource paths not beginning by `webjars/` are broken. If you use custom webjars resource reference (check `WebjarsJQueryPluginResourceReference`, `WebjarsJavaScriptResourceReference`, `WebjarsCoreJQueryPluginResourceReference` derived classes), ensure that your resource path begins by `webjars`.

  Example:

```
 private BootstrapAlertJavaScriptResourceReference() {
-  super("bootstrap/current/js/dist/alert.js");
```

(continues on next page)

(continued from previous page)

```
+  super("webjars/bootstrap/current/js/dist/alert.js");
 }
```

63.10.2 Dependencies

- We no longer override cglib-nodep dependency. It is managed exclusively by wicket-ioc.
- Compilation-time code quality annotations dependencies are moved to provided scope, so that it does not clutter war artifact.
- `com.sun.mail:javax.mail` is replaced by `com.sun.mail:jakarta.mail`. Check in that your dependencies are updated and does not contain old dependency.

63.10.3 Removed

- LessCss / Less4j is removed
- maven-enforcer-plugin `DependencyConvergence` rule is replaced by `requireUpperBoundDeps`: transitive dependencies versions no longer need to be consistent, but they needs to match the last version of candidate dependencies
- `glyphicons-halflings-white.png` and `glyphicons-halflings` are removed (used by bootstrap 3, also removed previously)

63.10.4 Bugfixes

- City : update xlsx init file - postalcode with 5 characters
- Feedbacks : update style (fatal + debug)

63.11 3.0.3 (2021-12-22)

Fix for CVE-2021-45105 Log4Shell. The only modification from 3.0.2 is the log4j dependency update (2.17.0).

63.12 2.7.6 (2021-12-22)

Fix for CVE-2021-45105 Log4Shell. The only modification from 2.7.5 is the log4j dependency update (2.17.0).

63.13 3.0.2 (2021-12-16)

Fix for CVE-2021-45046 Log4Shell. The only modification from 3.0.1 is the log4j dependency update (2.16.0).

63.14 2.7.5 (2021-12-16)

Fix for CVE-2021-45046 Log4Shell. The only modification from 2.7.4 is the log4j dependency update (2.16.0).

63.15 2.7.4 (2021-12-14)

Fix for CVE-2021-44228 Log4Shell. The only modification from 2.7.2 is the log4j dependency update (2.15.0).

Fix broken jgtiflow configuration introduced in 2.7.3.

63.16 3.0.1 (2021-12-13)

Fix for CVE-2021-44228 Log4Shell. The only modification from 3.0.0 is the log4j dependency update (2.15.0).

63.17 2.7.3 (2021-12-13)

Fix for CVE-2021-44228 Log4Shell. The only modification from 2.7.2 is the log4j dependency update (2.15.0).

This release breaks jgtiflow configuration. Please use 2.7.4.

63.18 3.0.0 (2021-11-25)

63.18.1 Updates

- Jackson `QueuedTaskHolder` serializer configuration is modified to remove deprecated APIs; task output in console modified to use Jackson nodetree (it allows to get rid of a real deserialization).
- Removed POI deprecated API calls.
- Fix `javax.annotations-api` dependency issue.
- QueryDSL 4.4.0 -> 5.0.0 (check your JPA / SQL query generation)

63.18.2 Breaking changes

- Removed Java < 11 support.
- Removed servlet < 4.0 support.
- Removed `externallinkchecker`: if you want to use it, fork the module and put it into you project.
- Removed `org.iglooproject.jpa.more.business.execution`: if you use it, fork the module and put it into your project.
- Removed `org.javatuples:javatuples` from `igloo-component-commons`; if you use it, add this dependency to your project.
- TrueZip replaced by TrueVFS.
- Jersey update: Jersey version implies Java EE 8 API, so it implies tomcat >= 9.x
-

63.18.3 Java 11 and Servlet 4.0

Igloo now uses Java 11 and Servlet 4.0. See here what you need to do to perform Java 11 and Servlet 4.0 migration. *Migrating to Java 11 and Servlet 4.0*

63.19 2.7.2 (2021-11-19)

63.19.1 Bugfixes

- Select2: temporary fix focus search input.

63.20 2.7.1 (2021-10-07)

63.20.1 Bugfixes

- Select2: temporary fix focus search input.

63.21 2.7.0 (2021-09-07)

63.21.1 Updates

- Font Awesome 5.15.1 -> 5.15.2
- commons-compress 1.20 -> 1.21
- jsoup 1.13.1 -> 1.14.2

63.21.2 Enhancements

- BasicApp: use `ExternalLink` in notification emails.
- BasicApp: add reference data read-only list feature.
- BasicApp: refactor condition on enable/disable actions.
- WicketTester : new test case for first sign in workflow
- Revert `@Basic` for user and user group generic classes, keep with `@Colmun` for override.

63.22 2.6.0 (2021-07-30)

63.22.1 Bugfixes

- BasicApp: fix `enabled` / `active` fields for `User` and `Announcement`.
- BasicApp: fix administration breadcrumb.
- `AbstractMapCollectionModel`: fix `.size()` method.

63.22.2 Updates

- wicket 8.10.0 -> 8.12.0
- wicketstuff-select2 8.10.0 -> 8.12.0

63.22.3 Enhancements

- Fix container max width mixin deprecated message.
- BasiApp: erros pages use same layout than application access pages.
- BasicApp: use `color-yiq` for color consistency (component active + navbar main).
- BasicApp: add rel noopener on target blank links.
- BasicApp: add meta description.
- `EnumDropDownMultipleChoice` : `Collection` instead of `List` for choices model.
- Fix HTML `<title>` to be on one line.

63.23 2.5.0 (2021-04-12)

63.23.1 Breaking changes

- User - rename field active to enabled for consistency.
- BasicApp: Announcement - rename field active to enabled for consistency.
- BasicApp: ReferenceData - update enabled properties for consistency.
- User - remove useless attributs (phone numbers)
- User - rename 2 comparators for consistency.
- User - rename groups join table.
- User - rename package person to user for consistency.
- User - fix create entity service method.
- User - clean flush in save
- BasicApp: clean useless spring component name value.
- User - update service and dao name value for consistency.

63.23.2 Enhancements

- BasicApp: consistency for `enabled` fields in `User` and `Announcement`.
- SqlUpdateScript: (used in <Project>SqlUpdateScriptMain) target file is now overwritten (previously, SQL script was append to the target file).

63.24 2.4.0 (2021-04-02)

63.24.1 Breaking changes

- **Bootstrap 4.5.3 -> 4.6.0**

63.24.2 Updates

- **Bootstrap 4.5.3 -> 4.6.0**
- Font Awesome 5.15.1 -> 5.15.2
- Clipboard.js 2.0.6 -> 2.0.8

63.24.3 Bugfixes

- Security: remove user active / enabled check, done by Spring Security during authentication.
- Fix Select2 css box shadow focus error state.
- BasicApp: fix scss import BS utilities.
- BasicApp: fix margin bottom application access.

63.24.4 Enhancements

- BasicApp: small changes on sidebar scss.
- BasicApp: use kobalt email address for admin user.
- BasicApp: use shared email address in filter mode.
- BasicApp: refactor sign in content page.

63.25 2.3.1 (2021-02-05)

63.25.1 Breaking changes

- **Bootstrap 4.5.0 -> 4.5.3**
- Refactor password security options.

63.25.2 Updates

- **Bootstrap 4.5.0 -> 4.5.3**
- Font Awesome 5.14.0 -> 5.15.1

63.25.3 Bugfixes

- BasicApp: fix `User` permission evaluator.
- BasicApp: fix ehcache xsd url.

63.25.4 Enhancements

- Add `hasPasswordHash()` in `GenericUser`.
- BasicApp: fix sign out on password security pages.
- BasicApp: add first sign in workflow.
- BasicApp: clean up user password recovery notification panel.
- BasicApp: use bypassPermissions for links in notification panels.
- BasicApp: add fallback url in mail notifications.
- BasicApp: add properties console page.
- BasicApp: rename resources properties packages and classes for consistency.
- BasicApp: wording for `HistoryLog` mandatory differences search query.
- BasicApp: Update resource key update password.
- BasicApp: properties for password length min max.
- BasicApp: refactor condition on enable/disable actions.
- Permission Evaluator: object no longer needs to be a `GenericEntity`.

63.26 2.3.0

Not released.

63.27 2.2.1 (2020-12-01)

63.27.1 Breaking changes

- BasicApp: rework batch report console page

63.28 2.2.0 (2020-11-19)

63.28.1 Breaking changes

- New default logging backend : log4j2
  - *Keep log4j 1.2*
  - *Migrate to log4j2*

- Spring boot update related change : if you have `new ApplicationContextRunner()` declared in your application or tests and you want to override existing beans, you now need to add `.withAllowBeanDefinitionOverriding(true)`
- Flyway update related change :
  - you must override `getEquivalentChecksum` either in `AbstractDataUpgradeMigration.java` or in each of your migrations. This function is used if you want to state that two of your migrations are doing the same thing and if one pass the other must not be executed. The default implementation is to call `getChecksum()`.
  - You can now have the possibility to override in each of your migrations the following functions :
    - `isUndo` if you want to flag your migration as undoing another (default is false)
    - `canExecuteInTransaction` if you want your migration not to be executed in a transaction (default is false)
  - You also need to modify the flyway locations in your properties file as dot-separated path are no longer supported by flyway, you need to refactor them in slash-separated path.
- Spring Security update related change : references to http://www.springframework.org/schema/security/spring-security-5.3.xsd url must be rewritten to https://www.springframework.org/schema/security/spring-security-5.4.xsd.

63.28.2 Updates

- **spring-framework 5.2.6.RELEASE -> 5.2.9.RELEASE**
- **spring-security 5.3.2.RELEASE -> 5.4.1**
- **spring-boot 2.2.7.RELEASE -> 2.3.4.RELEASE**
- **spring-ldap 2.3.2.RELEASE -> 2.3.3.RELEASE**
- **hibernate 5.4.16.Final -> 5.4.21.Final**
  - Hibernate 5.4.22 skipped, waiting for https://hibernate.atlassian.net/browse/HHH-14279 fix
- **hibernate-search 5.11.4.Final -> 5.11.5.Final**
- **wicket 8.8.0 -> 8.10.0**
- **wicketstuff-select2 8.8.0 -> 8.10.0**
- flyway 5.2.7 -> 7.0.2
- jackson 2.10.4 -> 2.11.3
- jackson-databind 2.10.4 -> 2.11.3
- commons-codec 1.14 -> 1.15
- commons-validator 1.6 -> 1.7
- commons-lang3 3.10 -> 3.11
- commons-text 1.8 -> 1.9
- aspectj 1.9.5 -> 1.9.6
- bouncycastle-jdk15on 1.65 -> 1.66
- postgresql 42.2.12 -> 42.2.14
- webjars-locator-core 0.45 -> 0.46

- flying-saucer 9.1.19 -> 9.1.20
- querydsl 4.3.1 -> 4.4.0
- httpclient 4.5.12 -> 4.5.13
- prettytime 4.0.5 -> 4.0.6
- allure-junit4 2.13.3 -> 2.13.6
- assertj 3.16.1 -> 3.17.2
- mockito 3.3.3 -> 3.5.13
- maven-failsafe-plugin 3.0.0-M4 -> 3.0.0-M5
- maven-project-info-reports-plugin 3.0.0 -> 3.1.1
- maven-resources-plugin 3.1.0 -> 3.2.0
- maven-surefire-plugin 3.0.0-M4 -> 3.0.0-M5
- maven-war-plugin 3.2.3 -> 3.3.1
- exec-maven-plugin 1.6.0 -> 3.0.0
- wagon-ssh-external 3.3.4 -> 3.4.1
- jacoco-maven-plugin 0.8.5 -> 0.8.6
- maven-javadoc-plugin 3.1.1 -> 3.2.0
- junit 4.13 -> 4.13.1
- log4j2 support 2.13.3

63.28.3 Bugfixes

- Wicket: allow *.webmanifest in SecurePackageResourceGuard
- Fix missing scope:test on igloo-component-web-jpa-test in igloo-component-rest-jersey2
- BS4 Popover: fix close button.
- Fix add-in elements css placements in `DataTableBuilder`.

63.28.4 Enhancements

- BasicApp: drop `init` module.
- BasicApp: move `BasicApplicationSqlUpdateScriptMain` to `cli` package in `core` module.
- BasicApp: rename `INotificationUserProfileUrlBuilderService` to `IBasicApplicationNotificationUrlBuilderService`.
- BasicApp: update favicons and conf.
- BasicApp: rework logo header sections in application access pages, error pages, and the home page.
- BasicApp: fix decorated table add-in elements margin.
- BasicApp: update user groups list page.
- BasicApp: fix permissions on users and usergroups.
- BasicApp: add `ReferenceDataAjaxDropDownSingleChoice` and `ReferenceDataAjaxDropDownMultipleChoice`.

- BS4 tabs: update url anchor and show tab from anchor on load.
- jQuery multivalued expand: fix toggle button html.
  - Explicit close for icons.
  - Use instead of <a> to wrap icons.

63.29 1.7.2 (2020-09-16)

63.29.1 Bugfixes

- Fix export Excel cell formula type.

63.30 2.1.1 (2020-09-15)

63.30.1 Bugfixes

- Fix export Excel cell formula type.

63.31 2.1.0 (2020-09-09)

63.31.1 Enhancements

- BasicApp: fix style notification password recovery.
- Animal-sniffer maven plugin is not disabled for JDK >1.8, as it is now managed since JDK 9.
- Add new MediaType `APPLICATION_MS_EXCEL_MACRO` to handle macros

63.32 2.0.0 (2020-07-29)

63.32.1 Breaking changes

- **Bootstrap 4.5.0.**
- **Disable Autoprefixer in development mode.**
- Rework `toString()` on `GenericEntity`. Drop `getNameForToString()` and `getDisplayName`.
- Drop Bootstrap 3 module.
- Remove JQuery Autosize plugin.
- `IComponentFactory` and parameterized ones are now functional interfaces. Drop `AbstractComponentFactory`, `AbstractParameterizedComponentFactory` and `AbstractDecoratingComponentFactory`.
- Fix null values display in Excel exports. For instance, a number cell will be blank instead of displaying zero.
- Remove `$sizes` scss variable override.

63.32.2 Updates

- **Bootstrap 4.3.1 -> 4.5.0**
- Font Awesome 5.11.2 -> 5.14.0
- Popper.js 1.16.0 -> 1.16.1-lts
- Clipboard.js 2.0.4 -> 2.0.6

63.32.3 Bugfixes

- BasicApp: fix close tag on static error pages.
- BasicApp: fix reference data sort type label.

63.32.4 Enhancements

- BasicApp: remove BS override shadow focus.
- BasicApp: fix markup custom check.
- BasicApp: forms - use `col-md-*` instead of `col-sm-*`.
- BasicApp: environment section in sidebar for advanced layout.
- BasicApp: hover on table disabled row + upstream scss to Igloo.
- BasicApp: fix sidebar sub menu collapse animation.
- BasicApp: clean up + update css on email notifications.
- BasicApp: add `.divider-light` css class.
- BasicApp: advanced layout as default.
- Fix some Sonar issues.
- Clean up some deprecated.
- `LinkDescriptor`: `bypassPermissions` method no longer deprecated.
- Add a debug stopwatch on Autoprefixer process.

63.33 1.7.1 (2020-06-17)

63.33.1 Enhancements

- Add `ConditionalOnMissingBean` annotation on default `AuthenticationProvider` to allow use of exclusively custom `AuthenticationProvider`.

63.34 1.7.0 (2020-06-16)

63.34.1 Bugfixes

- Fix `ReferenceData` comparator.

63.34.2 Enhancements

- Select2: force size 1 row.
- Select2 - BS4: override selected element background color.
- Make class `AbstractImmutableMaterializedPrimitiveValueUserType` public.
- `AbstractUnicityFormValidator`: all `FormComponent` are flagged on error.
- Hibernate identifier generator strategy can now be customized through property `hibernate.identifier_generator_strategy_provider`, with a fallback on the previous default `PerTableSequenceStrategyProvider`.
- `PredefinedIdSequenceGenerator` is a new sequence generator allowing to set entity ids manually base on a transient field `predefinedId`.

63.34.3 Updates

- **spring-framework 5.2.2.RELEASE -> 5.2.6.RELEASE**
- **spring-security 5.2.1.RELEASE -> 5.3.2.RELEASE**
- **spring-boot 2.1.3.RELEASE -> 2.2.7.RELEASE**
- **hibernate 5.4.10.Final -> 5.4.16.Final**
- *byte-buddy 1.10.2 -> 1.10.10*
- **wicket 8.6.1 -> 8.8.0**
- *wicketstuff-select2 8.6.0 -> 8.8.0*
- jackson 2.9.10 -> 2.10.4
- jackson-databind 2.9.10.1 -> 2.10.4
- guava 28.1-jre -> 29.0-jre
- ph-css 6.2.0 -> 6.2.3
- querydsl 4.2.2 -> 4.3.1
- HikariCP 3.4.1 -> 3.4.5
- commons-codec 1.13 -> 1.14
- commons-compress 1.19 -> 1.20
- commons-configuration2 2.6 -> 2.7
- commons-lang3 3.9 -> 3.10
- httpclient 4.5.10 -> 4.5.12
- httpcore 4.4.12 -> 4.4.13

- apache-poi 4.1.1 -> 4.1.2
- bouncycastle bcprov-jdk15on 1.64 -> 1.65
- freemarker 2.3.29 -> 2.3.30
- javassist 3.26.0-GA -> 3.27.0-GA
- jsoup 1.12.1 -> 1.13.1
- prettytime 4.0.2.Final -> 4.0.5.Final
- passay 1.5.0 -> 1.6.0
- postgresql 42.2.9 -> 42.2.12
- slf4j 1.7.29 -> 1.7.30
- webjars-locator-core 0.43 -> 0.45
- flying-saucer 9.1.19 -> 9.1.20
- jdk-serializable-functional 1.8.5 -> 1.8.6
- maven-antrun 1.5.0 -> 3.0.0
- maven-assembly-plugin 3.2.0 -> 3.3.0
- maven-dependency-plugin 3.1.1 -> 3.1.2
- maven-jar-plugin 3.1.1 -> 3.2.0
- maven-javadoc-plugin 3.1.1 -> 3.2.0
- maven-source-plugin 3.2.0 -> 3.2.1
- mockito 3.2.0 -> 3.3.3
- allure-junit4 2.13.0 -> 2.13.3
- junit 4.12 -> 4.13
- assertj 3.14.0 -> 3.16.1
- assertj-guava 3.3.0 -> 3.4.0

63.35 1.6.1 (2020-04-24)

63.35.1 Enhancements

- BasicApp: user - fix user group add form layout.
- BasicApp: user group - fix authorities list layout.
- BasicApp: users - remove useless `withNoRecordsResourceKey`.
- BasicApp: move bs breakpoint div to the bottom.
- Boostrap Override: remove `.card-${color}-full`.

63.36 1.6.0 (2020-03-13)

63.36.1 Enhancements

- BasicApp: major markup and scss changes.
- Fix jQuery UI datepicker positioning and input height value.

63.37 1.5.2 (2020-03-12)

63.37.1 Bugfixes

- Fix spring-security namespace; without this fix, network-less application start is not possible because spring-security namespace cannot be mapped with jar's provided .xsd.

  In your application, you need to replace in XML files http(s)://www.springframework.org/schema/security/spring-security*.xsd URL by https://www.springframework.org/schema/security/spring-security-5.2.xsd.

  This URLs are mapped by Spring to jar's provided files.

63.38 1.5.1 (2020-01-10)

63.38.1 Bugfixes

- Fix manifest resource finding error.

63.39 1.5.0 (2020-01-06)

Breaking changes and enhancements are introduced to allow usage of autoconfiguration and to prepare a future reorganization and splitting of Igloo modules, to ease future development and maintenance tasks.

63.39.1 Breaking changes

- Configuration system is modified to replace custom `@ConfigurationLocations` system by spring vanilla `@PropertySource`. See *Configuration migration guide* to find how to modify your application and check that configuration is correctly managed.
- Spring Security related change : references to http://www.springframework.org/schema/security/spring-security-4.2.xsd url must be rewritten to http://www.springframework.org/schema/security/spring-security.xsd (same file, but does not trigger a failed check on version done by Spring Security at startup time).

63.39.2 Updates

- **wicket 8.2.0 -> 8.6.0**
- **hibernate 5.4.2.Final -> 5.4.10.Final**
- **hibernate-search 5.11.1 -> 5.11.4**
- **spring-framework 5.1.6.RELEASE -> 5.2.2.RELEASE**
- **spring-security 5.1.4.RELEASE -> 5.2.1.RELEASE**
- cglib 3.2.10 -> 3.3
- jackson 2.9.8 -> 2.9.10
- gson 2.8.5 -> 2.8.6
- guava 27.1-jre -> 28.1-jre
- ph-css 6.1.2 -> 6.2.0
- HikariCP 3.3.1 -> 3.4.1
- wicket webjars 2.0.10 -> 2.0.16
- jsass 5.8.0 -> 5.10.3
- allure-junit4 2.10.0 -> 2.13.0
- ehcache-core 2.10.6.5.1 -> 2.10.7.0.62
- commons-codec 1.12 -> 1.13
- commons-beanutils 1.9.3 -> 1.9.4
- commons-collections4 4.3 -> 4.4
- commons-compress 1.18 -> 1.19
- commons-configuration 2.4 -> 2.6
- commons-lang3 3.8.1 -> 3.9
- commons-text 1.6 -> 1.8
- httpclient 4.5.8 -> 4.5.10
- httpcore 4.4.11 -> 4.4.12
- wicketstuff-select2 8.2.0 -> 8.6.0
- aspectj 1.9.2 -> 1.9.5
- assertj 3.12.2 -> 3.14.0
- assertj-guava 3.2.1 -> 3.3.0
- bouncycastle bcprov-jdk15on 1.61 -> 1.64
- jdk-serializable-functional 1.8.5 -> 1.9.0
- freemarker 2.3.28 -> 2.3.29
- javassist 3.24.1-GA -> 3.26.0-GA
- jboss-logging 3.3.2.Final -> 3.4.1.Final
- jsoup 1.11.3 -> 1.12.1
- mockito 2.25.1 -> 3.2.0

- passay 1.4.0 -> 1.5.0
- postgresql 42.2.5 -> 42.2.9
- slf4j 1.7.26 -> 1.7.29
- apache-poi 4.1.0 -> 4.1.1
- byte-buddy 1.9.10 -> 1.10.2
- h2database 1.4.199 -> 1.4.200
- querydsl 4.2.1 -> 4.2.2
- webjars-locator-core 0.37 -> 0.43
- maven-compiler-plugin 3.8.0 -> 3.8.1
- maven-javadoc-plugin 3.1.0 -> 3.1.1
- maven-source-plugin 3.0.1 -> 3.2.0
- maven-toolchains-plugin 1.1 -> 3.0.0
- maven-war-plugin 3.2.2 -> 3.2.3
- jacoco-maven-plugin 0.8.3 -> 0.8.5
- dependency-check-maven 5.2.1 -> 5.2.4
- animal-sniffer-maven-plugin 1.17 -> 1.18
- maven-antrun 1.4.0 -> 1.5.0
- maven-assembly-plugin 3.1.1 -> 3.2.0
- maven-failsafe-plugin 3.0.0-M3 -> 3.0.0-M4
- maven-surefire-plugin 3.0.0-M3 -> 3.0.0-M4
- wagon-ssh-external 3.3.3 -> 3.3.4
- maven-enforcer-plugin 3.0.0-M2 -> 3.0.0-M3

63.39.3 Enhancements

- basic-application now uses autoconfiguration
- `GenericEntity` can be used without hibernate dependency (this allow to use existing entity objects in third-party micro-services if needed)
- `WicketRendererServiceImpl`: add `renderPage(...)` method (similar to `renderComponent(...)` method)
- bindgen-functional now includes `java.time.*` bindings (jdk8+ date/time APIs)

63.40 1.4.0 (2019-11-28)

63.40.1 Breaking changes

- Remove Google Analytics jQuery plugin.
- Remove CarouFredSel jQuery plugin.
- Remove Hotkeys jQuery plugin.
- Remove Autocomplete jQuery plugin.
- Remove ItemIt jQuery plugin.
- Remove ListFilter jQuery plugin.
- Remove Modal Fancybox jQuery plugin.
- Remove Easing jQuery plugin.
- Remove Placeholder Polyfill jQuery plugin.
- Remove ScrollInViewport jQuery plugin.
- Remove SortableListUpdate jQuery plugin.
- Remove Waypoints jQuery plugin.
- Remove obfuscated email jQuery plugin.
- Remove FileUpload jQuery plugin.
- Remove JSON jQuery plugin.
- Remove CarouFredSel webjar.
- Remove Modal Fancybox webjar.
- Remove JSON jQuery webjar.
- BS4: Keep only jQuery UI datepicker resources (js and css).

63.40.2 Bugfixes

- Fix up jQuery UI MonthPicker.
- Fix up JavaScript inherited dependencies.
- Fix confirm modal dependency.

63.40.3 Enhancements

- BasicApp: add a custom `BasicApplicationUserDetailsService` to deal with permissions by role.
- BS3: Move Font Awesome package.

63.40.4 Updates

- jQuery Mask 1.11.2 -> 1.14.16

63.41 1.3.2 (2019-11-18)

63.41.1 Bugfixes

- Fix stackoverflow on `Announcement` with `getNameForToString()` and `getDisplayName()` methods.
- Use `Predicates2` instead of `Predicates` (guava).
- Add missing Bootstrap utility `.stretched-link`.
- Remove `position: relative` from Bootstrap cols.

63.42 1.3.1 (2019-10-23)

63.42.1 Bugfixes

- Transaction synchronization: unbind context before `doOnRollback` as synchronization is already removed by caller and remaining resources prevent correct transaction synchronization creation during `doOnRollback`.

63.42.2 Updates

- Font Awesome 5.10.2 -> 5.11.2
- Popper.js 1.15.0 -> 1.16.0

63.43 1.3.0 (2019-10-17)

63.43.1 Breaking changes

- `DataTableBuilder`: `.addRowCssClass(...)` has been removed. Use `.rows().withClass(...).end()` instead with proper indentation.
- Due to Flyway update, migration parent has changed. `AbstractDataUpgradeMigration.java` must now implement `IglooMigration.java`.
- Property `notification.test.emails` has been renamed `notification.mail.filter.emails`
- Property `notification.mail.recipientsFiltered` has been replaced by `notification.mail.send.mode`. It is no longer a boolean value. It is now an enumeration, with the following values :
  - `SEND`, emails are sent to their designated recipients
  - `FILTER_RECIPIENTS`, email recipients are filtered to a specific list given by the property `notification.mail.filter.emails`
  - `NO_EMAIL`, no email is sent by the application

63.43.2 Updates

- Select2 4.0.9 -> 4.0.10
- Flyway 5.0.7 -> 5.2.4

63.43.3 Bugfixes

- BasicApp: preload scss file for both themes.

63.43.4 Enhancements

- Add configuration property `autoprefixer.enabled` to enable or disable Autoprefixer.
- BasicApp: sidebar user quicksearch only visible for admins.
- Add `table-layout` css classes. Usage : `table-layout{-sm|-md|-lg|-xl}-(auto|fixed)`
- `DataTableBuilder`: row item model dependant behaviors and css classes on rows and actions columns elements + single element.

```
- IBuildState#addRowCssClass(IDetachableFactory<? super IModel<? extends T>, ?␣
↪extends String>);
- IActionColumnAddedElementState#withClass(String);
- IActionColumnCommonBuildState#withClassOnElements(String);
```

```
+ IDataTableRowsState#add(Collection<? extends IDetachableFactory<? super IModel<?␣
↪extends T>, ? extends Behavior>>);
+ IDataTableRowsState#add(IDetachableFactory<? super IModel<? extends T>, ? extends␣
↪Behavior> rowsBehaviorFactory);
+ IDataTableRowsState#add(Behavior, Behavior...);
+ IDataTableRowsState#withClass(Collection<? extends IDetachableFactory<? super␣
↪IModel<? extends T>, ? extends IModel<? extends String>>>);
+ IDataTableRowsState#withClass(IDetachableFactory<? super IModel<? extends T>, ?␣
↪extends IModel<? extends String>>);
+ IDataTableRowsState#withClass(IModel<? extends String>);
+ IDataTableRowsState#withClass(String, String...);
+ IDataTableRowsState#end();

+ IActionColumnAddedElementState#withClass(Collection<? extends IDetachableFactory<?
↪ super IModel<? extends T>, ? extends IModel<? extends String>>>);
+ IActionColumnAddedElementState#withClass(IDetachableFactory<? super IModel<?␣
↪extends T>, ? extends IModel<? extends String>>);
+ IActionColumnAddedElementState#withClass(IModel<? extends String>);
+ IActionColumnAddedElementState#withClass(String, String...);
+ IActionColumnAddedElementState#add(Collection<? extends IDetachableFactory<?␣
↪super IModel<? extends T>, ? extends Behavior>>);
+ IActionColumnAddedElementState#add(IDetachableFactory<? super IModel<? extends T>,
↪ ? extends Behavior>);
+ IActionColumnAddedElementState#add(Behavior, Behavior...);

+ IActionColumnCommonBuildState#withClassOnElements(Collection<? extends␣
↪IDetachableFactory<? super IModel<? extends T>, ? extends IModel<? extends String>
```

(continues on next page)

(continued from previous page)

```
↪>>);
+ IActionColumnCommonBuildState#withClassOnElements(IDetachableFactory<? super␣
↪IModel<? extends T>, ? extends IModel<? extends String>>);
+ IActionColumnCommonBuildState#withClassOnElements(IModel<? extends String>);
+ IActionColumnCommonBuildState#withClassOnElements(String, String...);
```

- `.gitlab-ci.yml` integrates an owasp / dependency check

63.44 1.2.0 (2019-09-05)

63.44.1 Updates

- Font Awesome 5.10.1 -> 5.10.2

63.44.2 Enhancements

- Add `BootstrapCollapseBehavior` to easily enable BS collapse plugin on components.
- BasicApp: sidebar is automatically displayed if there is enough space.
- BasicApp: add `-webkit-overflow-scrolling: touch` on sidebar.

63.45 1.1.28 (2019-08-30)

63.45.1 Breaking changes

- `QueuedTaskHolder`: remove `CREATION_DATE_SORT`, `TRIGGERING_DATE_SORT`, `START_DATE_SORT` and `END_DATE_SORT`. Use fields without `_SORT` suffix. **Warning**: `QueuedTaskHolder` needs to be reindexed.

63.45.2 Updates

- Bootstrap 3.3.6 -> 3.4.1
- Font Awesome 5.9.0 -> 5.10.1
- Popper.js 1.14.7 -> 1.15.0
- BS4: Select2 4.0.5 -> 4.0.9
- BS3: Select2 4.0.3 -> 4.0.9
- BS3: select2-bootstrap-theme 0.1.0-beta.8 -> 0.1.0-beta.10

63.45.3 Enhancements

- Add `list-group-sub` css class.

63.45.4 Bugfixes

- BS4 modal: remove fade animation on close.
- BS4 tooltip: set `window` as default `boundary` instead of `viewport`.
- BS4 select2: remove options tooltip.
- BS3 select2: update tab key behavior.
- Hibernate Search: use Lucene `missingValue` parameter on HS field context.

63.46 1.1.27 (2019-07-26)

63.46.1 Highlights

- BasicApp: update basic and advanced layouts + consistency. Revamp sidebar (style and positioning) in advanced layout.
- Add build tool **Autoprefixer**: css prefixes like `-webkit-`, `-moz-`, `-ms-`, `-o-`, etc. are automatically added if needed.
- Added PropertySourceLogger, for debugging/maintenance purpose.

63.46.2 Breaking changes

- Drop Igloo Infinispan maven module.

63.46.3 Bugfixes

- `FilterByModelItemModelAwareCollectionModel`: Use copy of `unfiltered` (iterator) to avoid concurrent modification exceptions.
- `AbstractJpaSearchQuery`: Method `containsIfGiven` use `CollectionPathBase` instead of `CollectionPath` to allow `SetPath` and `ListPath`.
- Fix wicket-more-jqplot `pom.xml` to embed Js files. May fix "resource not found" messages when using JQPlot charts.
- Feedback panel (BS4): fix unwanted overlay preventing users to interact with the bottom (or top) of the page.

63.47 1.1.26 (2019-07-03)

63.47.1 Bugfixes

- Transaction synchronization: `unbindContext()` must be called in a finally block. Otherwise, in rare case where previous call `doOnRollback()` throw an error, context will be bind for the current thread forever. If really needed, the new context will not be bind in future (for the same thread).

63.47.2 Enhancements

- Announcement: various enhancements and bugfixes.

63.47.3 Updates

- Font Awesome 5.8.1 -> 5.9.0

63.48 1.1.25 (2019-06-11)

63.48.1 Bugfixes

- `FilterByModelItemModelAwareCollectionModel`: Fix `size` method to use the filtered iterable instead of using the unfiltered model size.

63.48.2 Enhancements

- BS3 affix js: check position on dom ready.

63.49 1.1.24 (2019-05-03)

63.49.1 Updates

Warning:

- **wicket-webjars**: bug in latest versions from 2.0.11 to 2.0.14, don't use them.
- **wicket** and **wicketstuff-select2**: bug in latest version 8.3.0 in wicketstuff-select2 dependency.

- **spring-core 5.1.4.RELEASE -> 5.1.6.RELEASE**
- **hibernate-core 5.4.1 -> 5.4.2**
- hibernate-validator 5.4.2 -> 5.4.3
- wicket-webjars 2.0.8 -> 2.0.10
- webjars-locator-core 0.35 -> 0.37
- spring-security 5.1.3.RELEASE -> 5.1.4.RELEASE

- flying-saucer-pdf 9.1.16 -> 9.1.18
- guava 27.0-jre -> 27.1-jre
- commons-codec 1.11 -> 1.12
- jsass 5.7.3 -> 5.7.4
- aspectjrt 1.9.1 -> 1.9.2
- aspectjweaver 1.9.1 -> 1.9.2
- jsch 0.1.54 -> 0.1.55
- slf4j 1.7.25 -> 1.7.26
- cglib-nodep 3.2.8 -> 3.2.10
- ph-css 4.1.3 -> 6.1.2
- HikariCP 3.2.0 -> 3.3.1
- commons-collections4 4.2 -> 4.3
- commons-fileupload 1.3.3 -> 1.4
- commons-configuration2 2.3 -> 2.4
- httpcore 4.5.6 -> 4.5.7
- httpclient 4.4.10 -> 4.4.11
- assertj 3.11.1 -> 3.12.2
- assertj-guava 3.2.0 -> 3.2.1
- elasticsearch 5.6.9 -> 5.6.10
- elasticsearch-cluster-runner 5.6.9.0 -> 5.6.10.0
- flywaydb 5.0.7 -> 5.2.4
- javassist 3.24.0-GA -> 3.24.1-GA
- passay 1.3.1 -> 1.4.0
- allure-junit4 2.8.1 -> 2.10.0
- ehcache 2.10.6 -> 2.10.6.5.1
- allure-maven 2.9 -> 2.10.0
- mockito-core 2.23.0 -> 2.25.1
- jackson 2.9.7 -> 2.9.8
- h2database 1.4.197 -> 1.4.199
- maven-javadoc-plugin 3.0.1 -> 3.1.0
- jacoco-maven-plugin 0.8.0 -> 0.8.3
- maven-assembly-plugin 3.1.0 -> 3.1.1
- maven-clean-plugin 3.0.0 -> 3.1.0
- maven-compiler-plugin 3.7.0 -> 3.8.0
- maven-dependency-plugin 3.0.2 -> 3.1.1
- maven-deploy-plugin 2.8.2 -> 3.0.0-M1

- maven-enforcer-plugin 3.0.0-M1 -> 3.0.0-M2
- maven-install-plugin 2.5.5 -> 3.0.0-M1
- maven-failsafe-plugin 2.21.0 -> 3.0.0-M3
- maven-jar-plugin 3.0.2 -> 3.1.1
- maven-resources-plugin 3.0.2 -> 3.1.1
- maven-surefire-plugin 2.21.0 -> 3.0.0-M3
- maven-war-plugin 3.2.1 -> 3.2.2
- animal-sniffer-maven-plugin 1.16 -> 1.17
- wagon-maven-plugin 1.0 -> 2.0.0
- wagon-ssh-external 3.2.0 -> 3.3.1

63.49.2 Dependencies deleted

- pgjdbc-ng
- solr-core

63.49.3 Enhancements

Added Owasp Dependency-Check and Versions maven plugin for maven dependencies.

Refactor basic-application java configuration, now uses a custom Spring-boot annotation.

63.50 1.1.23 (2019-03-04)

63.50.1 Enhancements

- Excel init data: fallback on old xls format to avoid breaking change.

63.51 1.1.22 (2019-03-04)

63.51.1 Breaking changes

- Refactor `ReferenceData`:
  - Remove `*Simple*ReferenceData*` classes and references.
  - Rename `*Localized*GenericReferenceData*` classes and references to `*GenericReferenceData*`
  - BasicApp: rename `*LocalizedReferenceData*` classes and references to `*ReferenceData*`.
  - BasicApp: rename `*Simple*ReferenceData*` classes and references to `*Basic*ReferenceData*`.

63.51.2 Enhancements

Warning: This is a unwanted breaking change. Use 1.1.23 instead to keep using the old xls format.

- Excel init data: use xlsx format instead of xls.

63.52 1.1.21 (2019-03-29)

63.52.1 Updates

- Bootstrap 4.2.1 -> 4.3.1
- Font Awesome 5.7.0 -> 5.8.1
- popper.js 1.14.6 -> 1.14.7

63.52.2 Bugfixes

- BasicApp: fix `UserPasswordValidator` to check the username rule. It now has to be added to a `ModelValidatingForm` instead of a `Form`.
- BasicApp: fix email check on password reset page.

63.52.3 Enhancements

- Select2: override BS theme to make multiple selection choices more responsive.

63.53 1.1.20 (2019-03-22)

63.53.1 Bugfixes

- Fix Hibernate Search sort util to deal with score sort.
- Fix condition for `notEmpty` and `mapNotEmpty` predicates.

63.53.2 Enhancements

- BS3 module:
  - Custom Select2 4.0.3 js file.
  - Update Select2 Bootstrap 3 theme and clean up override.
  - Update JQuery UI to 1.12.1 with custom js and css files.
  - Change pagination default size (small) in panel add-in.
  - Update logo on console sign in page.
  - Change modal backdrop style.

- Fix popover html template.

63.54 1.1.19 (2019-02-25)

63.54.1 Updates

- Bindgen 4.0.1 -> 4.0.2

63.54.2 Enhancements

- Update and fix footer layout on BasicApp and console template.

63.55 1.1.18 (2019-02-13)

63.55.1 Updates

- Hibernate 5.3.7 -> 5.4.0
- Hibernate 5.10.4 -> 5.11.0
- Spring 5.0.10 -> 5.1.4
- Spring security 5.0.9 -> 5.1.3
- Font Awesome 5.6.3 -> 5.7.0

Hibernate & JAXB dependencies

From 5.4.0, Hibernate includes JAXB dependencies in pom.xml, so this new release transitively includes javax.xml.bind:jaxb-api and org.glassfish.jaxb:jaxb-runtime (and transitive dependencies). Please check your dependencies.

63.55.2 Enhancements

- Improve inclusion of tables into cards with new custom css classes (`.table-bordered-inner`, `.table-card-body`, `.card-body-table`). From now on every content in a `card` should be placed under a `card-body` element.
- Add new method `replaceAll` in `CollectionUtils` utilitary to provide the transformation to operate on the reverse collection.
- Creation of a new Igloo module, `igloo-component-jpa-more-test`, that was originally included in `igloo-component-jpa-more`. It includes utilitaries for tests and all tests present in `igloo-component-jpa-more` `src/test` package.
- Select2: Override `ChoiceProvider` to add `offset` and `limit` parameters to `query` method. Also, compute `hasMore` attribut for ajax response.

63.56 1.1.17 (2019-01-04)

63.56.1 Updates

- Bootstrap 4.1.3 -> 4.2.1
- Font Awesome 5.6.1 -> 5.6.3

63.57 1.1.16 (2018-12-28)

63.57.1 Bugfixes

- Fix partial reindexation form not submitted.
- BasicApp: fix email in import excel files.

63.57.2 Breaking changes

- Update scss custom grid:
  - Remove `.row-default` and `.row-compact`, use `.row-md` and `.row-xs` instead.
  - Change `$grid-gutter-widths` to `$grid-gutters` and update keys from `(0, 1, 2, 3, 4, 5, 6)` to `(0, xxs, xs, sm, md, lg, xl, xxl)`.
  - Add `$layout-container-padding-x` for consistency across containers in page sections.
  - Revamp css for description parts (label-value display).

63.57.3 Updates

- Allure (test reports) updated to version 2.8.1

63.58 1.1.15 (2018-12-14)

63.58.1 Bugfixes

- Fix issue 16 Webjars - too many open files

63.58.2 Updates

- Font Awesome 5.5.0 -> 5.6.1
- Wicket Stuff Select2 8.1.0 -> 8.2.0
- Apache POI 4.0.0 -> 4.0.1
- Popper.js 1.14.4 -> 1.14.6
- Clipboard.js 2.0.1 -> 2.0.4

63.58.3 Enhancements

- BasicApp: consistent use of default locale french.
- BasicApp: refactor users admin pages.
- BasicApp: add tabs in user detail pages.

63.58.4 WicketTester

WicketTester mecanism has been improved by providing new utilitary methods and somes modules were refactored in that way.

63.59 1.1.14 (2018-12-03)

63.59.1 Enhancements

- Bootstrap Modal changes:
  - Use custom js file `modal-more.js` to override modal behavior.
  - Move `_enforceFocus` method override in `modal-more.js`.
  - Override `show` and `hide` methods to move modal to body on show and put it back to its parent on hide.
  - Override `show` and `hide` methods to force modal to close on transition.
  - Remove custom `modal.js` override, no longer needed.
- BasicApp: minor scss updates.

63.60 1.1.13 (2018-11-23)

63.60.1 Bugfixes

- Fix Apache POI dependency: add missing commons-math3.
- Remove from html useless confirm modal on hidden event.
- BasicApp: add missing visible condition on navbar submenu items.

63.61 1.1.12 (2018-11-19)

Warning: Apache POI 4.0.0: dependency `commons-math3` is missing. Use Igloo 1.1.13 instead or add the dependency locally.

63.61.1 Bugfixes

- Add missing Bootstrap Util js dependency for Bootstrap Modal js.

63.61.2 Updates

- Wicket 8.1.0 -> 8.2.0
  - https://wicket.apache.org/news/2018/11/17/wicket-8.2.0-released.html
- javax.mail:mail 1.4.7 updated to com.sun.mail:javax.mail 1.6.2
  - javax.mail:mail added as a forbidden dependency
  - igloo-component-spring dependency modified to com.sun.mail:javax.mail
  - if you declare your own javax.mail:mail dependency in you project, please update groupId/artifactId with com.sun.mail/javax.mail
- poi 3.17.0 updated to poi 4.0.0; there's some breaking change that are not involved in API used by Igloo
  - http://poi.apache.org/changes.html#4.0.0
- Font Awesome 5.3.1 -> 5.5.0
  - https://github.com/FortAwesome/Font-Awesome/releases/tag/5.4.0
  - https://github.com/FortAwesome/Font-Awesome/releases/tag/5.4.1
  - https://github.com/FortAwesome/Font-Awesome/releases/tag/5.4.2
  - https://github.com/FortAwesome/Font-Awesome/releases/tag/5.5.0
- Bindgen 4.0.0 -> 4.0.1

63.61.3 Enhancements

- BasicApp: fix reference data permission check on add action.
- BasicApp: add build date and commit sha in footer.

63.61.4 WicketTester

- The use of `WicketTester` has been added to the BasicApplication. For now it's more a showcase and does not present an entire test coverage.
- This development required to create a new Igloo module, `igloo-component-wicket-more-test`, that was originally included in `igloo-component-wicket-more`.
- Note that the version of `igloo-component-jpa-test` has been declared globally, so it should not be present in project pom anymore.

63.62 1.1.11 (2018-11-06)

Warning: Wicket 8.1.0 websocket implementation is broken wicket Tomcat 8.5+ (https://github.com/apache/wicket/commit/5fc86bdd8628686ffcd124849750f327dccc0c77#diff-94114697955d73acae40bf0a21c6b961)
Please do not update if you use websocket.

63.62.1 Bugfixes

- Fix Select2 focus and dropdown results position in Bootstrap Modal.

63.63 1.1.10 (2018-10-29)

63.63.1 Dependencies

- Major updates:
  - hibernate 5.3.5 -> 5.3.17, hibernate-search 5.10.3 -> 5.10.4
  - spring 5.0.7 -> 5.0.10, spring-security 5.0.6 -> 5.0.9
  - wicket 8.0.0 -> 8.1.0

> **Warning:** Wicket 8.1.0 websocket implementation is broken wicket Tomcat 8.5+ (https://github.com/apache/wicket/commit/5fc86bdd8628686ffcd124849750f327dccc0c77#diff-94114697955d73acae40bf0a21c6b961)
> Please do not update if you use websocket.

- Details:
  - https://github.com/igloo-project/igloo-parent/commit/5fbfce45d2ea92c340dff6107c24a2de0e28e19b
  - https://github.com/igloo-project/igloo-parent/commit/80563f1a097d46fae2c3dfc310966265ecbf46db
  - https://github.com/igloo-project/igloo-parent/commit/d4c3a13fc28ff46c0802f3443b17940c01cb235a
  - https://github.com/igloo-project/igloo-parent/commit/e4107081d829c3f36106674fa778ba771a69d94f
  - https://github.com/igloo-project/igloo-parent/commit/d082937880f43dd076fd7615f15a902aaa00140b

63.64 1.1.9 (2018-10-29)

63.64.1 Bugfixes

- Fix JQuery UI datepicker absolute top position.
- Fix condition on edit button for `ReferenceData` list pages.

63.64.2 Enhancements

- Move Wicket JavaScript and Select2 custom settings to `CoreWicketApplication`.
- Add announcement feature into BasicApp.
- Update error pages (403, 404, 500, 503).

63.64.3 Breaking changes

- `DataTableBuilder`: rename method `when(SerializablePredicate2<? super T> predicate)` to `whenPredicate(SerializablePredicate2<? super T> predicate)`.

63.65 1.1.8 (2018-10-11)

63.65.1 Bugfixes

- Fix conflict between Bootstrap 4 tooltip and JQuery UI widget tooltip.

63.65.2 Breaking changes

- Override JQuery UI js ressource from WiQuery to remove widget tooltip.

63.66 1.1.7 (2018-10-10)

63.66.1 Bugfixes

- Fix inline enclosure component handler in BS modal.
- Fix limit 0 case in QueryDSL and HS search query (return empty list).

63.66.2 Breaking changes

- Custom Wicket tag `wicket:enclosure-container` is now deprecated and will be removed soon. Use Igloo component `EnclosureContainer` instead.

63.66.3 Enhancements

- added tests on rollback behavior in `igloo-component-jpa-test`

63.67 1.1.6 (2018-10-01)

63.67.1 Bugfixes

- Select2: attach component to the Bootstrap modal.

63.67.2 Breaking changes

- Fix Bootstrap variables override.

63.68 1.1.5 (2018-09-24)

63.68.1 Bugfixes

- Select2: prevent dropdown toggle (open) on clear (single + multiple).
- Select2: dispose tooltip on element clear (multiple).

63.68.2 Updates

- Font Awesome 5.3.1.

63.68.3 Enhancements

- Add build informations (date, commit sha, etc.).
- Consistency in use of Wicket `Session.get()`.
- Remove useless icon on cancel buttons.
- BasicApp: fix custom BS checkbox position.
- BasicApp: improve alignment on page title and back to btn.
- BasicApp: minor change on style (nav and pagination background colors).
- BasicApp: remove useless link to user detail page.

63.69 1.1.4 (2018-09-16)

63.69.1 Bugfixes

- issue 18 - fix grouping/splitting behavior when sending a notification to multiple recipients.
- issue 17 - use an explicit setting `notification.mail.sender.behavior` to control what is done when sender is not explictly set when a mail is sent. Get rid of an extraneous INFO message on PropertyServiceImpl when `notification.mail.sender` is empty.

63.69.2 Breaking changes

If you use a not-empty value for `notification.mail.sender`, you need to add to your configuration `notification.mail.sender.behavior=FALLBACK_TO_CONFIGURATION`.

63.70 1.1.3 (2018-09-12)

63.70.1 Bugfixes

- Fix off-request wicket generation (scheduler, async tasks). The issue broke all wicket-based API used outside of an HTTP request.
- Fix a problematic dependency declaration on igloo-dependency-hibernate-search that triggers (wrongly) SNAPSHOT detection by jgitflow plugin.

63.71 1.1.2 (2018-09-06)

63.71.1 Enhancements

This changes are backward-compatible.

- added JNDI's database support (*Use a JDNI datasource*)
- added `igloo.config` and `igloo.log4j` configuration overrides (*Configuration management*)
- drop some useless WARN messages
- AuthenticationManager now uses Spring to search AuthenticationProvider (instead of a static configuration).

63.71.2 Bugfixes

- fix logger's configuration overriding (higher precedence for last files).

63.71.3 Misc

- update developers' information (pom.xml)

63.72 1.1.1 (2018-09-03)

63.72.1 Enhancements

- [4747e20056678ae7300272a6bf9dd39d38ba7b9a] added !default on some styles
- [713cc732fce44c5b26e3cf9e46abf5aebcacb9c3] update some data for Excel-based initialization
- [c28ed4fccd9a25481123da2db48d34d54c031a98] basic-application: use raw bootstrap grid styling instead of custom styles
- [df3bcdb1f215e7005efba0fefcde751064bddb0b] prepare bootstrap-override resources to ease fix and workaround integration in Igloo on external styling resources (bootstrap, ...).

63.72.2 Bugfixes

- [e3007084ca90495cc4e8b9d875938f6d52c8a25c] workaround for bootstrap col-auto max width
- [ad0896a0ab4b28705e9bef122050bf330f557f9b] fix scroll to top (styles)

63.73 1.1.0 (2018-08-20)

Major rewrite of Igloo ; see Migrating to 1.1 guide.

CHAPTER

SIXTYFOUR

MIGRATION GUIDES

64.1 Migrating to Java 11 / Servlet 4.0

From version 3.0.0, Igloo is now build with Java 11. Igloo also uses API Servlet version 4.0 and it will lead to encoding character issues. Here are some vigilance points to be aware of.

64.1.1 Servlet Version

Before version 4.0, the character encoding was set by default. Now, it can be customised ; but it needs to be specified manually. If a character encoding is not specified, the Servlet specification requires that an encoding of `ISO-8859-1` is used.

To use `UTF-8` in your project :

- Update the **xsi:schemaLocation** and the **version** in `web.xml` :

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/
↪javaee/web-app_4_0.xsd"
  version="4.0" metadata-complete="true">
```

- Add character encoding in `web.xml` :

```
<request-character-encoding>UTF-8</request-character-encoding>
<response-character-encoding>UTF-8</response-character-encoding>
```

64.1.2 Java Version

Java 11+ is mandatory for building and running Igloo projects. Update :

- Your IDE setting
- Your Tomcat configuration (Tomcat 9 required) to run with Java 11
- Your maven setting (`JAVA_HOME=/usr/lib/jvm/java-11/ mvn`)
- Your runtime environment (ensure `JAVA_HOME` environment setting for your tomcat runtime)
- Your CI settings (JAVA_HOME, docker image, . . .)

64.1.3 Update your project

- Update parent pom version
- Update `igloo.version` property
- fix broken imports
  - fix wrong imports of `jersey.repackaged` replaced with `com.google...`
  - `de.schlichtherle/TFile` replaced with `net.java.truevfs...`
- optional - Apache POI: remove `setCellType` calls (setCellValue automatically override type)
- optional - Java: replace `Class.newInstance()` by `Class.getConstructor().newInstance()` and add `InvocationTargetException | NoSuchMethodException | IllegalArgumentException` to the existing catch block
- replace `lifecycle-mapping` maven plugin configuration (used to control how eclipse maps maven plugin lifecycle to eclipse build) by `<?m2e ...>` XML processing instruction (see https://www.eclipse.org/m2e/documentation/release-notes-17.html#new-syntax-for-specifying-lifecycle-mapping-metadata) This instruction may be added at `<plugin>` or `<execution>` level

64.1.4 Maven plugins

Compiler and processor plugin uses `--release` option to force java 11 usage. Check you do not override these setttings.

64.1.5 Dependencies

Several dependencies have been updated to Java 11-mandatory versions :

- bindgen : expected >= 5.0.0
- bindgen-bindings-java : expected >= 2.0.0
- bindgen-java : expected >= 2.0.0
- maven-processor-plugin : expected >= 4.4
- jersey : expected >= 2.32
- querydsl : expected >= 5.0.0

Be careful with these if you have one of them in a custom version.

64.1.6 Jersey

Jersey has been updated from jersey-spring3 to jersey-spring5. This update had several effects :

- Jersey do not longer repackage guava, so you have to refactor your `jersey.repackaged` imports if you have some and directly import guava.
- Due to some version conflicts with `jackson-module-jaxb-annotations` and `hk2-utils/runlevel`, `jakarta-xml-bind-api`, `jakarta-activation-api` and `jakarta-annotation-api` versions are now manually declared.

64.1.7 javax.annotations

From Java 9, `javax.annotation` is reworked.

- `javax.annotation` linked to injection management (@PostConstruct, @Managed) are moved in an external dependency, `javax.annotation:javax.annotation-api`. In Igloo context, this dependency is used when Spring beans are managed via `javax.annotation` annotations.
- JSR 305 `javax.annotation` are removed (@Nonnull, ...); we use instead `com.google.code.findbugs:jsr305` dependency.
- `javax.annotation.Generated` is replaced by runtime-provided `javax.annotation.processing.Generated` (updated bindgen version, querydsl 5.0.0 with custom configuration).

In your projects :

- If you encounter spring initialization problems with missing Spring beans, add a scope-provided `javax.annotation:javax.annotation-api` dependency (this dependency is provided by tomcat container).
- If you have `javax.annotation.Generated` references, regenerate code or search for an updated and `javax.annotation.processing.Generated`-friendly dependency.

64.2 Migrating to 1.1.0

1.1.0 is a big update of owsi-core 0.x versions. It includes a major package rename, a lot of updates :

- move to Java 8, Tomcat 8.5 stack
- introduce Java 8 lambdas, streams; reworked predicates, ...
- bindgen optimizations (at compilation time) with lambda
- hibernate update (5.2 -> 5.3)
- bootstrap 4 (wicket & bootstrap 3 code kept for compatibility)
- fontawesome 5 (fontawesome 4 kept for compatibility)
- wicket 8.x
- removed deprecated early target definition in linkdescriptor API
- deprecated GenericListItem removed and replaced by ReferenceData
- migration to select2 from wicketstuff
- maven-release-plugin is replaced by jgitflow plugin
- and others...

Detailed explanations and migration plan can be found from this page.

This is the first official release of igloo-project, forked from OWSI-Core project.

        * *Mockito*
    - *No longer supported*
        * *JDK 7*
        * *Tomcat 7*
        * *JFreeChart*
        * *Maven*
        * *Password encoding*
        * *YUI Compressor*
        * *maven-release-plugin*
        * *tomcat-jdbc*
        * *Joda-Time*
        * *Session - redirectUrl*
        * *Javascript*
    - *New features*
        * *Test tooling*
        * *Bootstrap 4*
        * *JNDI Datasource*
    - *Migration script*

64.2.1 Detailed modifications

Bindgen

Bindgen is updated to 4.0.0. This new version uses lambda for binding generation.

You need to handle the following issues:

- AbstractBinding must not be used; please use BindingRoot.
- AbstractCoreBinding must not be used; please use ICoreBinding.
- If you use a custom parent binding class in your project, you need to adapt it to work with the new version. Check AbstractCoreBinding.java history in https://github.com/igloo-project/bindgen-java project to help you.

This script should handle these updates:

```
#! /bin/bash

while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.commons.util.binding.AbstractCoreBinding/org.iglooproject.commons.
↪util.binding.ICoreBinding/g
s/\\\QAbstractCoreBinding/ICoreBinding/g
s/\\\Qorg.bindgen.binding.AbstractBinding/org.bindgen.BindingRoot/g
```

(continues on next page)

(continued from previous page)

```
s/\\\QAbstractBinding/BindingRoot/g
EOF
```

Functional / lambda

This new version enforces the use of `Serializable{Predicate,Function,Supplier}2` classes where we previously use `Serializable*` variants. This is also enforced in our custom wicket components methods.

The purpose of this move is :

- To explicitly state in codebase the `Serializable` constraint when it is needed by wicket page persistence model.
- To build types that enable both guava, java.util and Serializable contracts and so that they can be used easily with all these API.
- To implement once and consistently all the needed composition methods (for `SerializablePredicate2`, it is needed to override `and()`, `or()`, ... to enforce Serializable contract on generated Predicate).

To ease the transition, the following mechanisms can be used:

- This script allow to track all package/class renaming involved.

```
#! /bin/bash

while read line; do
find -name '*.java' -exec perl -i -pe "$line" {} ';'
done <<EOF
s/\\\Qorg.iglooproject.commons.util.functional/org.iglooproject.functional/g
s/SerializableFunction(?!2)/SerializableFunction2/g
s/SerializablePredicate(?!2)/SerializablePredicate2/g
s/SerializableSupplier(?!2)/SerializableSupplier2/g
EOF
```

- For `@FunctionalInterface` s, dynamic casting allow to obtain the new types. (beware that lambda must involve only `Serializable` objects).
- Helpers in `{Predicate,Function,Supplier}s2` allow to transform existing objects to the targetted type.

For an already working codebase, your `{Predicate,Function,Supplier}` used by wicket are surely `Serializable`, even if not explicitly enforced by type declaration, so it is safe to use provided helpers to wrap them to a `{Predicate,Function,Supplier}2` instance.

Configuration management

With version 1.1.x, configuration management is heavily rewritten. Here are the steps and precaution to migrate your application.

Class rename

Some classes need a package renaming. Following script should be used for this operation.

```
#! /bin/bash

while read line; do
find \( -name "*.java" -o -name "web.xml" \) -exec sed -i "$line" {} \;
done <<EOF
s/org.iglooproject.spring.config.AbstractExtendedApplicationContextInitializer/org.
↪iglooproject.config.bootstrap.spring.AbstractExtendedApplicationContextInitializer/g
s/org.iglooproject.spring.config.annotation.
↪ApplicationConfigurerBeanFactoryPostProcessor/org.iglooproject.config.bootstrap.spring.
↪ApplicationConfigurerBeanFactoryPostProcessor/g
s/org.iglooproject.spring.config.ExtendedApplicationContextInitializer/org.iglooproject.
↪config.bootstrap.spring.ExtendedApplicationContextInitializer/g
s/org.iglooproject.spring.config.spring.annotation.ApplicationDescription/org.
↪iglooproject.config.bootstrap.spring.annotations.ApplicationDescription/g
s/org.iglooproject.spring.config.spring.annotation.ConfigurationLocations/org.
↪iglooproject.config.bootstrap.spring.annotations.ConfigurationLocations/g
EOF
```

Maven profile resource filtering removal

New projects initialized with basic application no longer use maven profile to filter resources.

You can still use profiles in your project if you want to (as profiles are configured in the project, there is no impact on this point).

ConfigurationLocations

- `configurationLocationProvider()`: This attribute is removed from annotation. There is no longer customization of this behavior.
- `locations()`: placeholders available here are now handled differently.
  - `${user}` must be replaced by `${user.name}`
  - `${environment}` is no longer available
  - `${igloo.config}` is no longer available
  - `${applicationName}` must be renamed `${igloo.applicationName}` but it is recommended to load it with profile configurationLocations setting.

If you use `environment` or `igloo.config`, the preferred way to replace this behavior is to add the targeted location to your profile definition of `igloo.configurationLocations` in your *bootstrap* configuration.

With default bootstrap configuration, renaming your file configuration-env-*.properties (for environment development, preproduction, production) or configuration-user-*.properties (for user-related file) should be enough.

Default bootstrap configuration also load `file:/etc/${igloo.applicationName}/configuration.properties` with preproduction and production profiles.

log4j.configurationLocations

This configuration must be moved to your bootstrap configuration. Configuration in `configuration*.properties` are ignored.

If you want to use default bootstrap settings, renaming your files may be enough: * `log4j-<environment>.properties` in `log4j-env-<environment>.properties`` * `log4j-<user>.properties` in `log4j-user-<user>.properties``

Configuration file naming

With default bootstrap configuration, environment related files are named `environment-*.properties` and personal configurations are named `user-*.properties`.

configuration-private.properties

configuration-private.properties is no longer loaded from igloo default configurations. If you use a configuration-private.properties, please add it to your custom `ConfigurationLocations.locations`.

configuration-bootstrap*.properties

Copy `configuration-bootstrap.properties` from basic-application in your `application-core/src/{main,test}/resources/`.

Rename your `configuration-test.properties` to `configuration-env-test.properties`.

You should check in these files:

- ensure that configurations are available or defined by default for all the targetted profiles (i.e.: development, preproduction, production, ...).
- that `igloo.default.spring.profiles.active` or by profile setting `igloo.<profile>.spring.profiles.active` is correct.
- if different Spring profiles are needed in unit-test, you may use @ActiveProfiles to handle it test-by-test.
- that loaded log4j configurations are corrects: `igloo.<profile>.log4j.configurationLocations`

You should also check:

- that your base test class uses `ExtendedTestApplicationContextInitializer`
- that your base test class activates test profile with @PropertySource("igloo.profile=test")

Runtime

In each target environment, configure either `IGLOO_PROFILE` environment variable or `igloo.profile` system property.

IPropertyService

Property registration is modified to allow to use properties during Spring initialization.

Originally introduced to reference some properties during Flyway initialization (flyway is initialized very early, before hibernate and persistence layer), this modification allow other components to use IPropertyService lookup before application is fully initialized.

This modification need to change signature of `IPropertyRegistryConfig.register` method.

Method register(IPropertyRegistry propertyRegistry)

This script allow to quickly refactor your application, if your `IPropertyRegistryConfig` classes are named `*PropertyRegistryConfig.java`.

If not, the modification is to declare `IPropertyRegistryConfig.register` **public**.

```
#! /bin/bash

find -name "*PropertyRegistryConfig.java" -exec perl -p -i -e 's/\Qprotected void␣
↪register(IPropertyRegistry registry)/public void register(IPropertyRegistry registry)/g
↪' {} \;
```

hibernate.implicit_naming_strategy

`org.iglooproject.jpa.hibernate.model.naming.ImplicitNamingStrategyLegacyJpaComponentPathImpl` is removed and replaced with `ImplicitNamingStrategyJpaComponentPathImpl`.

To check if this change is problematic with your database model, use `*SqlUpdateScriptMain` to check and compare new database model with your previous version.

Hibernate 5.2.13 (upstream)

Changes in the @TableGenerator and @SequenceGenerator name scope

cf http://in.relation.to/2018/02/07/hibernate-orm-5213-final-release/

> In order to be compliant with the JPA specifications, the names of identity generators need now be considered global, and no longer scoped to the entity in which they are declared. This means existing applications might now have a naming conflict which needs to be addressed to upgrade. Configuring two generators, even with different types but with the same name will now cause a java.lang.IllegalArgumentException to be thrown at boot time.

TODO lazy-loading on identifier access

Hibernate 5.3.0 (upstream)

Hibernate 5.3 is mainly a maintenance release, with some important modification:

- Complete list here: http://in.relation.to/2018/01/18/hibernate-orm-530-beta1-release/
- HQL legacy positional parameters removed: https://hibernate.atlassian.net/browse/HHH-12101
- JPA compliance level added: https://github.com/hibernate/hibernate-orm/blob/master/hibernate-core/src/main/java/org/hibernate/jpa/JpaCompliance.java ; we choose to use strict compliance enforcement.
- Features for Caching an inheritance: https://hibernate.atlassian.net/browse/HHH-12146
- JPA 2.2 support

Bootstrap

Bootstrap 3

Bootstrap 3 is now deprecated. If you use it, you need to add igloo-component-wicket-bootstrap3 as a dependency, and to run the following script:

```
#! /bin/bash

while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManager/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.statement.BootstrapModalManager/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManagerStatement/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪js.bootstrap.modal.statement.BootstrapModalManagerStatement/g
s/\\\Qorg.iglooproject.wicket.more.WicketBootstrapPackage/org.iglooproject.wicket.
↪bootstrap3.WicketBootstrapPackage/g
s/\\\Qorg.iglooproject.wicket.more.application.WicketBootstrapModule/org.iglooproject.
↪wicket.bootstrap3.application.WicketBootstrapModule/g
s/\\\Qorg.iglooproject.wicket.more.config.spring.WicketBootstrapServiceConfig/org.
↪iglooproject.wicket.bootstrap3.config.spring.WicketBootstrapServiceConfig/g
s/\\\Qorg.iglooproject.wicket.more.config.spring.AbstractBootstrapWebappConfig/org.
↪iglooproject.wicket.bootstrap3.config.spring.AbstractBootstrapWebappConfig/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleSignInPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleSignInPage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleAccessDeniedPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleAccessDeniedPage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleLoginFailurePage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleLoginFailurePage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleLoginSuccessPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleLoginSuccessPage/g
s/\\\Qorg.iglooproject.wicket.more.console.resources.CoreWicketConsoleResources/org.
↪iglooproject.wicket.bootstrap3.console.resources.CoreWicketConsoleResources/g
s/\\\Qorg.iglooproject.wicket.more.console.common.util.LinkUtils/org.iglooproject.wicket.
↪bootstrap3.console.common.util.LinkUtils/g
s/\\\Qorg.iglooproject.wicket.more.console.common.component.
↪JavaClassesListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.common.
```

(continues on next page)

(continued from previous page)

```
↪component.JavaClassesListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.common.component.PropertyIdListPanel/org.
↪iglooproject.wicket.bootstrap3.console.common.component.PropertyIdListPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.common.form.PropertyIdEditPopup/org.
↪iglooproject.wicket.bootstrap3.console.common.form.PropertyIdEditPopup/g
s/\\\Qorg.iglooproject.wicket.more.console.template.ConsoleTemplate/org.iglooproject.
↪wicket.bootstrap3.console.template.ConsoleTemplate/g
s/\\\Qorg.iglooproject.wicket.more.console.template.ConsoleConfiguration/org.
↪iglooproject.wicket.bootstrap3.console.template.ConsoleConfiguration/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.
↪ConsoleLessCssResourceReference/org.iglooproject.wicket.bootstrap3.console.template.
↪style.ConsoleLessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.
↪ConsoleSignInLessCssResourceReference/org.iglooproject.wicket.bootstrap3.console.
↪template.style.ConsoleSignInLessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.CoreConsoleCssScope/org.
↪iglooproject.wicket.bootstrap3.console.template.style.CoreConsoleCssScope/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.page.
↪ConsoleMaintenanceEhCachePage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.page.ConsoleMaintenanceEhCachePage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheProgressBarComponent/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheProgressBarComponent/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheCachePortfolioPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheCachePortfolioPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheCacheModificationPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheCacheModificationPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.file.page.
↪ConsoleMaintenanceFilePage/org.iglooproject.wicket.bootstrap3.console.maintenance.file.
↪page.ConsoleMaintenanceFilePage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.search.page.
↪ConsoleMaintenanceSearchPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪search.page.ConsoleMaintenanceSearchPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.renderer.
↪QueueTaskRenderer/org.iglooproject.wicket.bootstrap3.console.maintenance.queuemanager.
↪renderer.QueueTaskRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.renderer.
↪QueueManagerRenderer/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪queuemanager.renderer.QueueManagerRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.page.
↪ConsoleMaintenanceQueueManagerPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.queuemanager.page.ConsoleMaintenanceQueueManagerPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.component.
↪ConsoleMaintenanceQueueManagerNodePanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.queuemanager.component.ConsoleMaintenanceQueueManagerNodePanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.authentication.page.
↪ConsoleMaintenanceAuthenticationPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.authentication.page.ConsoleMaintenanceAuthenticationPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.renderer.INodeRenderer/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.infinispan.renderer.
```

(continues on next page)

(continued from previous page)

```
↪INodeRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.page.
↪ConsoleMaintenanceInfinispanPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.page.ConsoleMaintenanceInfinispanPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanNodesPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanNodesPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanLocksPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanLocksPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanRolesPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesRequestsPanel/org.iglooproject.wicket.bootstrap3.
↪console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesRequestsPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanClusterPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanClusterPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.form.
↪NodeDropDownSingleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪infinispan.form.NodeDropDownSingleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.form.
↪ConsoleMaintenanceInfinispanRoleAssignPopup/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.form.ConsoleMaintenanceInfinispanRoleAssignPopup/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.page.
↪ConsoleMaintenanceTaskDescriptionPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.task.page.ConsoleMaintenanceTaskDescriptionPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.page.
↪ConsoleMaintenanceTaskListPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.page.ConsoleMaintenanceTaskListPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskTypeListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.task.
↪component.TaskTypeListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskQueueIdListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskQueueIdListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskStatusPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskStatusPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskExecutionResultPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.task.
↪component.TaskExecutionResultPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskResultListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskResultListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskFilterPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskFilterPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskStatusListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskStatusListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
```

(continues on next page)

(continued from previous page)

```
↪TaskManagerInformationPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskManagerInformationPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskResultPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskResultPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskResultsPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskResultsPanel/
↪g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.gestion.page.
↪ConsoleMaintenanceGestionPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪gestion.page.ConsoleMaintenanceGestionPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.upgrade.page.
↪ConsoleMaintenanceDonneesPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪upgrade.page.ConsoleMaintenanceDonneesPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.upgrade.component.
↪DataUpgradePanel/org.iglooproject.wicket.bootstrap3.console.maintenance.upgrade.
↪component.DataUpgradePanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.template.
↪ConsoleMaintenanceTemplate/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪template.ConsoleMaintenanceTemplate/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.WicketBootstrapComponentsModule/
↪org.iglooproject.wicket.bootstrap3.markup.html.bootstrap.
↪WicketBootstrapComponentsModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.component.BootstrapLabel/org.
↪iglooproject.wicket.bootstrap3.markup.html.bootstrap.component.BootstrapLabel/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.component.BootstrapBadge/org.
↪iglooproject.wicket.bootstrap3.markup.html.bootstrap.component.BootstrapBadge/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.collapse.
↪BootstrapCollapseJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.collapse.BootstrapCollapseJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.dropdown.
↪BootstrapDropdownModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.dropdown.BootstrapDropdownModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.dropdown.
↪BootstrapDropDownJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.dropdown.BootstrapDropDownJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.confirm.
↪BootstrapConfirmModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.confirm.BootstrapConfirmModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.confirm.
↪BootstrapConfirmJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.confirm.BootstrapConfirmJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tooltip.
↪BootstrapTooltipModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.tooltip.BootstrapTooltipModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tooltip.
↪BootstrapTooltipJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.tooltip.BootstrapTooltipJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.popover.
↪BootstrapPopoverJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.popover.BootstrapPopoverJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.popover.
↪BootstrapPopoverModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
```

(continues on next page)

(continued from previous page)

```
↪bootstrap.popover.BootstrapPopoverModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.alert.
↪BootstrapAlertJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.alert.BootstrapAlertJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.SimpleOptions/org.
↪iglooproject.wicket.bootstrap3.markup.html.template.js.bootstrap.SimpleOptions/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixOptions/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.affix.BootstrapAffixOptions/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixBehavior/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.affix.BootstrapAffixBehavior/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.affix.BootstrapAffixJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tab.
↪BootstrapTabModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.tab.BootstrapTabModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tab.
↪BootstrapTabJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.
↪template.js.bootstrap.tab.BootstrapTabJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.modal.BootstrapModalJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.BootstrapModalModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManager/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.statement.BootstrapModalManager/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManagerStatement/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪js.bootstrap.modal.statement.BootstrapModalManagerStatement/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalManagerJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.
↪markup.html.template.js.bootstrap.modal.
↪BootstrapModalManagerJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.button.
↪BootstrapButtonJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.button.BootstrapButtonJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.button.
↪BootstrapButtonModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.button.BootstrapButtonModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.
↪markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.scrollspy.BootstrapScrollSpyModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.select2.
↪Select2CssResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.template.
```

(continues on next page)

(continued from previous page)

```
↪css.select2.Select2CssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.fontawesome.
↪CoreFontAwesomeCssScope/org.iglooproject.wicket.bootstrap3.markup.html.template.css.
↪bootstrap.fontawesome.CoreFontAwesome4CssScope/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.
↪CoreBootstrap3CssScope/org.iglooproject.wicket.bootstrap3.markup.html.template.css.
↪bootstrap.CoreBootstrap3CssScope/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.bootstrap.
↪DefaultBootstrap3LessCssResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.css.bootstrap.bootstrap.DefaultBootstrap3LessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.jqueryui.
↪JQueryUiCssResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪css.bootstrap.jqueryui.JQueryUiCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.respond.
↪RespondJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.
↪template.js.respond.RespondJavaScriptResourceReference/g
EOF
```

Bootstrap Modal

- Remove Bootstrap Modal override plugin in BS4 module. Use default Bootstrap modal js file and css style. At this time, modal markup is in the html flow (according to Wicket components hierarchy), where the component is added - this may cause some display / style issues.
- In both BS3 and BS4 cases, remove `loading` and `removeLoading` statements. Use `BootstrapModalStatement` instead of `BootstrapModalManagerStatement`.
- Sizing: in BS4, use `AbstractModalPopupPanel#dialogCssClass(IModel<String> dialogCssClassModel)` instead of `AbstractModalPopupPanel#getCssClassNamesModel()` to add sizing css classes.

Font Awesome

- Rename `CoreFontAwesomeCssScope` to `CoreFontAwesome4CssScope`.
- Change static resource directory of Font Awesome 4 from `/font-awesome` to `/font-awesome-4`.

Wicket & UI

Note: Updated to Wicket 8. Read the Wicket 8 migration guide.

Model

- Remove `org.iglooproject.wicket.more.model.ClassModel.java`. This model was useless and potentially harmful in a multithreaded context. Use `Model.of()` instead.
- Remove `org.iglooproject.wicket.more.model.GenericEntityArrayListModel.java`. Use `CollectionCopyModel` with `Suppliers2` and `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.model.GenericEntityHashSetModel.java`. Use `CollectionCopyModel` with `Suppliers2` and `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.model.GenericEntityLinkedHashSetModel`.java. Use `CollectionCopyModel` with `Suppliers2` and `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.model.GenericEntityTreeSetModel.java`. Use `CollectionCopyModel` with `Suppliers2` and `GenericEntityModel` instead.
- Remove `CompositeSortModel#getOrder()`. Use `CompositeSortModel#getActiveOrder(ISort)` or `CompositeSortModel#getSelectedOrder(ISort)` instead.

Component

- Remove `org.iglooproject.wicket.markup.html.basic.HideableLabel.java`. Use `CoreLabel` with `hideIfEmpty()` instead.
- Remove `org.iglooproject.wicket.markup.html.basic.HideableMultiLineLabel.java`. Use `CoreLabel` with `multiline()` and `hideIfEmpty()` instead.
- Remove `org.iglooproject.wicket.more.markup.html.basic.BigDecimalToIntegerLabel.java`. Use `Renderer.fromStringFormat("%1$.0f").asLabel(id, model))` to get the exact same result.
- Remove `org.iglooproject.wicket.more.markup.html.basic.LocaleLabel.java`. Use `CoreLabel` with `Renderer` instead.
- Remove `org.iglooproject.wicket.more.markup.html.basic.PercentageValueLabel.java`. Use new `CoreLabel(id, CoreRenderers.percent().asModel(model))` in most case ('#0.00 %' with a ratio value (from 0 to 1)) instead. To display a non-ratio value (from 0 to 100 for instance), use your own `DecimalFormat` with `df.setMultiplier(1)`, see `CoreRenderers#percentDecimalFormatFunction(String, RoundingMode)`.
- Remove `org.iglooproject.wicket.more.markup.html.image.BooleanGlyphicon.java`. Use `BooleanIcon` instead.
- Remove `org.iglooproject.wicket.more.markup.html.image.BooleanImage.java`. Use `BooleanIcon` instead.
- Remove some public constructors from `AjaxConfirmButton`.
- Remove `org.iglooproject.wicket.more.markup.html.form.AbstractQuickSearchComponent.java`. Use Ajax Select2 with `UpdateOnChangeAjaxEventBehavior` instead.
- Remove `org.iglooproject.wicket.more.markup.html.form.AutocompleteAjaxComponent.java`. Use Ajax Select2 instead.

RepeatingView

- Remove `org.iglooproject.wicket.more.markup.html.collection.AbstractGenericCollectionView.java`. Use `CollectionView` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.AbstractGenericEntityCollectionView.java`. Use `CollectionView` with `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.GenericEntityCollectionView.java`. Use `CollectionView` with `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.GenericEntityListView.java`. Use `CollectionView` with `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.GenericEntitySetView.java`. Use `CollectionView` with `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.GenericEntitySortedSetView.java`. Use `CollectionView` with `GenericEntityModel` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.AbstractSerializedItemCollectionView.java`. Use `CollectionView` with `Models` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.SerializedItemCollectionView.java`. Use `CollectionView` with `Models` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.SerializedItemListView.java`. Use `CollectionView` with `Models` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.SerializedItemSetView.java`. Use `CollectionView` with `Models` instead.
- Remove `org.iglooproject.wicket.more.markup.html.collection.SerializedItemSortedSetView.java`. Use `CollectionView` with `Models` instead.
- Remove `org.iglooproject.wicket.more.markup.repeater.data.GenericEntityListModelDataProvider.java`. Use `ISequenceProvider` instead.
- Remove `org.iglooproject.wicket.more.markup.repeater.data.OddEvenDataView.java`. Use Bootstrap css classes instead.

Visibility

- Remove `org.iglooproject.wicket.more.markup.html.basic.EnclosureBehavior.java`. Use `Condition` or implement your own `AbstractConfigurableComponentBooleanPropertyBehavior` instead.
- Remove `org.iglooproject.wicket.more.markup.html.basic.PlaceholderBehavior.java`. Use `Condition` or implement your own `AbstractConfigurableComponentBooleanPropertyBehavior` instead.
- Remove `org.iglooproject.wicket.more.markup.html.basic.AbstractHidingBehavior.java`. Use `AbstractComponentBooleanPropertyBehavior` instead.
- Remove `AbstractConfigurableComponentBooleanPropertyContainer#collectionModel(IModel)`. Use `Condition#collectionModelNotEmpty(IModel)` instead.
- Remove `AbstractConfigurableComponentBooleanPropertyContainer#model(IModel)`. Use `Condition#modelNotNull(IModel)` instead.

- Remove AbstractConfigurableComponentBooleanPropertyContainer#model(Predicate, IModel). Use Condition#predicate(IModel, Predicate) instead.
- Remove AbstractConfigurableComponentBooleanPropertyContainer#models(IModel, IModel...). Use Condition#modelsAnyNotNull(IModel, IModel...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyContainer#models(Predicate, IModel, IModel...). Use Condition#predicateAnyTrue(Predicate, IModel, IModel...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyContainer#component(Component). Use Condition#componentVisible(Component) instead.
- Remove AbstractConfigurableComponentBooleanPropertyContainer#components(Component, Component...). Use Condition#componentsAnyVisible(Component, Component...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyContainer#components(Collection). Use Condition#componentsAnyVisible(Collection) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#collectionModel(IModel). Use Condition#collectionModelNotEmpty(IModel) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#model(IModel). Use Condition#modelNotNull(IModel) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#model(Predicate, IModel). Use Condition#predicate(IModel, Predicate) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#models(IModel, IModel...). Use Condition#modelsAnyNotNull(IModel, IModel...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#models(Predicate, IModel, IModel...). Use Condition#predicateAnyTrue(Predicate, IModel, IModel...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#component(Component). Use Condition#componentVisible(Component) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#components(Component, Component...). Use Condition#componentsAnyVisible(Component, Component...) instead.
- Remove AbstractConfigurableComponentBooleanPropertyBehavior#components(Collection). Use Condition#componentsAnyVisible(Collection) instead.

Factory

- Remove org.iglooproject.wicket.more.markup.html.factory.AbstractDetachableFactory. Use IDetachableFactory instead.
- Remove org.iglooproject.wicket.more.markup.html.factory.IOneParameterConditionFactory.java. Use IDetachableFactory instead.
- Remove org.iglooproject.wicket.more.markup.html.factory.AbstractOneParameterConditionFactory.java. Use IDetachableFactory instead.
- Remove org.iglooproject.wicket.more.markup.html.factory.IOneParameterModelFactory.java. Use IDetachableFactory instead.
- Remove org.iglooproject.wicket.more.markup.html.factory.AbstractOneParameterModelFactory.java. Use IDetachableFactory instead.
- Remove ComponentFactories#addAll(RepeatingView, Iterable). Use FactoryRepeatingView instead.

- Remove `ComponentFactories#addAll(RepeatingView, Iterable, P)`. Use `FactoryRepeatingView` instead.

Action

- Remove `org.iglooproject.wicket.more.markup.html.action.AbstractAction`. Use `IAction` instead.
- Remove `org.iglooproject.wicket.more.markup.html.action.AbstractAjaxAction`. Use `IAjaxAction` instead.
- Remove `org.iglooproject.wicket.more.markup.html.action.AbstractOneParameterAction`. Use `IOneParameterAction` instead.
- Remove `org.iglooproject.wicket.more.markup.html.action.AbstractOneParameterAjaxAction`. Use `IOneParameterAjaxAction` instead.
- Remove `org.iglooproject.wicket.more.markup.html.template.js.jquery.plugins.bootstrap.confirm.util.AjaxResponseAction`. Use `IOneParameterAjaxAction` instead.

Condition

- Remove `Condition#asValue(IModel<? extends T>, IModel<? extends T>)`. Use `.then(...).otherwise(...)` instead.
- Remove `Condition#asValue(T, T)`. Use `.then(...).otherwise(...)` instead.

DataTableBuilder

- Remove `IAddedLabelColumnState#withLink(LinkGeneratorFactory<T>)`. Use `IAddedLabelColumnState#withLink(ILinkDescriptorMapper)` instead.
- Remove `IAddedLabelColumnState#withLink(AbstractCoreBinding<? super T, C>, LinkGeneratorFactory<C>)`. Use `IAddedLabelColumnState#withLink(AbstractCoreBinding, ILinkDescriptorMapper)` instead.
- Remove `IAddedLabelColumnState#withSideLink(LinkGeneratorFactory<T>`. Use `IAddedLabelColumnState#withSideLink(ILinkDescriptorMapper)` instead.
- Remove `IAddedLabelColumnState#withSideLink(AbstractCoreBinding<? super T, C>, LinkGeneratorFactory<C>)`. Use `IAddedLabelColumnState#withSideLink(AbstractCoreBinding, ILinkDescriptorMapper)` instead.
- Remove `IBuildState#hideTopToolbar()`. Use `IBuildState#hideHeadersToolbar()` instead.
- Remove `IBuildState#hideBottomToolbar()`. Use `IBuildState#hideNoRecordsToolbar()` instead.

Renderer and Converter

- Remove `org.iglooproject.wicket.more.util.convert.converters.HumanReadableEnumConverter.java`
- Remove `org.iglooproject.wicket.more.util.convert.converters.HumanReadableLocaleConverter.java`
- **Remove `org.iglooproject.wicket.markup.html.model.EnumLabelModel.java`. Use `EnumRenderer` instead:**
  - `new EnumLabelModel(enumValueModel)` should become `EnumRenderer.get().asModel(enumValueModel)`
  - `new EnumLabelModel(enumValueModel, nullKeyValue)` should become `EnumRenderer.get().nullsAsResourceKey(nullKeyValue).asModel(enumValueModel)`
  - `new EnumLabelModel(enumValue)` should become `EnumRenderer.get().asModel(new Model<>(enumValue))`
  - `new EnumLabelModel(enumValue, nullKeyValue)` should become `EnumRenderer.get().nullsAsResourceKey(nullKeyValue).asModel(new Model<>(enumValue))`
- Remove `org.iglooproject.wicket.more.markup.html.bootstrap.label.renderer.BootstrapLabelRenderer.java`. Use `BootstrapRenderer` instead.
- Remove `BooleanRenderer#BooleanRenderer()` and `BooleanRenderer#BooleanRenderer(String, String)`. Use static factory methods instead.
- Remove `EnumRenderer#EnumRenderer()` and `EnumRenderer#EnumRenderer(String, String)`. Use static factory methods instead.
- Remove `LocaleRenderer#LocaleRenderer()`. Use `LocaleRenderer#get()` instead.
- Remove `Renderer#from(Renderer<? super T>)`. Use the parameter `Renderer` as-is.

Breadcrumb

`BreadCrumbElement`: remove deprecated support for page class + parameters. Use a `LinkDescriptor` instead. Remove `LinkBreadCrumbElementPanel` as well.

DatePickerSync

DatePickerSync exclusively use `precedents` (previous) and `suivants` (next) attributes. There is no longer `courant` (current) field.

FileUploadMediaTypeValidator

`FileUploadMediaTypeValidator#errorResourceKey` and `FileUploadMediaTypeValidator.setErrorResourceKey(String)` and related constructor are removed. If you use this property, you now need to use component-based resource naming (so `FileUploadMediaTypeValidator`, or `<fieldName>.FileUploadMediaTypeValidator` or `<form>.<fieldName>.FileUploadMediaTypeValidator`).

Bootstrap

- Move `BootstrapColor` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.model` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.model`.
- Move `IBootstrapColor` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.model` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.model`.
- Move `BootstrapRenderer` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.renderer` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.renderer`.
- Move `BootstrapRendererInformation` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.renderer` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.renderer`.
- Move `IBootstrapRendererModel` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.renderer` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.renderer`.
- Move `BootstrapColorBehavior` from `org.iglooproject.wicket.more.markup.html.bootstrap.label.behavior` to `org.iglooproject.wicket.more.markup.html.bootstrap.common.behavior`.

Misc

- Remove `AjaxListeners#refresh(MarkupContainer, Class<? extends Component>, Class<? extends Component>...)`. Use `AjaxListeners#refreshChildren(MarkupContainer, Class, Class...)` instead.
- Remove redirect methods from `CoreWebPage`. Theses methods hide the exception throwing, which makes dead code harder to spot. Just throw a `RestartResponseException` or a `RedirectToUrlException` yourself. Note that if you're using a `IPageLinkGenerator`, it can instantiate the exception for you.
- Remove `CoreWebPage#visible(Component, boolean)`. Just use `Component#setVisible(boolean)` or `Component#setVisibilityAllowed(boolean)`, or (better) add an `EnclosureBehavior` to manage the component's visibility declaratively.
- Remove `NavigationMenuItem#isAccessible()`. Use the `NavigationMenuItem#linkHidingIfInvalid(String)` to create a link that will be hidden when it is invalid, or a `BlankLink` when the `NavigationMenuItem` does not have any `LinkGenerator`.
- Remove `PredicateValidator#PredicateValidator(Predicate, String)`. Use `PredicateValidator#of(Predicate)` and then `PredicateValidator#errorKey(String)`.
- Remove `org.iglooproject.wicket.more.security.authorization.AuthorizeRenderIfPermissionOnModelObject java`. Use validation features in `LinkDescriptors` instead. See `IValidatorState#permission(IModel, String, String...)`.

LinkDescriptor

> **Warning:** Link descriptor early target definition is completely dropped.

- Remove `IBaseState`, `IBaseResourceState`, `IBasePageState`, `IBasePageInstanceState`, `IBaseImageResourceState`. Use late target definition (aka the new and clean way to create link descriptor).
- Remove `IBackwardCompatibleTerminalState`, `IEarlyTargetDefinitionTerminalState`. Use late target definition (aka the new and clean way to create link descriptor).

- Remove `AbstractLinkFactory#builder()`. Use `LinkDescriptorBuilder#start()` or `LinkDescriptorBuilder#toPageInstance(Page)` or `LinkDescriptorBuilder#toPageInstance(IModel)` instead.
- Remove `org.iglooproject.wicket.more.link.descriptor.mapper.LinkGeneratorFactoryToOneParameterLinkDe java`.
- Remove `org.iglooproject.wicket.more.link.descriptor.factory.LinkGeneratorFactory. java`. Instead of extending this class, implement `IOneParameterLinkDescriptorMapper` by extending `AbstractOneParameterLinkDescriptorMapper` or build one such object using a `LinkDescriptorBuilder`.
- Remove `org.iglooproject.wicket.more.link.descriptor.factory. BindingLinkGeneratorFactory.java`. Use `BindingOneParameterLinkDescriptorMapper` instead.
- Remove `ILinkGenerator#INVALID`. Use `LinkDescriptors#invalid()` instead.
- Remove `LinkParameterExtractionException#LinkParameterExtractionException(Throwable)`. Use `LinkParameterExtractionException#LinkParameterExtractionException(String, Throwable)` instead.
- Remove `LinkParameterInjectionException#LinkParameterInjectionException(Throwable)`. Use `LinkParameterInjectionException#LinkParameterInjectionException(String, Throwable)` instead.
- Remove `LinkParameterValidationException#LinkParameterValidationException()`. Use `LinkParameterValidationException#LinkParameterValidationException(String)` or `LinkParameterValidationException#LinkParameterValidationException(String, Throwable)` instead.
- Remove `LinkParameterValidationException#LinkParameterValidationException(Throwable)`. Use `LinkParameterValidationException#LinkParameterValidationException(String, Throwable)` instead.
- Remove `org.iglooproject.wicket.more.link.descriptor.parameter.validator. PermissionLinkParameterValidator.java`. Use `IValidatorState#permission(IModel, String, String...)` instead.
- Remove `org.iglooproject.wicket.more.link.descriptor.parameter.validator. PredicateLinkParameterValidator.java`. Use `IValidatorState#validator(Condition)` instead.
- Remove `AbstractDynamicBookmarkableLink#setAutoHideIfInvalid(boolean)`. Use `AbstractDynamicBookmarkableLink#hideIfInvalid()` instead.
- Remove `DynamicImage#setAutoHideIfInvalid(boolean)`. Use `DynamicImage#hideIfInvalid()` instead.

ReferenceData

Remove old `GenericListItem` and everything related. Use `*ReferenceData` from now on.

> **Danger:** This is a major breakdown.

- Remove `org.iglooproject.jpa.more.business.generic.model.EnabledFilter.java`
- Remove `org.iglooproject.jpa.more.business.generic.model.GenericListItem.java`
- Remove `org.iglooproject.jpa.more.business.generic.model.GenericLocalizedGenericListItem. java`

- Remove org.iglooproject.jpa.more.business.generic.model.IGenericListItemBindingInterface.java
- Remove org.iglooproject.jpa.more.business.generic.model.search.GenericListItemSort.java
- Remove org.iglooproject.jpa.more.business.generic.query.AbstractGenericListItemHibernateSearchSearc
java
- Remove org.iglooproject.jpa.more.business.generic.query.IGenericListItemSearchQuery.java
- Remove org.iglooproject.jpa.more.business.generic.query.ISimpleGenericListItemSearchQuery.java
- Remove org.iglooproject.jpa.more.business.generic.query.SimpleGenericListItemHibernateSearchSearchQ
java
- Remove org.iglooproject.jpa.more.business.generic.service.GenericListItemServiceImpl.java
- Remove org.iglooproject.jpa.more.business.generic.service.GenericLocalizedGenericListItemServiceImp
java
- Remove org.iglooproject.jpa.more.business.generic.service.IGenericListItemService.java
- Remove org.iglooproject.jpa.more.business.generic.service.IGenericLocalizedGenericListItemService.java
- Remove org.iglooproject.jpa.more.business.generic.util.AbstractGenericListItemComparator.java
- Remove org.iglooproject.jpa.more.business.generic.util.GenericListItemComparator.java
- Remove org.iglooproject.jpa.more.business.generic.dao.GenericListItemDaoImpl.java
- Remove org.iglooproject.jpa.more.business.generic.dao.GenericLocalizedGenericListItemDaoImpl.java
- Remove org.iglooproject.jpa.more.business.generic.dao.IGenericListItemDao.java
- Remove org.iglooproject.jpa.more.business.generic.dao.IGenericLocalizedGenericListItemDao.java
- Remove org.iglooproject.wicket.more.markup.html.model.LocalizedGenericListItemListModel.java
- Remove org.iglooproject.wicket.more.util.convert.converters.HumanReadableLocalizedGenericListItemConverter.java
- Remove org.iglooproject.wicket.more.util.convert.converters.HumanReadableLocalizedTextConverter.java
- Remove org.iglooproject.wicket.more.markup.html.basic.GenericListItemConverter.java
- Remove org.iglooproject.wicket.more.markup.html.basic.GenericListItemHideableLabel.java
- Remove org.iglooproject.wicket.more.markup.html.basic.GenericListItemLabel.java
- Remove org.iglooproject.wicket.more.markup.html.basic.GenericListItemListLabel.java
- Remove org.iglooproject.wicket.more.markup.html.basic.LocalizedGenericListItemLabel.java

- Remove org.iglooproject.wicket.more.markup.html.select2.DefaultLocalizedGenericListItemChoiceRender
java
- Remove org.iglooproject.basicapp.core.business.common.model.LocalizedGenericListItem.
java
- Remove org.iglooproject.basicapp.core.business.common.model.IHierarchicalListItem.
java
- Remove org.iglooproject.basicapp.core.business.common.model.comparator.
LocalizedGenericListItemComparator.java
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
AbstractGenericListItemListPanel.java
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
AbstractGenericListItemListPanel.html
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
SimpleGenericListItemListPanel.java
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
SimpleGenericListItemSearchPanel.java
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
SimpleGenericListItemListPanel.html
- Remove org.iglooproject.basicapp.web.application.referencedata.component.
SimpleGenericListItemSearchPanel.html
- Remove org.iglooproject.basicapp.web.application.referencedata.form.
AbstractGenericListItemPopup.java
- Remove org.iglooproject.basicapp.web.application.referencedata.form.
SimpleGenericListItemPopup.java
- Remove org.iglooproject.basicapp.web.application.referencedata.form.
AbstractGenericListItemPopup.html
- Remove org.iglooproject.basicapp.web.application.referencedata.form.
SimpleGenericListItemPopup.html
- Remove org.iglooproject.basicapp.web.application.referencedata.model.
AbstractGenericListItemDataProvider.java
- Remove org.iglooproject.basicapp.web.application.referencedata.model.
SimpleGenericListItemDataProvider.java

Select2

Danger: This is a major breakdown.

We dropped the dependency ivaynberg/wicket-select2 and now use select2-parent from wicektstuff to get the lastest v4 of Select2.

- Remove deprecated org.iglooproject.wicket.more.markup.html.select2.
AbstractSelect2MultipleChoice
- Remove deprecated org.iglooproject.wicket.more.markup.html.select2.
Select2CollectionMultipleChoice

- Remove deprecated `org.iglooproject.wicket.more.markup.html.select2.Select2ListMultipleChoice`
- Remove deprecated `org.iglooproject.wicket.more.markup.html.select2.Select2SortedSetMultipleChoice`

- Remove `org.iglooproject.wicket.more.markup.html.select2.AjaxSearchResourceReference`. Not the same API anymore, rebuild it from scratch if necessary.
- Remove `org.iglooproject.wicket.more.markup.html.select2.AbstractGenericEntitySelect2AjaxResourceRef` Not the same API anymore, rebuild it from scratch if necessary.

- Remove `select2.less`

- Remove `Select2Utils.CSS_DROP_MINI`;
- Remove `Select2Utils.String CSS_DROP_SMALL`;
- Remove `Select2Utils.String CSS_DROP_MEDIUM`;
- Remove `Select2Utils.CSS_DROP_LARGE`;
- Remove `Select2Utils.CSS_DROP_XLARGE`;
- Remove `Select2Utils.CSS_DROP_XXLARGE`;
- Remove `Select2Utils.CSS_DROP_XXXLARGE`

Misc cleaning

Parameter

- Remove old deprecated fields from `Parameter` entity, and everything related such as service methods. Use `PropertyId` and `PropertyService` from now on.
- Remove `org.iglooproject.spring.config.AbstractConfigurer.java`. Use `PropertyService` instead.
- Remove `org.iglooproject.spring.config.CoreConfigurer.java`. Use `PropertyService` instead.

Old table factory

- Remove `org.iglooproject.wicket.more.markup.html.list.AbstractGenericItemListActionButtons.java`
- Remove `org.iglooproject.wicket.more.markup.html.list.AbstractGenericItemListPanel.java`
- Remove `org.iglooproject.wicket.more.markup.html.list.GenericEntityListItemModel.java`
- Remove `org.iglooproject.wicket.more.markup.html.list.GenericPortfolioPanel.java`
- Remove `org.iglooproject.wicket.more.markup.html.list.PageablePortfolioPanel.java`
- Remove `org.iglooproject.wicket.more.markup.html.list.AbstractGenericItemListActionButtons_bs3.html`
- Remove `org.iglooproject.wicket.more.markup.html.list.AbstractGenericItemListActionButtons.html`
- Remove `org.iglooproject.wicket.more.markup.html.list.AbstractGenericItemListPanel.html`
- Remove `org.iglooproject.wicket.more.markup.html.list.GenericPortfolioPanel_bs3.html`
- Remove `org.iglooproject.wicket.more.markup.html.list.GenericPortfolioPanel.html`
- Remove `org.iglooproject.wicket.more.markup.html.list.PageablePortfolioPanel.html`

Use `DataTableBuilder` or make a custom tables from scratch.

Hibernate JPA

- Deprecated custom analyzer `HibernateSearchAnalyzer.TEXT_SORT`. Use Hibernate Search `Normalizer` instead, see `HibernateSearchNormalizer.TEXT`. For example:

```
// Before
@Field(
  name = FIELD_NAME,
  analyzer = @Analyzer(definition = HibernateSearchAnalyzer.TEXT_SORT)
)

// Now
@Field(
  name = FIELD_NAME,
  normalizer = @Normalizer(definition = HibernateSearchNormalizer.TEXT)
)
```

- Rename `GenericUser.USER_NAME_SORT_FIELD_NAME` to `GenericUser.USERNAME_SORT`.
- Rename `GenericSimpleUser.FIRST_NAME_SORT_FIELD_NAME` to `GenericSimpleUser.FIRST_NAME_SORT`.
- Rename `GenericSimpleUser.LAST_NAME_SORT_FIELD_NAME` to `GenericSimpleUser.LAST_NAME_SORT`.
- Rename `QueuedTaskHolder.NAME_SORT_FIELD_NAME` to `QueuedTaskHolder.NAME_SORT`.

- Remove SortNull from ISort. Use NullSortValue instead.
- Remove SortUtils#luceneStringSortField(ISort<SortField>, SortOrder, String, SortNull). Use SortUtils#luceneStringSortField(ISort, SortOrder, String, NullSortValue) instead.

- Remove GenericEntityReference#getEntityClass(). Use GenericEntityReference#getType() instead.
- Remove GenericEntityReference#getEntityId(). Use GenericEntityReference#getId() instead.

- Remove org.iglooproject.jpa.hibernate.usertype.AbstractMaterializedStringValue.java. Use AbstractMaterializedPrimitiveValue instead.

- Remove IGenericEntityDao#getEnti.wicket.more.markup.repeater.dataty(Class, K). Use IGenericEntityDao#getById(Class, Serializable) instead.
- Remove IGenericEntityService#getEntity(Class, K). Use IGenericEntityService#getById(Class, Serializable) instead.
- Update IGenericEntityService#save(E) not public anymore. Use specific method if you want to save entities without going through createEntity method.

- Remove old criteria query references from JpaDaoSupport. Use QueryDSL instead. - buildTypedQuery(CriteriaQuery criteria, Integer limit, Integer offset) - filterCriteriaQuery(CriteriaQuery, Expression) - rootCriteriaQuery(CriteriaBuilder, CriteriaQuery, Class) - getEntityByField(Class, SingularAttribute, V) - getEntityByFieldIgnoreCase(Class clazz, SingularAttribute, String) - listEntity(Class, Expression, Integer, Integer, Order...) - listEntity(Class, Expression) - listEntityByField(Class, SingularAttribute, V) - countEntityByField(Class, SingularAttribute, V) - countEntity(Class, Expression)
- Remove old query methods from IHibernateSearchDao, IHibernateSearchService, IGenericUserService, IGenericUserGroupService. Implement your own search query instead, either through a custom DAO or through ISearchQuery<T, S> as defined in igloo-component-jpa-more. See in particular AbstractHibernateSearchSearchQuery<T, S>.
- Remove AbstractHibernateSearchSearchQuery#getAnalyzer(). Use AbstractHibernateSearchSearchQuery#getDefaultAnalyzer() instead.

- Remove `Expressions2#map(Map, JPQLQuery, Expression, Expression)`. Use `map.putAll(query.transform(GroupBy2.transformer(GroupBy.map(key, value))))` instead.
- Remove `Expressions2#mapToTable(JPQLQuery, Expression, Comparator, Expression, Comparator, Expression)`. Use `query.transform(GroupBy2.transformer(GroupBy2.sortedTable(row, column, value, rowComparator, columnComparator)))` instead.
- Remove `Expressions2#mapToTable(Table, JPQLQuery, Expression, Expression, Expression)`. Use `table.putAll(query.transform(GroupBy2.transformer(GroupBy2.table(row, column, value))))` instead.
- Remove `Expressions2#mapToTable(Table, JPQLQuery, Expression, Expression, Expression)`. Use `table.putAll(query.transform(GroupBy2.transformer(GroupBy2.table(row, column, value))))` instead.

- `GenericUser.java` attribute `userName` has been renamed to `username`. The following script should handle this update :

```
#! /bin/bash

while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Q.userName(/.username(/g
s/\\\QgetUserName(/getUsername(/g
s/\\\QgetByUserName(/getByUsername(/g
s/\\\QsetUserName(/setUsername(/g
s/\\\Q.getByUserNameCaseInsensitive(/.getByUsernameCaseInsensitive(/g
s/\\\Qorg.iglooproject.jpa.security.service.AuthenticationUserNameComparison/org.
↪iglooproject.jpa.security.service.AuthenticationUsernameComparison/g
s/\\\QAuthenticationUserNameComparison/AuthenticationUsernameComparison/g
s/\\\QauthenticationUserNameComparison/authenticationUsernameComparison/g
s/\\\QsetAuthenticationUserNameComparison/setAuthenticationUsernameComparison/g
EOF
```

Import & Export

- Update `AbstractExcelTableExport#getLocalizedLabel(String)` to `AbstractExcelTableExport#localize(String)`.
- Remove `AbstractSimpleExcelTableExport#getLocalizedLabel(String)`. Use `AbstractSimpleExcelTableExport#localize(String)` instead.
- Remove `AbstractExcelTableExport#addHeadersToSheet(Sheet, int, Map)`. Use `AbstractExcelTableExport#addHeadersToSheet(Sheet, int, Collection)` instead.
- Remove `AbstractExcelTableExport#finalizeSheet(Sheet, Map)`. Use `AbstractExcelTableExport#finalizeSheet(Sheet, Collection)` instead.

- Remove `AbstractExcelTableExport#finalizeSheet(Sheet, Map, boolean)`. Use `AbstractExcelTableExport#finalizeSheet(Sheet, Collection, boolean)` instead.
- Remove `AbstractExcelTableExport#resizeMergedColumns(Sheet, Map)`. Use `AbstractExcelTableExport#resizeMergedColumns(Sheet, Collection)` instead.
- Remove `org.iglooproject.imports.table.common.event.SimpleTableImportEventHandler.java`. Use `LoggerTableImportEventHandler` instead.
- Remove `TableImportLocation#getSheetName()`. Use `TableImportLocation#getTableName()` instead.
- Remove `AbstractTableImportColumnSet#missingValue(String)`. Use `AbstractTableImportColumnSet#error(String, Object...)` instead.

Notification

- Remove `INotificationBuilderToState#to(String...)`. Use `INotificationBuilderToState#toAddress(String, String...)` instead.
- Remove `INotificationBuilderBuildState#cc(String...)`. Use `INotificationBuilderBuildState#ccAddress(Str String...)` instead.
- Remove `INotificationBuilderBuildState#bcc(String...)`. Use `INotificationBuilderBuildState#bccAddress(S String...)` instead.
- Remove `INotificationBuilderBuildState#subject(String, String)`. Use `INotificationBuilderBuildState#subjectPrefix(String)` and then `INotificationBuilderBuildState#subject(String)` instead.
- Remove `INotificationBuilderSendState#htmlBody(String)`. Use `INotificationBuilderBuildState#content(INotificationContentDescriptor)` instead.
- Remove `INotificationBuilderSendState#htmlBody(String, Locale)`. Use `INotificationBuilderBuildState#content(INotificationContentDescriptor)` instead.

A wicket default HTML notification variant can now be registered :

```
// Before
IHtmlNotificationCssService service = super.htmlNotificationCssService();
service.registerStyles(DEFAULT_NOTIFICATION_VARIATION,␣
↪NotificationLessCssResourceReference.get());

// Now
IHtmlNotificationCssService service = super.htmlNotificationCssService();
service.registerDefaultStyles(NotificationScssResourceReference.get());
```

Security

- Remove `org.iglooproject.jpa.security.service.IGenericEntityPermissionEvaluator.java`. Use `IGenericPermissionEvaluator` instead.

Lambda and functional

- **Remove org.iglooproject.commons.util.functional.SerializablePredicate.**
  Use org.iglooproject.functional.SerializablePredicate2 instead.
- **Remove org.iglooproject.commons.util.functional.SerializableFunction.**
  Use org.iglooproject.functional.SerializableFunction2 instead.
- **Remove org.iglooproject.commons.util.functional.SerializableSupplier.**
  Use org.iglooproject.functional.SerializableSupplier2 instead.
- Remove org.iglooproject.commons.util.functional.AbstractSerializablePredicate.java. Use SerializablePredicate2 instead.
- Remove org.iglooproject.wicket.more.util.functional.AbstractDetachablePredicate.java. Use Condition instead.
- Remove org.iglooproject.wicket.more.util.functional.DetachablePredicate.java. Use Condition instead.
- Remove Suppliers2#constant(T). Use Suppliers2.ofInstance(T) instead.

Properties resources keys

- Change console.signIn.button to console.signIn.action.signIn
- Change console.authentication.originalAuthentication.help to authentication.originalAuthentication.help
- Change signIn.button to signIn.action.signIn
- Change common.propertyId.actions.edit to common.propertyId.action.edit
- Change common.propertyId.actions.edit.title to common.propertyId.action.edit.title
- Change common.propertyId.actions.edit.success to common.propertyId.action.edit.success
- Change common.deleteConfirmation to common.action.delete.confirm.content
- Change common.deleteConfirmation.object to common.action.delete.confirm.content.object
- change common.confirmTitle to common.action.confirm.title
- Change common.save to common.action.save
- Change common.confirm to common.action.confirm
- Change common.applyFilters to common.action.filter
- Change common.emptyList to common.list.empty
- Change common.emptyField to common.field.empty

- Remove common.item.tableRow.edit
- Remove common.item.tableRow.delete
- Remove common.item.tableRow.cancel
- Remove common.item.tableRow.save

- Remove `common.item.tableRow.add`
- Remove `common.portfolio.action.viewDetails`
- Remove `common.itemList.action.edit`
- Remove `common.itemList.action.delete`
- Remove `common.editPopup.title`
- Remove `common.deletedItem`
- Remove `common.delete.success`
- Remove `common.delete.error`
- Remove `common.logout.tooltip`

DataUpgrade

DataUpgrade are no longer stored as a `parameter/property`. A new entity `DataUpgradeRecord` is created.

To prepare a migration :

- ensure that all your upgrade are done
- create new `DataUpgradeRecord` table :

```
CREATE SEQUENCE DataUpgradeRecord_id_seq start 1 increment 1;
CREATE TABLE DataUpgradeRecord (id int8 NOT NULL, autoPerform boolean NOT NULL, done␣
↪boolean NOT NULL, executionDate timestamp, name text NOT NULL, primary key (id));
CREATE TABLE DataUpgradeRecord ADD constraint UK_6q54k3x0axoc3n8ns55emwiev UNIQUE (name);
```

With this migration plan; you'll keep your old upgrade information in `parameter` table. New DatabaseUpgradeRecord will be stored in the dedicated new table.

> **Warning:** Beware that old DataUpgrade are going to be listed as runnable ones in administration console, as `DatabaseUpgradeRecord` is empty and entries in `parameter` ignored. Either drop your old upgrades in `DataUpgradeManagerImpl.listDataUpgrades()` or migrate `parameter` entries to `DataUpgradeRecord`.

The following script should handle data upgrade migration to `DataUpgradeRecord` :

```
INSERT INTO DataUpgradeRecord (id, autoperform, done, executiondate, name)
    SELECT nextval('dataupgraderecord_id_seq'), false, stringvalue::boolean, now(),␣
↪regexp_replace(name, 'dataUpgrade.', '')
    FROM parameter WHERE name ~ '^dataUpgrade\..*(?<!\.autoperform)$';
```

Once you have checked that DataUpgradeRecord contains every previously executed data upgrade, you can remove old values from the `parameter` table.

Lucene - French analyzer

Due to code refactor to eliminate duplicate code, some classes are renamed.

```
#! /bin/bash

while read line; do
find \( -name "*.java" -o -name "web.xml" \) -exec perl -p -i -e "${line}" {} +
done <<EOF
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemFilterFactory@org.
↪iglooproject.lucene.analysis.french.CoreFrenchMinimalStemFilterFactory@g
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemmer@org.iglooproject.
↪lucene.analysis.french.CoreFrenchMinimalStemmer@g
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemFilter@org.
↪iglooproject.lucene.analysis.french.CoreFrenchMinimalStemFilter@g
EOF
```

Warning: `org.iglooproject.spring.util.lucene.search.LuceneUtils.toFilterRangeQuery(...)` is modified to throw an exception when min and max are null (previously, it silently returned `null`).

SLF4JLoggingListener

Renamed from `org.iglooproject.commons.util.logging.SLF4JLoggingListener` to `org.iglooproject.slf4j.jul.bridge.SLF4JLoggingListener`.

```
#! /bin/bash

while read line; do
find \( -name "*.java" -o -name "web.xml" \) -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.commons.util.logging.SLF4JLoggingListener/org.iglooproject.slf4j.
↪jul.bridge.SLF4JLoggingListener/g
EOF
```

Commons split

igloo-component-commons is split in several package.

```
#! /bin/bash

while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s@\\\Qorg.iglooproject.commons.util.FileUtils@org.iglooproject.commons.io.FileUtils@g
s@\\\Qorg.iglooproject.commons.util.registry.TFileRegistry@org.iglooproject.truezip.
↪registry.TFileRegistry@g
EOF
```

Javascript libraries

Here is the list of removed obsolete javascript libraries, code is available in git history :

- jstree

ICloneable<T>

`Object.clone` use is discouraged. This interface, barely used, is removed.

Tests infrastructure

Test classes has been moved to separate modules. To keep using this classes, your core `pom.xml` should have the following dependencies :

```
<dependency>
      <groupId>org.iglooproject.dependencies</groupId>
      <artifactId>igloo-dependency-test</artifactId>
      <scope>test</scope>
      <type>pom</type>
</dependency>

<dependency>
      <groupId>org.iglooproject.components</groupId>
      <artifactId>igloo-component-jpa-test</artifactId>
      <version>${igloo.version}</version>
      <scope>test</scope>
</dependency>
```

Previous test dependencies (`junit`, `spring-test`) are now provided by the new modules and can be removed `pom.xml` if needed.

64.2.2 Others

Renamed configuration

`hibernate.defaultSchema` is renamed `db.schema` as it is used by flyway. You need to rename it in your `configuration.properties`.

Updated

Infinispan

With jgroups 4.0, Infinispan don't use any longer oob and internal threads. You have to remove all `internal_thread_pool.*`, `oob_thread_pool.*` and `thread_pool.queue_enabled` settings in your jgroups configuration (`*jgroup*.xml` files).

See http://planet.jboss.org/post/removing_thread_pools_in_jgroups_4_0

All references to `org.jgroups.Address` must be replaced with `org.iglooproject.infinispan.model.AddressWrapper` (as Address is no longer `Serializable`, AddressWrapper handles Serialization).

Mockito

Mockito is upgraded to 2.x version. You way need to rewrite some tests in your projects.

We recommend to exclude mockito 2.x dependency as a first step and check your test results without any rewrite, then to update mockito once all your tests are fixed.

No longer supported

JDK 7

JDK 7 support is removed as planned.

Tomcat 7

Tomcat 7 is no longer supported. Servlet 3.1 is targetted, and so Tomcat 8.5 is needed.

JFreeChart

Dependency management entry removed from parent pom. If your project use JFreeChart, declare dependency inside your project's pom.

Maven

- unused property `igloo.gson.version` is removed

Password encoding

Note: **CoreLowerCaseShaPasswordEncoder**, **Md5PasswordEncoder**, **CoreShaPasswordEncoder** removed (Spring Security 5 update related)

To know if your application is compatible with new password encoding, please check stored passwords. If your encoded passwords all start with `$2a$` (bcrypt marker), your application may be compatible.

If this is the case, **you need to update your hashed password** by prefixing it `{bcrypt}` with prefix as the new `PasswordEncoder` is a delegating one, that choose the correct PasswordEncoder based on this prefix. This PasswordEncoder uses `bcrypt` to hash new passwords.

If not you need to write your own password encoder based on code from previous versions. Please take care of case insensitive check if **CoreLowerCaseShaPasswordEncoder** was used.

This page https://en.wikipedia.org/wiki/Bcrypt, your application configuration, and hashed password patterns may allow you to identify password encoder behavior and identify needed use-cases.

If you upgrade your application, you should take into consideration to handle all new passwords with modern hashing (use encoded password prefix to switch encoder behavior).

You should also consider this paragraph from Spring documentation

Property `security.passwordSalt` and method `DefaultJpaSecurityConfig.getPasswordSalt()` are removed.

YUI Compressor

YUI Compressor (maven plugin, minification at build time) is removed as it was no longer used to provide minification (handled internally at runtime by wicket).

I you use YUI Compressor, you need to include your maven plugin configuration inside your project.

maven-release-plugin

Removed. Use jgitflow or reconfigure release plugin in your project.

tomcat-jdbc

We use HikariCP as database pool provider. tomcat-jdbc is no longer used. Switch to HikariCP.

As tomcat-jdbc is a provided dependency (included in tomcat), this may not affect your web-application. It may affect your tests or main scripts: if this is the case, you need to ensure that tomcat-jdbc dependency connfiguration is correct.

Joda-Time

Joda-Time is removed from dependency; you can continue to use it by re-adding this dependency to your project.

Session - redirectUrl

Igloo mechanisms to handle post-login redirectUrl are completely removed. You should use easily spring-security based one.

Removed methods are, on `AbstractCoreSession`:

- signOutWithoutCleaningUpRedirectUrl
- registerRedirectUrl
- getRedirectUrl
- consumeRedirectUrl
- registerRedirectPageLinkDescriptor
- getRedirectPageLinkDescriptor

If you use these methods, you should check how you handle your login success. If you use LoginSuccessPage (wicket-more), then Spring-Security redirect should work.

Here are the use-cases to check that there are no regressions on your application:

- login to default home page; logout
- visit a protected page; you should be redirect to it after login; logout
- login with a wrong password; check error message
- visit a protected and forbidden page; you should be redirected to default home page with an error message

Javascript

The following dependencies are split from igloo-component-wicket-more and marked as optional dependencies (you need to add them manually in your projects to use them):

- jquery.json
- jquery.carouFredSel
- jquery.fancybox

New features

Test tooling

A new **igloo-dependency-test** provides basic dependencies for tests. You can use this dependency in place of junit, mockito, spring-test, … dependencies.

org.iglooproject.jpa.junit.AbstractTestCase and **org.iglooproject.jpa.EntityManagerExecutionListener** are moved in a new **igloo-dependency-jpa-test** module. If you want to use them, add this new dependency with scope test, and fix your imports.

Bootstrap 4

Bootstrap 4 is available and used by basic-application archetype.

Codebase for bootstrap 3 is still available and unchanged.

JNDI Datasource

Configuration of JNDI can be done with a configuration switch. Please see *Use a JDNI datasource*

Migration script

The following script intends to help initiating migration by dealing with dump replacements.

> **Warning:** As is, the script move **from version 0.14 to 1.1**. Please replace version numbers in the first block to adjust to your version.

```
#! /bin/bash

## POM renaming
while read line; do
find . -type f -name "pom.xml" -exec sed -i "${line}" {} +
done <<EOF
s@Nexus OWSI Core@Nexus Igloo@g
s@fr\.openwide\.core@org.iglooproject@g
s@owsi-core@igloo@g
s@<version>0\.14@<version>1.1.0@g
s@<igloo.version>0\.14@<igloo.version>1.1.0@g
```

(continues on next page)

(continued from previous page)

```
s@projects\.openwide\.fr/services/nexus/content/repositories/igloo-snapshots@nexus.tools.
↪kobalt.fr/repository/igloo-snapshots/@g
s@projects\.openwide\.fr/services/nexus/content/repositories/igloo@nexus.tools.kobalt.fr/
↪repository/igloo/@g
EOF

## Package renaming
find . -type f \( -name "*.xml" -o -name "*.java" -o -name "*.properties" \)  -exec sed -
↪i 's@fr\.openwide\.core@org.iglooproject@g' {} +

## GenericListItem -> GenericBasicReferenceData
while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.jpa.more.business.generic.model.GenericListItem/org.iglooproject.
↪jpa.more.business.referencedata.model.GenericBasicReferenceData/g
s/\\\Qorg.iglooproject.jpa.more.business.generic.query.
↪AbstractGenericListItemHibernateSearchSearchQueryImpl/org.iglooproject.jpa.more.
↪business.referencedata.search.GenericReferenceDataSearchQueryImpl/g
s/\\\QAbstractGenericListItemHibernateSearchSearchQueryImpl/
↪GenericReferenceDataSearchQueryImpl/g
s/\\\Qorg.iglooproject.jpa.more.business.generic.query.IGenericListItemSearchQuery/org.
↪iglooproject.jpa.more.business.referencedata.search.
↪IGenericBasicReferenceDataSearchQuery/g
s/\\\QIGenericListItemSearchQuery/IGenericBasicReferenceDataSearchQuery/g
s/\\\Qorg.iglooproject.jpa.more.business.generic.service.IGenericListItemService/org.
↪iglooproject.jpa.more.business.referencedata.service.IGenericBasicReferenceDataService/
↪g
s/\\\QIGenericListItemService/IGenericBasicReferenceDataService/g
s/\\\Qorg.iglooproject.wicket.more.rendering.GenericListItemRenderer/org.iglooproject.
↪wicket.more.rendering.GenericBasicReferenceDataRenderer/g
s/\\\QGenericListItemRenderer/GenericBasicReferenceDataRenderer/g
s/\\\QGenericListItem/GenericBasicReferenceData/g
EOF

## Class moved
while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.jquery.plugins.bootstrap.
↪confirm.component.AjaxConfirmLink/org.iglooproject.wicket.more.markup.html.template.js.
↪bootstrap.confirm.component.AjaxConfirmLink/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.jquery.plugins.bootstrap.
↪modal.behavior.AjaxModalOpenBehavior/org.iglooproject.wicket.more.markup.html.template.
↪js.bootstrap.modal.behavior.AjaxModalOpenBehavior/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.jquery.plugins.bootstrap.
↪modal.component.AbstractAjaxModalPopupPanel/org.iglooproject.wicket.more.markup.html.
↪template.js.bootstrap.modal.component.AbstractAjaxModalPopupPanel/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.jquery.plugins.bootstrap.
↪modal.component.DelegatedMarkupPanel/org.iglooproject.wicket.more.markup.html.template.
↪js.bootstrap.modal.component.DelegatedMarkupPanel/g
s/\\\Qorg.iglooproject.jpa.more.business.generic.model.EnabledFilter/org.iglooproject.
```

(continues on next page)

(continued from previous page)

```
↪jpa.more.business.generic.model.search.EnabledFilter/g
s/\\\Qorg.iglooproject.spring.config.ExtendedTestApplicationContextInitializer/org.
↪iglooproject.config.bootstrap.spring.ExtendedTestApplicationContextInitializer/g
s/\\\Qorg.retzlaff.select2.Select2Behavior/org.wicketstuff.select2.Select2Behavior/g
s/\\\Qorg.iglooproject.wicket.more.common.WorkInProgressPopup/org.iglooproject.wicket.
↪more.common.component.WorkInProgressPopup/g
EOF

## Various renaming
while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\QOWSI_CORE_VERSION/IGLOO_VERSION/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.action.AbstractAjaxAction/org.
↪iglooproject.wicket.more.markup.html.action.IAjaxAction/g
s/\\\QAbstractAjaxAction/IAjaxAction/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.factory.AbstractDetachableFactory/org.
↪iglooproject.wicket.more.markup.html.factory.IDetachableFactory/g
s/\\\QAbstractDetachableFactory/IDetachableFactory/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.action.AbstractOneParameterAjaxAction/org.
↪iglooproject.wicket.more.markup.html.action.IOneParameterAjaxAction/g
s/\\\QAbstractOneParameterAjaxAction/IOneParameterAjaxAction/g
s/\\\Qorg.iglooproject.jpa.junit.AbstractTestCase/org.iglooproject.test.jpa.junit.
↪AbstractTestCase/g
s/\\\QFormPanelMode/FormMode/g
s/\\\Qorg.apache.wicket.ajax.AjaxRequestTarget.AbstractListener/org.apache.wicket.ajax.
↪AjaxRequestTarget.IListener/g
s/\\\QAbstractListener/IListener/g
s/\\\Qorg.apache.wicket.model.AbstractReadOnlyModel/org.apache.wicket.model.IModel/g
s/\\\QAbstractReadOnlyModel/IModel/g
s/\\\Qorg.retzlaff.select2.Select2Settings/org.wicketstuff.select2.Settings/g
s/\\\QSelect2Settings/Settings/g
s/\\\QfillSettings/fillSelect2Settings/g
EOF

## User modification
while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.jpa.security.service.AuthenticationUserNameComparison/org.
↪iglooproject.jpa.security.service.AuthenticationUsernameComparison/g
s/\\\QAuthenticationUserNameComparison/AuthenticationUsernameComparison/g
s/\\\QauthenticationUserNameComparison/authenticationUsernameComparison/g
s/\\\QsetAuthenticationUserNameComparison/setAuthenticationUsernameComparison/g
s/\\\QgetUserName(/getUsername(/g
s/\\\QgetByUserName(/getByUsername(/g
s/\\\QsetUserName(/setUsername(/g
s/\\\Q.userName(/.username(/g
s/\\\Q.getByUserNameCaseInsensitive(/.getByUsernameCaseInsensitive(/g
s/\\\QLAST_NAME_SORT_FIELD_NAME/LAST_NAME_SORT/g
s/\\\QFIRST_NAME_SORT_FIELD_NAME/FIRST_NAME_SORT/g
s/\\\QUSER_NAME_SORT_FIELD_NAME/USERNAME_SORT/g
```

(continues on next page)

(continued from previous page)

```
EOF

## Migration Bindgen
while read line; do
find .  -type f -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.commons.util.binding.AbstractCoreBinding/org.iglooproject.commons.
↪util.binding.ICoreBinding/g
s/\\\QAbstractCoreBinding/ICoreBinding/g
s/\\\Qorg.bindgen.binding.AbstractBinding/org.bindgen.BindingRoot/g
s/\\\QAbstractBinding/BindingRoot/g
EOF

## Configuration ciManagement
while read line; do
find .  -type f \( -name "*.java" -o -name "web.xml" \) -exec sed -i "${line}" {} \;
done <<EOF
s/org.iglooproject.spring.config.AbstractExtendedApplicationContextInitializer/org.
↪iglooproject.config.bootstrap.spring.AbstractExtendedApplicationContextInitializer/g
s/org.iglooproject.spring.config.annotation.
↪ApplicationConfigurerBeanFactoryPostProcessor/org.iglooproject.config.bootstrap.spring.
↪ApplicationConfigurerBeanFactoryPostProcessor/g
s/org.iglooproject.spring.config.ExtendedApplicationContextInitializer/org.iglooproject.
↪config.bootstrap.spring.ExtendedApplicationContextInitializer/g
s/org.iglooproject.spring.config.spring.annotation.ApplicationDescription/org.
↪iglooproject.config.bootstrap.spring.annotations.ApplicationDescription/g
s/org.iglooproject.spring.config.spring.annotation.ConfigurationLocations/org.
↪iglooproject.config.bootstrap.spring.annotations.ConfigurationLocations/g
EOF

## Property config
find . -type f \( -name "*PropertyRegistryConfig.java" -o -name
↪"*CoreApplicationPropertyConfig.java" \) \
 -exec perl -p -i -e 's/\Qprotected void register(/public void register(/g' {} \;

#! /bin/bash

while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManager/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.statement.BootstrapModalManager/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManagerStatement/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪js.bootstrap.modal.statement.BootstrapModalManagerStatement/g
s/\\\Qorg.iglooproject.wicket.more.WicketBootstrapPackage/org.iglooproject.wicket.
↪bootstrap3.WicketBootstrapPackage/g
s/\\\Qorg.iglooproject.wicket.more.application.WicketBootstrapModule/org.iglooproject.
↪wicket.bootstrap3.application.WicketBootstrapModule/g
s/\\\Qorg.iglooproject.wicket.more.config.spring.WicketBootstrapServiceConfig/org.
↪iglooproject.wicket.bootstrap3.config.spring.WicketBootstrapServiceConfig/g
```

(continues on next page)

(continued from previous page)

```
s/\\\Qorg.iglooproject.wicket.more.config.spring.AbstractBootstrapWebappConfig/org.
↪iglooproject.wicket.bootstrap3.config.spring.AbstractBootstrapWebappConfig/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleSignInPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleSignInPage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleAccessDeniedPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleAccessDeniedPage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleLoginFailurePage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleLoginFailurePage/g
s/\\\Qorg.iglooproject.wicket.more.console.navigation.page.ConsoleLoginSuccessPage/org.
↪iglooproject.wicket.bootstrap3.console.navigation.page.ConsoleLoginSuccessPage/g
s/\\\Qorg.iglooproject.wicket.more.console.resources.CoreWicketConsoleResources/org.
↪iglooproject.wicket.bootstrap3.console.resources.CoreWicketConsoleResources/g
s/\\\Qorg.iglooproject.wicket.more.console.common.util.LinkUtils/org.iglooproject.wicket.
↪bootstrap3.console.common.util.LinkUtils/g
s/\\\Qorg.iglooproject.wicket.more.console.common.component.
↪JavaClassesListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.common.
↪component.JavaClassesListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.common.component.PropertyIdListPanel/org.
↪iglooproject.wicket.bootstrap3.console.common.component.PropertyIdListPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.common.form.PropertyIdEditPopup/org.
↪iglooproject.wicket.bootstrap3.console.common.form.PropertyIdEditPopup/g
s/\\\Qorg.iglooproject.wicket.more.console.template.ConsoleTemplate/org.iglooproject.
↪wicket.bootstrap3.console.template.ConsoleTemplate/g
s/\\\Qorg.iglooproject.wicket.more.console.template.ConsoleConfiguration/org.
↪iglooproject.wicket.bootstrap3.console.template.ConsoleConfiguration/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.
↪ConsoleLessCssResourceReference/org.iglooproject.wicket.bootstrap3.console.template.
↪style.ConsoleLessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.
↪ConsoleSignInLessCssResourceReference/org.iglooproject.wicket.bootstrap3.console.
↪template.style.ConsoleSignInLessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.console.template.style.CoreConsoleCssScope/org.
↪iglooproject.wicket.bootstrap3.console.template.style.CoreConsoleCssScope/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.page.
↪ConsoleMaintenanceEhCachePage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.page.ConsoleMaintenanceEhCachePage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheProgressBarComponent/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheProgressBarComponent/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheCachePortfolioPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheCachePortfolioPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.ehcache.component.
↪EhCacheCacheModificationPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪ehcache.component.EhCacheCacheModificationPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.file.page.
↪ConsoleMaintenanceFilePage/org.iglooproject.wicket.bootstrap3.console.maintenance.file.
↪page.ConsoleMaintenanceFilePage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.search.page.
↪ConsoleMaintenanceSearchPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪search.page.ConsoleMaintenanceSearchPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.renderer.
```

(continues on next page)

(continued from previous page)

```
↪QueueTaskRenderer/org.iglooproject.wicket.bootstrap3.console.maintenance.queuemanager.
↪renderer.QueueTaskRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.renderer.
↪QueueManagerRenderer/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪queuemanager.renderer.QueueManagerRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.page.
↪ConsoleMaintenanceQueueManagerPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.queuemanager.page.ConsoleMaintenanceQueueManagerPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.queuemanager.component.
↪ConsoleMaintenanceQueueManagerNodePanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.queuemanager.component.ConsoleMaintenanceQueueManagerNodePanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.authentication.page.
↪ConsoleMaintenanceAuthenticationPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.authentication.page.ConsoleMaintenanceAuthenticationPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.renderer.INodeRenderer/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.infinispan.renderer.
↪INodeRenderer/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.page.
↪ConsoleMaintenanceInfinispanPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.page.ConsoleMaintenanceInfinispanPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanNodesPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanNodesPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanLocksPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanLocksPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanRolesPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesRequestsPanel/org.iglooproject.wicket.bootstrap3.
↪console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanRolesRequestsPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.component.
↪ConsoleMaintenanceInfinispanClusterPanel/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.component.ConsoleMaintenanceInfinispanClusterPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.form.
↪NodeDropDownSingleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪infinispan.form.NodeDropDownSingleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.infinispan.form.
↪ConsoleMaintenanceInfinispanRoleAssignPopup/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.infinispan.form.ConsoleMaintenanceInfinispanRoleAssignPopup/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.page.
↪ConsoleMaintenanceTaskDescriptionPage/org.iglooproject.wicket.bootstrap3.console.
↪maintenance.task.page.ConsoleMaintenanceTaskDescriptionPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.page.
↪ConsoleMaintenanceTaskListPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.page.ConsoleMaintenanceTaskListPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskTypeListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.task.
↪component.TaskTypeListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
```

(continues on next page)

(continued from previous page)

```
↪TaskQueueIdListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskQueueIdListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskStatusPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskStatusPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskExecutionResultPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.task.
↪component.TaskExecutionResultPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskResultListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskResultListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskFilterPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskFilterPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskStatusListMultipleChoice/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskStatusListMultipleChoice/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.
↪TaskManagerInformationPanel/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪task.component.TaskManagerInformationPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskResultPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskResultPanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.task.component.TaskResultsPanel/
↪org.iglooproject.wicket.bootstrap3.console.maintenance.task.component.TaskResultsPanel/
↪g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.gestion.page.
↪ConsoleMaintenanceGestionPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪gestion.page.ConsoleMaintenanceGestionPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.upgrade.page.
↪ConsoleMaintenanceDonneesPage/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪upgrade.page.ConsoleMaintenanceDonneesPage/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.upgrade.component.
↪DataUpgradePanel/org.iglooproject.wicket.bootstrap3.console.maintenance.upgrade.
↪component.DataUpgradePanel/g
s/\\\Qorg.iglooproject.wicket.more.console.maintenance.template.
↪ConsoleMaintenanceTemplate/org.iglooproject.wicket.bootstrap3.console.maintenance.
↪template.ConsoleMaintenanceTemplate/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.WicketBootstrapComponentsModule/
↪org.iglooproject.wicket.bootstrap3.markup.html.bootstrap.
↪WicketBootstrapComponentsModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.component.BootstrapLabel/org.
↪iglooproject.wicket.bootstrap3.markup.html.bootstrap.component.BootstrapLabel/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.bootstrap.component.BootstrapBadge/org.
↪iglooproject.wicket.bootstrap3.markup.html.bootstrap.component.BootstrapBadge/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.collapse.
↪BootstrapCollapseJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.collapse.BootstrapCollapseJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.dropdown.
↪BootstrapDropdownModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.dropdown.BootstrapDropdownModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.dropdown.
↪BootstrapDropDownJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.dropdown.BootstrapDropDownJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.confirm.
```

(continues on next page)

(continued from previous page)

```
↪BootstrapConfirmModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.confirm.BootstrapConfirmModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.confirm.
↪BootstrapConfirmJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.confirm.BootstrapConfirmJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tooltip.
↪BootstrapTooltipModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.tooltip.BootstrapTooltipModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tooltip.
↪BootstrapTooltipJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.tooltip.BootstrapTooltipJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.popover.
↪BootstrapPopoverJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.popover.BootstrapPopoverJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.popover.
↪BootstrapPopoverModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.popover.BootstrapPopoverModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.alert.
↪BootstrapAlertJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.alert.BootstrapAlertJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.SimpleOptions/org.
↪iglooproject.wicket.bootstrap3.markup.html.template.js.bootstrap.SimpleOptions/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixOptions/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.affix.BootstrapAffixOptions/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixBehavior/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.affix.BootstrapAffixBehavior/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.affix.
↪BootstrapAffixJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.affix.BootstrapAffixJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tab.
↪BootstrapTabModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.tab.BootstrapTabModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.tab.
↪BootstrapTabJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.
↪template.js.bootstrap.tab.BootstrapTabJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.modal.BootstrapModalJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.BootstrapModalModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManager/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.modal.statement.BootstrapModalManager/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.statement.
↪BootstrapModalManagerStatement/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪js.bootstrap.modal.statement.BootstrapModalManagerStatement/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.modal.
↪BootstrapModalManagerJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.
↪markup.html.template.js.bootstrap.modal.
```

(continues on next page)

(continued from previous page)

```
↪BootstrapModalManagerJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.button.
↪BootstrapButtonJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.js.bootstrap.button.BootstrapButtonJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.button.
↪BootstrapButtonModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.button.BootstrapButtonModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.
↪markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyJavaScriptResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.bootstrap.scrollspy.
↪BootstrapScrollSpyModule/org.iglooproject.wicket.bootstrap3.markup.html.template.js.
↪bootstrap.scrollspy.BootstrapScrollSpyModule/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.select2.
↪Select2CssResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪css.select2.Select2CssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.fontawesome.
↪CoreFontAwesomeCssScope/org.iglooproject.wicket.bootstrap3.markup.html.template.css.
↪bootstrap.fontawesome.CoreFontAwesome4CssScope/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.
↪CoreBootstrap3CssScope/org.iglooproject.wicket.bootstrap3.markup.html.template.css.
↪bootstrap.CoreBootstrap3CssScope/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.bootstrap.
↪DefaultBootstrap3LessCssResourceReference/org.iglooproject.wicket.bootstrap3.markup.
↪html.template.css.bootstrap.bootstrap.DefaultBootstrap3LessCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.css.bootstrap.jqueryui.
↪JQueryUiCssResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.template.
↪css.bootstrap.jqueryui.JQueryUiCssResourceReference/g
s/\\\Qorg.iglooproject.wicket.more.markup.html.template.js.respond.
↪RespondJavaScriptResourceReference/org.iglooproject.wicket.bootstrap3.markup.html.
↪template.js.respond.RespondJavaScriptResourceReference/g
EOF

## CoreFrenchMiniamStem
while read line; do
find \( -name "*.java" -o -name "web.xml" \) -exec perl -p -i -e "${line}" {} +
done <<EOF
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemFilterFactory@org.
↪iglooproject.lucene.analysis.french.CoreFrenchMinimalStemFilterFactory@g
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemmer@org.iglooproject.
↪lucene.analysis.french.CoreFrenchMinimalStemmer@g
s@\\\Qorg.iglooproject.jpa.search.analysis.fr.CoreFrenchMinimalStemFilter@org.
↪iglooproject.lucene.analysis.french.CoreFrenchMinimalStemFilter@g
EOF

## SLF4J
while read line; do
find \( -name "*.java" -o -name "web.xml" \) -exec perl -p -i -e "${line}" {} \;
done <<EOF
s/\\\Qorg.iglooproject.commons.util.logging.SLF4JLoggingListener/org.iglooproject.slf4j.
↪jul.bridge.SLF4JLoggingListener/g
```

(continues on next page)

(continued from previous page)

```
EOF

## Igloo commons split
while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s@\\\Qorg.iglooproject.commons.util.FileUtils@org.iglooproject.commons.io.FileUtils@g
s@\\\Qorg.iglooproject.commons.util.registry.TFileRegistry@org.iglooproject.truezip.
↪registry.TFileRegistry@g
EOF

## Wicket 8
while read line; do
find -name "*.java" -exec perl -p -i -e "${line}" {} \;
done <<EOF
s@\\\QonError(AjaxRequestTarget target, Form<?> form)@onError(AjaxRequestTarget target)@g
s@\\\QonSubmit(AjaxRequestTarget target, Form<?> form)@onSubmit(AjaxRequestTarget␣
↪target)@g
s@\\\QonAfterSubmit(AjaxRequestTarget target, Form<?>␣
↪form)@onAfterSubmit(AjaxRequestTarget target)@g
EOF

## Serializable
while read line; do
find -name '*.java' -exec perl -i -pe "$line" {} ';'
done <<EOF
s/\\\Qorg.iglooproject.commons.util.functional/org.iglooproject.functional/g
s/SerializableFunction(?!2)/SerializableFunction2/g
s/SerializablePredicate(?!2)/SerializablePredicate2/g
s/SerializableSupplier(?!2)/SerializableSupplier2/g
EOF

## Hibernate Search Sort
find . -type f -name "*.java" -exec sed -i 's/analyzer = @Analyzer(definition =␣
↪HibernateSearchAnalyzer\.TEXT_SORT)/normalizer = @Normalizer(definition =␣
↪HibernateSearchNormalizer.TEXT)/g' {} +
```

64.3 Migrating to 0.14

This guide aims at helping OWSI-Core users migrate an application based on OWSI-Core 0.13 to OWSI-Core 0.14.

In order to migrate from an older version of OWSI-Core, please refer to *Migrating to 0.13* first.

64.3.1 Java

This version only supports **Java 8**.

64.3.2 External changes (libraries)

Poi

- `HSSFColor.WHITE.index` is now `HSSFColorPredefined.WHITE.getIndex()`.

Spring & Spring Security

- `isTrue(boolean)` from the type Assert is now `isTrue(boolean, String)` with String = the exception message to use if the assertion fails.
- `notNull(boolean)` from the type Assert is now `notNull(boolean, String)` with String = the exception message to use if the assertion fails.

For the upgrade of Spring Security we had to update the schema from `spring-security-4.1.xsd` to `spring-security-4.2.xsd`.

Guava

- `CharMatcher.WHITESPACE` is now `CharMatcher.whitespace()`.

Hibernate

- `session.setFlushMode(FlushMode)` is now `session.setHibernateFlushMode(FlushMode)`.
- `SessionImplementor` class is replaced by `SharedSessionContractImplementor` class.

The AvailableSettings libray now is `org.hibernate.cfg.AvailableSettings` instead of `org.hibernate.jpa.AvailableSettings`.

- `AvailableSettings.SHARED_CACHE_MODE` is now `AvailableSettings.JPA_SHARED_CACHE_MODE`.
- `AvailableSettings.VALIDATION_MODE` is now `AvailableSettings.JPA_SHARED_CACHE_MODE`.

The EmbeddableTypeImpl library is now `org.hibernate.metamodel.internal.EmbeddableTypeImpl` instead of `org.hibernate.jpa.internal.metamodel.EmbeddableTypeImpl`.

The upgrade of hibernate-core forced us to explicitly specify the **default_schema** for the database. Every tables are created in this schema and it is no longer based on the search_path from PostgreSQL configuration.

By default, default_schema = db_user. If you need to change it, you have to add the variable `hibernate.defaultSchema` in `owsi-core-component-jpa.properties` and its value will override the default value.

Class `PostgresqlSequenceStyleGenerator` is renamed `PerTableSequenceStyleGenerator` as it is not postgresql-related; class content is unchanged. If you use it, just retarget the new class.

Hibernate Search

Hibernate Search & Lucene

We have upgraded Hibernate Search to the 5.7.0.Final which is not yet compatible with Lucene 6 but requires at least Lucene 5.5.X so we have upgraded Lucene to the 5.5.4 version.

The utilization of `setBoost(float)` and `getBoost()` directly to a Query is now deprecated. Instead we use the type `BoostQuery` to apply boost.

Configuration

- The `ExplicitJpaConfigurationProvider` class no longer exists, all the configuration is now exclusively provided by the `DefaultJpaConfigurationProvider` class.

Behavior checking

Some structural changes are done so that old applications are not broken. Make sure that expected behavior is still here:

- **Hibernate:** database's sequence is now handled with the *new-style* hibernate configuration. Verify that the sequence are style named *table_pk_seq*. Give a special attention to your specialized configurations:
  - ensure that *hibernate.id.new_generator_mappings=true* (if you do not override this setting, it is fine)
  - custom *@GeneratedValue.strategy()*
  - custom *@GeneratedValue.generator()*
  - custom *@SequenceGenerator*
  - custom *@GenericGenerator*

Hibernate Search & ElasticSearch

You can now choose between Lucene and ElasticSearch for your Hibernate Search requests. In order to do use ElasticSearch, you have first to install ElasticSearch 2.4.

Secondly, you have to specify 3 things in the file **app-core/configuration.properties** :

```
##
## Hibernate search Elasticsearch
##
hibernate.search.elasticsearch.enabled=true
hibernate.search.default.elasticsearch.host=http://127.0.0.1:9310
hibernate.search.default.elasticsearch.index_schema_management_strategy=CREATE
```

You have to set the first line value to `true` to enable ElasticSearch. The second line is the address of your installed ElasticSearch, and finally the third line is schema management strategy.

Lucene and ElasticSearch analyzers

Now that the analyzers are changing when you switch between Lucene and ElasticSearch, they are no longer in the annotation form in the class `Parameter.java`. You can find them respectively in `CoreLuceneAnalyzersDefinitionProvider.java` and `CoreElasticSearchAnalyzersDefinitionProvider.java`.

Due the exportation of analyzers definitions in external separate classes, you can add your own analyzers definitions by extending one of these two classes and override the function `register`. After that, you have to add a property in the file `hibernate-extra.properties` (create this file if it doesn't exists). If you want to use your own ElasticSearch analyzers add this line :

```
hibernate.search.elasticsearch.analyzer_definition_provider=package.to.yourclass.
↪ClassName
```

If you want to use your own Lucene analyzers add this line :

```
hibernate.search.lucene.analyzer_definition_provider=package.to.yourclass.ClassName
```

Note that when you choose to use ElasticSearch, Lucene's analyzers definitions are still instanciated but only used internally.

Date SortField and ElasticSearch

related commit

In ElasticSearch, Date SortField is of type STRING, but with Lucene, it is of type LONG. If you perform sort with `FullTextQuery.setSort(Sort sort)` with a Date field configured for one of the backends, it'll throw an exception with the other backend.

- **Solution 1:** Use only one backend, and initialize correctly and statically needed SortFields
- **Solution 2:** Use QueryBuilder to build your Sort object. QueryBuilder use field metadata to determine the right type to use.
  - `fr.openwide.core.jpa.search.util.SortFieldUtil` provides examples on the ways to obtain a Sort object or to perform a setSort(...) that use QueryBuilder and circumvent this issue.
  - replacing FullTextQuery.setSort(Sort sort) by SortFieldUtil.setSort(...) can be done quickly
  - beware that this workaround use field metadata to determine the right type; not deterministic and silent errors may become fatal errors with this workaround.

Wicket

ConsoleConfiguration.build()

`ConsoleConfiguration.build()` parameters are modified; you now need to provide a `IPropertyService`. This method call is generally done in you `<MyApplication>Application.java`. Just add IPropertyService as a `@SpringBean` field, and add it to the method call.

64.4 Migrating to 0.13

This guide aims at helping OWSI-Core users migrate an application based on OWSI-Core 0.12 to OWSI-Core 0.13.

In order to migrate from an older version of OWSI-Core, please refer to Migrating to 0.12 first.

64.4.1 owsi-core version numbering policy

owsi-core version 0.12.5 is the last version published exclusively for jdk 7. From owsi-core 0.13, vanilla versions will be built for java 1.8, and jdk 8 dependant starting owsi-core 0.14.

owsi-core 0.13.0 is planned to be a 0.12.5 isofunctionnal release, but published both for jdk 7 and 8.

Knowing this, you have two solutions for migrating from 0.12 to 0.13 : migrate to jdk 8 at the same time or keep a jdk 7 environment.

64.4.2 Solution 1: continue to use jdk 7

This solution allows you to upgrade project towards post or equals 0.13 release in a jdk 7 environment.

Please note that this version can be run in a java 8 environment.

Note: If your code base is already upgraded toward jdk7 version, you can switch directly to step 3 to upgrade your Eclipse's configuration.

Step 1 · udpate dependencies

Change project parent and owsi-core version to switch to jdk 7 dependency:

```
<parent>
    <groupId>fr.openwide.core.parents</groupId>
    <artifactId>owsi-core-parent-core-project-jdk7</artifactId>
    <version>0.13.jdk7-SNAPSHOT</version>
</parent>

[...]

<properties>
    <owsi-core.version>0.13.jdk7-SNAPSHOT</owsi-core.version>
</properties>
```

Note: owsi-core 0.13 codebase and all its upgrades will continue to support the use of java 1.7. You just have to add the **.jdk7** modifier after the version. Therefore you'll need to use **0.13.0.jdk7**, **0.13.1.jdk7**... owsi-core SNAPSHOT version will be flagged **0.13.jdk7-SNAPSHOT**.

Step 2 · clean your codebase

Perform a global maven clean so that generated source code is cleaned.

No more steps are needed to enable maven build.

Step 3 (for Eclipse IDE) · install m2e integration and reconfigure m2e-apt

Code generation, configured with maven-processor-plugin, is modified to facilitate m2e plugin integration. It is now recommended to use jboss m2e maven-processor-plugin integration.

Plugin intallation's instructions are available here: https://github.com/jbosstools/m2e-apt

It is easily installable via Eclipse Marketplace: m2e-apt (at least from Eclipse 4.5.2)

It is installed by default in Eclipse's team bundles from version 4.6.0

In *Windows → Preferences → Maven → Annotation Processing*, choose *Experimental: Delegate annotation processing to maven plugins…* (this option is known to work correctly for our use-cases)

If not done automatically, you need to reconfigure Maven projects (right-click on parent project, *Maven → Update project… → OK*

Note: Nothing to do with jdk version, but you may need to pay attention to the following setting : Windows → Preferences → Team → Git → Projects → **Uncheck** Automatically ignore derived resources by adding theme to .gitignore ; this setting prevents Eclipse to alter .gitignore configurations

---

Step 4 · optimization

If source code generation is to heavy on your project, you can restrain regeneration on project reconfiguration by adding the following m2e's configuration in your parent pom.xml (*dependencyManagement* section)

```
<pluginsManagement>
  <plugins>
    <plugin>
      <groupId>org.eclipse.m2e</groupId>
      <artifactId>lifecycle-mapping</artifactId>
      <version>1.0.0</version>
      <configuration>
            <lifecycleMappingMetadata>
              <pluginExecutions>
                <pluginExecution>
                  <pluginExecutionFilter>
                    <groupId>org.bsc.maven</groupId>
                    <artifactId>maven-processor-plugin</artifactId>
                    <versionRange>[0,)</versionRange>
                    <goals>
                      <goal>process</goal>
                      <goal>process-test</goal>
                    </goals>
                  </pluginExecutionFilter>
                  <action>
                    <execute>
```

(continues on next page)

(continued from previous page)

```
                    <runOnConfiguration>true</runOnConfiguration>
                    <runOnIncremental>false</runOnIncremental>
                  </execute>
                </action>
              </pluginExecution>
            </pluginExecutions>
          </lifecycleMappingMetadata>
      </configuration>
    </plugin>
  </plugins>
</pluginManagement>
```

64.4.3 Solution 2 · switch to jdk 8 version

This version use the same code base than jdk 7 (for 0.13 versions), but use a jdk 8 runtime. As jdk 7 version is compatible with jdk 8, this version is mainly provided to prepare your migration to jdk 8.

Step 1 · update dependencies

Simply make sure you use a post or equals 0.13 owsi-core version (without the .jdk7 modifier).

Step 2 to 4

Follow the same steps 2 to 4 than jdk 7 version.

64.5 Migrating to 0.12

This guide aims at helping OWSI-Core users migrate an application based on OWSI-Core 0.11 to OWSI-Core 0.12.

In order to migrate from an older version of OWSI-Core, please refer to *Migrating to 0.11* first.

64.5.1 Tools

Animal sniffer

JDK level validation using Animal sniffer is now enabled by default. This will probably force you to use JDK7 to build your project.

If the default JDK version (1.7) does not suit you, you should:

- change the value of the `jdk.version` property (as usual)
- and change the value of the `jdk.signature.artifactId` property to match one of the artifacts found here: http://search.maven.org/#search|ga|1|g%3A"org.codehaus.mojo.signature"

If this is somehow impossible and you want to disable these checks completely, you should disable the Animal Sniffer execution with id "check-java-version".

Bindings & code processors

There has been some changes regarding code processors. You will have to replace the goals of your `*.launch` files: instead of `generate-sources`, use `generate-sources generate-test-sources`.

64.5.2 External changes (libraries)

Spring & Spring Security

- change the `-4.0.xsd` schema to `-4.1.xsd` (Spring Security namespaces)

Wicket

- It seems like Wicket changed the implementation of URL mapping. There isn't many side effects, but one of them is the following: if you have mounted your two-parameter page twice, once with a trailing "$/param1/$" and once without a trailing "/{param1}/#{param2}", then Wicket will fail miserably and perform infinite redirections.
  - That's why it is now recommended, **for every page whose URL ends with a path parameter**, to mount the page with **no trailing slash**. Then the case mentioned above will work as a charm, and this will have the added benefit of allowing clients to use both versions of the URL: with or without a trailing slash.
  - Please note that if you skip the trailing slash for a page whose URL **does not** end with a path parameter, then Wicket will not allow accessing this page with a trailing slash (which is probably a bug). So **do not do this for pages whose URL does not end with a path parameter**.

64.5.3 Internal changes

Core

- Some classes' attributes have been renamed:
- `GenericEntityReference.getEntityId()` became `GenericEntityReference.getId()`
- `GenericEntityReference.getEntityClass()` became `GenericEntityReference.getType()`
- `HistoryValue.getEntityReference()` became `HistoryValue.getReference()`
- This could cause column name changes in your schema
- See https://github.com/openwide-java/owsi-core-parent/commit/33173617a905bdb110ee840acdad46fd20127b7f for details
- There's been some changes around notification descriptors in order to allow for applications to define user-specific context (`fr.openwide.core.spring.notification.model.INotificationContentDescriptor.withContext(INotificationRecipient)`). Thus:
- You're encouraged to use `NotificationContentDescriptors.explicit("defaultSubject", "defaultTextBody", "defaultHtmlBody")` as your default descriptor in your `EmptyNotificationContentDescriptorFactoryImpl`.
- Your notification content descriptor factories should not have a generic return type anymore, they should simply return `INotificationContentDescriptor`. Check out `NotificationDemoPage` and `ConsoleNotificationDemoIndexPage` from the basic application and use similar code in your own `NotificationDemoPage` and `ConsoleNotificationDemoIndexPage` in order to not depend on `IWicketNotificationDescriptor` anymore.

- JPAModelGen support is dropped in IGenericEntityDao/GenericEntityDaoImpl (*ByField) ; queries should be written with QueryDSL API, or compatibility layer may be extracted from GenericEntityDaoImpl 0.11.

64.5.4 Security

The unauthorized access mechanisms have been revamped, for more consistency:

- `AccessDeniedPage` is now accessed whenever a Wicket authorization error occurs.
- It is now clearer that `AccessDeniedPage` is **not** used when an anonymous user tries to access a protected resource.
- OWSI-Core's own redirection mechanism has been deprecated in favor of more standard ones (Wicket's and Spring Security's). On this particular subject, see *UI Redirecting*.

In order for your application to continue to work properly:

- You will need to add both `REQUEST` and `FORWARD` dispatchers to your Wicket application's filter mapping, so that Spring Security may forward requests when an access is denied.

  This:

```
<filter-mapping>
        <filter-name>MyApplication</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

  will have to become this:

```
<filter-mapping>
        <filter-name>MyApplication</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
</filter-mapping>
```

- If you overrode the default exception mapper, and you did not extend `CoreDefaultExceptionMapper` you may want to add this at the very top of your `map` method, outside of any `try` block:

```
        if (e instanceof AuthorizationException) {
                throw new AccessDeniedException("Access denied by Wicket's␣
↪security layer", e);
        }
```

  This will translate a wicket exception into something Spring Security can understand.

- If you overrode the default exception mapper, and you **did** extent `CoreDefaultExceptionMapper`, beware that your call to `super.map(e)` may now throw an `org.springframework.security.access.AccessDeniedException` which should not be caught. Ensure this call is made outside of a `try` block.

Webapp

- Some more logs have been added to `GenericEntityModel` and `AbstractThreadSafeLoadableDetachableModel`. See https://github.com/openwide-java/owsi-core-parent/wiki/UI-Models#debugging for more information.
- `DynamicImage`s, obtained through `IImageResourceLinkGenerator`s, now have their anticache parameter disabled by default. This may increase performance in Ajax refreshes where the same image appears multiple times. But it also means you will have to add a sensible anticache parameter to your image resources, such as `?t=<the last time your image was changed>`. You may do this when building your link descriptor, for instance with `fr.openwide.core.wicket.more.link.descriptor.builder.state.parameter.chosen.common.IOneChosenParameterState.renderInUrl(String, AbstractBinding<? super TChosenParam1, ?>)`.
- `IFormModelValidator` now extends `IDetachable`. You should implement `detach` as necessary.
- `ModelValidatingForm.addFormModelValidator(IFormModelValidator, IFormModelValidator ...)` has been renamed to simply `add`.
- `ModelValidatingForm.addFormModelValidator(Collection)` has been removed.
- `GenericEntityCollectionView` has been deprecated in favor of the more generic `CollectionView`. See `GenericEntityCollectionView`'s javadoc for information on migrating existing code.
- `SerializedItemCollectionView` has been deprecated in favor of the more generic `CollectionView`. See `SerializedItemCollectionView`'s javadoc for information on migrating existing code.
- `GenericEntity` collection models (`GenericEntityArrayListModel`, `GenericEntityTreeSetModel`, ...) have been deprecated in favor of the more generic `CollectionCopyModel`. See each older model's javadoc for information on migrating existing code.
- `IWicketContextExecutor` has been deprecated in favor of the more flexible `IWicketContextProvider`. Here are the main consequences to existing applications:
  - An object of type `IWicketContextProvider` is now available in the Spring context. You may `@Autowire` it in your own beans, or redefine it by overriding `fr.openwide.core.wicket.more.config.spring.AbstractWebappConfig.wicketContextProvider(WebApplication)` in your own webapp configuration.
  - The signature of `fr.openwide.core.wicket.more.config.spring.AbstractWebappConfig.wicketContextExecutor(WebApplication)` has changed and is now `fr.openwide.core.wicket.more.config.spring.AbstractWebappConfig.wicketContextExecutor(IWicketContextProvider)`. It cannot be overridden anymore. Please override `fr.openwide.core.wicket.more.config.spring.AbstractWebappConfig.wicketContextProvider(WebApplication)` instead.
  - Classes extending `AbstractWicketRendererServiceImpl`, `AbstractNotificationContentDescriptorFactory`, `AbstractNotificationUrlBuilderServiceImpl`, `AbstractNotificationPanelRendererServiceImpl` must now provide a `IWicketContextProvider` to their super constructor instead of a `IWicketContextExecutor`.
  - Classes extending `AbstractBackgroundWicketThreadContextBuilder` should instead rely on a `IWicketContextProvider`.
  - Classes relying on a `IWicketContextExecutor` should instead rely on a`IWicketContextProvider`. Here are a few examples of code refactoring:

```
String result = wicketExecutor.runWithContext(
        new Callable<String>() {
                public String call() throws Exception {
                        return doSomethingThatRequiresAWicketContext();
```

(continues on next page)

(continued from previous page)

```
                }
        },
        locale
);
```

becomes

```
String result;
try (ITearDownHandle handle = wicketContextProvider.context(locale).open()) {
        result = doSomethingThatRequiresAWicketContext();
}
```

```
And if you really must use a `Callable`:
```

```
String result = wicketExecutor.runWithContext(someCallable, locale);
```

becomes

```
String result = wicketContextProvider.context(locale).run(someCallable);
```

- `IOneParameterConditionFactory`, `IOneParameterModelFactory`, `AbstractOneParameterConditionFactory` and `AbstractOneParameterModelFactory` have been deprecated and replaced by `IDetachableFactory` and `AbstractDetachableFactory`. See their respective Javadoc for more information on migrating existing code.
- `LinkDescriptorBuilder`'s syntax changed slightly.

Previously, the entry point to building a link descriptor was `LinkDescriptorBuilder#LinkDescriptorBuilder()`, which was followed by a call to determine the type of the target, then various stuff around parameters, and finally a call to the `build()` method.

Now, the entry point is `LinkDescriptorBuilder#start()`, followed by various stuff around parameters, and finally a call to one of the build methods: `page`, `resource`, or `imageResource`.

The old syntax is still valid, but has been deprecated and will be removed in the future.

So something that we previously wrote this way:

```
    public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor, User>␣
↪MAPPER =
            new LinkDescriptorBuilder().page(MyPage.class)
                    .model(User.class).map("id").permission(READ)
                    .build();
```

... will now have to be written this way:

```
    public static final IOneParameterLinkDescriptorMapper<IPageLinkDescriptor, User>␣
↪MAPPER =
            LinkDescriptorBuilder.start()
                    .model(User.class).map("id").permission(READ)
                    .page(MyPage.class);
```

For existing projects, this perl script may help. Execute it this way from your project root:

```
find . -name '*.java' | xargs -n 1 -I{} bash -c 'TMP="$(mktemp)" ; perl -- ../
↪linkdescriptorbuilder_migrate.pl "{}" > $TMP ; diff -Zq "$TMP" {} >/dev/null || mv "
↪$TMP" {}'
```

It will try its best to convert most uses of `new LinkDescriptorBuilder` to `LinkDescriptorBuilder.start()`, converting early target definitions to late target definitions in the process. It should leave compilation errors wherever the conversion was not easy enough, so you can detect the places where you should edit code manually.

- **IPageLinkGenerator** implementations enforce permission checking in `getValidPageClass()`, hence in `fullUrl()` ; if you used `fullUrl()` to bypass permission checking (for example for email notification sent to another user than the one connected), replace `fullUrl()` by `bypassPermissions().fullUrl()`. NOTE: this backward compatibility is available only on former implementations, `CorePageInstanceLinkGenerator` and `CorePageLinkDescriptorImpl`; if you use newer implementions, you already should conform to the new behavior.
- **Ajax confirm link builder**: Ajax confirm link builder is now « form submit » aware ; current AjaxSubmitLink may be rewritten with `AjaxConfirmLink.build().[...].submit(form)`. AjaxSubmitLink still available.
- **Confirm link builder**: introduced `ConfirmLink.build()` builder. Unified syntax with ajax confirm link and ajax confirm submit. Confirm submit not supported (it was already a missing functionnality).
- **Confirm link**: introduced custom styles for yes / no buttons. Default values constructors were added to enable back-compatibility.
- **Condition and behavior**: EnclosureBehavior and PlaceholderBehavior are deprecated and replaced by behavior generation's methods on `Condition` object. This pattern allows to use more easily and consistently any `Condition` to control component's visibility or enabled property. More documentation on this pattern and the way to rewrite your code *UI Placeholder and Enclosure*

64.6 Migrating to 0.11

This guide aims at helping OWSI-Core users migrate an application based on OWSI-Core 0.9 to OWSI-Core 0.11.

OWSI-Core 0.10 was never released, so you should not have to migrate from this version, but if you did, then you could use this guide. Some parts would be irrelevant, but nothing should be missing.

64.6.1 Tools

Maven

Use the latest version of maven (at least Maven 3.3.1).

Java

~~Use JDK8. Earlier versions won't work.~~ Actually, use JDK7. A frequent VM crash was spotted when using Java 8 on a 32-bit OS and a bug report is currently pending review.

Bindings & code processors

You may have trouble with your bindings. Know that:

1. QueryDSL completely revamped its bindings; you'll need to delete them completely and regenerate them.
2. QueryDSL bindings generation will now fail if your project does not compile. Worse, not having QueryDSL bindings will add an enormous amount of build errors, which will make it much harder to spot actual errors that you must fix. It might be easier to first override QueryDSL's version and the processor used for QueryDSL bindings generation in your project (while sticking with OWSI-Core 0.9), and only then attempt migrating to OWSI-Core 0.11. The new processor is `com.querydsl:querydsl-apt` version 4.0.7. See below for changes in the new version of QueryDSL (`com.querydsl:querydsl-jpa` v4.0.7)
3. *This page* might help if you encounter errors when generating bindings.

A new feature was introduced in OWSI-Core 0.11 that allows the bindings to be completely wiped before generation, so that you won't need to manually delete them after a refactoring, for instance. In order to benefit from this feature, you must change your `eclipse/*.launch` files so that the invoked goals are simply `generate-sources`, but with the `eclipse-processor` profile enabled. On Linux, you may use the following command line from your project root:

```
find eclipse -name '*.launch' -print0 | xargs -0 sed -r -i 's/org.bsc.maven:maven-
↪processor-plugin:process( org.bsc.maven:maven-processor-plugin:process-test)?/generate-
↪sources/;s,<stringAttribute key="M2_PROFILES" value=""/>,<stringAttribute key="M2_
↪PROFILES" value="eclipse-processor"/>,'
```

64.6.2 External changes (libraries)

OpenCSV

- The maven artifact has changed. `net.sf.opencsv:opencsv` is now `com.opencsv:opencsv`.
- The main package changed, too. `au.com.bytecode.opencsv` is now `com.opencsv`.

FlyingSaucer (XHtmlRenderer)

- The renderer now supports border-radius. Please check that the rendering output in your project still suits you.

QueryDsl

- the root package has changed
- the general logic is now `query.select(...).from(...).fetch()`
- `count()` -> `fetchCount()`
- `uniqueResult()` -> `fetchOne()`
- `singleResult()` -> `fetchFirst()`
- `fetch().fetchAll()` -> `fetchJoin().fetchAll()`
- `query.map(keyExpression, valueExpression)` becomes `query.transform(GroupBy.groupBy(key).as(value))`
- Be careful with `mapExpression.containsKey`. From QueryDSL 4 on (maybe before?), QueryDSL will *not* create a subquery, but instead will try doing a join in the main query, with mixed results (especially considering

that Hibernate is notoriously buggy when it comes to cross-joins mixed with left/right/inner joins). To be safe, do not do this:

```
new JPAQuery<MyEntity>(getEntityManager())
        .select(qMyEntity)
        .from(qMyEntity)
        .from(qMyOtherEntity)
        .where(qMyOtherEntity.mapProperty.containsKey(qMyEntity))
        .orderBy(qMyEntity.id.asc())
        .fetch();
```

But instead do this:

```
new JPAQuery<MyEntity>(getEntityManager())
.select(qMyEntity)
        .from(qMyEntity)
        .where(
                        JPAExpressions.selectOne()
                        .from(qMyOtherEntity)
                        .where(qMyOtherEntity.mapProperty.containsKey(qMyEntity))
                        .exists()
        )
        .orderBy(qMyEntity.id.asc())
        .fetch();
```

Hibernate

- When using `javax.persistence.Index` with Hibernate, now the column names in columnList are *really* column names, not some mix-up of physical and logical names: if you had written myEmbeddable.myProperty_id, it becomes myEmbeddable_myProperty_id (or whatever name your column has)
- If you were using the deprecated datatype `org.hibernate.type.StringClobType`, then be sure to switch to `fr.openwide.core.jpa.hibernate.usertype.StringClobType` (as the former has been removed). Note that you may now use `fr.openwide.core.jpa.hibernate.usertype.StringClobType.TYPENAME` instead of duplicating the same inline String constant each time you need it.
- The naming strategy system changed in Hibernate 5. You must now add this to your `configuration-private.properties`

```
hibernate.implicit_naming_strategy=fr.openwide.core.jpa.hibernate.model.naming.
↪ImplicitNamingStrategyLegacyJpaComponentPathImpl
hibernate.physical_naming_strategy=fr.openwide.core.jpa.hibernate.model.naming.
↪PostgreSQLPhysicalNamingStrategyImpl
hibernate.id.new_generator_mappings=false
```

Hibernate Search & Lucene

- `SortField.STRING` is now `SortField.Type.STRING`
- Dates are now stored as `Long` so you need to sort them using `Sort.Type.LONG`
- Lucene has been upgraded. You will have to wipe clean your indexes and reindex everything
- For any field on which you perform a sort, you should now use the `@SortableField` (or `@SortableFields`) annotation. This is not mandatory, but will offer better performance and avoid annoying logs.
  - Note that, to sort GenericEntities by ID, you should now use the `GenericEntity.ID_SORT` field (or you'll get annoying warnings in your logs).
- The use of Lucene's `TokenStream` is now more safeguarded; this may lead to exceptions where you were not using it properly. Make sure that:
  - You always instantiate them in a try-with-resource (`try (TokenStream = /* ... */) { /* ... */ }`)
  - You always call `.reset()` before use
  - You always call `.end()` after use

WiQuery

The project has been moved to wicketstuff. Thus:

- The maven artifacts have changed:
- `org.odlabs.wiquery:wiquery-core` is now `org.wicketstuff.wiquery:wiquery-core`
- `org.odlabs.wiquery:wiquery-jquery-ui` is now `org.wicketstuff.wiquery:wiquery-jquery-ui`
- The main package changed, too. `org.odlabs.wiquery` is now `org.wicketstuff.wiquery`. If you're running an Unix-like OS, you may fix this in your project automatically with this command (to be run from the root of your project): `find . -name '*.java' | xargs sed -i 's/^import org.odlabs.wiquery/import org.wicketstuff.wiquery/'`

Wicket

- You no longer need to depend on `fr.openwide.core.components:000-owsi-core-component-wicket-override`. Plus, it will probably harm to do so. Just remove this dependency.
- `StringResourceModel` now has a fluid API: you should use `setModel`, `setParameters` and `setDefaultValue`
- You need to look for Ajax links whose markup is an <a> the click event is not blocked by Wicket anymore, which will result in a scroll to the top of the page each time the link is clicked. This is always true for Google Chrome, but only if there is a `href` attribute for Firefox. To avoid any kind of trouble, just follow the guidelines detailed *here*.
- There is now a high-level integration of JQPlot built in OWSI-Core. See the docs or the pull request for more information.

Spring & Spring Security

- in `security-http`, `use-expressions` is now `true` by default. Thus, you have to use expressions like `hasRole('xxx')` and `permitAll` or define it explicitely to `false`. Be careful that the error is triggered only when you effectively access a secured page.
- change the `-4.0.xsd` schema to `-4.2.xsd` (Spring namespaces)
- change the `-3.2.xsd` schema to `-4.0.xsd` (Spring Security namespaces) - obviously, you need to follow the order
- in **every** `security:http` (even those related to simple REST API calls), you need to add:

```
<security:headers disabled="true"/>
<security:csrf disabled="true"/>
```

64.6.3 Internal changes

Core

- @PermissionObject has been moved to package `fr.openwide.core.commons.util.security`. Run the following command from the root of your project to update your imports: `find . -type f -name '*.java' -print0 | xargs -0 sed -r -i 's/fr.openwide.core.jpa.business.generic.annotation.PermissionObject/fr.openwide.core.commons.util.security.PermissionObject/g'`
- `TransactionSynchronizationTaskManagerServiceImpl` now executes `afterRollback` on tasks implementing it in **reverse** order. See https://github.com/openwide-java/owsi-core-parent/commit/b431545ce20c8c5a182617e7e93b9f044086d4b1

Webapp

- The JQPlot/WQPlot dependency has been moved to a separate module. If you were using JQPlot/WQPlot, add this to your webapp's dependencies:

```
        <dependency>
                <groupId>fr.openwide.core.components</groupId>
                <artifactId>owsi-core-component-wicket-more-jqplot</artifactId>
                <version>${owsi-core.version}</version>
        </dependency>
```

- The `FormErrorDecoratorListener` has been pulled from various projects to OWSI-Core. Use OWSI-Core's version.
- The `DataTableBuilder` and related classes have moved. You may use the following sed script to convert your source code. Just create a file, put the following snippet in there, then run `find . -type f -name '*.java' -print0 | xargs -0 sed -r -i -f ./thescriptfile`. Here's the content of this file:

```
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.
↪DecoratedCoreDataTablePanel/fr.openwide.core.wicket.more.markup.repeater.table.
↪DecoratedCoreDataTablePanel/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.
↪DecoratedCoreDataTablePanel.AddInPlacement/fr.openwide.core.wicket.more.markup.
↪repeater.table.DecoratedCoreDataTablePanel.AddInPlacement/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.builder.
```

(continues on next page)

(continued from previous page)

```
↪DataTableBuilder/fr.openwide.core.wicket.more.markup.repeater.table.builder.
↪DataTableBuilder/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.AbstractCoreColumn/
↪fr.openwide.core.wicket.more.markup.repeater.table.column.AbstractCoreColumn/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.CoreDataTable/fr.
↪openwide.core.wicket.more.markup.repeater.table.CoreDataTable/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.util.DataTableUtil/
↪fr.openwide.core.wicket.more.markup.repeater.table.util.DataTableUtil/g
s/fr.openwide.core.wicket.more.markup.html.repeater.data.table.util.
↪IDataTableFactory/fr.openwide.core.wicket.more.markup.repeater.table.builder.
↪IDataTableFactory/g
```

- The `DataTableBuilder` and related classes are now based on the `ISequenceProvider` instead of `IDataProvider`. You may still use `IDataProvider` as an input to the `DataTableBuilder` (it will be wrapped).
- A new interface was introduced in order to address code execution in a wicket context: `IWicketContextExecutor`. Here are the main consequences to existing applications:
  - An object of type `IWicketContextExecutor` is now available in the Spring context. You may `@Autowire` it in your own beans, or redefine it by overriding `fr.openwide.core.wicket.more.config.spring.AbstractWebappConfig.wicketContextExecutor(WebApplication)` in your own webapp configuration.
  - Classes extending `AbstractWicketRendererServiceImpl`, `AbstractNotificationContentDescriptorFactory`, `AbstractNotificationUrlBuilderServiceImpl`, `AbstractNotificationPanelRendererServiceImpl` must now provide a `IWicketExecutor` to their super constructor and must not override `getApplicationName()` anymore.
  - Classes extending `AbstractBackgroundWicketThreadContextBuilder` should instead rely on a `IWicketContextExecutor`.

External link checker

The external link checker now has its own Maven module. See *ExternalLinkChecker* if you use it in your app.

Related to the new `PropertyService`: you also have to use `JpaExternalLinkCheckerConfig` (import) in your app.

Properties

Both immutable and mutable properties are now handled by `PropertyService`. See *PropertyService* to use it in your app.

- `CoreConfigurer`: getter methods are deprecated and redirect to `propertyService`. Utility methods are also deprecated.
- `AbstratParameterServiceImpl`: getter and setter methods are deprecated and redirect to `propertyService`. Utility methods are also deprecated.

Important notes

- Properties wrapping a date (or a date time) and registered in `PropertyService` must respect the following format 'yyyy-MM-dd' (or 'yyyy-MM-dd HH:mm(:ss)'). See `StringDateConverter` and `StringDateTimeConverter`.

- `ConfigurationLogger`: As previously it uses *propertyNamesForInfoLogLevel* property but it is based now on `PropertyService`. That's why all the properties you want to display must be registered in the `PropertyService`.
- To display a warning message in case of `null` value while retrieving a property, add the following entry in your log4j file: `log4j.logger.fr.openwide.core.spring.property.service.PropertyServiceImpl=DEBUG`.

1st option: keeping the old school properties management

This case is not tested yet and is **not recommended**. Please, as much as possible, migrate to the `PropertyService`.

Get the latest version of both `CoreConfigurer` and `AbstractParameterServiceImpl` (+ `IAbstractParameterService`) from the previous version of OWSI-Core and bring back all methods and attributes needed in your own `YourAppConfigurer` and `ParameterServiceImpl` (+ `IParameterService`).

Also, in `YourAppCorePropertyConfig`, make sure the `mutablePropertyDao` method returns a `IParameterDao` and not simply a `IMutablePropertyDao`.

2nd (better) option: migrating to PropertyService

See *PropertyService*

- Create a `YourAppCorePropertyIds` and a `YourAppApplicationPropertyConfig` in your `core` module.
- Create a `YourAppWebappPropertyIds` and a `YourAppApplicationPropertyRegistryConfig` in your `webapp` module.
- Register your properties.
- Deprecate everything in `YourAppConfigurer` and `ParameterServiceImpl`
- Fix all deprecated warnings caused by the configurer and the parameter service. See Javadoc on deprecated methods in OWSI-Core to make it easier.
- Remove `ParameterServiceImpl` and `IParameterService`.
- Remove everything from `YourAppConfigurer`.

Audit

The audit classes have been removed.

You should either:

- Copy the old Audit base classes in your own project
- Or (better) use the brand-new HistoryLog framework. See *HistoryLog & Audit*

PasswordEncoder

From now on, we use bcrypt method to encode new passwords. However, old passwords / hashes still use previous encryption method.

SecurityPasswordRules

SecurityPasswordRules is now a builder and provide a `Set<Rule>`.

```
SecurityPasswordRules
        .builder()
        .minMaxLength(..., ...)
        .forbiddenUsername()
        .rule(YourCustomRule())
        .build();
```

Also, replace `SecurityPasswordRules.DEFAULT` :

```
SecurityPasswordRules
        .builder()
        .minMaxLength(User.MIN_PASSWORD_LENGTH, User.MAX_PASSWORD_LENGTH)
        .build();
```

64.6.4 Configuration

- Ensure to give a value to notification.mail.recipientsFiltered property (true or false). If true, mail's recipients are replaced by notification.test.emails property's content
- Replace this : `hibernate.search.analyzer=org.hibernate.search.util.impl.PassThroughAnalyzer` with this `hibernate.search.analyzer=org.apache.lucene.analysis.core.KeywordAnalyzer`
- The content of configuration-private.properties should be:

```
hibernate.implicit_naming_strategy=fr.openwide.core.jpa.hibernate.model.naming.
↪ImplicitNamingStrategyLegacyJpaComponentPathImpl
hibernate.physical_naming_strategy=fr.openwide.core.jpa.hibernate.model.naming.
↪PostgreSQLPhysicalNamingStrategyImpl
hibernate.id.new_generator_mappings=false
```

64.6.5 Database

- You may have missing columns in the tables mapped to your `GenericLocalizedGenericListItem` entities. Please check them out.
- The position in `GenericLocalizedGenericListItems` is not nullable anymore. Execute this for each table:

```
update XXX set position=0 where position is null;
```

- The hash generated for foreign key constraints name has changed. Therefore, you will probably end up with duplicate foreign keys. After checking that this is effectively the case, you can use the following query to generate a cleanup script:

```
SELECT
        'ALTER TABLE ' || pclsc.relname || ' DROP CONSTRAINT ' || pc.conname || ';'
FROM
        (
        SELECT
                 connamespace,conname, unnest(conkey) as "conkey", unnest(confkey)
                  as "confkey" , conrelid, confrelid, contype
         FROM
                pg_constraint
        ) pc
        JOIN pg_namespace pn ON pc.connamespace = pn.oid
        -- and pn.nspname = 'panmydesk4400'
        JOIN pg_class pclsc ON pc.conrelid = pclsc.oid
        JOIN pg_class pclsp ON          pc.confrelid = pclsp.oid
        JOIN pg_attribute pac ON pc.conkey = pac.attnum        and pac.attrelid␣
↪=           pclsc.oid
        JOIN pg_attribute pap ON pc.confkey = pap.attnum and pap.attrelid = pclsp.
↪oid
WHERE pc.conname ilike 'fk\_%' or pc.conname ilike '%_fkey'
ORDER BY pclsc.relname;
```

- The hash generated for unique constraints name has changed when using table level annotation (`uk_mykeyhash` becomes `ukmykeyhash`). Therefore, you will probably end up with duplicate unique constraints. After checking that this is effectively the case, you will need to identify them and create a cleanup script. To identify these constraints, you should search for `@UniqueConstraint` annotation references in your project.
- If the application is old, you might even have a third naming scheme which you can detect with the following query:

```
SELECT
        'ALTER TABLE ' || pclsc.relname || ' DROP CONSTRAINT ' || pc.conname || ';'
FROM
        (
        SELECT
                 connamespace,conname, unnest(conkey) as "conkey", unnest(confkey)
                  as "confkey" , conrelid, confrelid, contype
         FROM
                pg_constraint
        ) pc
        JOIN pg_namespace pn ON pc.connamespace = pn.oid
        -- and pn.nspname = 'panmydesk4400'
        JOIN pg_class pclsc ON pc.conrelid = pclsc.oid
        JOIN pg_class pclsp ON  pc.confrelid = pclsp.oid
        JOIN pg_attribute pac ON pc.conkey = pac.attnum        and pac.attrelid =␣
↪pclsc.oid
        JOIN pg_attribute pap ON pc.confkey = pap.attnum and pap.attrelid = pclsp.
↪oid
WHERE char_length(pc.conname) = 18 and pc.conname ilike 'fk%'
ORDER BY pclsc.relname;
```

64.6.6 Wicket Resource Security

Until now security context was not set in Wicket Resource because we used this snippet:

```
<security:http pattern="/wicket/resource/**" security="none" />
```

However, since `DropDownChoice` may now use Wicket Resources to fetch data:

1. We need a security context for some of the resources (e.g. to retrieve current authenticated user, or to prevent some users to access that resource)
2. We need to take care of which resources are publicly accessible

That's why you should now use `intercept-url` to protect resources. Add something like this before your default `security:http`:

```
        <!-- An entry point to respond with a 403 error if Spring Security wants the␣
↪user to log in.
                Useful in situations where loging in is not an option, such as when␣
↪serving CSS.
         -->
        <bean id="entryPoint403" class="org.springframework.security.web.authentication.
↪Http403ForbiddenEntryPoint"/>

        <security:http request-matcher="regex"
                       pattern="^/wicket/resource/.*"
                       create-session="never" entry-point-ref="entryPoint403"␣
↪authentication-manager-ref="authenticationManager"
                       auto-config="false" use-expressions="true">
                <security:headers disabled="true"/>
                <security:csrf disabled="true"/>

                <security:intercept-url pattern="^/wicket/resource/fr.openwide.core.
↪basicapp.web.application.common.template.js.[^/]+.*" access="hasRole('ROLE_ANONYMOUS')
↪" />
                <security:intercept-url pattern="^/wicket/resource/fr.openwide.core.
↪basicapp.web.application.common.template.styles.[^/]+.*" access="hasRole('ROLE_
↪ANONYMOUS')" />
                <security:intercept-url pattern="^/wicket/resource/fr.openwide.core.
↪basicapp.web.application.common.template.images.[^/]+.*" access="hasRole('ROLE_
↪ANONYMOUS')" />
                <security:intercept-url pattern="^/wicket/resource/fr.openwide.core.
↪basicapp.web.application.[^/]+.*" access="hasRole('ROLE_AUTHENTICATED')" />
                <security:intercept-url pattern="^/wicket/resource/.*" access="hasRole(
↪'ROLE_ANONYMOUS')" />
        </security:http>
```

Please note that, if you have to make some other resources publicly available (for example on the login page), you should change the above to suit your needs. As is, only JS files, CSS files, static image files and Resources defined in packages other than those of your app (OWSI-Core, various dependencies like Select2) are made publicly available.

64.6.7 Ajax confirm link builder

- `AjaxConfirmLink#build(String)` and `AjaxConfirmLink#build(String, IModel<O>)` no longer exist. Use `AjaxConfirmLink#build()` instead.
- `AjaxConfirmLinkBuilder#create()` no longer exists. Use `AjaxConfirmLinkBuilder#create(String)` or `AjaxConfirmLinkBuilder#create(String, IModel<O>)`.
- `AjaxConfirmLinkBuilder#onClick(SerializableFunction<AjaxRequestTarget, Void>)` and `AjaxConfirmLinkBuilder#onClick(AjaxResponseAction)` no longer exist. Use `AjaxConfirmLinkBuilder#onClick(IOneParameterAjaxAction<IModel<O>>)` or `AjaxConfirmLinkBuilder#onClick(IAjaxAction)` (no parameters) instead. You can use `AbstractAjaxAction` or `AbstractOneParameterAjaxAction`.

CHAPTER

SIXTYFIVE

DEVELOPMENT ENVIRONMENT

65.1 Prerequisites

- **PostgreSQL** (default configuration); other databases may be used
- **Eclipse/Intellij**
- **JRE 11** (Open JDK - maven build / runtime compatible with Java 11)

65.2 Database initialization

To run the basic-application or any project that use the basic-application as an outline, you need to initialize a database. To do so, you need to create a user first, and then create a database with the user you have created as its owner.

```
createuser -U postgres -P basic_application
createdb -U postgres -O basic_application basic_application
psql -U postgres basic_application
#Here you are connected to the database as the user postgres
DROP SCHEMA public;
\q
psql -U basic_application application
#Here you are connected to the database as the user hello_world
CREATE SCHEMA basic_application;
```

Database's content is automatically initialized (schema, tables, index, data) with default basic_application configuration (spring profile **flyway** enabled, see `spring.profiles.active` configuration).

65.3 Eclipse configuration

Igloo uses Maven build system. Some shortcuts, configurations and tools are provided for Eclipse integration.

The following documentation needs a running Eclipse instance with m2e Maven integration installed.

This documentation is known to work with Eclipse 4.7 release.

- Clone https://github.com/igloo-project/igloo-oomph-project
- Start eclipse
- `Package Explorer` > contextual menu > `Import...`
- `Oomph` > `Projects into workspace` > `Next`

- Add user projects (green + icon) > Browse File System > igloo-oomph-project/org.iglooproject.eclipse.igloo.setup > Validate
- Select Igloo development in <Eclipse Projects> > <User> tree
- In Variables' form, check the fields. You may customize filesystem clone folder name (~/git/<folder name>), git branch or tomcat binary location; validate
- Accept to restart when Eclipse asks to
- At restart, igloo project is cloned and imported. Wait for build completion (~ 4 minutes), then stop-start Eclipse again
- Add Servers view (Window > Show view > Other...)
- tomcat85 may be listed, with basic-application-webapp module, and is ready to be run

65.4 Intellij

Maven build process is compatible with Intellij maven integration.

Keep attention to switch indentation configuration to tabs for *.java* and *.xml* files. Deactivate automatic import reorganization.

When using a launcher setup to start *main* application, include provided/runtime dependencies in classpath.

65.5 Other IDE

You may use the tools provided by your IDE for Maven's integration.

Igloo relies on the following requirements :

- **maven-processor-plugin**: code generation with java builtin annotation processing. It implies a custom generate-resources task, and added source folders for generated files
- **maven-surefire-plugin**: unit tests, managed by junit. Tests may also be run by junit IDE integration (known to work with Eclipse)
- **com.github.eirslett/frontend-maven-plugin**: npm build for javascript/css resources
- (optional) **maven-enforcer-plugin**: various check about dependencies
- (optional) **animal-sniffer-maven-plugin**: Java API checks

CHAPTER

SIXTYSIX

DOCUMENTATION MODIFICATION

66.1 Miscelleaneous

The documentation is located at http://igloo-doc.readthedocs.io/en/latest/index.html

66.2 Contributing to the doc

66.2.1 Install the documentation

To edit and update documentation;

```
git clone git@github.com:igloo-project/igloo-doc.git
cd igloo-doc
rm -rf .tools
pipenv install
pipenv shell
```

66.2.2 Build the documentation locally

A few commands to interact with the documentation locally:

```
# build documentation
clickable sphinx build html
# start a local server and preview documentation in browser (http://localhost:8000)
clickable sphinx live
# clean
clickable sphinx clean
```

66.2.3 Build the documentation on ReadTheDocs

To modify the online documentation, you just have to push your modifications on the igloo-doc git repository. A webhook is set and will automatically rebuild the documentation everytime you push something.

CHAPTER

SIXTYSEVEN

CONTRIBUTING TO UPSTREAM

67.1 Hibernate

Our cloned repo: https://github.com/igloo-project/hibernate-orm

67.2 Resources

- Full contribution procedure (also here, but it seems to be almost the same)
- How to develop using Eclipse (see below for more concrete explanations)

67.3 Developing

Hibernate uses Gradle. This means some pain if you haven't had to work with it in Eclipse, ever.

In order to build using gradle:

- Check that your default JRE is recent enough (tested with JRE8 on Hibernate 5.0, it should work)
- Generate the Eclipse `.project` files: `./gradlew clean eclipse --refresh-dependencies`
- Install the Gradle Eclipse plugin from this update site: `http://dist.springsource.com/release/TOOLS/gradle`
- Import the projects **as standard Eclipse projects** (Gradle import seems to mess things up, at least with Eclipse 4.3)
- Pray that everything builds right. I personally couldn't make every project compile, but what I had to work on did, so…

67.4 Testing

67.4.1 Running tests locally

Launch your test this way (example for a test in hibernate-core):

```
./gradlew :hibernate-core:test --tests 'MyTestClassName'
```

67.4.2 Running tests locally, with database vendor dependency

If your test relies on a specific database vendor, you'll need to do the following in order to run it locally (examples for PostgreSQL):

- Specify the Dialect to use with the following option `-Dhibernate.dialect=org.hibernate.dialect.PostgreSQL9Dialect`
- Specify JDBC information: `-Dhibernate.connection.url=...`, `-Dhibernate.connection.username=...`, `-Dhibernate.connection.password=...`, `-Dhibernate.connection.driver_class=...`
- Provide the vendor-specific driver jar. I couldn't find a way to do it other than changing the `hibernate-core/hibernate-core.gradle` file and adding this line in the `dependencies` block: `testCompile( 'org.postgresql:postgresql:9.4-1200-jdbc41' )`

You'll end up launching your test this way (example for a test in hibernate-core):

```
./gradlew -Dhibernate.dialect=org.hibernate.dialect.PostgreSQL9Dialect -Dhibernate.
↪connection.url=jdbc:postgresql://localhost:5432/hibernate_test -Dhibernate.connection.
↪username=hibernate -Dhibernate.connection.password=hibernate -Dhibernate.connection.
↪driver_class=org.postgresql.Driver :hibernate-core:test --tests 'MyTestClassName'
```

67.5 Hibernate Search

Our cloned repo: https://github.com/igloo-project/hibernate-search

67.5.1 Resources

- Full contribution procedure

CHAPTER

SIXTYEIGHT

INFRASTRUCTURE APACHE

68.1 Default configuration for an Apache Vhost

```
<VirtualHost *:443>
        ServerName www.siteurl.com
        ServerAlias technical.alias.net

        DocumentRoot /data/services/web/<sitename>/site

        ProxyErrorOverride On
        ProxyPass /static-content/ !
        ProxyPass /errors/ !
        ProxyPass /server-status !
        ProxyPass         / ajp://localhost:8009/ keepalive=on timeout=3000 ttl=300
        ProxyPassReverse  / ajp://localhost:8009/

        ErrorDocument 403 /errors/403.html
        ErrorDocument 404 /errors/404.html
        ErrorDocument 500 /errors/500.html
        ErrorDocument 503 /errors/503.html

        AddOutputFilterByType DEFLATE application/x-javascript text/html text/xml text/
↪css text/javascript

        AddDefaultCharset UTF-8
        AddCharset UTF-8 .js .css

        <Directory /data/services/web/<sitename>/site>
                Require all granted
        </Directory>

        SSLEngine on

        SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
        SSLCipherSuite SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
↪SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
↪POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
↪SHA384
        SSLHonorCipherOrder on
```

(continues on next page)

(continued from previous page)

```
        SSLCertificateFile /etc/httpd/ssl/<sitename>.crt
        SSLCertificateKeyFile /etc/httpd/ssl/<sitename>.key.unsecure
        SSLCACertificateFile /etc/httpd/ssl/ca-thawte.crt

        CustomLog /data/log/web/<sitename>-access_log combined
        ErrorLog /data/log/web/<sitename>-error_log
</VirtualHost>
```