

1.3 Découverte avec esup-example



Bon pour relecture

Ce paragraphe non exhaustif permet de découvrir *esup-commons* V2 au travers du projet *esup-example*. Ce dernier servira de bac à sable permettant de toucher du doigt les aspects incontournables pour le développement d'un projet.

Sommaire :

- [Installation](#)
- [Démarrage](#)

Installation

A l'aide du plugin Subversion (en perspective "SVN Repository Exploring") faire un checkout de la dernière version stable de *esup-example* :

```
https://subversion.cru.fr/esup-commons/tags/<dernière version>/esup-example
```

Pour connaître la dernière version release voir : [Changelog V2](#)

? Pièce jointe inconnue

Faire un **Checkout...**

Choisir **Check out as a project in the workspace**

Nommer le projet "esup-example" et laisser les paramètres par défaut puis **Finish**

? Pièce jointe inconnue

Passer en perspective "Java"

Clic-droit sur le projet **Maven => Enable Dependency Management**.

? Pièce jointe inconnue

Comme on l'a dit précédemment, un projet *esup-commons* est composé de plusieurs sous-projets ou modules situés ici dans des sous-répertoires contenant chacun un fichier pom.xml, qui peuvent être des projets indépendants :

Faire clic-droit sur le projet **Import... => Maven => Existing Maven Projects** :


Ceci permet de transformer chaque module de _esup-commons en projet à part entière grâce aux fichiers pom.xml déclarés dans les différents répertoires du projet père_

? Pièce jointe inconnue

Là Eclipse retrouve tous les pom.xml : les laisser tous cochés

? Pièce jointe inconnue

Cliquer sur **Finish** : il construit alors tous les projets.

Ceci permet de lier le projet à ses dépendances Maven déclarées dans les fichiers pom.xml situés dans chaque module. Il crée un répertoire .m2 s'il n'existe pas. Enfin un petit "M" apparaît sur l'icône du projet 

On constate que cette tâche entraîne le téléchargement de toutes les librairies nécessaires, c'est pourquoi elle peut-être relativement longue suivant le nombre de librairies pré-existantes dans votre répertoire .m2 (téléchargées pour d'autres projets)

? Pièce jointe inconnue

*Il peut arriver que les chemins vers des sources soient calculés depuis la racine alors que celles-ci se trouvent dans src/main/java, src/main/resources etc. Plutôt que de faire cela manuellement (**clic-droit sur le projet => Build Path => Configure Build Path**), il suffit de faire un **clic-droit sur le projet => Maven => Update Project Configuration**.*

Puis faire **clic-droit sur le projet => Run As => Maven install**

? Pièce jointe inconnue

Résultat :

? Pièce jointe inconnue

Démarrage

Un projet *esup-commons* peut proposer un serveur d'application embarqué (pour un démarrage *standalone*), de manière presque transparente grâce à Maven et à l'utilisation du plugin Jetty.

Ce plugin se base sur la structuration préconisée par défaut dans un projet maven et retrouve automatiquement la webapp à lancer. Ainsi, à partir du moment où le projet respecte les standards Maven, l'utilisation du plugin jetty ne nécessite pas de configuration supplémentaire (hormis éventuellement un `contextPath`)

```
<plugin>
  <groupId>org.mortbay.jetty</groupId>
  <artifactId>maven-jetty-plugin</artifactId>
  <version>6.1.26</version>
  <configuration>
    <contextPath></contextPath>
  </configuration>
</plugin>
```

Comme on l'a vu précédemment dans [1.2 Méthodologie de développement](#), *esup-commons* propose différents types de projets (jsf-mixte, jsf-servlet etc.). Le démarrage ne peut se faire que sur les modules packagés sous forme de WAR (qui sont donc dépendants des modules sous-jacents packagés en jar)

Dans *esup-exemple*, plusieurs types de projet sont implémentés il faudra donc choisir.

Via Eclipse : Lancer le projet par un clic-droit sur le projet *esup-exemple-jsf-servlet* par exemple **Run AS => Maven build...** et choisir l'action **jetty:run**

? Pièce jointe inconnue

? Pièce jointe inconnue

*On pourra modifier par **Run => Run configurations...** et sélectionner "Maven Build"*

Lancer la commande.


*Au premier démarrage on constate dans la console le téléchargement d'un certain nombre de librairies et la création d'une base de données(1637 INFO org.hibernate.tool.hbm2ddl.DatabaseMetadata - table not found: User
1639 INFO org.hibernate.tool.hbm2ddl.SchemaUpdate - schema update complete).*

Dans la console on obtient

```
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building esup-example
[INFO]    task-segment: [jetty:run]
[INFO] -----
[INFO] Preparing jetty:run
[INFO] [resources:resources {execution: default-resources}]
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform
dependent!
[INFO] Copying 33 resources
[INFO] [compiler:compile {execution: default-compile}]
[...]
```

[INFO] Started Jetty Server
2010-11-24 12:04:26.289::INFO: Started SelectChannelConnector@0.0.0.0:8080
[INFO] Starting scanner at interval of 3 seconds.

Dans un navigateur lancer localhost :8080... on obtient ceci :

 Pièce jointe inconnue

On constate que l'on est authentifié en **bourges**.

On va maintenant changer la méthode d'authentification afin de pouvoir se connecter en CAS.

Pour cela, ouvrir /src/main/resources/properties/auth/auth.xml en paramétrer un **CasFilterAuthenticationService** à la place du **OfflineFixedUserAuthenticationService**

```
<bean id="authenticator" lazy-init="true"
      class="org.esupportail.example.services.authentication.AuthenticatorImpl">
  <property name="authenticationService" ref="servletAuthenticationService" />
</bean>

<bean id="servletAuthenticationService" lazy-init="true"
      class="org.esupportail.commons.services.authentication.CasFilterAuthenticationService">
</bean>
```

Modifier ensuite fichier /src/main/webapp/WEB-INF/web.xml et y indiquer l'adresse du votre serveur CAS

Exemple :

```
<!-- CAS -->
<filter>
  <filter-name>CAS Authentication Filter</filter-name>
  <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  <init-param>
    <param-name>casServerLoginUrl</param-name>
    <param-value>https://cas.uhp-nancy.fr/cas/login</param-value>
  </init-param>
  <init-param>
    <param-name>serverName</param-name>
    <param-value>http://localhost:8080</param-value>
  </init-param>
</filter>
```

... et

```
<filter>
    <filter-name>CAS Validation Filter</filter-name>
    <!--
class>        <filter-class>org.jasig.cas.client.validation.Saml11TicketValidationFilter</filter-
-->
    <filter-class>org.jasig.cas.client.validation.Cas10TicketValidationFilter</filter-class>
    <init-param>
        <param-name>casServerUrlPrefix</param-name>
        <param-value>https://cas.uhp-nancy.fr/cas</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value>http://localhost:8080</param-value>
    </init-param>
</filter>
```

A l'enregistrement on constate dans la console que ça recharge le projet.

```
[...]
[INFO] Restart completed at Wed Nov 24 12:20:14 CET 2010
```

Retenter un localhost:8080 et vérifier que l'authentification fonctionne