3.15 Déploiement en production

Bon pour relecture

Sommaire :

Utilisation d'un "profile" Maven nommé "Production"

Ce "profile" va nous permettre de:

- réaliser un filtre sur les fichiers de configuration de manière à substituer les variables réelles de production à celles qui ont été préparées.
- produire un fichier WAR déployable en mode servlet et/ou portlet grace à la tâche ant "portlet.deploy" livrée avec le portail Esup.

Voici un exemple de code:

```
<profile>
        <id>Production</id>
        <activation>
                <property>
                        <name>Production</name>
                        <value>true</value>
                </property>
        </activation>
        <build>
                <filters>
                        <filter>src/main/resources/properties/defaults.properties</filter>
                        <filter>src/main/resources/properties/config.properties</filter>
                </filters>
                <resources>
                        <resource>
                                <directory>src/main/resources</directory>
                                <filtering>true</filtering>
                        </resource>
                </resources>
                <defaultGoal>package</defaultGoal>
                <plugins>
                        <plugin>
                                <artifactId>maven-war-plugin</artifactId>
                                <version>2.1</version>
                                <configuration>
                                        <webResources>
                                                <resource>
                                                        <filtering>true</filtering>
                                                        <directory>src/main/webapp</directory>
                                                        <includes>
                                                                <include>WEB-INF/web.xml</include>
                                                                <include>WEB-INF/portlet.xml</include>
                                                        </includes>
                                                </resource>
                                        </webResources>
                                        <warName>esup-blank-${misc.version}</warName>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</profile>
```

Dans l'**environnement de production**, lancer Maven pour ce "profile" (depuis le module web de l'application):

```
mvn -DProduction=true clean package
```

On peut alors lancer la tâche ant "portlet.deploy" qui va assembler dans un nouveau web.xml le web.xml livré dans le WAR avec le fichier portlet.xml.

L'application pourra alors fonctionner dans le portail **en mode servlet et en mode portlet**.

Voici la commande à lancer (depuis l'environnement Esup-package):

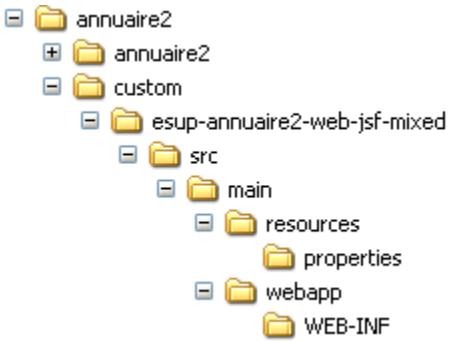
```
ant portlet.deploy -DportletApp=...monFichierWar...
```

Il est bien sûr nécessaire de réaliser un fichier "chanpub" pour que la portlet apparaisse au bon endroit et avec les bon droits associés dans le portail.

Personnaliser le déploiement

Pour mettre pleinement en œuvre la procédure ci-dessus dans le cadre d'un établissement, tout en gardant vos paramètres de configuration particuliers, vous pouvez suivre ces différentes étapes:

1. Si ce n'est déjà fait, mettre en place un espace de BUILD qui soit différent de la production.
2. Créer dans cet espace un répertoire spécifique pour l'application (ici nommé annuaire2). Et dans celui-ci un répertoire "custom".
3. Y construire une hiérarchie de répertoires contenant vos fichiers personnalisés. En voici un exemple:



Dans le répertoire WEB-INF, vous aurez le "porlet.xml" personnalisé.
Dans le répertoire properties, vous aurez le fichier config.properties de l'application personnalisé.

1. Depuis ce répertoire, lancer un script avec les commandes suivantes (ici avec l'application esup-annuaire2):

   ```
   svn co https://subversion.cru.fr/esup-annuaire2/trunk/esup-annuaire2
   zip -r packages/esup-annuaire2.zip esup-annuaire2/
   rm -rf esup-annuaire2/
   ```

2. puis lancer un autre script avec les commandes suivantes:

   ```
   ant init
   cd=`pwd`
   cd annuaire2
   mvn clean install
   cd esup-annuaire2-web-jsf-mixed/
   mvn -DProduction=true clean package
   cd /home/tomcat/portail/BUILD/esup-package
   ant portlet.deploy -DportletApp=/home/tomcat/portail/BUILD/applis/annuaire2/annuaire2/esup-annuaire2-web-
   jsf-mixed/target/esup-annuaire2-0.1.0.war
   cd $cd
   ```

   Commentaires:
   La cible "init" va partir du .zip construit par le script précédent pour y incorporer les fichiers que vous avec personnalisés, elle est constituée ainsi:

```
<target name="init" depends="unzip">
        <copy overwrite="true" todir="${appli.base}/">
                <fileset dir="custom" includes="**/*" />
        </copy>
</target>

<target name="unzip">
        <delete dir="${appli.base}"/>
        <delete dir="${appli.tmp}"/>
        <mkdir dir="${appli.tmp}"/>
        <unzip src="${appli.packages}/${appli.package-name}" dest="${appli.tmp}"   />
        <move todir="${appli.base}/">
                <fileset dir="${appli.tmp}/${appli.name-version}"/>
        </move>
</target>
```

Les propriétés utilisées sont celles-ci:

```
appli.name-version=esup-annuaire2
appli.package-name=${appli.name-version}.zip
appli.name=annuaire2
appli.root=/home/tomcat/portail/BUILD/applis/${appli.name}
appli.base=/home/tomcat/portail/BUILD/applis/${appli.name}/${appli.name}
appli.tmp=${appli.root}/temp
appli.packages=${appli.root}/packages
```

La commande Maven "clean install" va créer un répertoire "target", compiler les différents modules de l'application
On se place ensuite dans le module web de l'application.
La commande Maven -DProduction=true clean package réalise la construction du ficher WAR.
La suite, c'est l'assemblage dans un nouveau web.xml du web.xml livré dans le WAR avec le fichier portlet.xml.