1.5 Du développement à l'exploitation

A relire, compléter

Sommaire :

Utilisation de Maven pour générer le package à distribuer

Maven permet de générer un fichier war directement utilisable dans un serveur d'application Tomcat.

Pour réaliser cette opération, on utilise un "profile" Maven nommé "Production", placé dans le pom.xml du module "vues" de l'application, comme illustré ici:

```
<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>
      <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</warName>
               </configuration>
            </plugin>
         </plugins>
      </build>
</profile>
```

Ce profile va permettre de construire un fichier war prenant en compte les propriétés décrites dans default.properties, éventuellement recouvertes par celles de config.properties pour la partie src/main/resources. Les fichiers web.xml et portlet.xml seront affectées via le plugin "maven-war-plugin" qui permet aussi de nommer le fichier war résultant.

Le lancement de ce profile est obtenu avec cette commande maven (ou son équivalent avec le plugin Eclipse:

```
mvn \-DProduction=true \-PProduction \-B clean package
```

Il faut aussi avoir décrit le packaging comme étantde type war dan s le début du pom.xml en question:

```
<packaging>war</packaging>
```

On renforcera la substitution des variables au moment de l'exécution en plaçant ceci dans le fichier "applicationContext.xml":

```
<context:property-placeholder
                location="classpath:/properties/defaults.properties,classpath:/properties/config.properties" />
```

Utiliser Maven pour travailler sur un dossier de type "webapps"

En complément, il vous faudra modifier les fichiers "log4j.properties" et "web.xml" pour parvenir à exploiter correctement l'application...

Utilisez un profile comme celui-ci dans le pom.xml du module "vues" de l'application:

```
<profile>
   <id>webConfigure</id>
   <activation>
      <property>
         <name>webConfigure</name>
         <value>true</value>
      </property>
   </activation>
   <build>
      <filters>
         <filter>./../../../../WEB-INF/classes/properties/defaults.properties</filter>
         <filter>./../../../../WEB-INF/classes/properties/config.properties</filter>
      </filters>
      <resources>
         <resource>
            <directory>./../../../../WEB-INF</directory>
            <filtering>true</filtering>
         </resource>
      </resources>
      <defaultGoal>compile</defaultGoal>
   </build>
</profile>
```

Le lancement de ce profile est obtenu avec cette commande maven (ou son équivalent avec le plugin Eclipse):

```
mvn compile -DWebConfigure=true
```

depuis le répertoire META-INF/maven/org.esupportail/nom-appli/

Cas particulier du paramétrage de classe

Vous devrez peut-être paramétrer l'utilisation de classe plutôt que l'utilisation de simples variables.
Un exemple: depuis le "domain" d'une application on désire utiliser une classe "maison" à la place d'une classe fournie...
Si le bean Spring du "domain" est inclus dans le module maven du "domain", la substitution de variable ne peut s'appliquer sur lui (il n'est pas dans le module maven "vues" de l'application).

Voici un bean Spring de cet ordre:

```
<bean id="domainService" class="org.esupportail.annuaire2.domain.DomainServiceImpl">
                <property name="structuresService" ref="ldapStructuresService${annuaire2.ldap.schema}" />
...
```

"annuaire2.ldap.schema" predra une valeur via une variable d'environnement (à l'exécution) comme ceci:

```
export "annuaire2.ldap.shema"=Lille1
```

On aura alors le bean Spring suivant qui sera utilisée: ldapStructuresServiceLille1, celui-ci faisant référence à la bonne classe...

```
<bean id="ldapStructuresServiceLille1"
                class="fr.univlille1.annuaire2.ldap.services.StructuresServiceImpl">
                <description>
                        This bean provides LDAP facilities to manage structures.
                </description>
```

Utilisation de la variable @file + passage de paramétre au conteneur d'application : permet de sortir la config de l'arbo de l'appli