|  |
| --- |
| Formation JSF par la pratique  Partie 1 |
| Développement d'une application web avec JSF, JPA et WildFly |

Sommaire

[1. Objet 3](#_Toc546642)

[2. Pré requis 3](#_Toc546643)

[3. Partie 1 : Préparation de l'environnement de développement 4](#_Toc546644)

[4. Partie 2 : Développement de l'application web 39](#_Toc546645)

[**A.** **Le modèle :** 43](#_Toc546646)

[**B.** **Couche DAO :** 44](#_Toc546647)

[**C.** **Couche Métier :** 46](#_Toc546648)

[**D.** **Le Value Object** 47](#_Toc546649)

[**E.** **Couche Présentation :** 50](#_Toc546650)

[**F.** **web.xml :** 55](#_Toc546651)

[**G.** **Test** 55](#_Toc546652)

[5. Partie 3 : Intégration JSF avec JPA 62](#_Toc546653)

[6. Partie 4 : Enrichissement de la couche présentation 71](#_Toc546654)

[6.1. L'internationalisation 71](#_Toc546655)

# 1. Objet

Le présent document a pour objet de vous montrer étape par étape comment développer et configurer une application web sous Wildfly avec l'api JSF (Java Server Faces).

Ci-après les étapes que nous allons suivre :

* **Etape 1: Préparer l'environnement de développement.** Par rapport à cette première étape, nous allons effectuer les tâches suivantes :
* Installer MySQL (le serveur et l'interface graphique de l'administrateur).
* Créer un schéma.
* Installer WildFly.
* Créer un compte administrateur au niveau de WildFly.
* Créer un compte utilisateur au niveau de WildFly.
* Ajouter WildFly au niveau d'Eclipse Luna.
* **Etape 2 : Développer l'application web**.
* **Etape 3 : Développer la couche DAO avec JPA**. Dans cette 3ième partie, vous allez apprendre l'intégration de JSF et JPA dans le même projet WEB.
* **Etape 4 : Implémenter le principe de l'internationalisation** pour que l'utilisateur puisse choisir la langue qui sera utilisée par l'application.

# 2. Pré requis

Les outils que nous allons utiliser pour réaliser ce TP sont présentés au niveau du tableau ci-après :

|  |  |
| --- | --- |
| Outil | Chemin du fichier  voir dossier : *formation\_Framework* |
| Eclipse Luna | formation\_Framework\tools\eclipse-luna.rar |
| WildFly 8.2.0 | formation\_Framework\tools\wildfly-8.2.0.Final.zip |
| Le Driver MySQL 5.0 | formation\_Framework\tools\lib\_divers\mysql-connector-java-5.1.17-bin.jar |
| Le serveur MySQL 5.0 | formation\_Framework\tools\MySQL 5.0\Setup.exe |
| Le client (l'interface graphique) | formation\_Framework\tools\MySQL 5.0\mysql-gui-tools-5.0-r17-win32.msi |
| JDK 1.7 | formation\_Framework\tools\ jdk-7u75-windows-x64.exe |
| L'implémentation Hibernate de JPA. | Les jars sont disponibles au niveau du chemin :  formation\_Framework\tools\lib-hibernate |

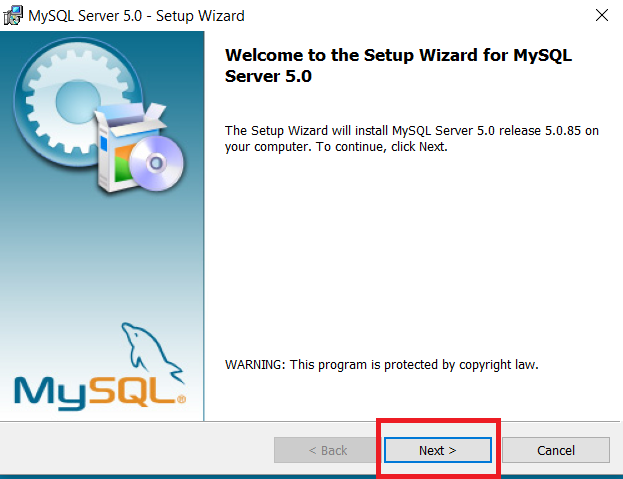
**NB1 :** **Nous n'avons pas besoin de télécharger une implémentation de JSF étant donné que WildFly utilise par défaut Mojara comme implémentation de JSF. Au cas ou vous voulez utiliser Tomcat, vous devriez télécharger une implémentation de JSF.**

**NB2 : De même pour JPA. En effet, WildFly utilise par défaut Hibernate comme implémentation de JPA. Au cas ou vous voulez utiliser Tomcat, vous devriez télécharger une implémentation de JPA.**

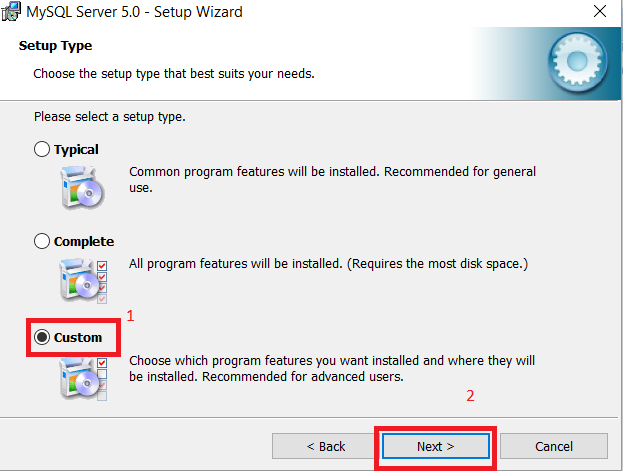
# 3. Partie 1 : Préparation de l'environnement de développement

**Installation de la base de données MySQL 5.0:**

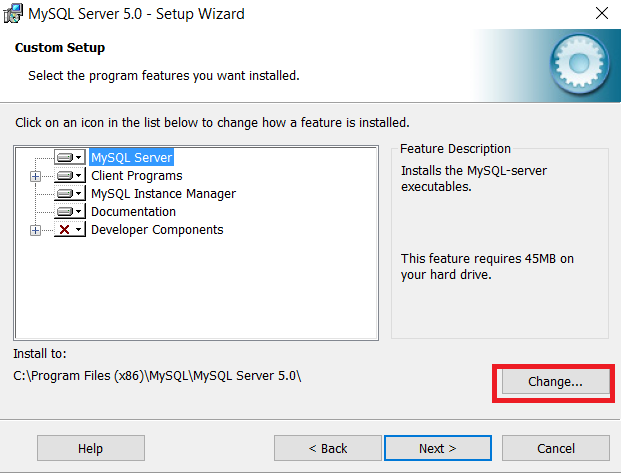
Double cliquer sur le fichier **setup.exe** et suivre les étapes suivantes :



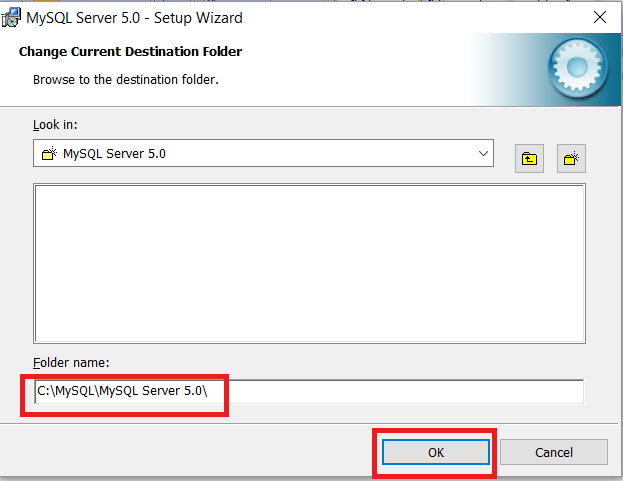
cliquer sur **Next>:**

****

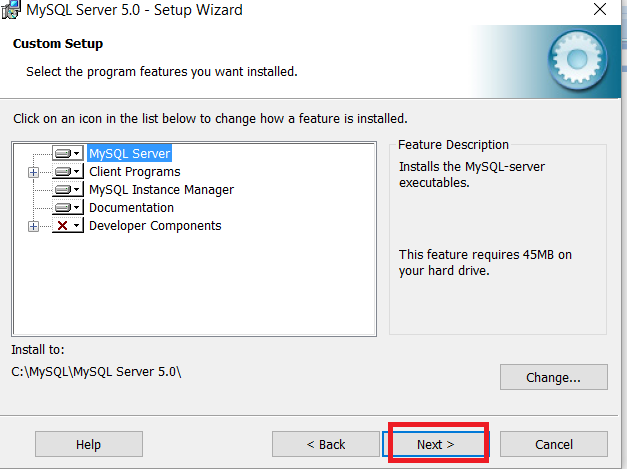
Cocher ***Custom*** et cliquer sur le bouton ***Next>*** :



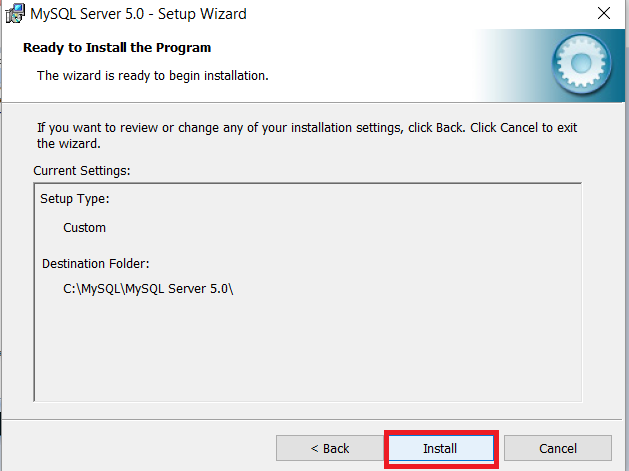
Cliquer sur ***Change... :***

******

Entrer le dossier dans lequel sera installé MySQL (vous devriez avoir le droit en lecture et en écriture sur ce répertoire) et cliquer sur le bouton ***OK*** :



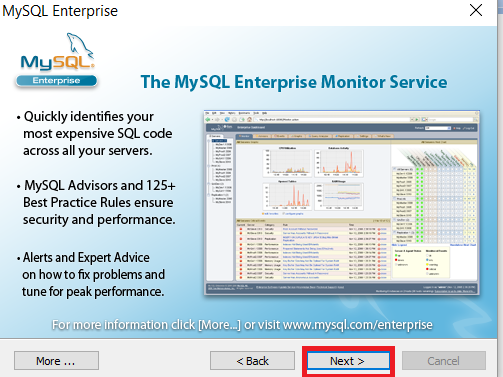
Cliquer sur le bouton ***Next>*** :



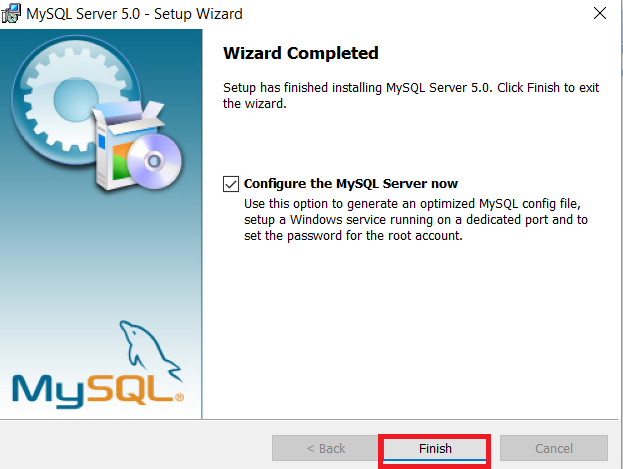
Cliquer sur ***Install*** :

Une fois l'écran suivant est affiché, cliquer sur le bouton ***Next>*** :

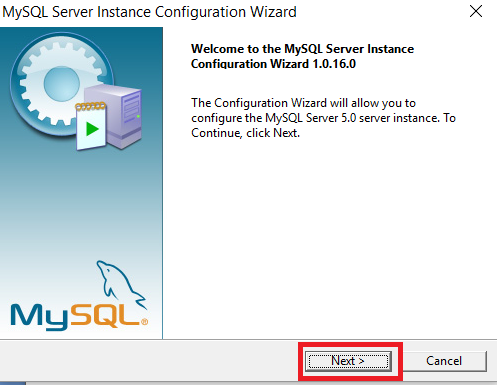




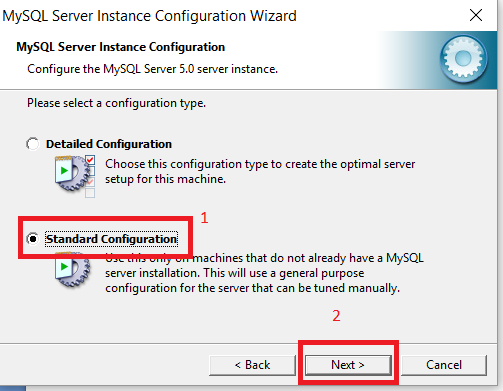
Cliquer sur ***Next> :***



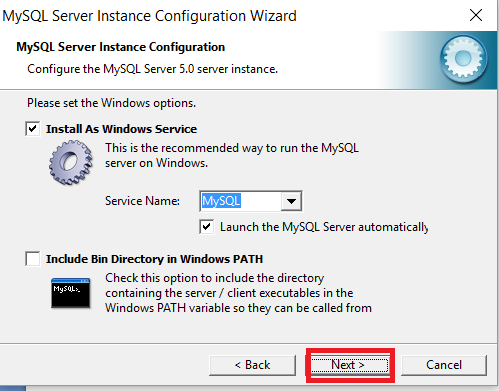
Cliquer sur ***Finish*** :



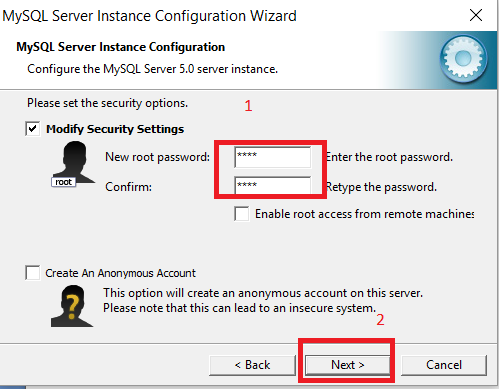
Cliquer sur le bouton ***Next>*** :



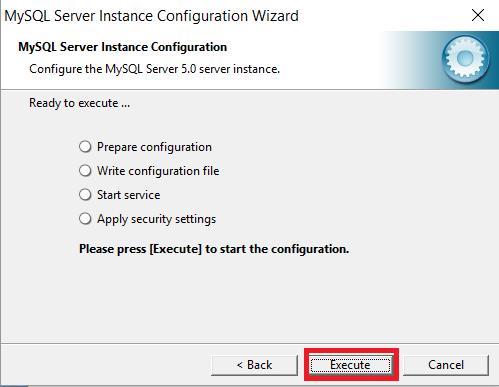
Cocher "***Standard Configuration"*** et cliquer sur le bouton ***Next> :***



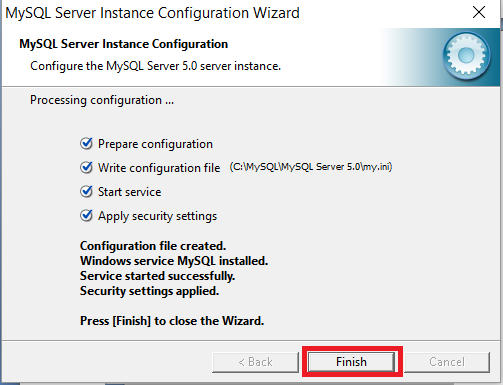
Cliquer sur le bouton ***Next>*** :



Entrer le mot de passe du compte ***root*** et cliquer sur ***Next >*** :



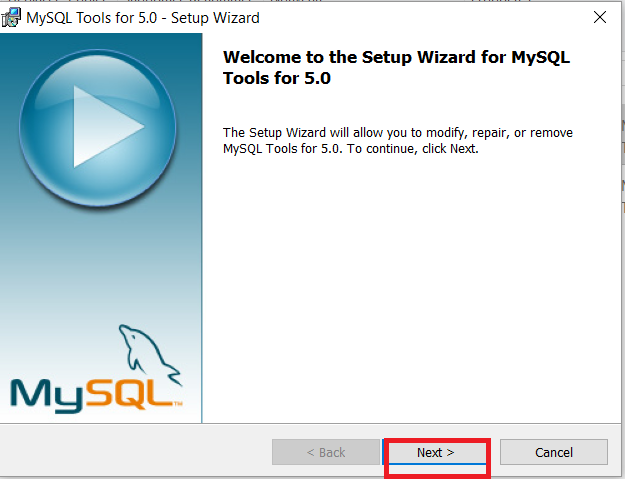
Cliquer sur le bouton ***Execute*** :



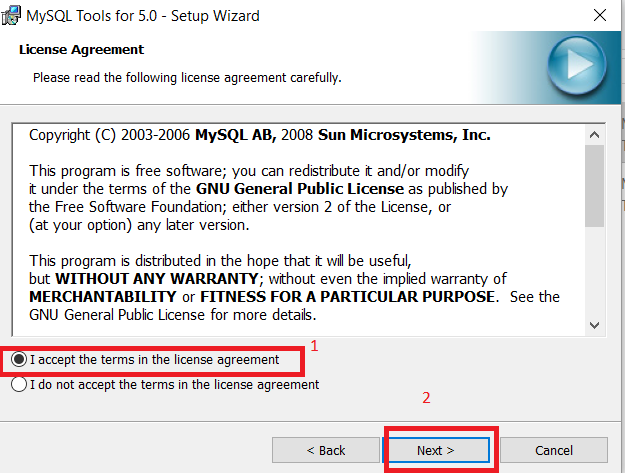
Une fois l'installation terminée, cliquer sur le bouton ***Finish***.

**Installation de l'administrateur (interface client) de la base de données MySQL 5.0:**

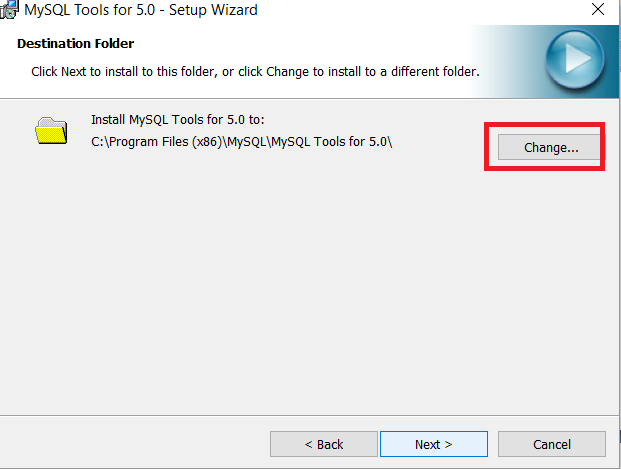
Cliquer sur *mysql-gui-tools-5.0-r17-win32 .msi* et suivre les étapes suivantes :



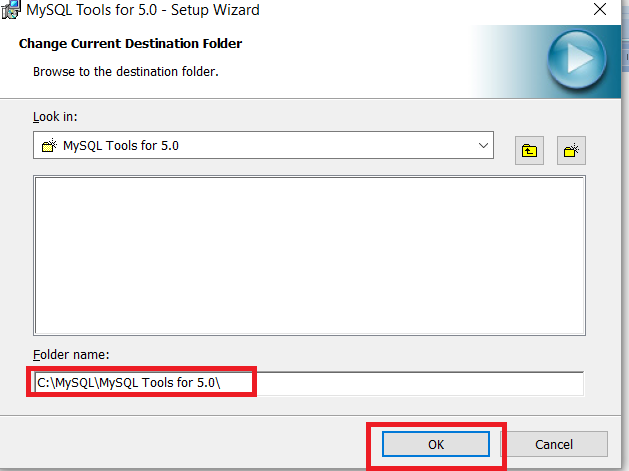
Cliquer sur le bouton ***Next>*** :



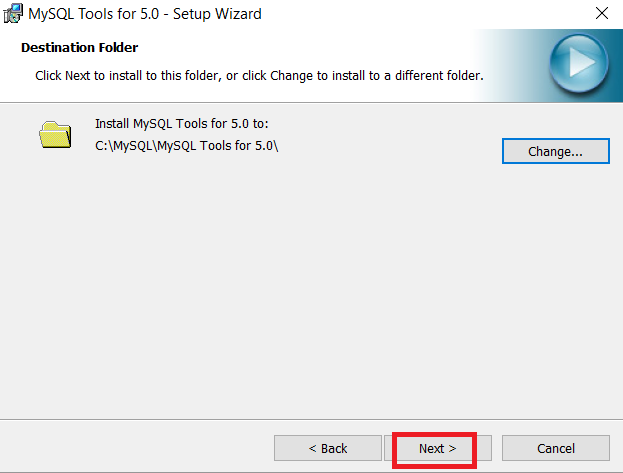
Cocher "***I accept ..."*** et cliquer sur le bouton ***Next>*** :



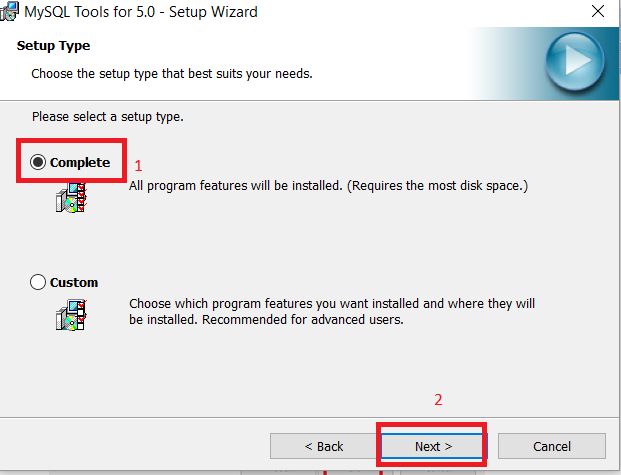
Cliquer sur le bouton ***Change...*** :



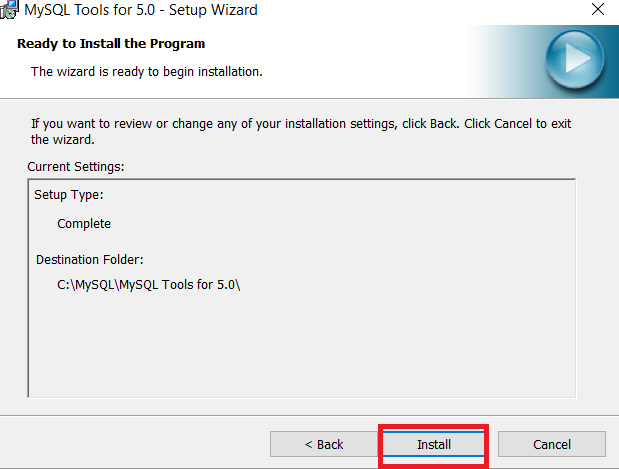
Entrer le chemin d'installation et cliquer sur le bouton ***OK*** :



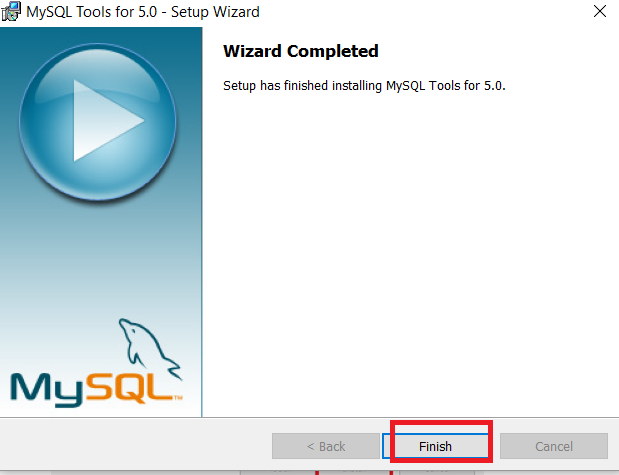
Cliquer sur le bouton ***Next>*** :



Cocher la case à cocher ***Complete*** et cliquer sur le bouton ***Next>*** :



Cliquer sur le bouton ***Install*** :

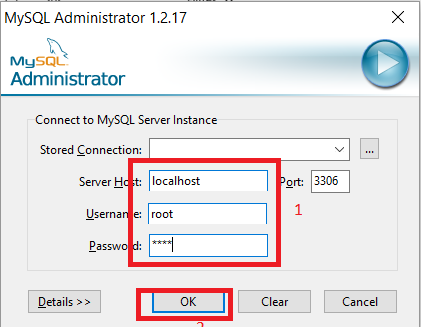


Cliquer sur le bouton ***Finish*** :

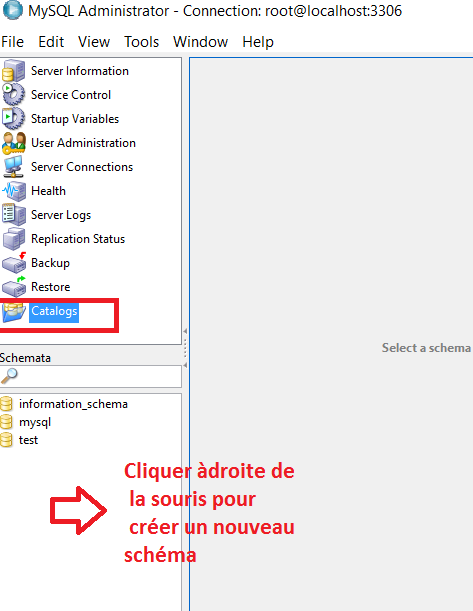
**Création d'un schéma :**

**Lancer "MySQL Administrator" :**

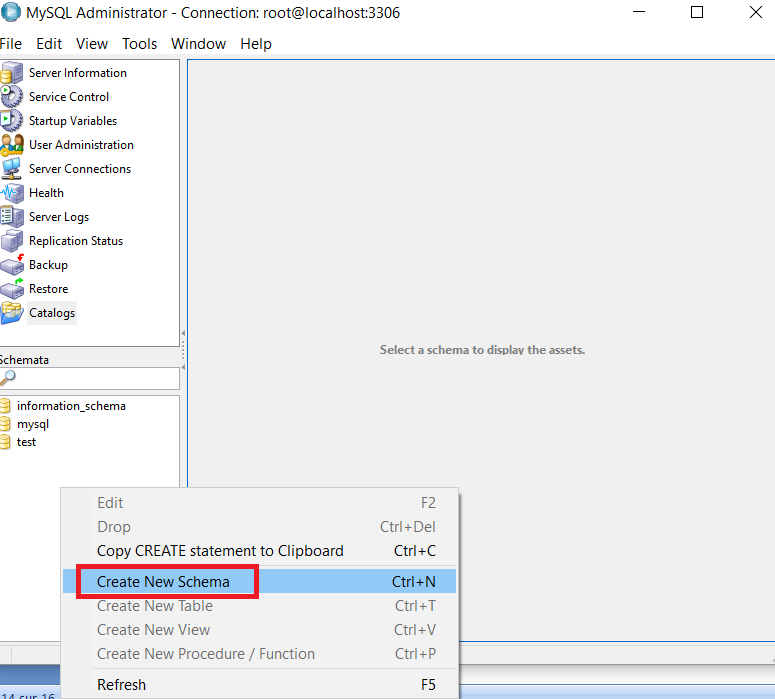
****



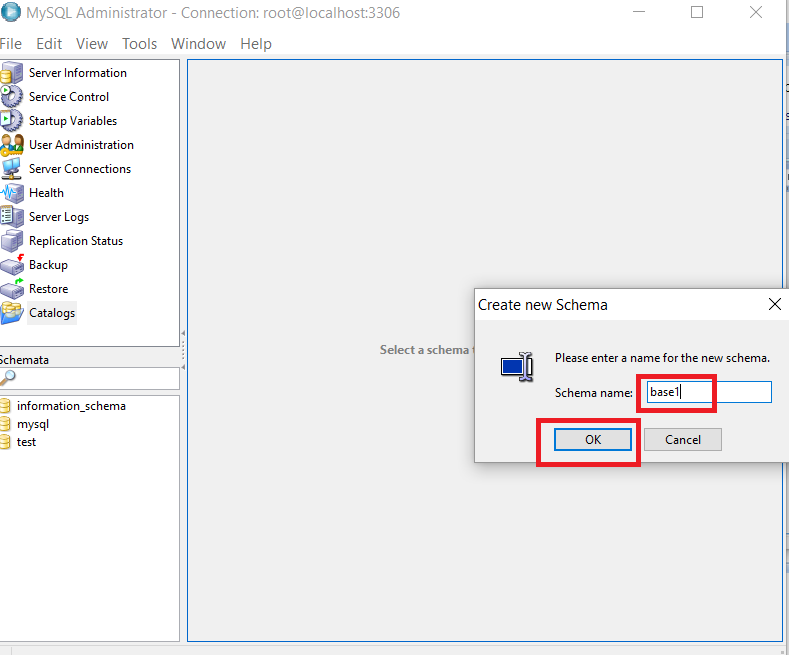
entrer le login/password et cliquer sur le bouton ***OK*** :



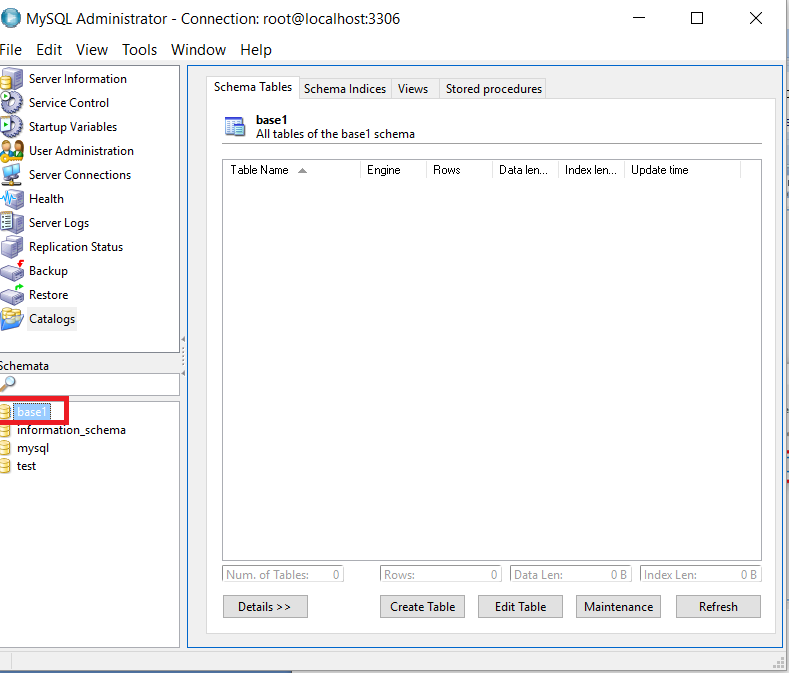
Cliquer sur ***Catalogs*** et cliquer ensuite à droit de la souris comme illustré ci-dessus :



Cliquer sur ***Create New Schema*** :

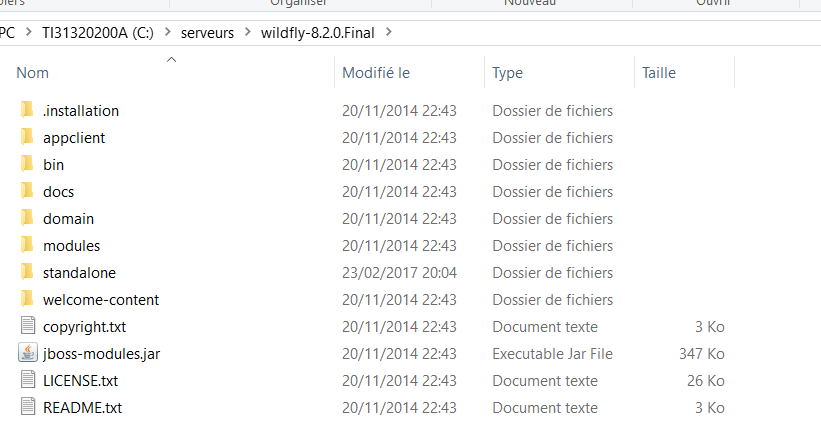


Entrer le nom de votre nouveau schéma et cliquer sur le bouton ***OK***. le schéma sera finalement crée comme illustre l'écran suivant :



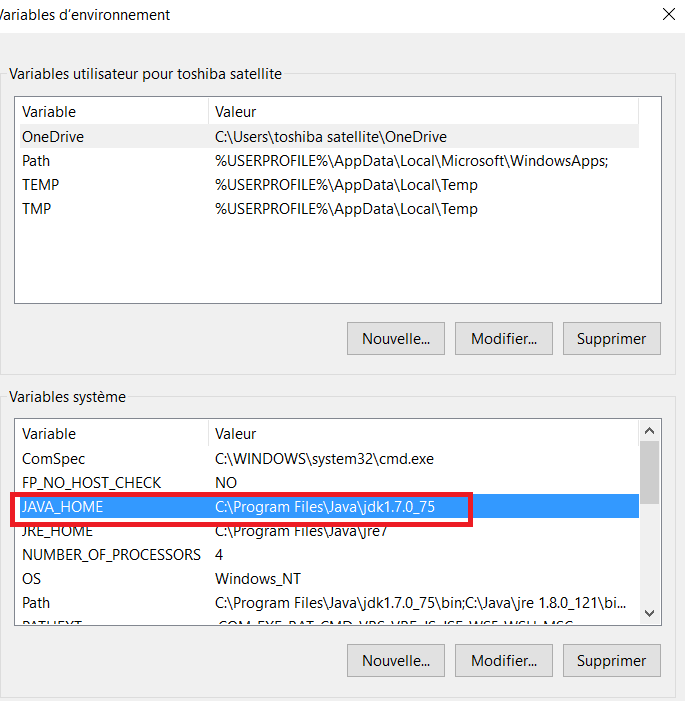
**Installation et configuration de WildFly 8.2.0:**

Décompresser le fichier *wildfly-8.2.0.Final.zip* et copier le dans un dossier de votre choix (par exemple : c:\serveurs\wildfly-8.2.0.Final) :

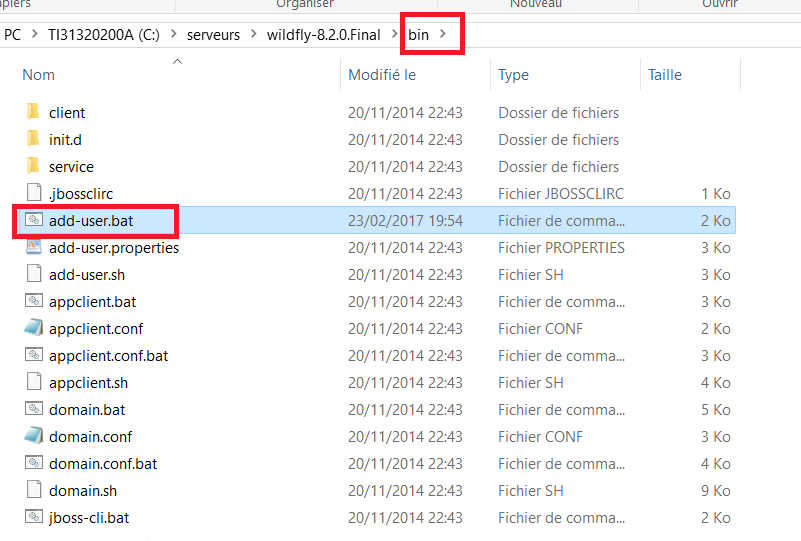


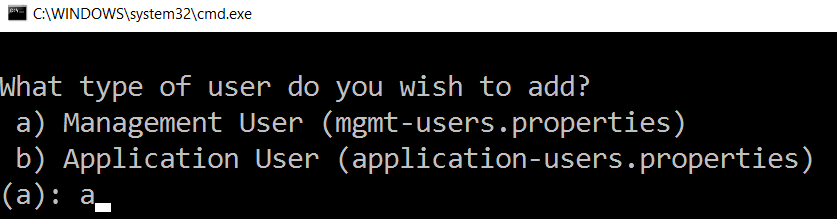
**Ajouter un compte administrateur :**

Avant d'ajouter votre compte administrateur, il faut ajouter la variable d'environnement ***JAVA\_HOME*** comme illustre l'écran suivant :

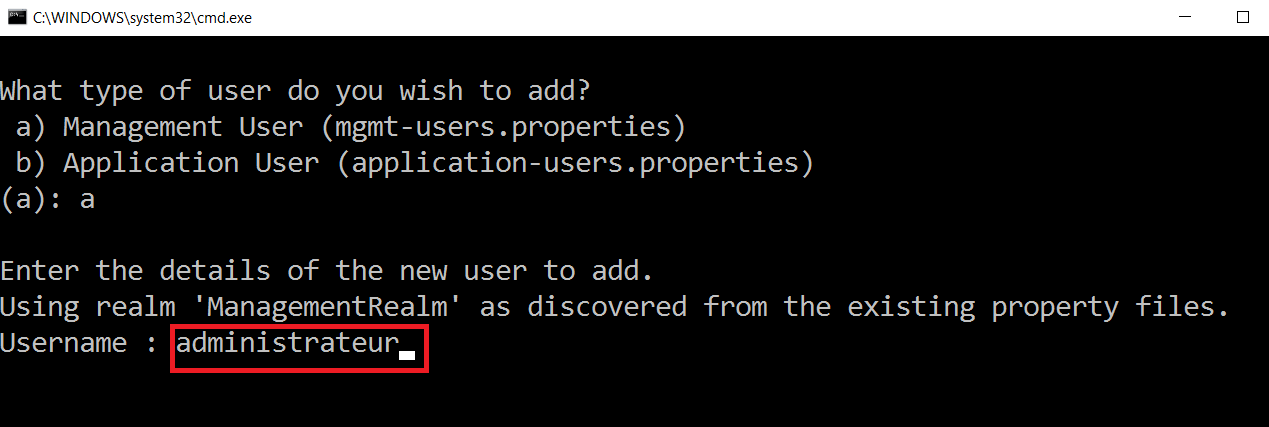


Double cliquer le fichier ***add-user.bat*** (ce fichier se trouve dans le dossier c:\serveurs\wildfly-8.2.0.Final\bin) et suivez les étapes suivantes :

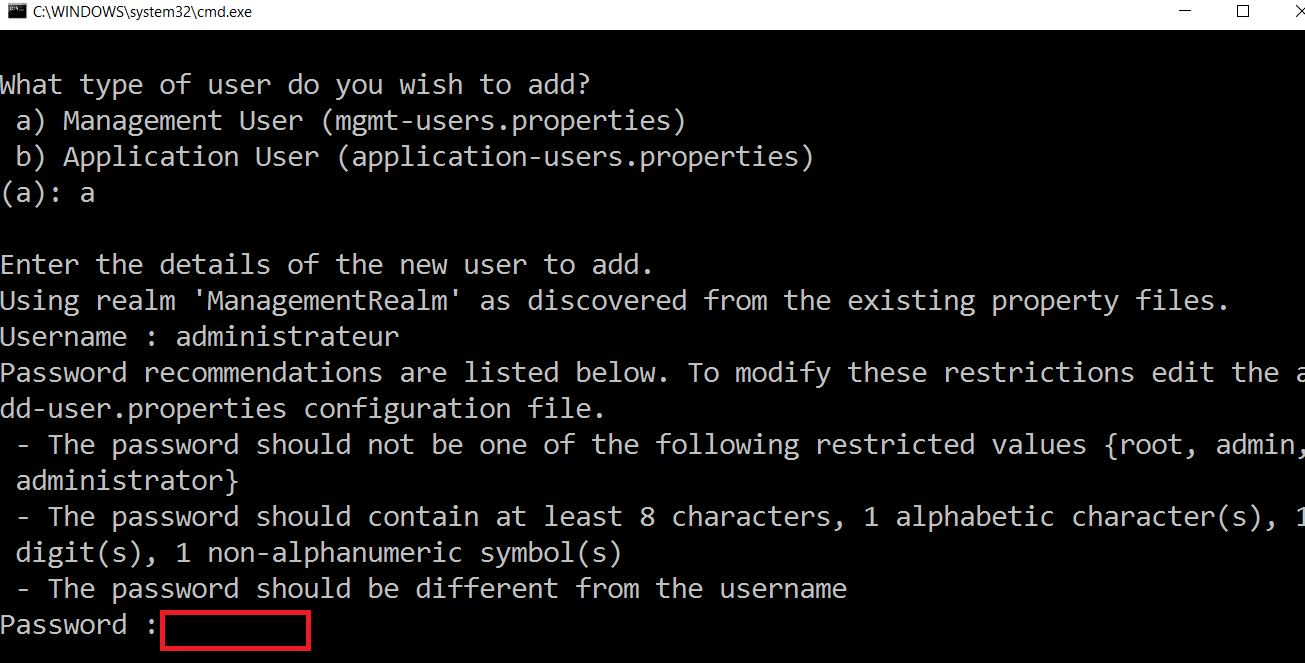
****

****

Saisir *a* et cliquer sur ***Entrer*** :

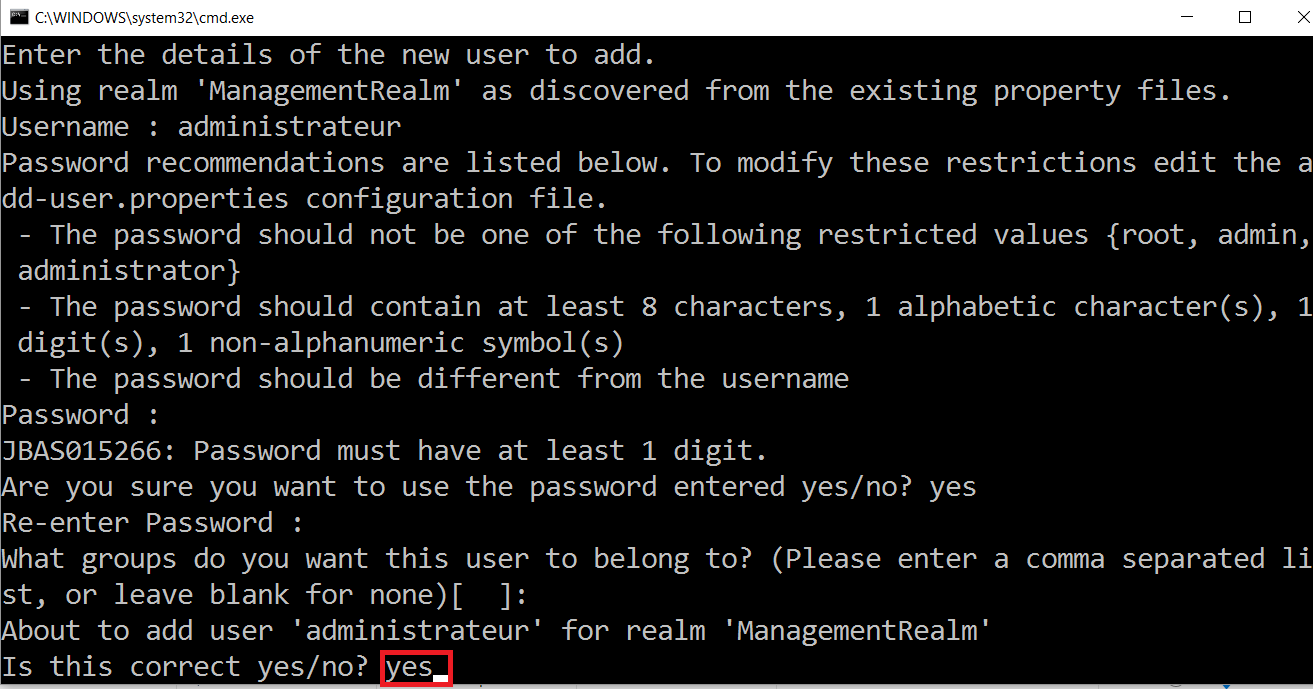
****

Saisir le nom de votre administrateur et cliquer sur ***Entrer*** :

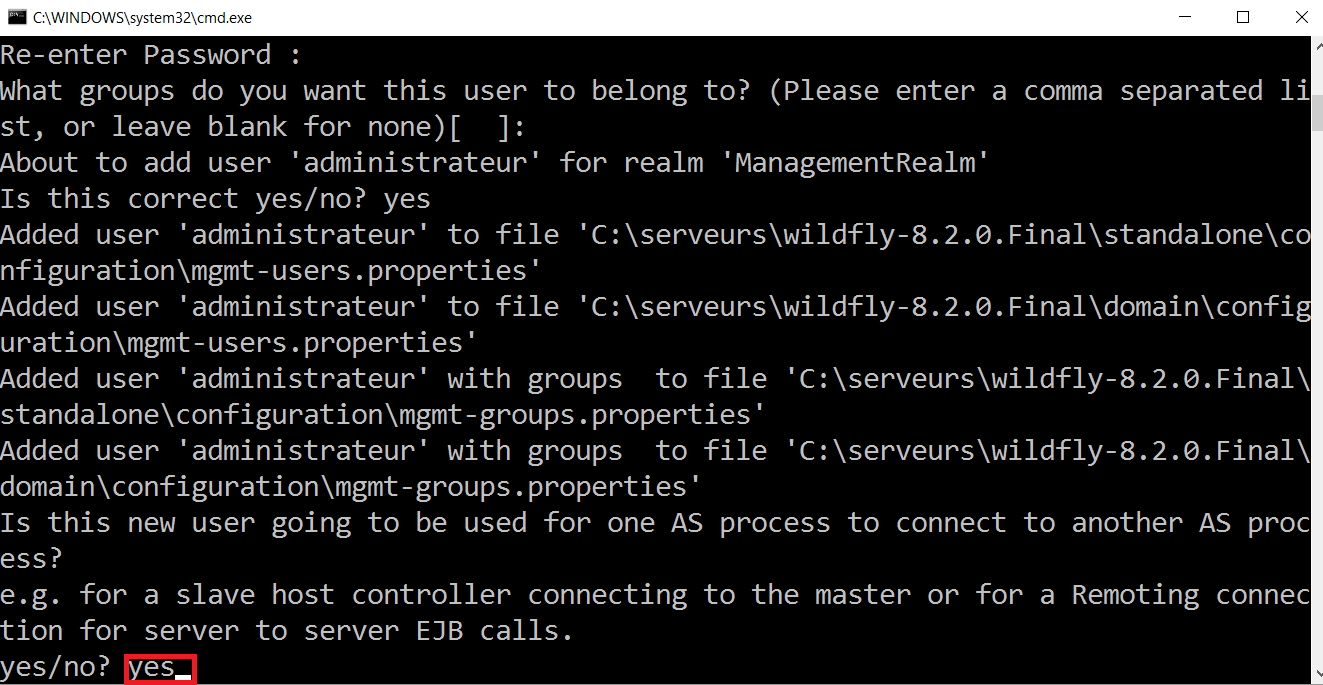
****

Saisir votre Password et cliquer sur ***Entrer*** :

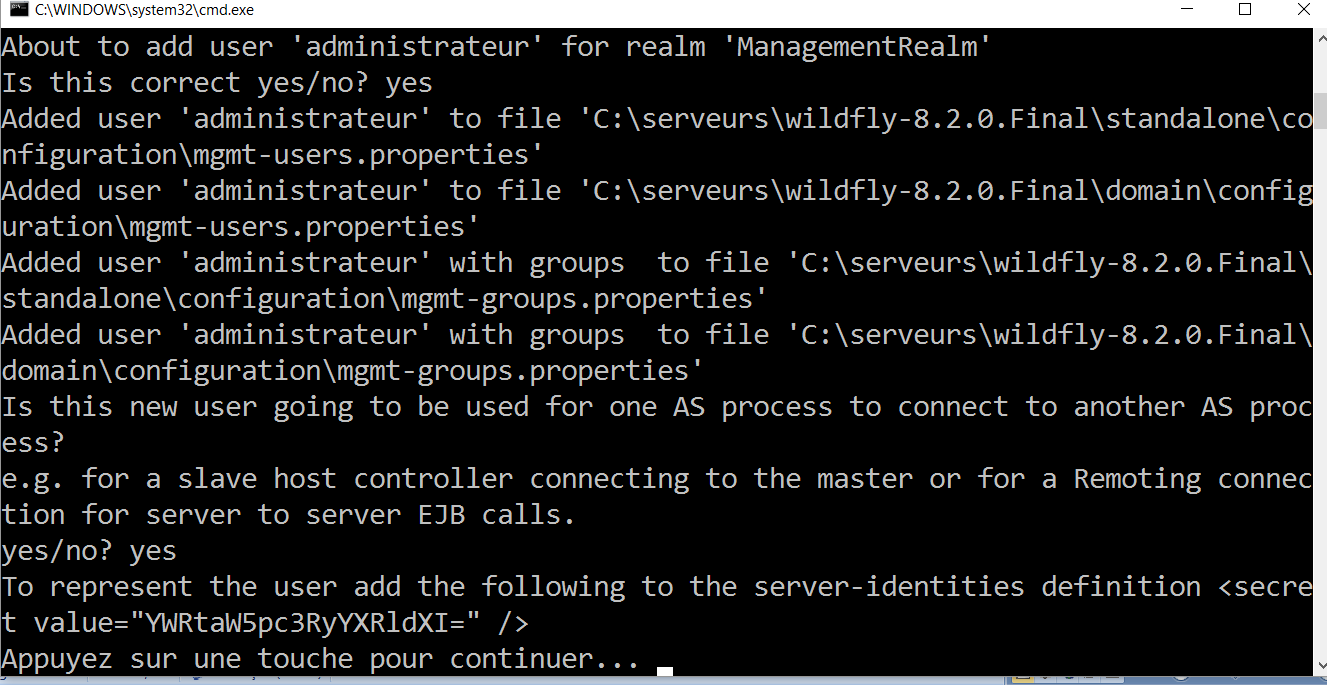
L'outil vous demandera de confirmer le mot de passe. Saisir le même mot de passe :



Saisir ***yes*** et cliquer sur ***Entrer*** :

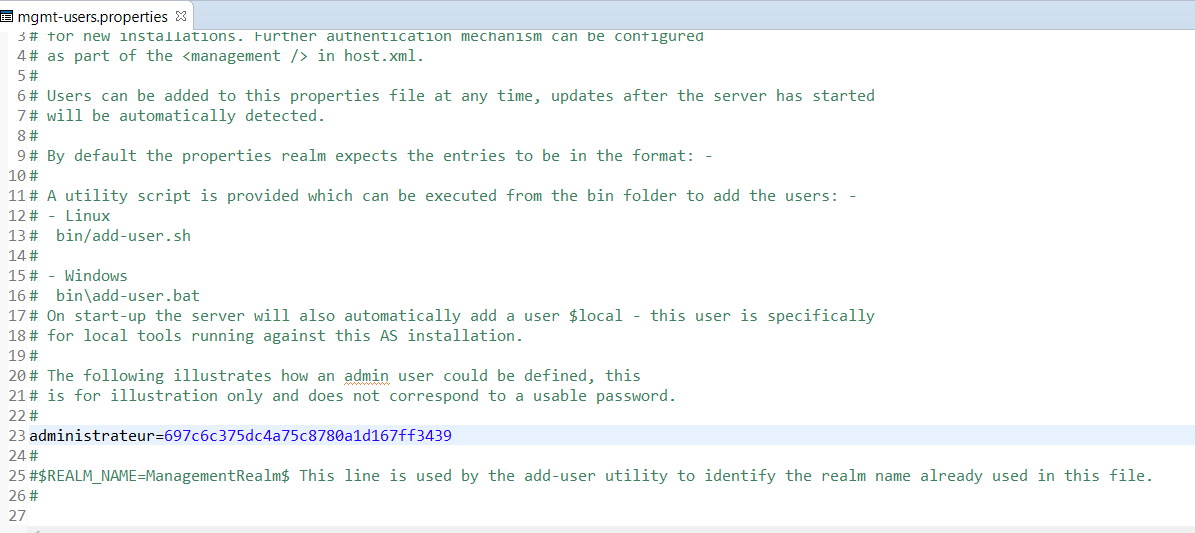


Saisir ***yes*** et cliquer sur ***Entrer*** :

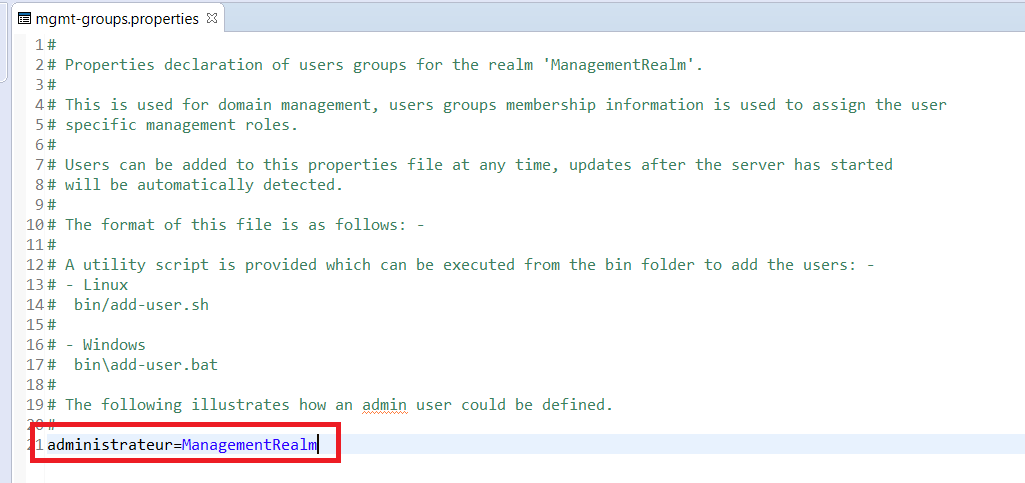
****

Cliquer ensuite sur ***Entrer***.

Vérifier que votre administrateur a été bien ajouté au niveau du fichier ***mgmt-users.properties*** (ce fichier se trouve dans le répertoire [VOTRE\_DOSSIER\_INSTALLATION]\domain\configuration) comme illustre l'écran suivant :

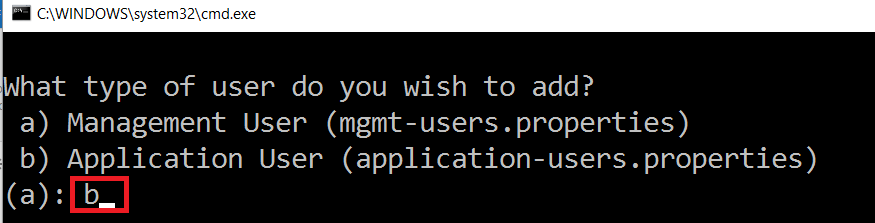


Vérifier également que le rôle de votre administrateur a été bien ajouté au niveau du fichier ***mgmt-groups.properties*** (ce fichier se trouve dans le répertoire c:\serveurs\wildfly-8.2.0.Final \domain\configuration) comme illustre l'écran suivant :

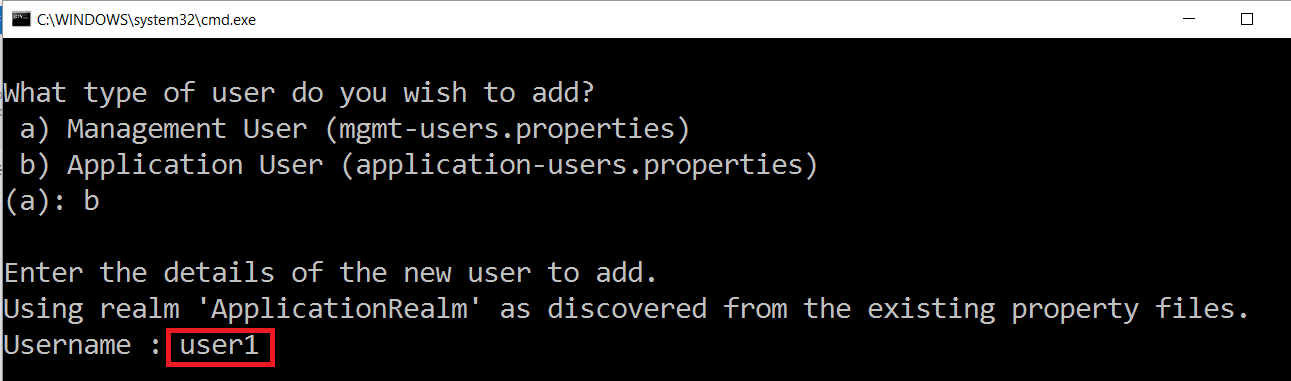


**Ajouter un compte utilisateur :**

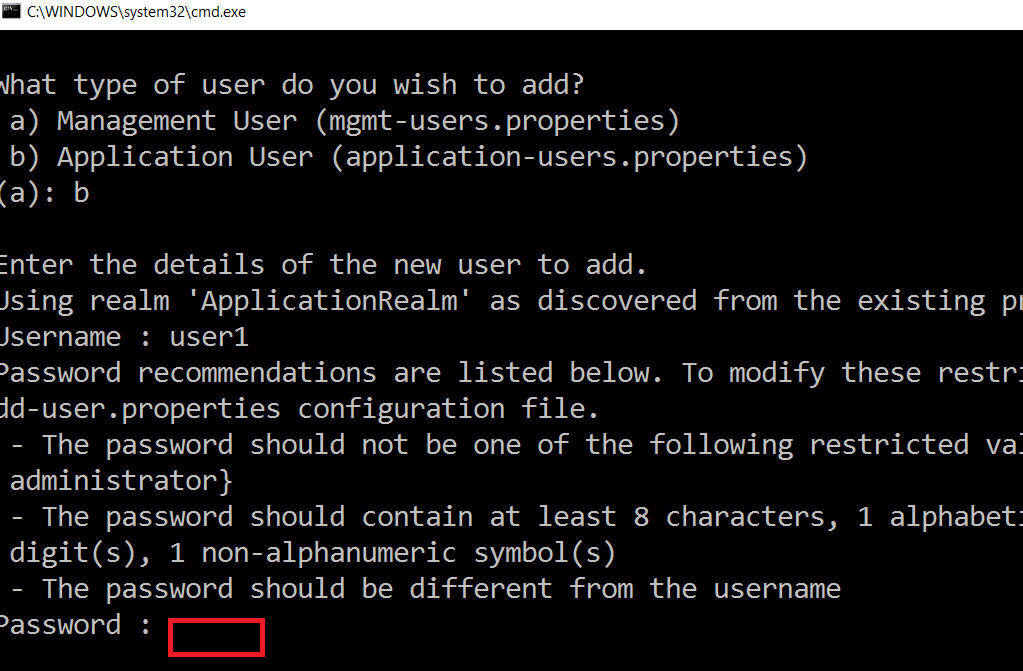
Pour ajouter un compte utilisateur, double cliquer sur le fichier ***add-user.bat*** (ce fichier se trouve dans le dossier c:\serveurs\wildfly-8.2.0.Final \bin) et suivez les étapes suivantes :



Saisir ***b*** et cliquer sur ***Entrer*** :

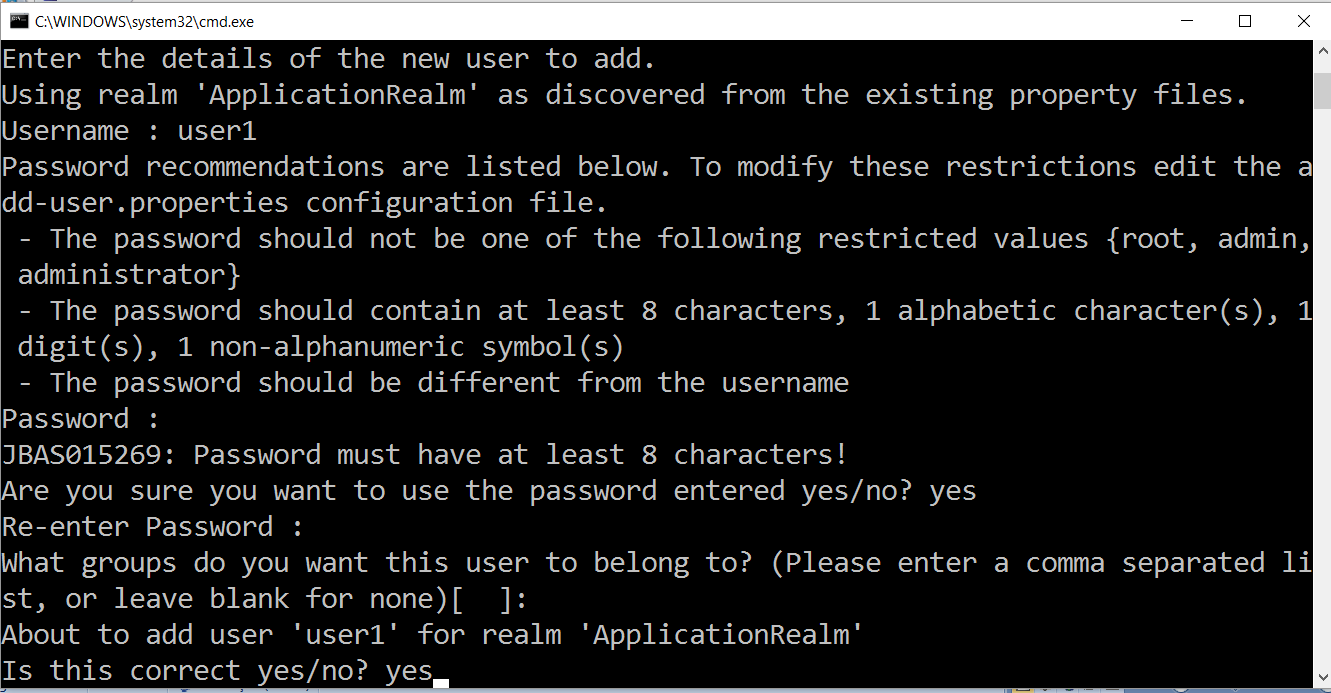


Entrer votre utilisateur et cliquer sur ***Entrer*** :

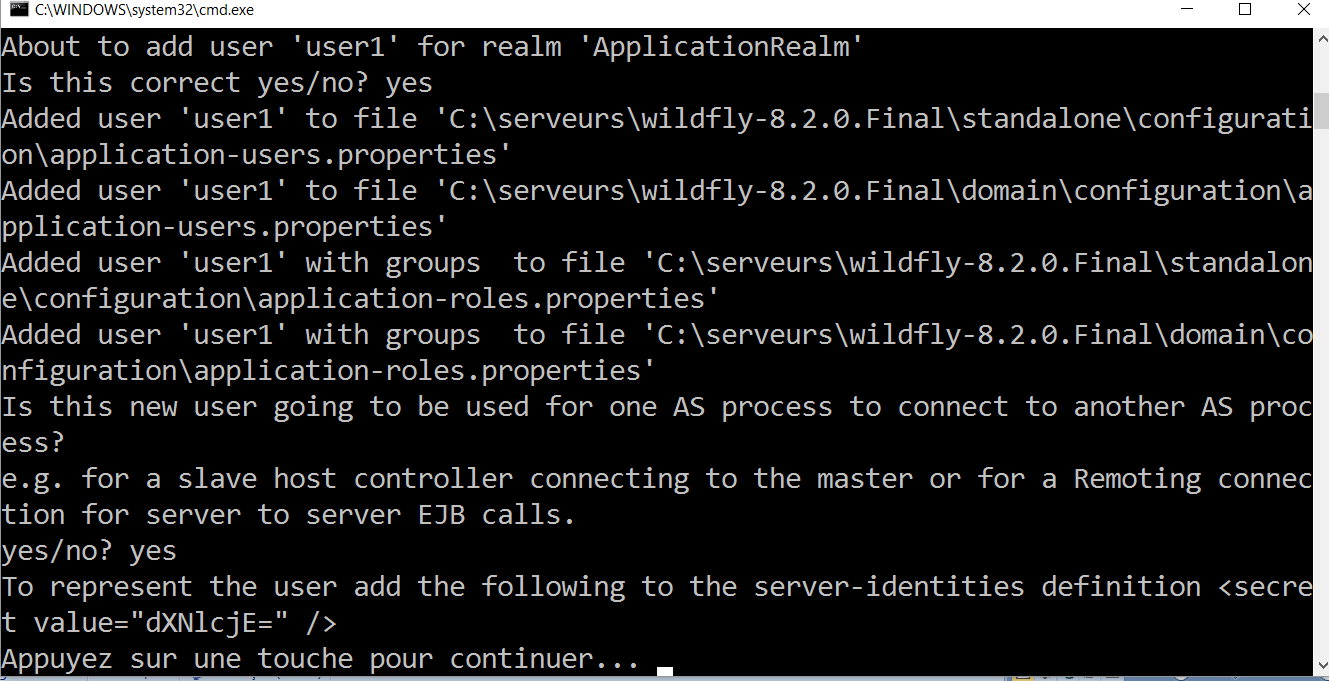


Entrer le mot de passe et cliquer sur ***Entrer*** :

L'application demandera par la suite de confirmer le mot de passe.

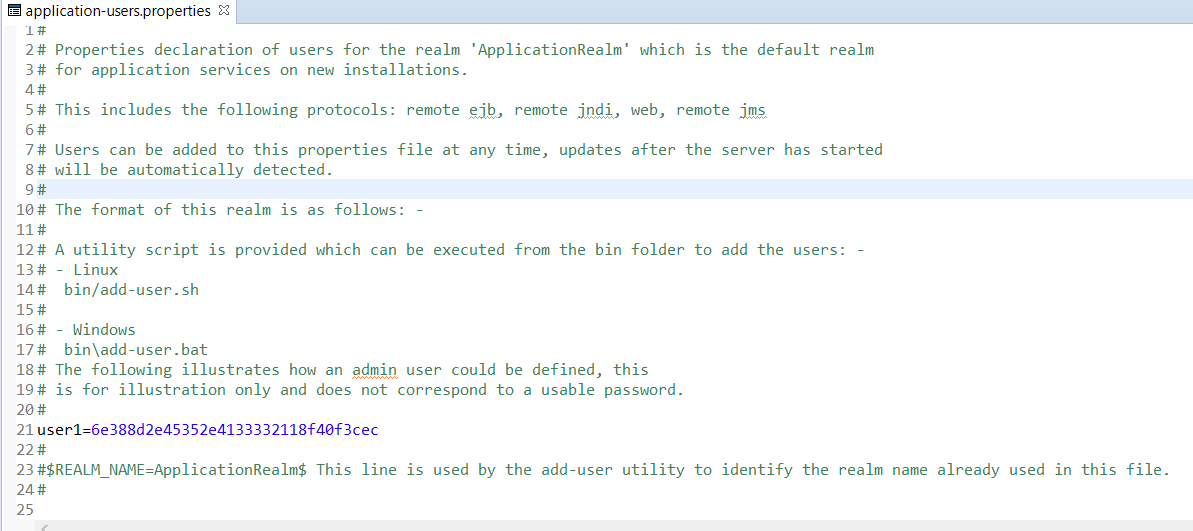


Entrer ***yes*** et cliquer sur ***Entrer***.

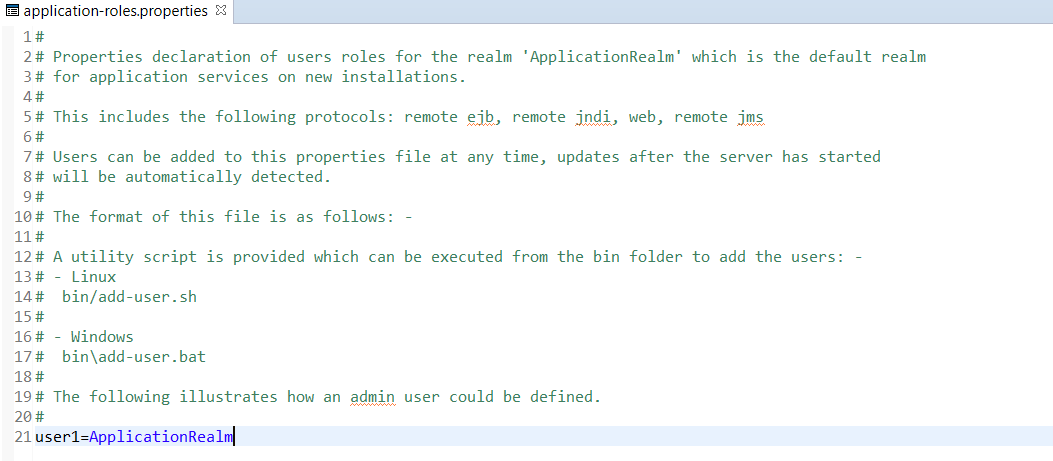


Cliquer enfin sur ***Entrer***.

Vérifier que votre utilisateur a été bien ajouté au niveau du fichier ***application-users.properties*** :

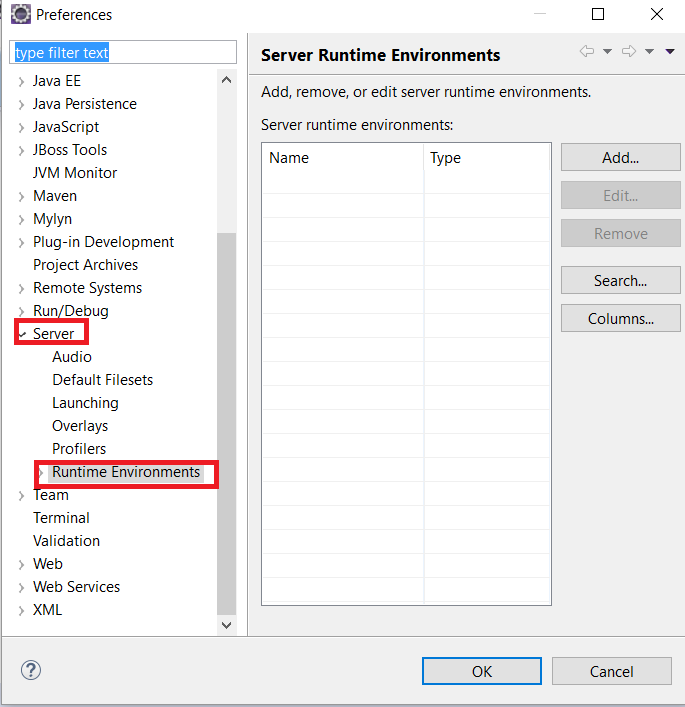


Vérifier également que le fichier ***application-roles.properties*** a été mis à jour avec le rôle de votre utilisateur :

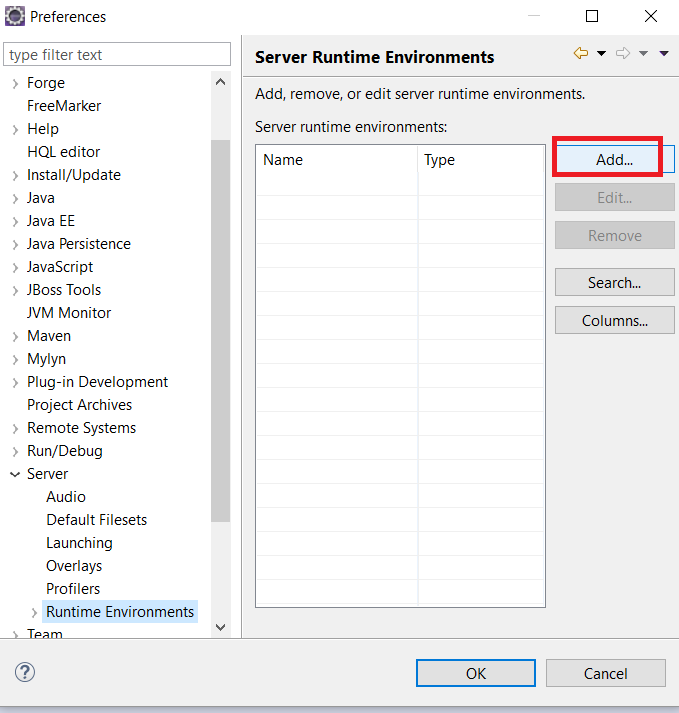


**Configuration de WildFly au niveau d'Eclipse:**

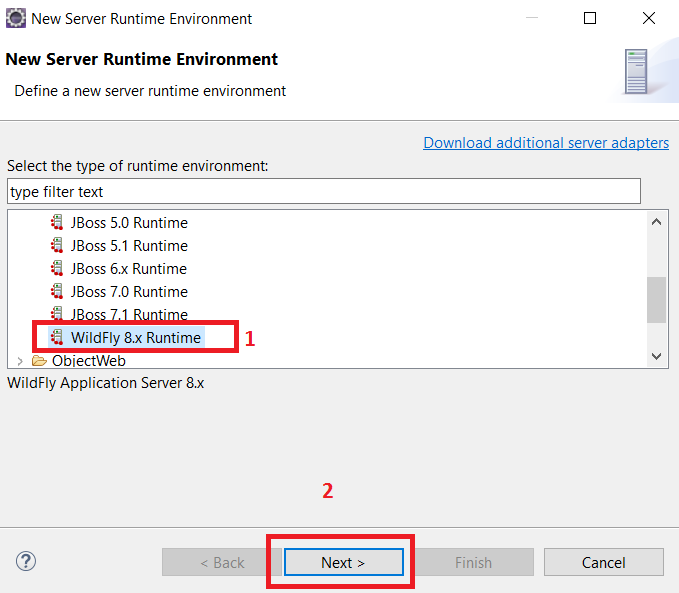
Dans Eclipse Luna, cliquer sur ***Window ==>Preferences*** :

****

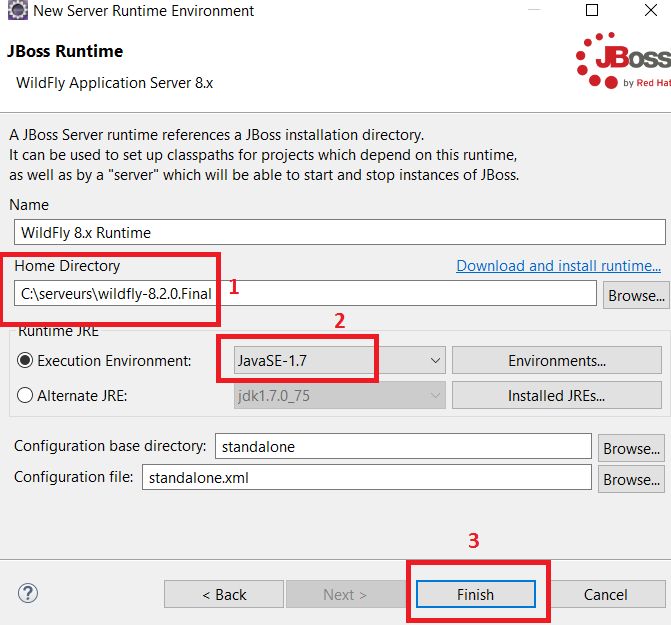
Cliquer sur ***Server=> Runtime Environments :***

******

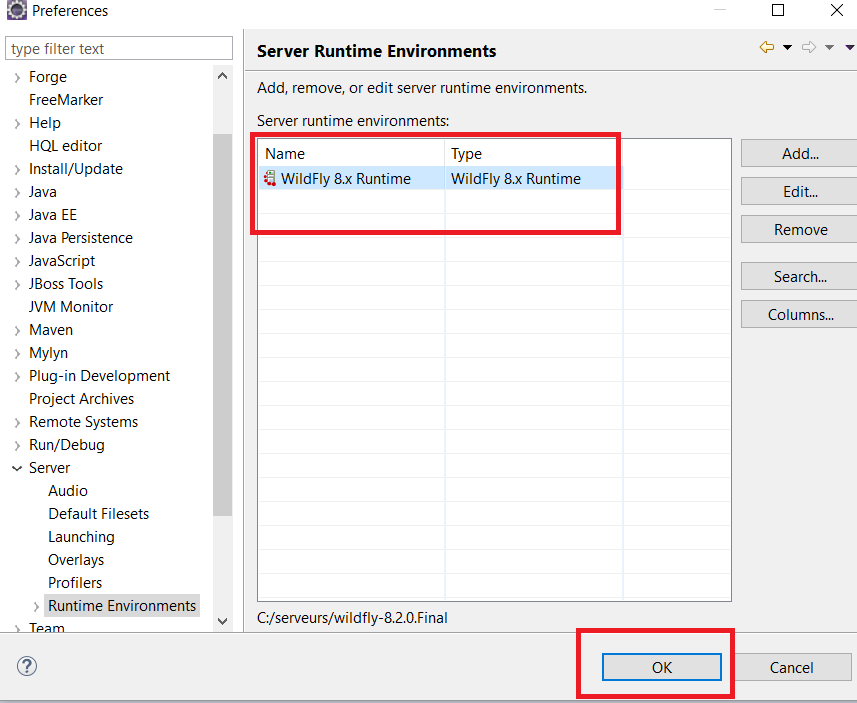
Cliquer sur le bouton ***Add... :***

******

Sélectionner ***WildFly 8.x Runtime*** et cliquer sur ***Next>*** :



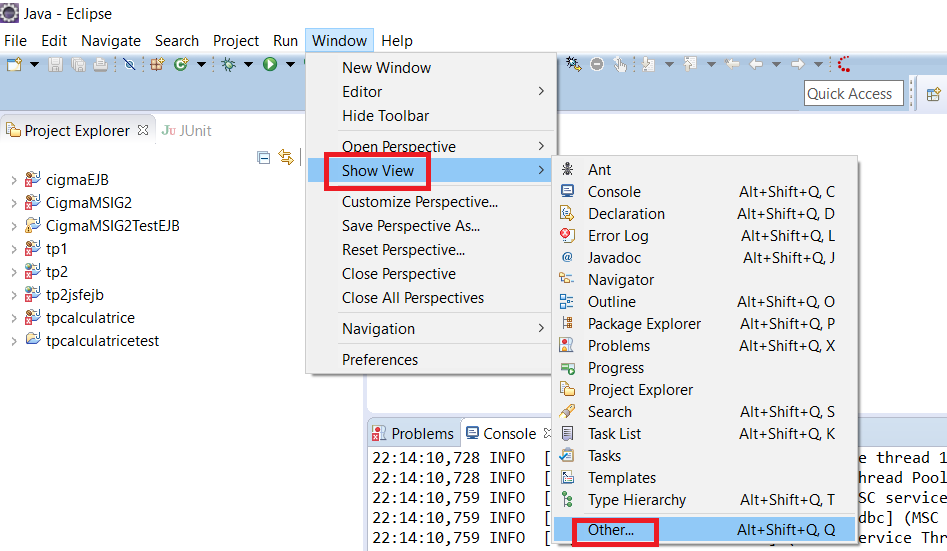
Entrer le chemin d'installation de WildFly, la JVM et cliquer ensuite sur ***Finish*** :



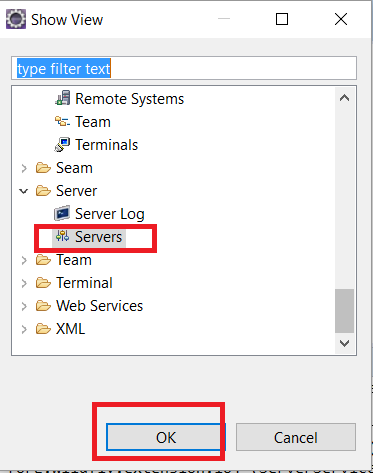
Cliquer ensuite sur ***OK***.

**Ajouter le serveur WildFly au niveau de la vue Servers :**

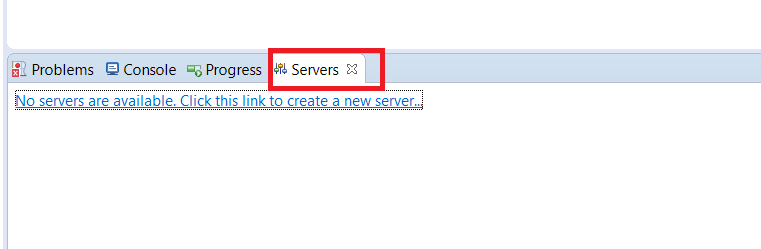
Cliquer sur **Winow==>Show View==>Other...** comme illustre l'écran suivant :



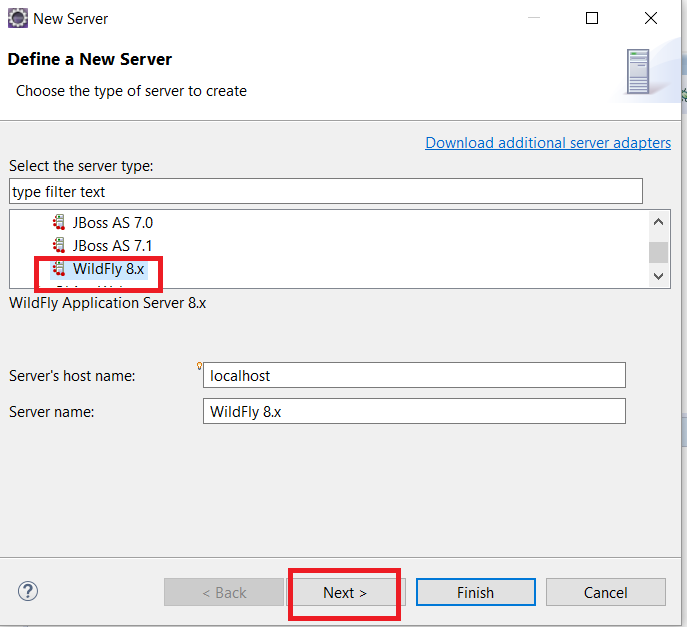
Cliquer sur ***Other...***

****

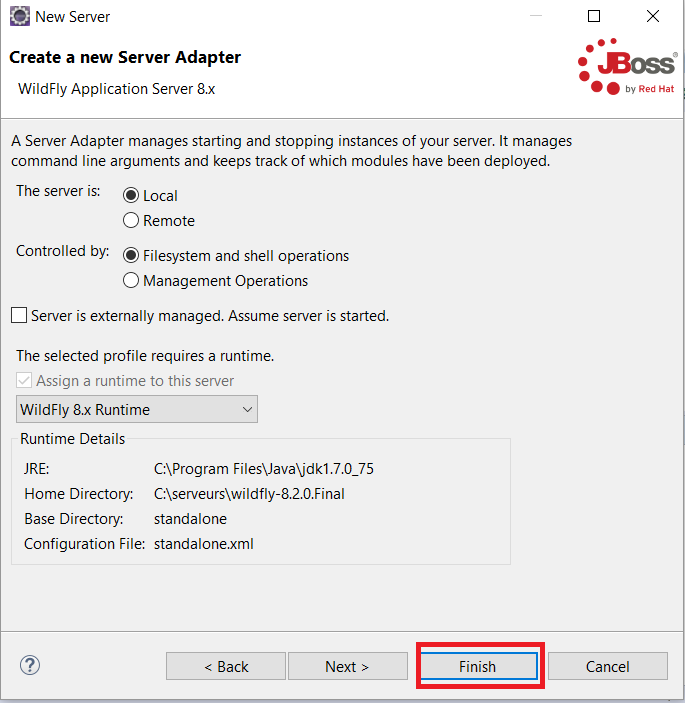
Cliquer sur **Servers** et cliquer ensuite sur le bouton ***OK*** :



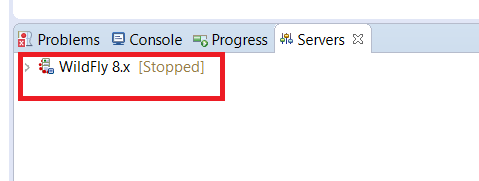
Cliquer sur le lien ci-dessus et suivez les tapes suivantes :



Sélectionner ***WildFly 8.x*** et cliquer sur ***Next>*** :



Cliquer sur le bouton ***Finish*** :

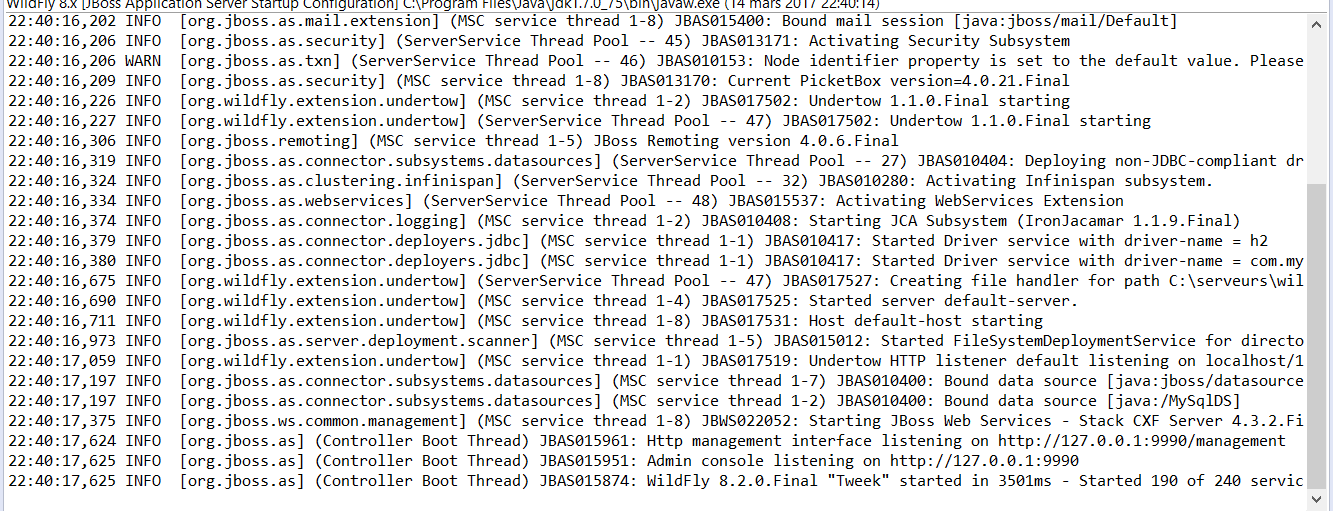


Le serveur est ajouté comme illustré dans l'écran ci-dessus.

Pour démarrer le serveur, cliquer sur l'icon comme illustré dans l'écran ci-dessous :

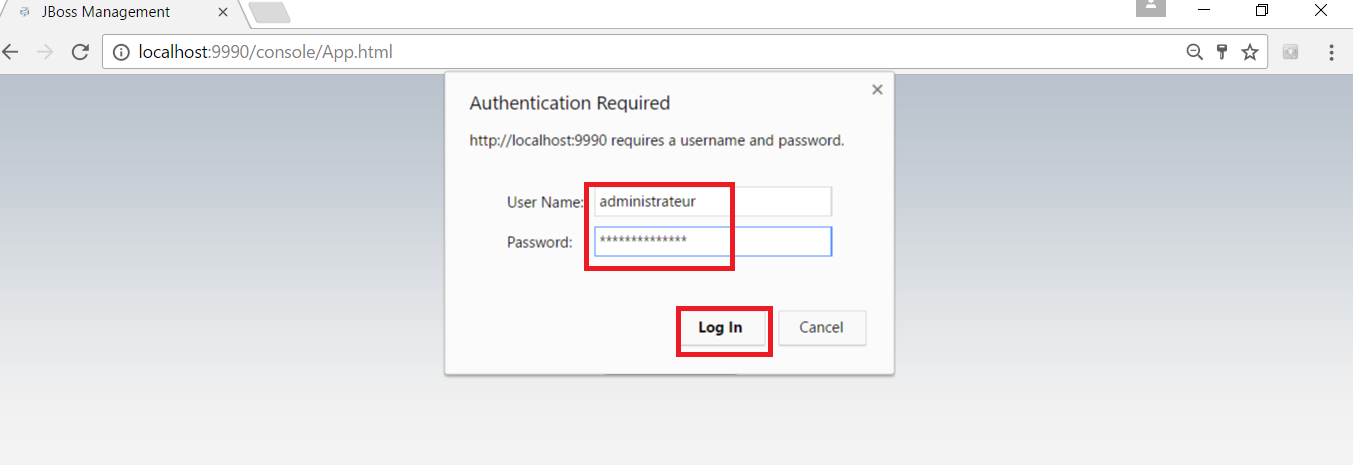


Vérifier que le serveur est bien démarré et qu'il n'y a aucune erreur au niveau de la log (voir la vue Console) :

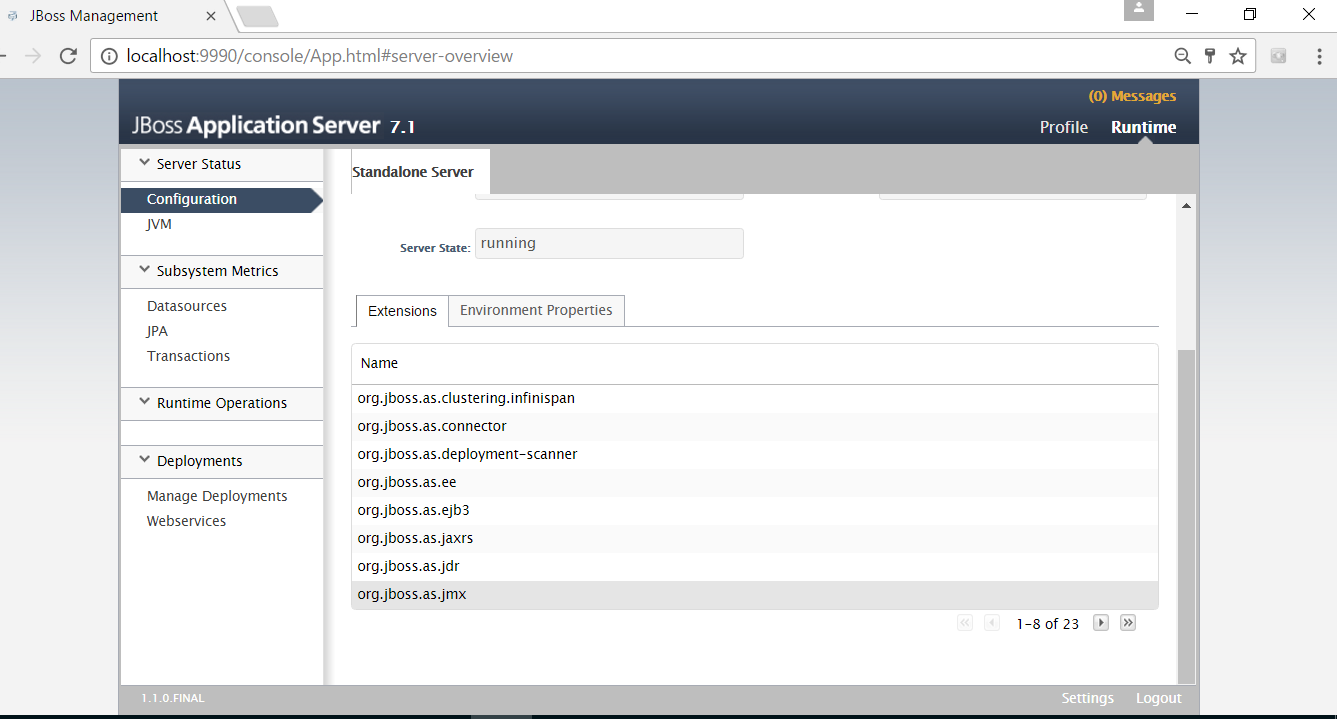


**Accès à la console d'administration de WildFly :**

Lancer le lien ***http://localhost:8080*** :

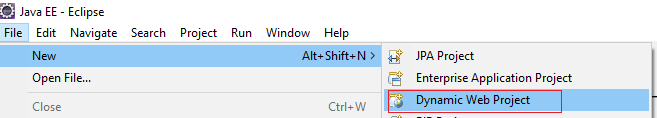


Saisir votre compte administrateur et cliquer sur le bouton **Log In :**

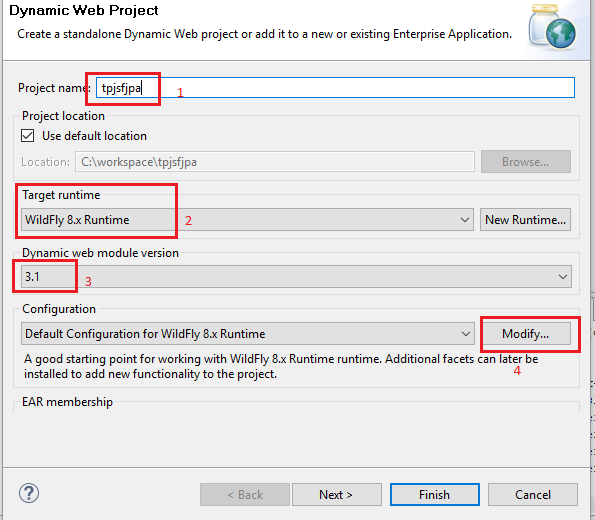


# 4. Partie 2 : Développement de l'application web

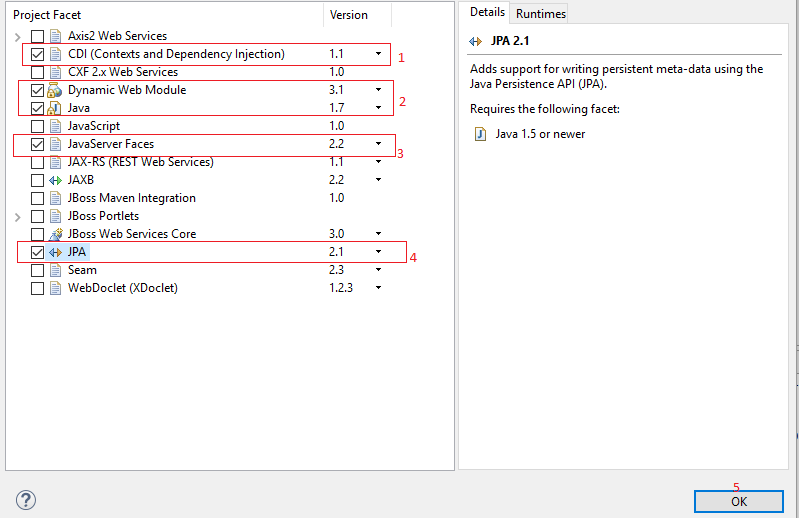
Au niveau d'Eclipse, créer un projet web comme illustre la fenêtre suivante :



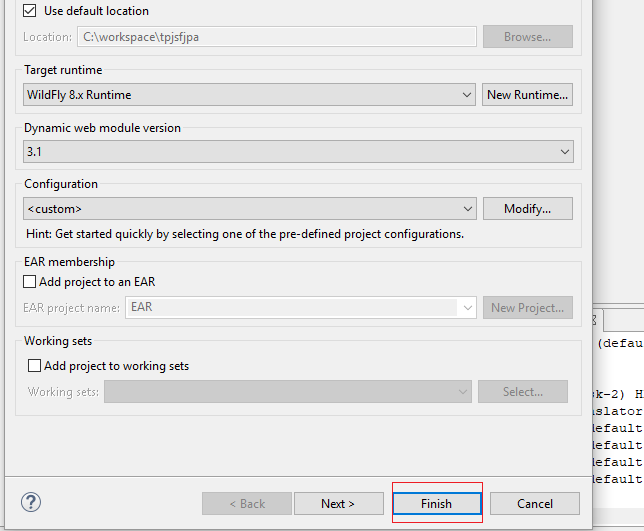
Cliquer sur "Dynamic Web Project" :



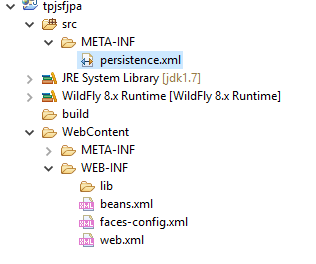
1. Saisir le nom de votre projet (ici : tpjsfjpa);
2. Préciser la "Target runtime" Wildfly 8.x Runtime;
3. Préciser la version de la Servlet (ici : 3.1) ;
4. Cliquer ensuite sur Modifiy.... L'écran suivant sera affiché :



1. Cocher CDI (Context and Dependency Injection). L'api CDI implémente le Design Pattern IOC, nous en parlerons par la suite.
2. Vérifier que la version de "Dynamic Web Module" est bien 3.1 et la version de java est bien 1.7 ;
3. Cocher JavaServerFaces et préciser la version 2.2 ;
4. Cocher JPA et préciser la version 2.1 ;
5. Enfin, cliquer sur le bouton OK.



- Cliquer sur Finish. L'arborescence du projet ci-dessous est créée :



1. **Le modèle :**

Créer la classe Article dans le package *ma.gov.formation.cigma.service.* Le code source de la classe Article est le suivant :

|  |
| --- |
| **package** ma.gov.formation.cigma.service;  **import** java.io.Serializable;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  @Entity  @Table(name="Article")  **public** **class** Article **implements** Serializable {  **private** **static** **final** **long** ***serialVersionUID*** = 6161046340508532663L;  @Id  @GeneratedValue  **private** Integer id;  @Override  **public** String toString() {  **return** "Article [id=" + id + ", designation=" + designation + ", prix=" + prix + ", quantiteEnStock=" + quantiteEnStock + "]";  }    @Column(name="libelle")  **private** String designation;  **private** Double prix;  **private** Integer quantiteEnStock;  **public** Article() {  }  **public** Article(String designation, Double prix, Integer quantiteEnStock) {  **super**();  **this**.designation = designation;  **this**.prix = prix;  **this**.quantiteEnStock = quantiteEnStock;  }    **public** Article(Integer id,String designation, Double prix, Integer quantiteEnStock) {  **this**(designation, prix, quantiteEnStock);  **this**.id=id;  }  **public** Integer getId() {  **return** id;  }  **public** **void** setId(Integer id) {  **this**.id = id;  }  **public** String getDesignation() {  **return** designation;  }  **public** **void** setDesignation(String designation) {  **this**.designation = designation;  }  **public** Double getPrix() {  **return** prix;  }  **public** **void** setPrix(Double prix) {  **this**.prix = prix;  }  **public** Integer getQuantiteEnStock() {  **return** quantiteEnStock;  }  **public** **void** setQuantiteEnStock(Integer quantiteEnStock) {  **this**.quantiteEnStock = quantiteEnStock;  }  } |

1. **Couche DAO :**

Pour la couche DAO, nous allons créer l'interface **IDao** et deux implémentations :

* ***DaoImplList*** : Permet de stocker les articles dans une liste statique.
* ***DaoImplJPA*** : Permet de stocker les articles dans une table au niveau de la base de données Mysql.

Dans un premier temps, nous allons commencer par l'implémentation **DaoImplList**.

**L'interface IDao:**

Créer l'interface IDao dans le package ma.gov.formation.cigma.dao. Le code source de cette interface est :

|  |
| --- |
| **package** ma.gov.formation.cigma.dao;  **import** java.util.List;  **import** ma.gov.formation.cigma.service.Article;  **public** **interface** IDao {  **void** save(Article e);  **void** remove(Integer id);  List<Article> getAll();  **public** Article getById(Integer id);  } |

**La classe DaoImplList :**

Créer la classe DaoImplList dans le package ma.gov.formation.cigma.dao. Le code source de cette classe est :

|  |
| --- |
| **package** ma.gov.formation.cigma.dao;  **import** java.util.ArrayList;  **import** java.util.List;  **import** ma.gov.formation.cigma.service.Article;  **public** **class** DaoImpList **implements** IDao {  **private** **static** List<Article> *modeles* = **new** ArrayList<Article>();  **private** **static** **int** *count*=0;  **static** {  *modeles*.add(**new** Article(1, "PC SONY", 15000.0, 20));  *modeles*.add(**new** Article(2, "TV LG 32 p", 4000.0, 20));  *modeles*.add(**new** Article(3, "TV Samsung 43 p", 65000.0, 35));  *modeles*.add(**new** Article(4, "Camera Sony", 5000.0, 18));  *count*=4;  }  @Override  **public** **void** save(Article article) {  **if** (exists(article.getId())) {  Article a = getById(article.getId());  *modeles*.remove(a);  *modeles*.add(article);  **return**;  }  *count*++;  article.setId(*count*);  *modeles*.add(article);  }  @Override  **public** **void** remove(Integer id) {  **if** (exists(id)) {  Article a = getById(id);  *modeles*.remove(a);  *count*--;  }  }  @Override  **public** List<Article> getAll() {  **return** *modeles*;  }  @Override  **public** Article getById(Integer id) {  **if** (id == **null**)  **return** **null**;  **for** (Article modele : *modeles*) {  **if** (id.equals(modele.getId())) {  **return** modele;  }  }  **return** **null**;  }  **private** **boolean** exists(Integer id) {  **if** (id == **null**)  **return** **false**;  **for** (Article modele : *modeles*)  **if** (id.equals(modele.getId())) {  **return** **true**;  }  **return** **false**;  }  } |

1. **Couche Métier :**

Pour la couche métier, nous allons créer l'interface IService et l'implémentation ServiceImpl.

**L'interface IService :**

Créer l'interface IService dans le package ma.gov.formation.cigma.service :

|  |
| --- |
| **package** ma.gov.formation.cigma.service;  **import** java.util.List;  **public** **interface** IService {  **public** **void** save(Article a);  **public** Article getById(Integer id);  **public** List<Article> getAll();  **public** **void** remove(Integer id);  } |

**La classe ServiceImpl :**

Créer la classe ServiceImpl dans le package ma.gov.formation.cigma.service :

|  |
| --- |
| **package** ma.gov.formation.cigma.service;  **import** java.util.List;  **import** ma.gov.formation.cigma.dao.DaoImpList;  **import** ma.gov.formation.cigma.dao.IDao;  **public** **class** ServiceImpl **implements** IService {  **private** IDao dao = **new** DaoImpList();  @Override  **public** **void** save(Article a) {  dao.save(a);  }  @Override  **public** Article getById(Integer id) {  **return** dao.getById(id);  }  @Override  **public** List<Article> getAll() {  **return** dao.getAll();  }  @Override  **public** **void** remove(Integer id) {  dao.remove(id);  }  } |

1. **Le Value Object**

Dans cette application, nous allons implémenter une bonne pratique qui est le Value Object (VO). Le VO permet de ne pas utiliser les objets externes dans la couche présentation. Donc, si on modifie le modèle (et donc la table), la couche présentation ne sera pas impactée.

**La classe ArticleVo :**

Créer la classe ArticleVo dans le package ma.gov.formation.cigma.domaine :

|  |
| --- |
| **package** ma.gov.formation.cigma.domaine;  **import** java.io.Serializable;  **public** **class** ArticleVo **implements** Serializable {  **private** **static** **final** **long** ***serialVersionUID*** = 6161046340508532663L;  **private** Integer id;  **private** String designation;  **private** Double prix;  **private** Integer quantiteEnStock;  **public** ArticleVo() {  }  **public** ArticleVo(Integer id, String designation, Double prix,  Integer quantiteEnStock) {  **this**.id = id;  **this**.designation = designation;  **this**.prix = prix;  **this**.quantiteEnStock = quantiteEnStock;  }  **public** Integer getId() {  **return** id;  }  **public** **void** setId(Integer id) {  **this**.id = id;  }  **public** String getDesignation() {  **return** designation;  }  **public** **void** setDesignation(String designation) {  **this**.designation = designation;  }  **public** Double getPrix() {  **return** prix;  }  **public** **void** setPrix(Double prix) {  **this**.prix = prix;  }  **public** Integer getQuantiteEnStock() {  **return** quantiteEnStock;  }  **public** **void** setQuantiteEnStock(Integer quantiteEnStock) {  **this**.quantiteEnStock = quantiteEnStock;  }  } |

**La classe ArticleConverter :**

Créer la classe ArticleConverter dans le package ma.gov.formation.cigma.domaine :

|  |
| --- |
| **package** ma.gov.formation.cigma.domaine;  **import** java.util.ArrayList;  **import** java.util.List;  **import** ma.gov.formation.cigma.service.Article;  **public** **class** ArticleConverter {  **public** **static** Article createModele(ArticleVo vo) {  Article modele = **new** Article();  modele.setId(vo.getId());  modele.setDesignation(vo.getDesignation());  modele.setPrix(vo.getPrix());  modele.setQuantiteEnStock(vo.getQuantiteEnStock());  **return** modele;  }  **public** **static** ArticleVo createArticleVo(Article modele) {  ArticleVo vo = **new** ArticleVo();  vo.setId(modele.getId());  vo.setDesignation(modele.getDesignation());  vo.setPrix(modele.getPrix());  vo.setQuantiteEnStock(modele.getQuantiteEnStock());  **return** vo;  }    **public** **static** List<ArticleVo> getArticleVoList(List<Article> modeles) {  List<ArticleVo> vos=**new** ArrayList<>();  **for** (Article modele : modeles) {  vos.add(*createArticleVo*(modele));  }  **return** vos;  }  } |

1. **Couche Présentation :**

**La classe ArticleBean :**

La classe ArticleBean est le ***"Managed Bean"*** qui est géré par JSF. Il s'agit du sous contrôleur auquel le contrôleur principal de JSF lui envoi la requête pour traitement. C'est le "Managed Bean" qui prépare le modèle et choisi la vue.

Créer la classe ArticleBean dans le package ma.gov.formation.cigma.presentation :

|  |
| --- |
| **package** ma.gov.formation.cigma.presentation;  **import** java.util.ArrayList;  **import** java.util.List;  **import** javax.faces.bean.ManagedBean;  **import** javax.faces.bean.SessionScoped;  **import** ma.gov.formation.cigma.domaine.ArticleConverter;  **import** ma.gov.formation.cigma.domaine.ArticleVo;  **import** ma.gov.formation.cigma.service.IService;  **import** ma.gov.formation.cigma.service.ServiceImpl;  @ManagedBean(name = "articleBean")  @SessionScoped  **public** **class** ArticleBean {  **private** IService service = **new** ServiceImpl();  **private** List<ArticleVo> articlesVo = **new** ArrayList<>();  **private** ArticleVo articleVo = **new** ArticleVo();  **public** ArticleVo getArticleVo() {  **return** articleVo;  }  **public** **void** setArticleVo(ArticleVo articleVo) {  **this**.articleVo = articleVo;  }  **public** **void** setArticlesVo(List<ArticleVo> articlesVo) {  **this**.articlesVo = articlesVo;  }  **public** IService getService() {  **return** service;  }  **public** **void** setService(IService service) {  **this**.service = service;  }  **public** List<ArticleVo> getArticlesVo() {  articlesVo = ArticleConverter.*getArticleVoList*(service.getAll());  **return** articlesVo;  }  **public** String addArticle() {  service.save(ArticleConverter.*createModele*(articleVo));  articleVo = **new** ArticleVo();  **return** "articles";  }  **public** String editArticle(Integer id) {  articleVo = ArticleConverter.*createArticleVo*(service.getById(id));  **return** "edit";  }  **public** String removeArticle(Integer id) {  service.remove(id);  **return** **null**;  }  } |

**La page articles.xhtml :**

Créer la page articles.xhtml dans WebContent :

|  |
| --- |
| <html lang=*"en"* xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:f=*"http://xmlns.jcp.org/jsf/core"*  xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets"*  xmlns:a=*"http://xmlns.jcp.org/jsf/passthrough"*  xmlns:h=*"http://xmlns.jcp.org/jsf/html"*>  <h:head>  <title>Exemple d'application web avec JSF+JPA+Wildfly</title>  <h:outputStylesheet library=*"css"* name=*"styles.css"* />  </h:head>  <h:body>  <h:form>  <h:panelGrid columns=*"2"* border=*"1"* cellpadding=*"10"* cellspacing=*"1"*>  <f:facet name=*"header"*>  <h:outputText value=*"Ajouter un nouvel article :"* />  </f:facet>  <h:outputLabel value=*"Désignation de l'article"* />  <h:inputText value=*"#{articleBean.articleVo.designation}"*></h:inputText>  <h:outputLabel value=*"Prix"* />  <h:inputText value=*"#{articleBean.articleVo.prix}"*></h:inputText>  <h:outputLabel value=*"Quantité en stock"* />  <h:inputText value=*"#{articleBean.articleVo.quantiteEnStock}"*></h:inputText>  <f:facet name=*"footer"*>  <h:panelGroup style="display:*block*; text-align:*center*">  <h:commandButton value=*"Ajouter"*  action=*"#{articleBean.addArticle}"*></h:commandButton>  </h:panelGroup>  </f:facet>  </h:panelGrid>  <h2>Liste des articles :</h2>  <h:dataTable value=*"#{articleBean.articlesVo}"* var=*"articleVo"*  styleClass=*"articleTable"* headerClass=*"articleTableHeader"*  rowClasses=*"articleTableOddRow,articleTableEvenRow"*>  <h:column>  <f:facet name=*"header"*>Identifiant</f:facet>  #{articleVo.id}  </h:column>  <h:column>  <f:facet name=*"header"*>Désignation</f:facet>  #{articleVo.designation}  </h:column>  <h:column>  <f:facet name=*"header"*>Prix</f:facet>  #{articleVo.prix}  </h:column>  <h:column>  <f:facet name=*"header"*>Quantité en stock</f:facet>  #{articleVo.quantiteEnStock}  </h:column>  <h:column>  <h:commandLink action=*"#{articleBean.removeArticle(articleVo.id)}"* value=*"Supprimer"* />  </h:column>  <h:column>  <h:commandLink action=*"#{articleBean.editArticle(articleVo.id)}"* value=*"Modifier"* />  </h:column>  </h:dataTable>  </h:form>  </h:body>  </html> |

**La page edit.xhtml :**

Créer la page edit.xhtml dans WebContent :

|  |
| --- |
| <html lang=*"en"* xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:f=*"http://xmlns.jcp.org/jsf/core"*  xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets"*  xmlns:a=*"http://xmlns.jcp.org/jsf/passthrough"*  xmlns:h=*"http://xmlns.jcp.org/jsf/html"*>  <h:head>  <title>Exemple d'application web avec JSF+JPA+Wildfly</title>  <h:outputStylesheet library=*"css"* name=*"styles.css"* />  </h:head>  <h:body>  <h:form>  <h:panelGrid columns=*"2"* border=*"1"* cellpadding=*"10"* cellspacing=*"1"*>  <f:facet name=*"header"*>  <h:outputText value=*"Modifier un article :"* />  </f:facet>  <h:outputLabel value=*"Désignation de l'article"* />  <h:inputText value=*"#{articleBean.articleVo.designation}"*></h:inputText>  <h:outputLabel value=*"Prix"* />  <h:inputText value=*"#{articleBean.articleVo.prix}"*></h:inputText>  <h:outputLabel value=*"Quantité en stock"* />  <h:inputText value=*"#{articleBean.articleVo.quantiteEnStock}"*></h:inputText>  <f:facet name=*"footer"*>  <h:panelGroup style="display:*block*; text-align:*center*">  <h:commandButton value=*"Modifier"*  action=*"#{articleBean.addArticle}"*></h:commandButton>  </h:panelGroup>  </f:facet>  </h:panelGrid>  </h:form>  </h:body>  </html> |

**La feuille de style style.css :**

Dans WebContent, créer le dossier resources. Puis dans resources, créer le dossier css et en fin dans css, créer le fichier style.css suivant :

|  |
| --- |
| *.articleTable* {  border-collapse:*collapse*;  border:*1px* *solid* *#000000*;  }  *.articleTableHeader* {  text-align:*center*;  background:*none* *repeat* *scroll* *00#B5B5B5*;  border-bottom:*1px* *solid* *#000000*;  padding:*2px*;  }  *.articleTableOddRow* {  text-align:*center*;  background:*none* *repeat* *scroll* *00#FFFFFFF*;  }  *.articleTableEvenRow* {  text-align:*center*;  background:*none* *repeat* *scroll* *00#D3D3D3*;  } |

1. **web.xml :**

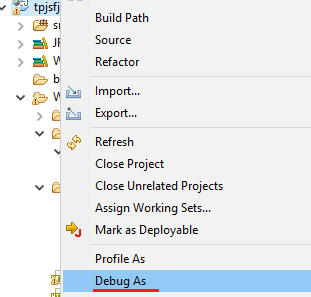
Le contenu du fichier web.xml est le suivant :

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* version=*"3.1"*>  <display-name>tpjsfjpa</display-name>  <welcome-file-list>  <welcome-file>articles.xhtml</welcome-file>  </welcome-file-list>  <servlet>  <servlet-name>Faces Servlet</servlet-name>  <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>Faces Servlet</servlet-name>  <url-pattern>\*.xhtml</url-pattern>  </servlet-mapping>  </web-app> |

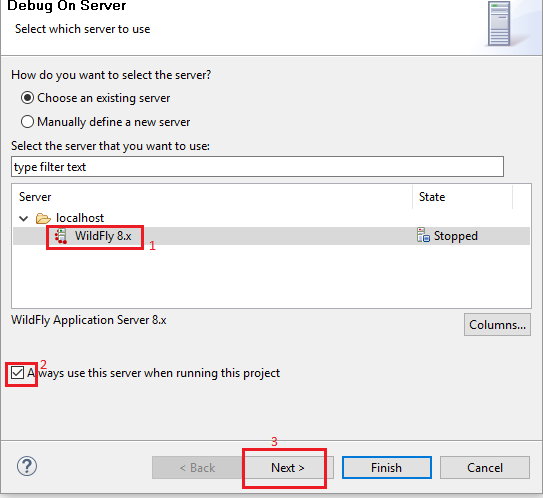
La classe ***javax.faces.webapp.FacesServlet*** est le contrôleur principal (c'est le C dans le MVC).

1. **Test**

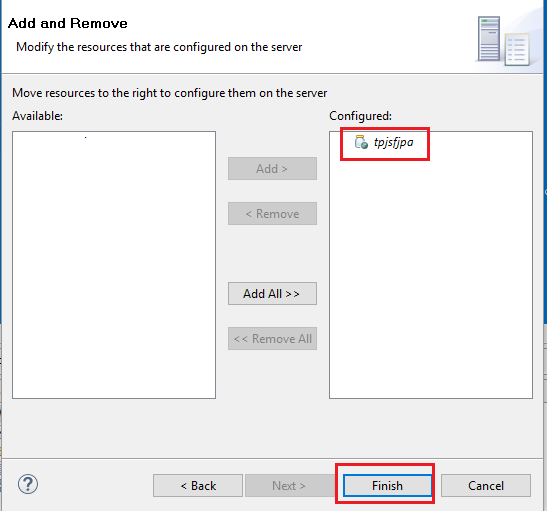
Pour tester l'application, cliquer à droite sur le projet :

****

- Cliquer sir Debug As :



1. Sélectionner WildFly 8.x ;
2. Cocher "Always use this server when running this project" ;
3. Cliquer sur Next>. L'écran suivant sera affiché :

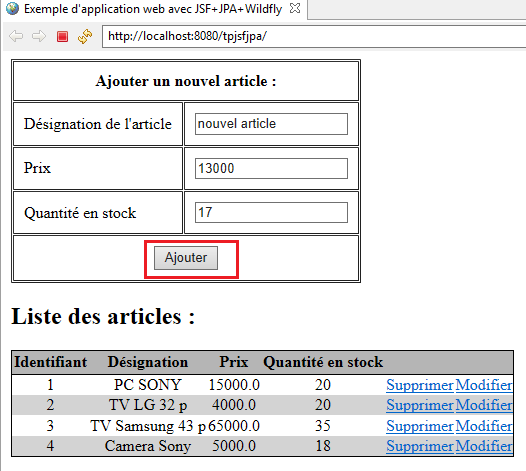


- Cliquer sur Finish. La page articles.xhtml sera exécutée comme illustre la fenêtre suivante :

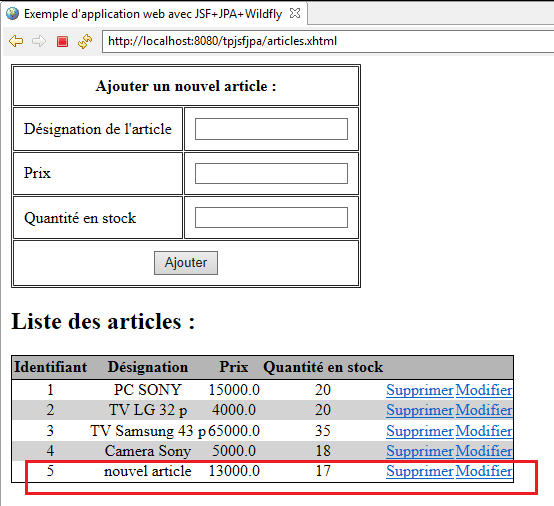


* **Cas de l'ajout :**

Pour ajouter un nouvel article, entrer la désignation, le prix et la quantité en stock comme illustre l'écran suivant :



* Cliquer sur Ajouter. Le nouvel article sera ajouté comme illustré ci-dessous :

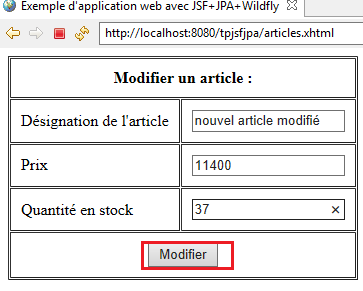


* **Cas de la modification :**

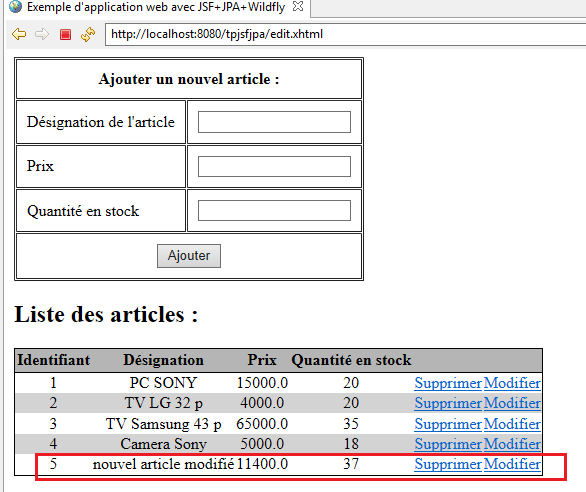
Pour modifier un article, cliquer sur le lien "Modifier" qui corresponds à l'article que vous voulez modifier :



Modifier les données que vous voulez modifier et cliquer sur le bouton Modifier :

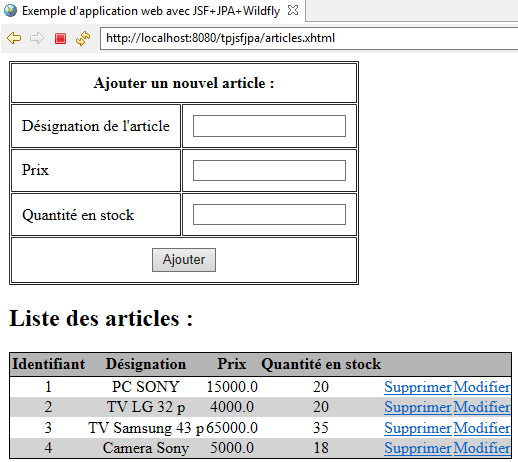


- Cliquer sur le bouton Modifier. l'écran suivant sera affiché. vérifier que l'article a été bien modifié :



* **Cas de la suppression :**

Pour supprimer un article, cliquer sur le lien Supprimer qui corresponds à l'article que vous voulez supprimer (ici par exemple l'article dont l'id est 5) :



Vérifier que l'article 5 a été bien supprimé.

# 5. Partie 3 : Intégration JSF avec JPA

Dans ce qui suit, nous allons développer une deuxième implémentation de la couche DAO en utilisant JPA.

Nous allons suivre les étapes suivantes :

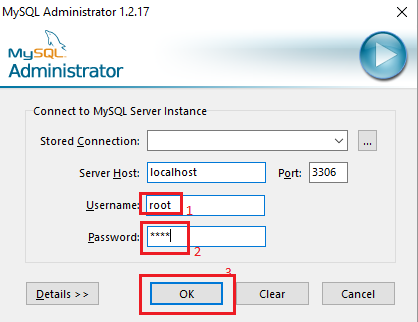
* Ajouter le driver de la base de données Mysql dans WebContent/Web-INF/lib ;
* Créer le schéma ;
* Compléter le fichier persistence.xml ;
* Créer la classe SessionBuilder ;
* et enfin créer la classe DaoImplJPA.

**Ajouter le Driver :**

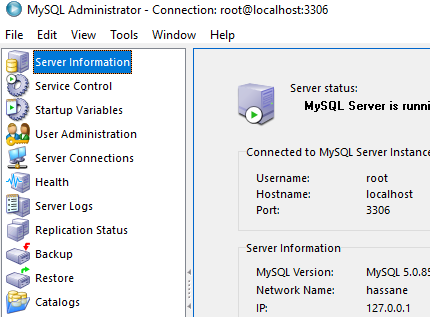
Copier le fichier mysql-connector-java-5.1.17-bin.jar dans le dossier WebContent/WEB-INF/lib de votre projet.

**Création du schéma :**

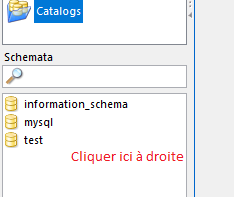
Lancer Mysql Administrator :



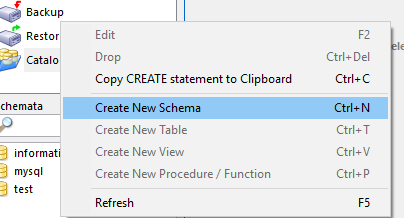
1. Entrer Username (ici c'est root) ;
2. Entrer le mot de passe ;
3. Cliquer ensuite sur OK. La fenêtre suivante sera créée :



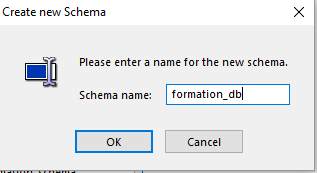
- Cliquer sur Catalogs :



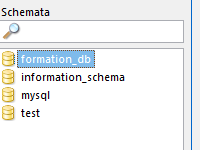
- Cliquer à droite de la souris comme le montre l'écran ci-dessus :



- Cliquer ensuite sur "Create New Schema" :



- Entrer le nom de votre schéma et cliquer sur OK. Le schéma sera crée comme le montre la fenêtre ci-dessous :



* **Le fichier persistence.xml :**

Compléter le fichier persistence.xml comme suit :

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <persistence version=*"2.1"* xmlns=*"http://xmlns.jcp.org/xml/ns/persistence"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd"*>  <persistence-unit name=*"unite1"*>  <provider>org.hibernate.ejb.HibernatePersistence</provider>  <class>ma.gov.formation.cigma.service.Article</class>  <properties>  <!-- Classes persistantes -->  <property name=*"hibernate.show\_sql"* value=*"true"* />  <property name=*"hibernate.format\_sql"* value=*"true"* />  <property name=*"use\_sql\_comments"* value=*"true"* />  <!-- connexion JDBC -->  <property name=*"hibernate.connection.driver\_class"* value=*"com.mysql.jdbc.Driver"* />  <property name=*"hibernate.connection.url"* value=*"jdbc:mysql://localhost:3306/formation\_db"* />  <property name=*"hibernate.connection.username"* value=*"root"* />  <property name=*"hibernate.connection.password"* value=*"root"* />  <!-- création automatique du schéma -->  <property name=*"hibernate.hbm2ddl.auto"* value=*"update"* />  <!-- Dialecte -->  <property name=*"hibernate.dialect"* value=*"org.hibernate.dialect.MySQL5InnoDBDialect"* />  </properties>  </persistence-unit>  </persistence> |

Vérifier votre URL, le nom utilisateur (username) et le mot de passe (password).

**NB :** La balise **<property name=*"hibernate.hbm2ddl.auto"* value=*"update"* />** permet à JPA de créer automatiquement les tables. Dans cet exemple, la table ARTICLE sera créée automatiquement.

**- La classe SessionBuilder :**

Créer la classe SessionBuilder dans le package ma.gov.formation.cigma.dao :

|  |
| --- |
| **package** ma.gov.formation.cigma.dao;  **import** javax.persistence.EntityManager;  **import** javax.persistence.EntityManagerFactory;  **import** javax.persistence.Persistence;  **public** **class** SessionBuilder {  **private** **static** EntityManager *session* = **null**;  **private** **static** SessionBuilder *instance* = **null**;  **private** SessionBuilder() {  }  **public** **static** SessionBuilder getInstance() {  **if** (*instance* == **null**) {  **try** {  *instance* = **new** SessionBuilder();  *openSession*();  } **catch** (Exception e) {  System.***err***.println("session error " + e);  e.printStackTrace();  }  }  **return** *instance*;  }  **private** **static** **void** openSession() **throws** Exception {  EntityManagerFactory sessionFactory = Persistence.*createEntityManagerFactory*("unite1");  *session* = sessionFactory.createEntityManager();  }  **public** EntityManager getSession() {  **return** *session*;  }  } |

**NB :** Remarquer que la méthode createEntityManagerFactory de la classe Persistence prends en paramètre le nom de l'unité de persistance que vous avez précisé dans le fichier persistence.xml.

* **La classe DaoImplJPA :**

Créer la classe DaoImplJPA dans le package ma.gov.formation.cigma.dao :

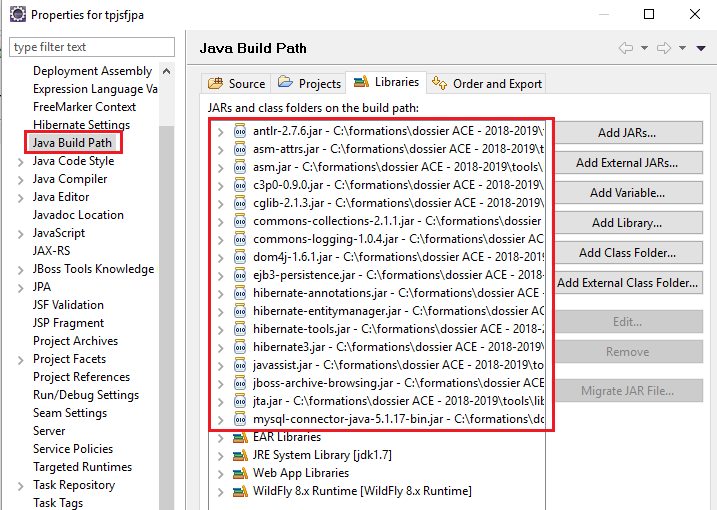
|  |
| --- |
| **package** ma.gov.formation.cigma.dao;  **import** java.util.List;  **import** javax.persistence.EntityManager;  **import** javax.persistence.EntityTransaction;  **import** ma.gov.formation.cigma.service.Article;  **public** **class** DaoImplJPA **implements** IDao {  **private** **static** EntityManager *session* = SessionBuilder.*getInstance*().getSession();  **public** **void** save(Article article) {  **try** {  EntityTransaction tx = *session*.getTransaction();  tx.begin();  *session*.merge(article);  tx.commit();  } **catch** (Exception e) {  e.printStackTrace();  }  }  @SuppressWarnings("unchecked")  **public** List<Article> getAll() {  List<Article> liste = **null**;  **try** {  liste = (List<Article>) *session*.createQuery("from Article")  .getResultList();  **return** liste;  } **catch** (Exception e) {  e.printStackTrace();  **return** **null**;  }  }  @Override  **public** **void** remove(Integer id) {  **try** {  EntityTransaction tx = *session*.getTransaction();  tx.begin();  Article a = *session*.find(Article.**class**, id);  **if** (a != **null**)  *session*.remove(a);  tx.commit();  } **catch** (Exception e) {  e.printStackTrace();  }  }  @Override  **public** Article getById(Integer id) {  Article a = **null**;  **try** {  EntityTransaction tx = *session*.getTransaction();  tx.begin();  a = *session*.find(Article.**class**, id);  tx.commit();  **return** a;  } **catch** (Exception e) {  e.printStackTrace();  **return** **null**;  }  }  } |

* **Test :**

Pour tester la classe DaoImplJPA, créer la classe Test suivant :

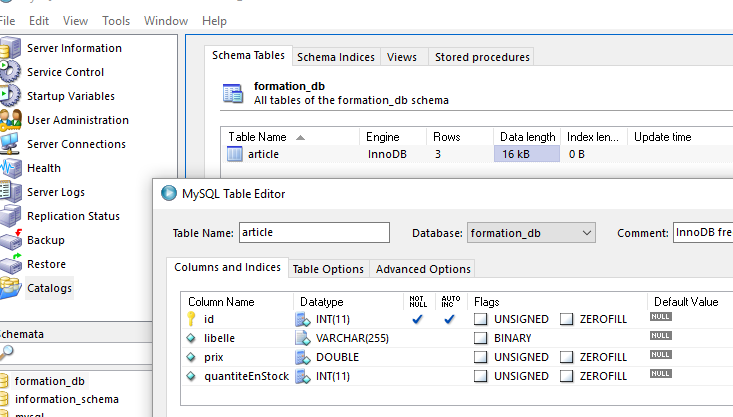
|  |
| --- |
| **package** test;  **import** ma.gov.formation.cigma.dao.DaoImplJPA;  **import** ma.gov.formation.cigma.dao.IDao;  **import** ma.gov.formation.cigma.service.Article;  **public** **class** Test {  **private** **static** IDao *dao* = **new** DaoImplJPA();  **private** **static** **void** addArticle(String designation, **double** prix, Integer quantite) {  *dao*.save(**new** Article(designation,prix,quantite));  }    **public** **static** **void** main(String[] args) {  *addArticle*("ddd",12.0,12);  *addArticle*("ddzzzz",18.0,136);  *addArticle*("vvvvv",19.0,120);  }  } |

- Ajouter dans le Build Path, les librairies d'Hibernante ainsi que le driver de Mysql :



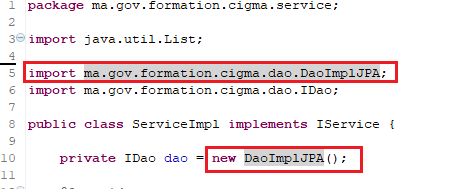
et exécuter ensuite la méthode main.

- Vérifier que la table Article a été bien créée et que les articles ont été bien insérés.



**Basculer de DaoImplList vers DaoImplJPA :**

Pour utiliser la deuxième implémentation avec JPA, opérer la seule modification suivante au niveau de la classe ServiceImpl :



Nous avons fini. refaire les mêmes tests des CRUD pour vérifier que la communication avec la base de données fonctionne sans problème.

# 6. Partie 4 : Enrichissement de la couche présentation

## 6.1. L'internationalisation

Maintenant que l'application web avec JSF et JPA a été bien développée, nous allons enrichir la couche présentation avec l'implémentation du principe de l'internationalisation.

En effet, nous allons donner la main à l'utilisateur de choisir la langue qui sera utilisée par l'application pour afficher tous les messages: les noms des boutons, les liens hypertext, les zones de text, etc. Dans cet exemple, 03 langues seront gérées : l'arable, le français et l'anglais. Bien sûr, vous pouvez par la suite ajouter une autre langue.

**Etape 1 :**

Modifier le fichier faces-config.xml (/WebContent/WEB-INF/faces-config.xml) comme suit :

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <faces-config xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig\_2\_2.xsd"*  version=*"2.2"*>  <application>  <locale-config>  <default-locale>en</default-locale>  <supported-locale>fr</supported-locale>  <supported-locale>ar</supported-locale>  </locale-config>  <resource-bundle>  <base-name>messages</base-name>  <var>msg</var>  </resource-bundle>  </application>  </faces-config> |

- ici, la langue par défaut de notre application est l'anglais :

==> <default-locale>en</default-locale>

- L'application supporte également le français et l'arabe :

==> <supported-locale>fr</supported-locale>

==> <supported-locale>ar</supported-locale>

- La balise <base-name> permet de configurer le nom du fichier à travers lequel l'application épuisera tous les messages à afficher. Ici, le fichier est **messages\_\*.properties.** Ce fichier doit être mis dans le package que vous avez précisé dans la balise<base-name>. Ici, nous avons utilisé le package par défaut, donc c'est **src**. Sinon, nous pouvons mettre à titre d'exemple ma.cigma.resource.messages. Dans ce cas, le fichier messages\_\*.properties doit être mis dans le package ma.cigma.resource.

- La balise <var> permet à JSF de préciser le nom du bean à travers lequel JSF accédera au contenu des fichiers \*.properties. Ici, le nom de ce bean est **msg**.

Le tableau suivant montre le fichier que JSF utilise pour chaque langue choisie :

|  |  |
| --- | --- |
| **La langue** | **Le fichier utilisé par JSF** |
| Anglais | **messages.properties** ou bien **messages\_en.properties**.  puisque nous avons précisé au niveau du fichier faces-config.xml que l'anglais étant la langue par défaut. |
| Français | **messages\_fr.properties** |
| Arable | **messages\_ar.properties** |

Vous remarquez le nom du fichier : **messages\_X.properties** avec X est le code de la locale. En effet, la norme ISO 639 décrit le code standard de chaque langue (pour plus de détails, vous pouvez se référer au lien suivant :

https://www.loc.gov/standards/iso639-2/php/code\_list.php).

**Etape 2 :**

Dans src, créer les fichiers messages\_en.properties, messages\_fr.properties et messages\_ar.properties dont les contenus respectifs sont comme suit :

**messages\_en.properties :**

|  |
| --- |
| #page articles.xhtml et edit.xhtml  titre=Web Application with : JSF, JPA and Wildfly  formulaire.article.designation=Article Designation  formulaire.article.prix=Price  formulaire.article.quantiteEnStock=Quantity in stock  #page articles.xhtml  ajouter.nouvel.article=Add a new article :  formulaire.bouton.ajouter=Add  message.listeDesArticles=Articles list :  intitule.colonne.identifiant.listeDesArticles=Identifier  intitule.colonne.designation.listeDesArticles=Designation  intitule.colonne.prix.listeDesArticles=Price  intitule.colonne.quantiteEnStock.listeDesArticles=Quantity in stock  intitule.lien.supprimer=Delete  intitule.lien.modifier=Update  #page edit.xhtml  modifier.article=Update an article :  formulaire.bouton.modifier=Update |

**messages\_fr.properties :**

|  |
| --- |
| #page articles.xhtml et edit.xhtml  titre=Exemple d'application web avec JSF+JPA+Wildfly  formulaire.article.designation=Désignation de l'article  formulaire.article.prix=Prix  formulaire.article.quantiteEnStock=Quantité en stock  #page articles.xhtml  ajouter.nouvel.article=Ajouter un nouvel article :  formulaire.bouton.ajouter=Ajouter  message.listeDesArticles=Liste des articles :  intitule.colonne.identifiant.listeDesArticles=Identifiant  intitule.colonne.designation.listeDesArticles=Désignation  intitule.colonne.prix.listeDesArticles=Prix  intitule.colonne.quantiteEnStock.listeDesArticles=Quantité en stock  intitule.lien.supprimer=Supprimer  intitule.lien.modifier=Modifier  #page edit.xhtml  modifier.article=Modifier un article :  formulaire.bouton.modifier=Modifier |

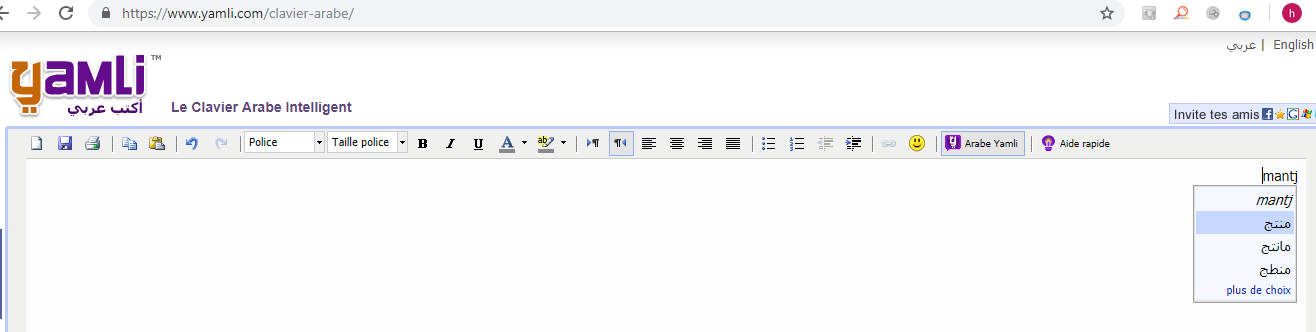
**messages\_ar.properties :**

|  |
| --- |
| #page articles.xhtml et edit.xhtml  titre=\u0628\u0631\u0646\u0627\u0645\u062C \u0645\u0639\u0644\u0648\u0645\u0627\u062A\u064A \u0645\u0646\u062C\u0632 \u0628\u061B JSF\u060CJPA \u0648 WILDFLY  formulaire.article.designation=\u0625\u0633\u0645 \u0627\u0644\u0645\u0646\u062A\u062C  formulaire.article.prix=\u0627\u0644\u0633\u0639\u0631  formulaire.article.quantiteEnStock=\u0643\u0645\u064A\u0629 \u0627\u0644\u0645\u0633\u062A\u0648\u062F\u0639  #page articles.xhtml  ajouter.nouvel.article=: \u0625\u0636\u0627\u0641\u0629 \u0645\u0646\u062A\u062C \u062C\u062F\u064A\u062F  formulaire.bouton.ajouter=\u0625\u0636\u0627\u0641\u0629  message.listeDesArticles=: \u0644\u0627\u0626\u062D\u0629 \u0627\u0644\u0645\u0646\u062A\u062C\u0627\u062A  intitule.colonne.identifiant.listeDesArticles=\u0645\u0631\u062C\u0639  intitule.colonne.designation.listeDesArticles=\u0625\u0633\u0645 \u0627\u0644\u0645\u0646\u062A\u062C  intitule.colonne.prix.listeDesArticles=\u0627\u0644\u0633\u0639\u0631  intitule.colonne.quantiteEnStock.listeDesArticles=\u0643\u0645\u064A\u0629 \u0627\u0644\u0645\u0633\u062A\u0648\u062F\u0639  intitule.lien.supprimer=\u0645\u0633\u062D  intitule.lien.modifier=\u062A\u063A\u064A\u064A\u0631  #page edit.xhtml  modifier.article=: \u062A\u063A\u064A\u064A\u0631 \u0645\u0646\u062A\u062C  formulaire.bouton.modifier=\u062A\u063A\u064A\u064A\u0631 |

Pour la traduction du texte en arabe, vous pouvez utiliser le site

https://www.yamli.com/clavier-arabe

Vous saisissez le texte avec votre clavier en français et le site affichera les propositions des mots en arabe. Voir l'écran ci-après :



Ensuite, il suffit de copier le texte en arabe et le coller dans votre fichier messages\_ar.properties au niveau d'eclipse. Ce dernier, permettra d'afficher le texte en code ASCII.

**Etape 3 :**

Il reste à remplacer dans vos pages \*.xhtml chaque valeur avec sa clé. Exemple, pour permettre à JSF d'afficher le prix au niveau du formulaire, utiliser la formule suivante :

**#{msg['formulaire.article.prix']}**

**Règles à observer :**

- Si le nom de la clé ne contient pas ".", vous pouvez utiliser la formule : **#{msg.votre\_cle\_sans\_le\_point}**

- Si le nom de la clé contient le ".", vous devez utiliser la formule suivante :

**#{msg['votre\_cle\_avec\_le\_ou\_les\_points']}**

**Etape 4 :**

Pour donner la possibilité à l'utilisateur de choisir la langue, ajouter les lignes suivantes au niveau de vos pages :

|  |
| --- |
| <h3>Language switcher:</h3>  <tr>  <td>  <h:commandLink action=*"#{langue.changeLanguage('ar')}"*>  <h:graphicImage value=*"resources/images/maroc.gif"* />  </h:commandLink>  </td>    <td>  <h:commandLink action=*"#{langue.changeLanguage('fr')}"*>  <h:graphicImage value=*"resources/images/france.gif"* />  </h:commandLink>  </td>    <td>  <h:commandLink action=*"#{langue.changeLanguage('en')}"*>  <h:graphicImage value=*"resources/images/usa.gif"* />  </h:commandLink>  </td>  </tr> |

**Explication :**

Le composant <h:commandLink> permet de créer un lien hypertext et de préciser l'action qui sera exécutée lorsque l'utilisateur cliquera sur le lien.

Vous remarquez que nous avons utilisé le managedBean "**langue**" avec la méthode "**changeLanguage"***.*Ce bean sera crée dans l'étape suivante.

Copier le code suivant de la page articles.xhtml :

|  |
| --- |
| <html lang=*"#{langue.language}"* xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:f=*"http://xmlns.jcp.org/jsf/core"*  xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets"*  xmlns:a=*"http://xmlns.jcp.org/jsf/passthrough"*  xmlns:h=*"http://xmlns.jcp.org/jsf/html"*>  <f:view locale=*"#{langue.locale}"*>  <h:head>  <title>#{msg.titre}</title>  <h:outputStylesheet library=*"css"* name=*"styles.css"* />  </h:head>  <h:body>  <h:form>  <h3>Language switcher:</h3>  <tr>  <td>  <h:commandLink action=*"#{langue.changeLanguage('ar')}"*>  <h:graphicImage value=*"resources/images/maroc.gif"* />  </h:commandLink></td>  <td>  <h:commandLink action=*"#{langue.changeLanguage('fr')}"*>  <h:graphicImage value=*"resources/images/france.gif"* />  </h:commandLink>  </td>  <td>  <h:commandLink action=*"#{langue.changeLanguage('en')}"*>  <h:graphicImage value=*"resources/images/usa.gif"* />  </h:commandLink>  </td>  </tr>  <h:panelGrid columns=*"2"* border=*"1"* cellpadding=*"10"* cellspacing=*"1"*>  <f:facet name=*"header"*>  <h:outputText value=*"#{msg['ajouter.nouvel.article']}"* />  </f:facet>  <h:outputLabel value=*"#{msg['formulaire.article.designation']}"* />  <h:inputText value=*"#{articleBean.articleVo.designation}"*></h:inputText>  <h:outputLabel value=*"#{msg['formulaire.article.prix']}"* />  <h:inputText value=*"#{articleBean.articleVo.prix}"*></h:inputText>  <h:outputLabel value=*"#{msg['formulaire.article.quantiteEnStock']}"* />  <h:inputText value=*"#{articleBean.articleVo.quantiteEnStock}"*></h:inputText>  <f:facet name=*"footer"*>  <h:panelGroup style="display:*block*; text-align:*center*">  <h:commandButton value=*"#{msg['formulaire.bouton.ajouter']}"*  action=*"#{articleBean.addArticle}"*></h:commandButton>  </h:panelGroup>  </f:facet>  </h:panelGrid>  <h2>#{msg['message.listeDesArticles']}</h2>  <h:dataTable value=*"#{articleBean.articlesVo}"* var=*"articleVo"*  styleClass=*"articleTable"* headerClass=*"articleTableHeader"*  rowClasses=*"articleTableOddRow,articleTableEvenRow"*>  <h:column>  <f:facet name=*"header"*>#{msg['intitule.colonne.identifiant.listeDesArticles']}</f:facet>  #{articleVo.id}  </h:column>  <h:column>  <f:facet name=*"header"*>#{msg['intitule.colonne.designation.listeDesArticles']}</f:facet>  #{articleVo.designation}  </h:column>  <h:column>  <f:facet name=*"header"*>#{msg['intitule.colonne.prix.listeDesArticles']}</f:facet>  #{articleVo.prix}  </h:column>  <h:column>  <f:facet name=*"header"*>#{msg['intitule.colonne.quantiteEnStock.listeDesArticles']}</f:facet>  #{articleVo.quantiteEnStock}  </h:column>  <h:column>  <h:commandLink action=*"#{articleBean.removeArticle(articleVo.id)}"*  value=*"#{msg['intitule.lien.supprimer']}"* />  </h:column>  <h:column>  <h:commandLink action=*"#{articleBean.editArticle(articleVo.id)}"*  value=*"#{msg['intitule.lien.modifier']}"* />  </h:column>  </h:dataTable>  </h:form>  </h:body>  </f:view>  </html> |

Copier le code suivant de la page edit.xhtml :

|  |
| --- |
| <html lang=*"#{langue.language}"* xmlns=*"http://www.w3.org/1999/xhtml"*  xmlns:f=*"http://xmlns.jcp.org/jsf/core"*  xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets"*  xmlns:a=*"http://xmlns.jcp.org/jsf/passthrough"*  xmlns:h=*"http://xmlns.jcp.org/jsf/html"*>  <f:view locale=*"#{langue.locale}"*>  <h:head>  <title>#{msg.titre}</title>  <h:outputStylesheet library=*"css"* name=*"styles.css"* />  </h:head>  <h:body>  <h:form>  <h3>Language switcher:</h3>  <tr>  <td><h:commandLink action=*"#{langue.changeLanguage('ar')}"*>  <h:graphicImage value=*"resources/images/maroc.gif"* />  </h:commandLink></td>  <td><h:commandLink action=*"#{langue.changeLanguage('fr')}"*>  <h:graphicImage value=*"resources/images/france.gif"* />  </h:commandLink></td>  <td><h:commandLink action=*"#{langue.changeLanguage('en')}"*>  <h:graphicImage value=*"resources/images/usa.gif"* />  </h:commandLink></td>  </tr>  <h:panelGrid columns=*"2"* border=*"1"* cellpadding=*"10"* cellspacing=*"1"*>  <f:facet name=*"header"*>  <h:outputText value=*"#{msg['modifier.article']}"* />  </f:facet>  <h:outputLabel value=*"#{msg['formulaire.article.designation']}"* />  <h:inputText value=*"#{articleBean.articleVo.designation}"*></h:inputText>  <h:outputLabel value=*"#{msg['formulaire.article.prix']}"* />  <h:inputText value=*"#{articleBean.articleVo.prix}"*></h:inputText>  <h:outputLabel value=*"#{msg['formulaire.article.quantiteEnStock']}"* />  <h:inputText value=*"#{articleBean.articleVo.quantiteEnStock}"*></h:inputText>  <f:facet name=*"footer"*>  <h:panelGroup style="display:*block*; text-align:*center*">  <h:commandButton value=*"#{msg['formulaire.bouton.modifier']}"*  action=*"#{articleBean.addArticle}"*></h:commandButton>  </h:panelGroup>  </f:facet>  </h:panelGrid>  </h:form>  </h:body>  </f:view>  </html> |

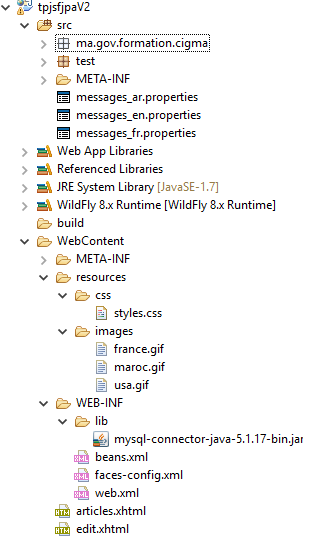
Copier les trois images (france.gif, maroc.gif et usa.gif) se trouvant dans le dossier ***formation\_Framework/images*** dans le dossier : WebContent/resources/images/.

**Etape 5 :**

Créer la classe *ma.gov.formation.cigma.presentation.LangueBean* suivante :

|  |
| --- |
| **package** ma.gov.formation.cigma.presentation;  **import** java.io.Serializable;  **import** java.util.Locale;  **import** javax.faces.bean.ManagedBean;  **import** javax.faces.bean.SessionScoped;  **import** javax.faces.context.FacesContext;  @ManagedBean(name = "langue")  @SessionScoped  **public** **class** LangueBean **implements** Serializable {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **private** Locale locale = **new** Locale("en");  **public** Locale getLocale() {  **return** locale;  }  **public** String getLanguage() {  **return** locale.getLanguage();  }  **public** **void** changeLanguage(String language) {  locale = **new** Locale(language);  FacesContext.*getCurrentInstance*().getViewRoot().setLocale(locale);  }  } |

L'arborescence finale de votre projet est :



Lancer le serveur et exécuter par la suite le lien : http://localhost:8080/tpjsfjpa. La page suivante sera affichée :

****

Pour choisir la langue arabe, cliquer sur le drapeau du Maroc :

****

Pour choisir la langue française, cliquer sur le drapeau de la France :

# 