JAVA JEE

**Table des matières**

[DocPapier](#_topic_DocPapier) 5

[Toutes les installations](#_topic_Touteslesinstallations) 8

[Marques pages utilisés durant la formation](#_topic_Marquespagesutilisesdurantlaform) 12

[NOTIONS GENERALES](#_topic_NOTIONSGENERALES) 15

[A connaitre / Tutoriel](#_topic_AconnaitreTutoriel) 17

[Raccourcis](#_topic_Raccourcis) 20

[Syntaxe dans un java](#_topic_Syntaxedansunjava) 24

[Syntaxe jstl](#_topic_Syntaxejstl) 28

[Paramétrage Eclipse](#_topic_ParametrageEclipse) 32

[Config Markers](#_topic_ConfigMarkers) 34

[Cartographie](#_topic_Cartographie) 38

[Solutions erreurs](#_topic_Solutionserreurs) 41

[MAVEN](#_topic_MAVEN) 44

[Installation](#_topic_Installation) 47

[1) Déclaration des variables d'environnement dans windows](#_topic_1Declarationdesvariablesdenviron) 50

[2) Dans éclipse](#_topic_2Danseclipse) 54

[3) Définition du pom.xml](#_topic_3Definitiondupomxml) 57

[4) Les liens](#_topic_4Lesliens) 60

[Nouveau chapitre](#_topic_Nouveauchapitre) 63

[SPRING](#_topic_SPRING) 66

[Installation](#_topic_Installation1) 68

[Obtenir de l'aide](#_topic_Obtenirdelaide) 70

[Fonctionnement du déploiement](#_topic_Fonctionnementdudeploiement) 72

[@Annotation](#_topic_Annotation) 75

[Projet Cocktail avant BDD](#_topic_ProjetCocktailavantBDD) 78

[pom.xml](#_topic_pomxml) 81

[web.xml](#_topic_webxml) 84

[applicationContext.xml](#_topic_applicationContextxml) 87

[menu.properties](#_topic_menuproperties) 90

[IngredientController.java](#_topic_IngredientControllerjava) 93

[Menu.java](#_topic_Menujava) 96

[IngredientService.java](#_topic_IngredientServicejava) 99

[IngredientDAO.java](#_topic_IngredientDAOjava) 102

[Ingredient.java](#_topic_Ingredientjava) 105

[MainController.java](#_topic_MainControllerjava) 108

[Placer correctement les fichiers](#_topic_Placercorrectementlesfichiers) 111

[ingredient.jsp](#_topic_ingredientjsp) 114

[index.jsp](#_topic_indexjsp) 117

[pom.properties](#_topic_pomproperties) 120

[menu.properties](#_topic_menuproperties1) 122

[Structure](#_topic_Structure) 125

[Projet Cocktail avec BDD](#_topic_ProjetCocktailavecBDD) 129

[orm.xml](#_topic_ormxml) 133

[persistence.xml](#_topic_persistencexml) 136

[pom.xml](#_topic_pomxml1) 139

[applicationContext.xml](#_topic_applicationContextxml1) 143

[IngredientController.java](#_topic_IngredientControllerjava1) 146

[IngredientDAO.java](#_topic_IngredientDAOjava1) 149

[IngredientService.java](#_topic_IngredientServicejava1) 152

[AddIngredient.jsp](#_topic_AddIngredientjsp) 155

[Ingredient.jsp](#_topic_Ingredientjsp1) 158

[Mysql](#_topic_Mysql) 161

[Config de départ](#_topic_Configdedepart) 165

[Config tjs JPA](#_topic_ConfigtjsJPA) 167

[Vue/Controller/](#_topic_VueController) 170

[GIT](#_topic_GIT) 173

[Liste des commandes](#_topic_Listedescommandes) 176

[Création de mon Git pour la doc](#_topic_CreationdemonGitpourladoc) 179

[Projet Cocktail de Jeremy](#_topic_ProjetCocktaildeJeremy) 182

[src/main/java](#_topic_srcmainjava) 185

[CONTROLLER](#_topic_CONTROLLER) 187

[CocktailController](#_topic_CocktailController) 189

[IngredientController](#_topic_IngredientController) 192

[MainController](#_topic_MainController) 195

[DAO](#_topic_DAO) 198

[CocktailDAO](#_topic_CocktailDAO) 200

[IngredientDao](#_topic_IngredientDao) 203

[ENTITY](#_topic_ENTITY) 206

[Cocktail](#_topic_Cocktail) 208

[Ingredient](#_topic_Ingredient) 210

[MODEL](#_topic_MODEL) 212

[Menu](#_topic_Menu) 214

[SERVICE](#_topic_SERVICE) 216

[CocktailService](#_topic_CocktailService) 218

[IngredientService](#_topic_IngredientService) 220

[sr/main/resources](#_topic_srmainresources) 222

[menu.properties](#_topic_menuproperties2) 224

[META-INF](#_topic_METAINF) 226

[orm.xml](#_topic_ormxml1) 228

[persistence.xml](#_topic_persistencexml1) 231

[webapp](#_topic_webapp) 233

[css](#_topic_css) 235

[Nouveau chapitre](#_topic_Nouveauchapitre1) 237

[views](#_topic_views) 239

[WEB-INF](#_topic_WEBINF) 241

[Outils](#_topic_Outils) 243

[STAN](#_topic_STAN) 245

[Organigramme choix Objet Collection](#_topic_OrganigrammechoixObjetCollection) 248

[GlassFish](#_topic_GlassFish) 251

[Install](#_topic_Install) 253

[Lancer la console Glassfish déclaration du path (variable d'environnement JAVA\_HOME)](#_topic_LancerlaconsoleGlassfishdeclarat) 256

[Changement port 8080](#_topic_Changementport8080) 259

[Paramétrage des logs](#_topic_Parametragedeslogs) 262

[Mysql pool](#_topic_Mysqlpool) 265

[Dans eclipse](#_topic_Danseclipse) 269

[Schéma Appli](#_topic_SchemaAppli) 272

[Remote debugging](#_topic_Remotedebugging) 275

[Configuration Glassfish et éclipse](#_topic_ConfigurationGlassfisheteclipse) 277

[Déclaration des Logs en java](#_topic_DeclarationdesLogsenjava) 280

[Classe EntityManager et UserTransaction](#_topic_ClasseEntityManageretUserTransac)  283

[Schéma de vie d'une exception](#_topic_Schemadevieduneexception) 286

[Schéma des ERROR / EXCEPTION](#_topic_SchemadesERROREXCEPTION) 289

[Lancer Glassfish par un Bat](#_topic_LancerGlassfishparunBat) 292

[Liste des tutos donnés par Jérémy](#_topic_ListedestutosdonnesparJeremy) 296

[JFS](#_topic_JFS) 299

[Annotations](#_topic_Annotations) 301

[Annotations BEAN](#_topic_AnnotationsBEAN) 304

[Annotations CONTROLLER](#_topic_AnnotationsCONTROLLER) 307

[Annotations DAO](#_topic_AnnotationsDAO) 310

[Annotations ENTITY](#_topic_AnnotationsENTITY) 313

[@ManageBean](#_topic_ManageBean) 316

[Ajax](#_topic_Ajax) 319

[Qq tags](#_topic_Qqtags) 322

[Web Services](#_topic_WebServices) 325

[Spring security](#_topic_Springsecurity) 328

[Config Java](#_topic_ConfigJava) 331

[Nouveau chapitre](#_topic_Nouveauchapitre2) 334

[Test unitaire](#_topic_Testunitaire) 338

[Service rest](#_topic_Servicerest) 340

[Ecriture d'un stream](#_topic_Ecrituredunstream) 343

[Cycles de développement](#_topic_Cyclesdedeveloppement) 348

[Modèle en V](#_topic_ModeleenV) 351

[JUnit](#_topic_JUnit) 354

[Création d'un projet Mavec via éclipse](#_topic_CreationdunprojetMavecviaeclipse) 357

[Test de Spring MVC](#_topic_TestdeSpringMVC) 360

[Export JAR](#_topic_ExportJAR) 363

[Vérifier la qualité du code](#_topic_Verifierlaqualiteducode) 366

[Résumé de la formation!](#_topic_Resumedelaformation) 369

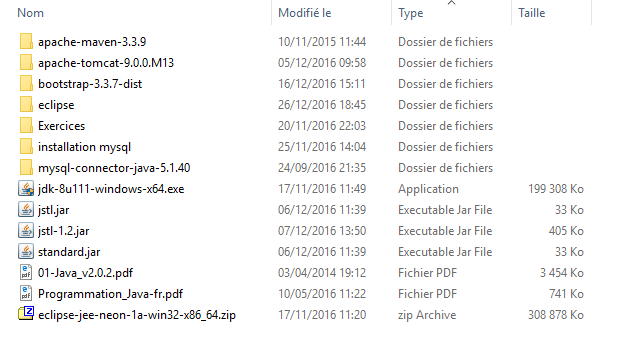
**DocPapier**

documentation word: [C:\Users\hb-asus\Documents\HelpNDoc\Projects\JAVA JEE.docx](file:///C:/Users/hb-asus/Documents/HelpNDoc/Projects/JAVA%20JEE.docx)

documentation pdf: [C:\Users\hb-asus\Documents\HelpNDoc\Projects\JAVA JEE.pdf](file:///C:/Users/hb-asus/Documents/HelpNDoc/Projects/JAVA%20JEE.pdf)

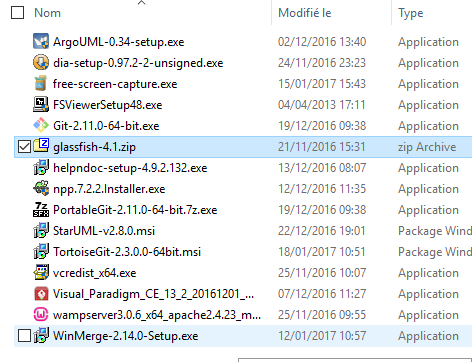
*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides CHM gratuit*](http://www.helpndoc.com/fr)

**Toutes les installations**



Ajouter Glassfish

**Liste des applis installées :**



*Créé avec HelpNDoc Personal Edition:* [*Outil de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**Marques pages utilisés durant la formation**

[C:\Users\hb-asus\Documents\HelpNDoc\Projects\Java JEE\bookmarks.html](file:///C:/Users/hb-asus/Documents/HelpNDoc/Projects/Java%20JEE/bookmarks.html)

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation iPhone gratuit*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**NOTIONS GENERALES**

*Créé avec HelpNDoc Personal Edition:* [*Outil de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**A connaitre / Tutoriel**

Pour debbuger il faut mettre les points d'arret sur \*.java

JSP JavaServer Page..... : <https://www.tutorialspoint.com/jsp/jsp_overview.htm>

Maven repository........... :

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation d'aide HTML gratuit*](http://www.helpndoc.com/fr)

**Raccourcis**

ctrl /T --> pour vérifier les arborescences

alt F5 --> pour mettre à jour les dépendances

ctrl 1 --> pour assigner une variable

ctrl 2 L --> locale

ctrl 2 F --> field

*Créé avec HelpNDoc Personal Edition:* [*Écrire des livres électroniques ePub pour l'iPad*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Syntaxe dans un java**

Ajouter les annotations de spring @ :

Instancier un objet ModelAndView en final

Lui ajouter les objets avec addObject("nomUtiliséDansLaJSP", VariableLocale)

Ajouter le nom de la vue avec setViewName

@Controller

**public** **class** ProduitController {

@RequestMapping("ListeProduit")

**public** ModelAndView getList() {

**final** ModelAndView mav = **new** ModelAndView();

Produit produit1 = **new** Produit("art1", 21, "rouge");

Produit produit2 = **new** Produit("art2", 21, "bleu");

Produit produit3 = **new** Produit("art3", 21, "vert");

Produit produit4 = **new** Produit("art4", 21, "noir");

ArrayList<Produit> tableau = **new** ArrayList<>();

tableau.add(produit1);

tableau.add(produit2);

tableau.add(produit3);

tableau.add(produit4);

mav.addObject("liste", tableau);

mav.setViewName("ListeProduit");

**return** mav;

}

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Syntaxe jstl**

Dans un fichier jsp, il faut ajouter :

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<head><meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*></head>

Utilisation de la librairie core :

<c:forEach items=*""* var=*""*> </c:forEach>

<c:if test=*""* > </c:if>

<c:out value=*""*></c:out>

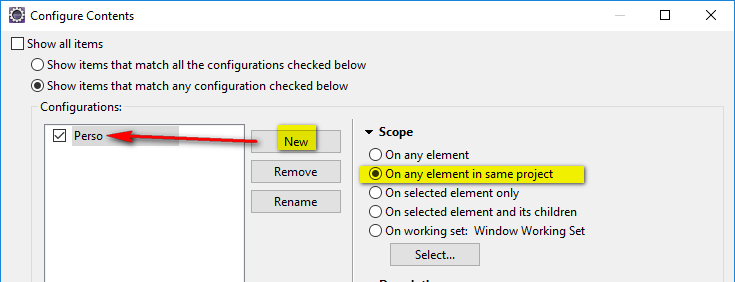
*Créé avec HelpNDoc Personal Edition:* [*Générateur facile de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Paramétrage Eclipse**

*Créé avec HelpNDoc Personal Edition:* [*Générateur facile de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Config Markers**

Pour ne voir dans la console **Markers** que les erreurs du projet en cour : (à partir de la petite flèche)



*Créé avec HelpNDoc Personal Edition:* [*Environnement de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**Cartographie**

|  |  |  |
| --- | --- | --- |
|  |  |  |

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Solutions erreurs**

|  |  |
| --- | --- |
|  |  |
| org.springframework.beans.factory.NoSuchBeanDefinitionException:  **No qualifying bean** of type **'fr.formation.SERVICE.IngredientService'** available:  expected at least 1 bean which qualifies as autowire candidate.  Dependency annotations: {@org.springframework.beans.factory.annotation.Autowired(required=true)} | Le problème était que je n'avais pas déclarer mes packages dao et services dans mon fichier servlet-dispatch.xml>  Du coup j'en ai déduit que chaque package qui contient des classes utilisant des annotations du framework Spring doit être déclaré dans ce fichier XML ! |
|  |  |

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation iPhone gratuit*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**MAVEN**

Quelques commandes utilisées lors du projet :

* mvn package
* mvn dependency:purge-local-repository
* **mvn clean package**

*Créé avec HelpNDoc Personal Edition:* [*Créer facilement des fichiers Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Installation**

MAVEN : Outil de gestion de projet

POM : Project Object Model

*Créé avec HelpNDoc Personal Edition:* [*Créer facilement des fichiers Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**1) Déclaration des variables d'environnement dans windows**

Après avoir téléchargé Maven

Ajouter JAVA\_HOME lien vers le répertoire Java/jdk1.8.0\_11

Ajouter dans Path le répertoire bin de Maven

# Créer l'arborescence des dossier sous le nx projet sous workspace :

Structure du dossier

C:\Users\hb-asus\workspace\CocktailBar

C:.

└───src

├───main

│ ├───java

│ ├───resources

│ └───webapp

│ └───WEB-INF

└───test

├───java

└───resources

# Par la suite avec la BDD

C:.

│ .classpath

│ .project

│ pom.xml

│

├───src

│ ├───main

├───java

│ └───fr

│ └───formation

│ ├───CONTROLLER

│ │ IngredientController.java

│ │ MainController.java

│ ├───DAO

│ │ IngredientDAO.java

│ ├───entity

│ │ Ingredient.java

│ ├───MODEL

│ │ Menu.java

│ └───service

│ IngredientService.java

│

├───resources

│ │ menu.properties

│ └───META-INF

│ orm.xml

│ persistence.xml

│

└───webapp

├───css

│ paper.css

├───inc

│ footer.jsp

│ header.jsp

├───views

│ addIngredient.jsp

│ index.jsp

│ ingredients.jsp

└───WEB-INF

applicationContext.xml

web.xml

liste settings

.jsdtscope

org.eclipse.core.resources.prefs

org.eclipse.jdt.core.prefs

org.eclipse.jpt.core.prefs

org.eclipse.m2e.core.prefs

org.eclipse.wst.common.component

org.eclipse.wst.common.project.facet.core.prefs.xml

org.eclipse.wst.common.project.facet.core.xml

org.eclipse.wst.jsdt.ui.superType.container

org.eclipse.wst.jsdt.ui.superType.name

org.eclipse.wst.validation.prefs

*Créé avec HelpNDoc Personal Edition:* [*Générer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**2) Dans éclipse**

Dans éclipse il faut créer deux fichiers :

**pom.xml et web.xml**

C:.workspace/projet

|

├───.settings

├───src

│ ├───main

│ │ ├───java

│ │ ├───resources

│ │ └───webapp

│ │ └───WEB-INF --------------------------> **web.xml**

│ └───test

│ ├───java

│ └───resources

│--------------------------------------------------------------> **pom.xml**

*Créé avec HelpNDoc Personal Edition:* [*Écrire des livres électronique Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

**3) Définition du pom.xml**

Aller sous mvn repository pour copier les repository

Faire alt+F5 pour mettre à jour les libraries dans le projet

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*

*http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>fr.formation</groupId>

<artifactId>cocktailBar</artifactId>

<version>0.0.1</version>

<packaging>war</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet.jsp.jstl/jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.4.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.4.RELEASE</version>

</dependency>

</dependencies>

<!-- Configuration des plugins -->

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

*Créé avec HelpNDoc Personal Edition:* [*Créer des fichiers d'aide pour la plateforme Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

**4) Les liens**

<https://mvnrepository.com/artifact/org.springframework/spring-webmvc/4.3.4.RELEASE>

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides CHM gratuit*](http://www.helpndoc.com/fr)

**Nouveau chapitre**

Pour utiliser Jquery, ajouter les dépendances suivantes dans pom.xml

JQUERY-UI 1.12.1

JQUERY-UI datatables 1.12.1

JQUERY 3.11.1

JQUERY datatables colreorder 1.2.0

Le code Jquery peut être copié depuis datatables.net puis

-> Examples -->Advanced initialisation

-> DOM/JQuery events

*Créé avec HelpNDoc Personal Edition:* [*Création d'aide CHM, PDF, DOC et HTML d'une même source*](http://www.helpndoc.com/fr/outils-creation-aide)

**SPRING**

*Créé avec HelpNDoc Personal Edition:* [*Avantages d'un outil de création d'aide*](http://www.helpauthoringsoftware.com/fr)

**Installation**

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation complet*](http://www.helpndoc.com/fr)

**Obtenir de l'aide**

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation et EPub facile*](http://www.helpndoc.com/fr)

**Fonctionnement du déploiement**

Désolé cette prise de note est en relation directe avec ma compréhension sur le moment... vous comprendrez en la lisant....

Merci Jéremy d'être aller si vite ;-(

Projet/src/main/WEB-INF/applicationContext.xml

Ajouter un rép source/main/**views**

**1) Fonctionnement du déploiement**

Tomcat/bin > startup.bat

lib

log

firstservlet.war sous webapp

invalidelockHeader -> vider le repository local de maven

tmp wtpwebapps

temp sous tomcat

work fichier temp mais peut être supprimer et se régénère cache de tomcat cote appli

conf

2) Commande mvn :

**mvn clean package**

**3) properies cocktail**

**web project settings ==> nom du projet en minuscule**

**Menu Projet > Clean**

**4) Création file général : Java resources/src/main/resources/menu.properties**

*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation CHM facile*](http://www.helpndoc.com/fr)

**@Annotation**

# Les annotations

@Autowired : permet de voir qu'on surcharge la méthode

final classe et methode empèche la surcharge ou extends

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide PDF facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Projet Cocktail avant BDD**

Cocktail

--> Nom

--> Ingrédients

--> Prix

--> Alcoolisé

Ingrédient

--> nom

--> Quantité

--> Etat

Menu :

Liste des cocktails

Liste des ingrédients

Ajouter un cocktail

Ajouter un ingrédient

Rechercher

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**pom.xml**

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<!-- Aller sous mvn repository pour copier les repository -->

<!-- Faire alt+F5 pour mettre à jour les libraries dans le projet -->

<!-- Posé à la racine du projet -- >

<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0*

*http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>fr.formation</groupId>

<artifactId>cocktailBar</artifactId>

<version>0.0.1</version>

<packaging>war</packaging>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet.jsp.jstl/jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.4.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.4.RELEASE</version>

</dependency>

</dependencies>

<!-- Configuration des plugins -->

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.5.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentations PDF gratuit*](http://www.helpndoc.com/fr)

**web.xml**

<web-app 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-app\_3\_1.xsd"*

version=*"3.1"*>

<display-name>Appli Cocktail</display-name>

<servlet>

<servlet-name>SpringServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/applicationContext.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>SpringServlet</servlet-name>

<url-pattern>\*.html</url-pattern>

</servlet-mapping>

</web-app>

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation et EPub gratuit*](http://www.helpndoc.com/fr)

**applicationContext.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<context:component-scan base-package=*"fr.formation.CONTROLLER,fr.formation.DAO,fr.formation.MODEL,fr.formation.service"*></context:component-scan>

<!-- nom de la propriété -->

<!-- view classe -->

<bean id=*"viewResolver"*

class=*"org.springframework.web.servlet.view.UrlBasedViewResolver"*>

<property name=*"viewClass"*

value=*"org.springframework.web.servlet.view.JstlView"* />

<property name=*"prefix"* value=*"/views/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

<bean id=*"messageSource"*

class=*"org.springframework.context.support.ReloadableResourceBundleMessageSource"*>

<property name=*"basename"* value=*"classpath:/menu"* />

</bean>

</beans>

*Créé avec HelpNDoc Personal Edition:* [*Créer de la documentation iPhone facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**menu.properties**

# Liste des menus de l'application

menu.list=cocktailList, ingredientList, addCocktail, addIngredient, search

menu.cocktailList.title=Liste des cocktails

menu.cocktailList.url=/cocktails

menu.ingredientList.title=Liste des ingredients

menu.ingredientList.url=/ingredients

menu.addCocktail.title=Ajouter un cocktail

menu.addCocktail.url=/cocktail/add

menu.addIngredient.title=Ajouter un ingredient

menu.addIngredient.url=/ingredient/add

menu.search.title=Recherche

menu.search.url=/search

|  |  |  |
| --- | --- | --- |
| persistant | Business | Presentation |
| menu.properties --> | ---> MessageSource  |  List<Menu>  |  ModelAndView --> | -->index.jsp |
|  |  |  |

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**IngredientController.java**

**package** fr.formation.CONTROLLER;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.servlet.ModelAndView;

**import** fr.formation.service.IngredientService;

@Controller

@RequestMapping("/ingredients")

**public** **class** IngredientController {

@Autowired

**private** IngredientService service;

@RequestMapping

**public** ModelAndView list() {

**final** ModelAndView mav = **new** ModelAndView();

mav.setViewName("ingredients");

mav.addObject("ingredients", **this**.service.getAll());

**return** mav;

}

}

*Créé avec HelpNDoc Personal Edition:* [*Générer facilement des livres électroniques Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

**Menu.java**

**package** fr.formation.MODEL;

**public** **class** Menu {

**final** **private** String title;

**final** **private** String url;

/\*\*

\*

\* **@param** title titre du menu

\* **@param** url url de lien d'ouverture du menu

\*/

**public** Menu(String title, String url){

**this**.title = title;

**this**.url = url;

}

**public** String getTitle() {

**return** title;

}

**public** String getUrl() {

**return** url;

}

}

*Créé avec HelpNDoc Personal Edition:* [*Création d'aide CHM, PDF, DOC et HTML d'une même source*](http://www.helpndoc.com/fr/outils-creation-aide)

**IngredientService.java**

package fr.formation.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import fr.formation.DAO.IngredientDAO;

import fr.formation.entity.Ingredient;

@Service

public class IngredientService {

@Autowired

private IngredientDAO dao;

public List<Ingredient> getAll(){

return this.dao.readAll();

}

}

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**IngredientDAO.java**

package fr.formation.DAO;

import java.util.Arrays;

import java.util.List;

import org.springframework.stereotype.Component;

import fr.formation.entity.Ingredient;

@Component

public class IngredientDAO {

public List<Ingredient> readAll(){

return Arrays.asList(new Ingredient(0, "Rhum"),

new Ingredient(0,"Whiskey"),

new Ingredient(0,"Tequila"),

new Ingredient(1, "Ice cubes"),

new Ingredient(1, "Sugar"),

new Ingredient(2, "CO2"));

}

}

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aide complet*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Ingredient.java**

package fr.formation.entity;

import java.io.Serializable;

import org.springframework.web.servlet.mvc.method.annotation.ResponseBodyEmitterReturnValueHandler;

public class Ingredient implements Serializable {

private static final long serialVersionUID = 1L;

private int etat;

private String name;

public Ingredient(){

}

public Ingredient(int etat, String name) {

this.etat = etat;

this.name = name;

}

/\*\*

\* @return the etat

\*/

public int getEtat() {

return etat;

}

public void setEtat(int etat) {

this.etat = etat;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

*Créé avec HelpNDoc Personal Edition:* [*Nouvelles et informations sur les outils de logiciels de création d'aide*](http://www.helpauthoringsoftware.com/fr)

**MainController.java**

package fr.formation.CONTROLLER;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.MessageSource;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;

import fr.formation.MODEL.Menu;

@Controller

public class MainController {

@Autowired

private MessageSource messages;

// Object model and view, pas d'importance sur le nom de la méthode

@RequestMapping("/index")

**public ModelAndView index() {**

final ModelAndView mav = new ModelAndView() ;

mav.setViewName("index");

// recup list de menu

final List<String> menuKeys = Arrays.asList(getMessage("menu.list").split(","));

final List<Menu> menus = new ArrayList<>();

for (final String menuKey : menuKeys) {

final String prefix = "menu."+ menuKey.trim();

final String title = getMessage(prefix+ ".title");

final String url = getMessage(prefix+ ".url");

menus.add(new Menu(title, url));

}

mav.getModel().put("menus", menus);

return mav;

}

**private String getMessage(final String key) {**

return this.messages.getMessage(key, null,null);

}

}

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Placer correctement les fichiers**

C:.

├───.settings

├───src

│ ├───main

│ │ ├───java

│ │ │ └───fr

│ │ │ └───formation

│ │ │ ├───CONTROLLER

│ │ │ ├───DAO

│ │ │ ├───entity

│ │ │ ├───MODEL

│ │ │ └───service

│ │ ├───resources

│ │ └───webapp

│ │ ├───views

│ │ └───WEB-INF

│ └───test

│ ├───java

│ └───resources

└───target

├───classes

│ └───fr

│ └───formation

│ ├───CONTROLLER

│ ├───DAO

│ ├───entity

│ ├───MODEL

│ └───service

├───cocktailBar-0.0.1

│ ├───META-INF

│ ├───views

│ └───WEB-INF

│ ├───classes

│ │ └───fr

│ │ └───formation

│ │ └───CONTROLLER

│ └───lib

├───generated-sources

│ └───annotations

├───m2e-wtp

│ └───web-resources

│ └───META-INF

│ └───maven

│ └───fr.formation

│ └───cocktailBar

├───maven-archiver

└───test-classes

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation et EPub facile*](http://www.helpndoc.com/fr)

**ingredient.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Liste des ingrédients</title>

</head>

<body>

<h1>Liste des ingrédients</h1>

<table>

<thead>

<tr>

<th>Nom</th>

<th>Etat</th>

</tr>

</thead>

<tbody>

<c:forEach items=*"*${ingredients}*"* var=*"ingredient"*>

<tr>

<td>${ingredient.name}</td>

<td>${ingredient.etat}</td>

</tr>

</c:forEach>

</tbody>

</table>

</body>

</html>

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide HTML facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**index.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Gestion des cocktails</title>

<META HTTP-EQUIV=*"Refresh"* CONTENT=*"5"*>

</head>

<body>

<h1>Bienvenue dans l'applicationCocktail</h1>

<div>

Menu :

<ul>

<c:forEach items=*"*${menus}*"* var=*"menu"*>

<c:url value=*"*${menu.url}*.html"* var=*"menuUrl"*></c:url>

<li><a href=*"*${menuUrl}*"*>${menu.title}</a></li>

</c:forEach>

</ul>

</div>

</body>

</html>

*Créé avec HelpNDoc Personal Edition:* [*Produire des aides en ligne pour les applications Qt*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

**pom.properties**

*Créé avec HelpNDoc Personal Edition:* [*Qu'est-ce qu'un outil de création d'aide ?*](http://www.helpauthoringsoftware.com/fr)

**menu.properties**

# Liste des menus de l'application

menu.list=cocktailList, ingredientList, addCocktail, addIngredient, search

menu.cocktailList.title=Liste des cocktails

menu.cocktailList.url=/cocktails

menu.ingredientList.title=Liste des ingredients

menu.ingredientList.url=/ingredients

menu.addCocktail.title=Ajouter un cocktail

menu.addCocktail.url=/cocktail/add

menu.addIngredient.title=Ajouter un ingredient

menu.addIngredient.url=/ingredient/add

menu.search.title=Recherche

menu.search.url=/search

*Créé avec HelpNDoc Personal Edition:* [*Générateur gratuit de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Structure**

C:.

│ .classpath

│ .project

│ pom.xml

│

├───src

│ ├───main

│ │ ├───java

│ │ │ └───fr

│ │ │ └───formation

│ │ │ ├───CONTROLLER

│ │ │ │ IngredientController.java

│ │ │ │ MainController.java

│ │ │ │

│ │ │ ├───DAO

│ │ │ │ IngredientDAO.java

│ │ │ │

│ │ │ ├───entity

│ │ │ │ Ingredient.java

│ │ │ │

│ │ │ ├───MODEL

│ │ │ │ Menu.java

│ │ │ │

│ │ │ └───service

│ │ │ IngredientService.java

│ │ │

│ │ ├─── resources

│ │ │ menu.properties

│ │ │

│ │ └───webapp

│ │ ├───views

│ │ │ index.jsp

│ │ │ ingredients.jsp

│ │ │

│ │ └───WEB-INF

│ │ applicationContext.xml

│ │ web.xml

│ │

│ └───test

│ ├───java

│ └───resources

└───target

│ cocktailBar-0.0.1.war

│

├───classes

│ │ menu.properties

│ │

│ └───fr

│ └───formation

│ ├───CONTROLLER

│ │ IngredientController.class

│ │ MainController.class

│ │

│ ├───DAO

│ │ IngredientDAO.class

│ │

│ ├───entity

│ │ Ingredient.class

│ │

│ ├───MODEL

│ │ Menu.class

│ │

│ └───service

│ IngredientService.class

│

├───cocktailBar-0.0.1

│ ├───META-INF

│ ├───views

│ │ index.jsp

│ │

│ └───WEB-INF

│ │ applicationContext.xml

│ │ web.xml

│ │

│ ├───classes

│ │ └───fr

│ │ └───formation

│ │ └───CONTROLLER

│ │ MainController.class

│ │

│ └───lib

│ commons-logging-1.2.jar

│ javax.servlet-api-3.1.0.jar

│ jstl-1.2.jar

│ spring-aop-4.3.4.RELEASE.jar

│ spring-beans-4.3.4.RELEASE.jar

│ spring-context-4.3.4.RELEASE.jar

│ spring-core-4.3.4.RELEASE.jar

│ spring-expression-4.3.4.RELEASE.jar

│ spring-web-4.3.4.RELEASE.jar

│ spring-webmvc-4.3.4.RELEASE.jar

│

├───generated-sources

│ └───annotations

├───m2e-wtp

│ └───web-resources

│ └───META-INF

│ │ MANIFEST.MF

│ │

│ └───maven

│ └───fr.formation

│ └───cocktailBar

│ pom.properties

│ pom.xml

│

*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation CHM facile*](http://www.helpndoc.com/fr)

**Projet Cocktail avec BDD**

|  |  |
| --- | --- |
| orm.xml | Déclaration de la structure des tables mysql  <entity class=*"fr.formation.entity.Ingredient"*>  <table name=*"ingredient"*></table>  <attributes>  <id name=*"id"*>  <column name=*"id\_ing"* />..................-> nom du champ  <generated-value strategy=*"IDENTITY"* />...-> indique que  c'est une clé  </id>  <basic name=*"etat"*>........................-> nom dans Java  <column name=*"state"* />.............-> nom dans mysql  </basic>  </attributes>  </entity>  <entity class=*"chemin complet jusqu'à la classe java"*>  <table name=*"nom\_table"*></table>    <basic name=*"idJava"*>  <column name=*"id\_mysql"* />  </basic>  </attributes>  </entity> |
| persistence.xml | Définit le "DriverManager" |
| pom.xml | Ajout des dépendences utiles pour la base :  1) "mysql connector"  2) "spring data jpa  3) "hibernate core" dernière version  4) Copie "hibernate core" changer core en hibernate-entitymanager |
| applicationContext.xml | Spring attent entityManagerEntity Name pris dans persistance.xml -->   1. Ajout de 2 beans :    * entityManagerFactory    * transactionManager 2. Ajout <jpa:repositories>   base-package=*"fr.formation.DAO"*  <bean id=*"entityManagerFactory"*  class=*"org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"*>  <property name=*"persistenceUnitName"* value=*"bar"*></property>  </bean>    <bean id=*"transactionManager"*  class=*"org.springframework.orm.jpa.JpaTransactionManager"*>  <property name=*"entityManagerFactory"* ref=*"entityManagerFactory"* />  </bean> |
| IngredientController.java | Ajout de ModelAndView avec un appel à la jsp qui fait action |
|  | package fr.formation.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import fr.formation.DAO.IngredientDAO;  import fr.formation.entity.Ingredient;  @Service  public class IngredientService {  @Autowired  private IngredientDAO dao;    public List<Ingredient> getAll(){  return this.dao.findAll();    }  @Transactional  public Ingredient create(final Ingredient ingredient){  return this.dao.save(ingredient);  }  } |

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides CHM gratuit*](http://www.helpndoc.com/fr)

**orm.xml**

<?xml version=*"1.0"*?>

<entity-mappings

xmlns=*"http://www.eclipse.org/eclipselink/xsds/persistence/orm"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.eclipse.org/eclipselink/xsds/persistence/orm http://www.eclipse.org/eclipselink/xsds/eclipselink\_orm\_2\_1.xsd"*

version=*"2.1"*>

<entity class=*"fr.formation.entity.Ingredient"*>

<table name=*"ingredient"*></table>

<attributes>

<id name=*"id"*>

<column name=*"id\_ing"* />

<generated-value strategy=*"IDENTITY"* />

</id>

<basic name=*"etat"*>

<column name=*"state"* />

</basic>

<basic name=*"nom"*>

<column name=*"name"* />

</basic>

</attributes>

</entity>

</entity-mappings>

*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation CHM facile*](http://www.helpndoc.com/fr)

**persistence.xml**

<persistence 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"*

version=*"2.1"*>

<!-- On met ce que l'on veut -->

<persistence-unit name=*"bar"* >

<!-- une classe d'hibernate qui va nous servir -->

<provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

<mapping-file>META-INF/orm.xml</mapping-file>

<properties>

<property name=*"hibernate.connection.driver\_class"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"hibernate.connection.url"* value=*"jdbc:mysql://localhost:3306/cocktail"*/>

<property name=*"hibernate.connection.user"* value=*"root"*/>

<property name=*"hibernate.connection.password"* value=*""*/>

<property name=*"hibernate.dialect"* value=*"org.hibernate.dialect.MySQL5Dialect"*/>

</properties>

</persistence-unit>

</persistence>

*Créé avec HelpNDoc Personal Edition:* [*Outil de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**pom.xml**

*<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->*

*<!-- Aller sous mvn repository pour copier les repository -->*

*<!-- Faire alt+F5 pour mettre à jour les libraries dans le projet -->*

*<!-- Posé à la racine du projet -->*

*<!-- \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ -->*

*<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xsi:schemaLocation="http://maven.apache.org/POM/4.0.0*

*http://maven.apache.org/xsd/maven-4.0.0.xsd">*

*<modelVersion>4.0.0</modelVersion>*

*<groupId>fr.formation</groupId>*

*<artifactId>cocktailBar</artifactId>*

*<version>0.0.1</version>*

*<packaging>war</packaging>*

*<properties>*

*<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>*

*</properties>*

*<dependencies>*

*<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->*

*<dependency>*

*<groupId>javax.servlet</groupId>*

*<artifactId>javax.servlet-api</artifactId>*

*<version>3.1.0</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/javax.servlet.jsp.jstl/jstl -->*

*<dependency>*

*<groupId>javax.servlet</groupId>*

*<artifactId>jstl</artifactId>*

*<version>1.2</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-context</artifactId>*

*<version>4.3.4.RELEASE</version>*

*</dependency>*

*<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->*

*<dependency>*

*<groupId>org.springframework</groupId>*

*<artifactId>spring-webmvc</artifactId>*

*<version>4.3.4.RELEASE</version>*

*</dependency>*

<!-- Dépendance à ajouter pour les accés à la base -->

<!-- 1) "mysql connector" Ajout BDD -->

<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.40</version>

</dependency>

<!-- 2) "spring data jpa" -->

<!-- https://mvnrepository.com/artifact/org.springframework.data/spring-data-jpa -->

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>1.10.5.RELEASE</version>

</dependency>

<!-- 3) "hibernate core" dernière version -->

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.5.Final</version>

</dependency>

<!-- 4) Copie "hibernate core" changer core en hibernate-entitymanager-->

<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>5.2.5.Final</version>

</dependency>

</dependencies>

*<!-- Configuration des plugins -->*

*<build>*

*<plugins>*

*<plugin>*

*<groupId>org.apache.maven.plugins</groupId>*

*<artifactId>maven-compiler-plugin</artifactId>*

*<version>2.5.1</version>*

*<configuration>*

*<source>1.8</source>*

*<target>1.8</target>*

*</configuration>*

*</plugin>*

*</plugins>*

*</build>*

*</project>*

*Créé avec HelpNDoc Personal Edition:* [*Environnement de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**applicationContext.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

**xmlns:jpa=*"http://www.springframework.org/schema/data/jpa"***

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

***http://www.springframework.org/schema/data/jpa***

***http://www.springframework.org/schema/data/jpa/spring-jpa.xsd"*>**

**<**context:component-scan base-package=*"fr.formation.CONTROLLER,fr.formation.MODEL,fr.formation.service"*></context:component-scan>

<!-- nom de la propriété -->

<!-- view classe -->

<bean id=*"viewResolver"*

.....

<bean id=*"messageSource"*

.....

<!-- Spring attent entityManagerEntity NAme pris dans persistance.xml -->

<bean id=*"entityManagerFactory"*

class=*"org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"*>

<property name=*"persistenceUnitName"* value=*"bar"*></property>

</bean>

<bean id=*"transactionManager"*

class=*"org.springframework.orm.jpa.JpaTransactionManager"*>

<property name=*"entityManagerFactory"* ref=*"entityManagerFactory"* />

</bean>

<jpa:repositories base-package=*"fr.formation.DAO"*></jpa:repositories>

</beans>

*Créé avec HelpNDoc Personal Edition:* [*Créer des fichiers d'aide pour la plateforme Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

**IngredientController.java**

package fr.formation.CONTROLLER;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.servlet.ModelAndView;

import fr.formation.entity.Ingredient;

import fr.formation.service.IngredientService;

**@Controller**

@RequestMapping("/ingredients")

public class IngredientController {

**@Autowired**

private IngredientService service;

**@RequestMapping**

public ModelAndView list() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("ingredients");

mav.addObject("ingredients", this.service.getAll());

return mav;

}

**@RequestMapping("/add")**

public ModelAndView add() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("addIngredient");

mav.addObject("ingredients", this.service.getAll());

return mav;

}

**@RequestMapping(value = "/add", method = RequestMethod.POST)**

public String newIngredient(final HttpServletRequest request) {

final String name = request.getParameter("name");

final Integer state = Integer.parseInt(request.getParameter("state"));

// ----> ajout @Transactional dans IngredientService.java

this.service.create(new Ingredient(state, name));

return "redirect:/ingredients/add.html";

}

**@RequestMapping(value = "/add2", method = RequestMethod.POST)**

public String newIngredient(@RequestParam final String name, @RequestParam final Integer state ) {

// ----> ajout @Transactional dans IngredientService.java

this.service.create(new Ingredient(state, name));

return "redirect:/ingredients/add.html";

}

}

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**IngredientDAO.java**

**package** fr.formation.DAO;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** org.springframework.stereotype.Repository;

**import** fr.formation.entity.Ingredient;

@Repository

**public** **interface** IngredientDAO **extends** JpaRepository<Ingredient, Integer>{

}

*Créé avec HelpNDoc Personal Edition:* [*Produire des livres électroniques facilement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**IngredientService.java**

package fr.formation.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import fr.formation.DAO.IngredientDAO;

import fr.formation.entity.Ingredient;

**@Service**

public class IngredientService {

**@Autowired**

private IngredientDAO dao;

public List<Ingredient> getAll(){

return this.dao.findAll();

}

**@Transactional**

public Ingredient create(final Ingredient ingredient){

return this.dao.save(ingredient);

}

}

*Créé avec HelpNDoc Personal Edition:* [*Éditeur complet de livres électroniques ePub*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**AddIngredient.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Ajout d'un ingrédient</title>

</head>

<body>

<h1>AJOUT D'UN INGREDIENT</h1>

<c:url value=*"/ingredients/add.html"* var=*"addUrl"* />

<form action=*"*${addUrl}*"* method=*"POST"*>

<label for=*"name"*>Nom : </label>

<input id=*"name"* name=*"name"* class=*"form-control"* />

<label for=*"state"*>Etat : </label>

<input id=*"state"* name=*"state"* type=*"number"* min=*"0"* max=*"2"* class=*"form-control"* />

<button>VALIDER</button>

</form>

<div style="position: *fixed*; bottom: *0*; left: *0*; padding: *20px*; font-size: *18px*;">

<a href=*"*<c:url value=*'/'* />*"* >RETOUR</a>

</div>

</body>

</html>

*Créé avec HelpNDoc Personal Edition:* [*Générateur gratuit de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Ingredient.jsp**

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aide complet*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Mysql**

JPA **J**ava **P**ersistant **A**PI

--> persistance.xml : config qui utilise JPA , nous JPA puis spring par les repositories

ORM

Object Relational Mapping

Hibernate

1) Ajouter 3 dépendances dans pom.xml

1) "mysql connector" Ajout BDD -->

https://mvnrepository.com/artifact/mysql/mysql-connector-java

2) "spring data jpa" Recherche -->

3) "hibernate core" dernière version -->

4) Copier Hibernate core et modifier artifact en hibernate-entitymanager

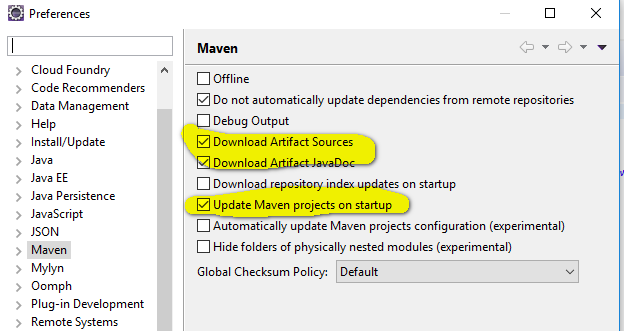
<artifactId>hibernate-entitymanager</artifactId>

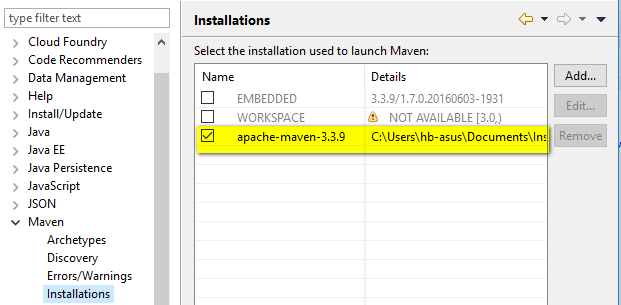
2) Sous /src/java/resources

* 1. Créer un dossier META-INF
  2. puis persistence.xml
  3. puis orm.xml

===> Fichier de config JPA

3) Dans windows / preferences/ MAVEN





Pour chaque entité il faut un **id.**

*Créé avec HelpNDoc Personal Edition:* [*Nouvelles et informations sur les outils de logiciels de création d'aide*](http://www.helpauthoringsoftware.com/fr)

**Config de départ**

*Créé avec HelpNDoc Personal Edition:* [*Créer des sites web d'aide facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Config tjs JPA**

|  |  |  |
| --- | --- | --- |
| JPA | JAVA | SQL |
| PersistenceUnit  "Cocktail" (\*) | IngredientJava  |--> new Ingredient("",0) | Table ingredient  Ligne dans la table |
| PersistenceContext (\*\*)  --> contient les informations Unit | **EntityManagerFactory** : on lui configure le nom de notre entité = "Cocktail" (\*) (\*\*)  en JPA on a un persistence objet  **EntityManager** : gère les objetsJava qui sont des entités  (0,"Whiskey"),  (0,"Tequila"),  (1, "Ice cubes"),  (1, "Sugar")  (2, "CO2"));    EntityManaget soccupe de créer les requêtes sql  **TransactionManager** |  |
|  | Requête JPQL = équivalent de requêtage SQL mais en java | Passe les requêtes en SQL via le JPQL |

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides CHM gratuit*](http://www.helpndoc.com/fr)

**Vue/Controller/**

|  |  |  |  |
| --- | --- | --- | --- |
| **VUE** | **CONTROLLER** |  |  |
| ingredients.jsp<----------------- | IngredientController  RequestMapping  ModelAndView  view name  model  List<Ingredient> <-------- | **@Transactional**  IngredientService  DAO<---------------------------------------------- | IngredientDAO  JPARepository  Spring  EntityManager  BDD  Requêtes JPQL |

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:jpa=*"http://www.springframework.org/schema/data/jpa"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/data/jpa/spring-jpa.xsd"*>

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation d'aide HTML gratuit*](http://www.helpndoc.com/fr)

**GIT**

|  |  |
| --- | --- |
| GIT | REMOTES |
| Gestionnaire de source  --> repository distant  --> Histoire de toutes les modifications  --> Commit --> nouvelle révision / Version | Github --> Origin  Branches   * + MASTER   + tags |
| Local (clone) |
|  | PC  --> Staging  --> Working Directory |

Différence entre SVN et GIT : plusieurs repository avant l'original

Branche principale :

tag : copie figée des sources

|  |  |  |  |
| --- | --- | --- | --- |
| 3 couches en local | | | Distant |
| Workspace | staging | Repository local | Repository distant (remote) |
| src/pom.xml-------  |---------------------------------> | git add  src/pom.xml |----------------------------------> | git commit  |--------------------------------------------->  révision #1 | git push  révision #1 |
| pom.xml | git add | révision #2 | révision #1 |
|  |  |  | Si projet #32 ne fonctionne plus on peut revenir sur projet #28 |

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation iPhone gratuit*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**Liste des commandes**

# Créer un repositiry de son poste sur github

C:/user/workspace > mkdir <MonProjet>

* + **git clone https://github/flo1012/nom\_du\_repository**
  + **git add** <fichier>
  + **git commit -m "commentiare"**
  + **git push origin master** --------------> demande mot de passe

# Récupérer un repository

Sur github :

Aller sur le repository qui nous interesse , récupérer l'@ https sous clone puis :

* + **git clone https://github/flo1012/nom\_du\_repository**

Pour mettre à jour :

* + **git pull**

# Autres commandes

* + **git status**
  + **git remote**
  + **git rm nom\_fichier**
  + **git reset**

Pour voir la différence :

**git diff**

# Avec plusieurs branches

* + git branch
  + git checkout security
  + git status
  + git branch

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide CHM facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Création de mon Git pour la doc**

…or create a new repository on the command line

echo "# JAVA\_JEE" >> README.md

git init

git add \*

git commit -m "**Documentation chronologique de la formation Java JEE**"

git remote add origin https://github.com/Flo1012/JAVA\_JEE.git

git push -u origin master

*Créé avec HelpNDoc Personal Edition:* [*Générateur gratuit de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Projet Cocktail de Jeremy**

org.webjars

* + jquery-ui 1.12.1
  + datatables 1.10.12-1
  + datatables-colreorder 1.2.0
  + bootstrap 3.1.0

jquery datatables (jquery est un framework de javascript)

*Créé avec HelpNDoc Personal Edition:* [*Créer des aides HTML, DOC, PDF et des manuels depuis une même source*](http://www.helpndoc.com/fr/outils-creation-aide)

**src/main/java**

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide PDF facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**CONTROLLER**

*Créé avec HelpNDoc Personal Edition:* [*Produire des livres Kindle gratuitement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

CocktailController

package fr.formation.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.servlet.ModelAndView;

import fr.formation.entity.Cocktail;

import fr.formation.entity.CocktailPart;

import fr.formation.service.CocktailService;

@Controller

@RequestMapping("/cocktails")

public class CocktailController {

@Autowired

private CocktailService service;

@RequestMapping("/add")

public ModelAndView add() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("addCocktail");

return mav;

}

@RequestMapping

public ModelAndView list() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("cocktails");

mav.addObject("cocktails", this.service.getAll());

return mav;

}

@RequestMapping(value = "/add", method = RequestMethod.POST)

public String newCocktail(@RequestParam final String name,

@RequestParam final Float price,

@RequestParam(required = false) final Boolean withAlcohol) {

final Cocktail cocktail = new Cocktail();

cocktail.setName(name);

System.out.println("Cocktail name : " + name);

cocktail.setPrice(price);

cocktail.setWithAlcohol(withAlcohol != null);

this.service.create(cocktail);

return "redirect:/cocktails/add.html";

}

@RequestMapping("/test")

public void test() {

final List<CocktailPart> parts = this.service.getCocktailParts();

System.out.println("parts size : " + parts.size());

}

}

*Créé avec HelpNDoc Personal Edition:* [*Produire des aides en ligne pour les applications Qt*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

IngredientController

package fr.formation.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.servlet.ModelAndView;

import fr.formation.entity.Ingredient;

import fr.formation.service.IngredientService;

@Controller

@RequestMapping("/ingredients")

public class IngredientController {

@Autowired

private IngredientService service;

@RequestMapping("/add")

public ModelAndView add() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("addIngredient");

return mav;

}

@RequestMapping

public ModelAndView list() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("ingredients");

mav.addObject("ingredients", this.service.getAll());

return mav;

}

@RequestMapping(value = "/add", method = RequestMethod.POST)

public String newIngredient(final HttpServletRequest request) {

final String name = request.getParameter("name");

final Integer state = Integer.parseInt(request.getParameter("state"));

this.service.create(new Ingredient(name, state));

return "redirect:/ingredients/add.html";

}

@RequestMapping(value = "/add2", method = RequestMethod.POST)

public String newIngredient2(@RequestParam final String name,

@RequestParam final Integer state) {

this.service.create(new Ingredient(name, state));

return "redirect:/ingredients/add.html";

}

}

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation d'aide HTML gratuit*](http://www.helpndoc.com/fr)

MainController

package fr.formation.controller;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.MessageSource;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;

import fr.formation.model.Menu;

@Controller

public class MainController {

@Autowired

private MessageSource messages;

private String getMessage(final String key) {

return this.messages.getMessage(key, null, null);

}

@RequestMapping("/index")

public ModelAndView index() {

final ModelAndView mav = new ModelAndView();

mav.setViewName("index");

final List<String> menuKeys = Arrays

.asList(this.getMessage("menu.list").split(","));

final List<Menu> menus = new ArrayList<>();

for (final String menuKey : menuKeys) {

final String prefix = "menu." + menuKey.trim();

final String title = this.getMessage(prefix + ".title");

final String url = this.getMessage(prefix + ".url");

menus.add(new Menu(title, url));

}

mav.getModel().put("menus", menus);

return mav;

}

}

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**DAO**

*Créé avec HelpNDoc Personal Edition:* [*Générateur complet de livres électroniques Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

CocktailDAO

package fr.formation.dao;

import org.springframework.data.jpa.repository.JpaRepository;

import fr.formation.entity.Cocktail;

public interface CocktailDao extends JpaRepository<Cocktail, Integer> {

}

*Créé avec HelpNDoc Personal Edition:* [*Outils facile d'utilisation pour créer des aides HTML et des sites web*](http://www.helpndoc.com/fr/outils-creation-aide)

IngredientDao

package fr.formation.dao;

import org.springframework.data.jpa.repository.JpaRepository;

import fr.formation.entity.Ingredient;

public interface IngredientDao extends JpaRepository<Ingredient, Integer> {

}

*Créé avec HelpNDoc Personal Edition:* [*Sites web iPhone faciles*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**ENTITY**

*Créé avec HelpNDoc Personal Edition:* [*Générateur facile de livres électroniques et documentation*](http://www.helpndoc.com/fr)

Cocktail

*Créé avec HelpNDoc Personal Edition:* [*Environnement de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

Ingredient

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**MODEL**

*Créé avec HelpNDoc Personal Edition:* [*Avantages d'un outil de création d'aide*](http://www.helpauthoringsoftware.com/fr)

Menu

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**SERVICE**

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation Qt Help gratuit*](http://www.helpndoc.com/fr)

CocktailService

*Créé avec HelpNDoc Personal Edition:* [*Générateur complet de livres électroniques Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

IngredientService

*Créé avec HelpNDoc Personal Edition:* [*Produire des aides en ligne pour les applications Qt*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

**sr/main/resources**

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aide complet*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**menu.properties**

*Créé avec HelpNDoc Personal Edition:* [*Produire des aides en ligne pour les applications Qt*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

**META-INF**

*Créé avec HelpNDoc Personal Edition:* [*Créer des fichiers d'aide pour la plateforme Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

orm.xml

*Créé avec HelpNDoc Personal Edition:* [*Créer des fichiers d'aide pour la plateforme Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creez-des-fichiers-daide-pour-le-framework-qt-help)

persistence.xml

*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation Qt Help facile*](http://www.helpndoc.com/fr)

**webapp**

*Créé avec HelpNDoc Personal Edition:* [*Avantages d'un outil de création d'aide*](http://www.helpauthoringsoftware.com/fr)

css

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

Nouveau chapitre

*Créé avec HelpNDoc Personal Edition:* [*Générateur complet de livres électroniques ePub*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

views

*Créé avec HelpNDoc Personal Edition:* [*Produire facilement des livres électroniques Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

WEB-INF

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**Outils**

*Créé avec HelpNDoc Personal Edition:* [*Écrire des livres électronique Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

**STAN**

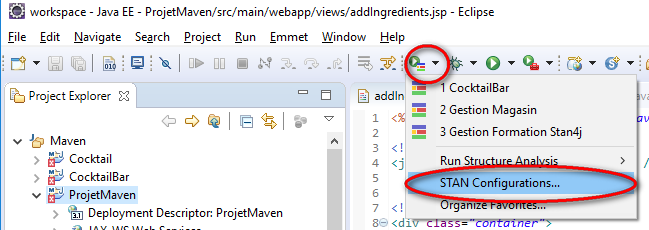
## Télécharger Plugins pour Eclipse

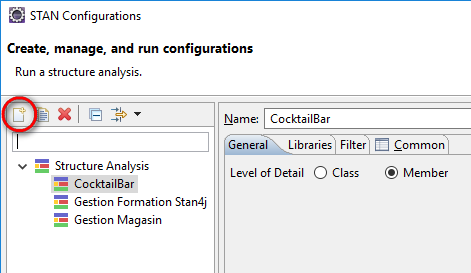
Installation de la fonction Eclipse

STAN est disponible via notre site de mise à jour Eclipse. Veuillez suivre les instructions ci-dessous.

* + - Depuis Eclipse, allez à Aide - Installer un nouveau logiciel ...
    - Ajouter un nouveau site pour STAN avec l'URL http://update.stan4j.com/ide
    - Sélectionnez la fonction STAN IDE et appuyez sur "Suivant"
    - Suivez l'assistant, lisez et acceptez les termes de la licence
    - Redémarrer Eclipse

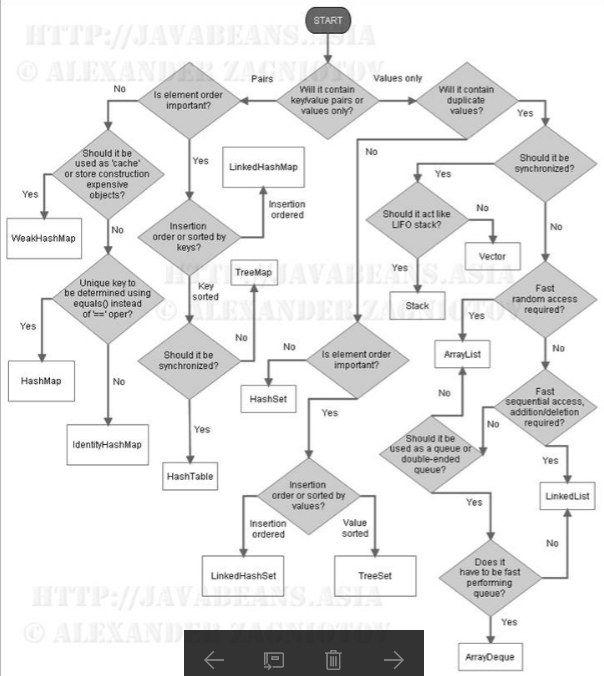
## Générer un projet STAN à partir d'un projet éclipse :



 ==> 

*Créé avec HelpNDoc Personal Edition:* [*Générer facilement des livres électroniques Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

**Organigramme choix Objet Collection**



*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation Qt Help facile*](http://www.helpndoc.com/fr)

**GlassFish**

*Créé avec HelpNDoc Personal Edition:* [*Créer des sites web d'aide facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Install**

Télécharger <https://glassfish.java.net/download.html>

Sous unix : **jar xvf glassfish-4.1.zip**

Lire le fichier readme.txt :

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Lancer la console Glassfish déclaration du path (variable d'environnement JAVA\_HOME)**

La commande "asadmin" est utilisé pour controler et manager GlassFish

* + start,
  + stop,
  + configure,
  + deploy applications,
  + etc...

Lancer les commandes GlassFish sous **glassfish4\glassfish\bin** !

Pour démarrer : asadmin start-domain

Pour arrêter : asadmin stop-domain

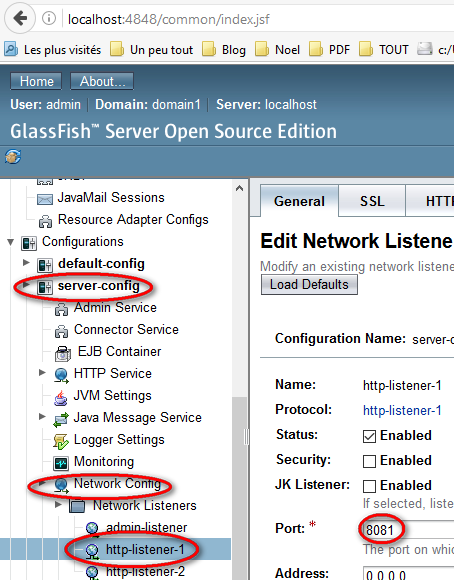
Ouvrir GlassFish: **http://localhost:4848**

*Créé avec HelpNDoc Personal Edition:* [*Qu'est-ce qu'un outil de création d'aide ?*](http://www.helpauthoringsoftware.com/fr)

**Changement port 8080**

Aller sous la console Glassfish

changer



ajouter la variable d'environnement GLASSFISH\_AUTODEPLOY

qui pointe vers

C:\Users\hb-asus>echo %GLASSFISH\_AUTODEPLOY%

C:\Users\hb-asus\Documents\Install\glassfish4\glassfish\domains\domain1\autodeploy

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentations PDF gratuit*](http://www.helpndoc.com/fr)

**Paramétrage des logs**

Plusieurs niveaux de logs :

* 1. fatal
  2. error
  3. warning
  4. debug
  5. info

## 1) Historique des API

### API de Login JUL

Pour gérer les logs on peut utiliser l'API de Login JUL : Java Util Login

Celle-ci utilise la sortie standard

Un peu vieille

### API Log4J

Nouvelle librairy Log4J puis Log4J2 et SLF4J qui centralise toutes les logs en seul point pour créer un pont entre toutes les api java.

### Implémentation LogBack

Puis LogBack (implementation) plus pratique de SLF4J

Dans les prog, on utilise SLF4J.

L'utilisation de <scope> :

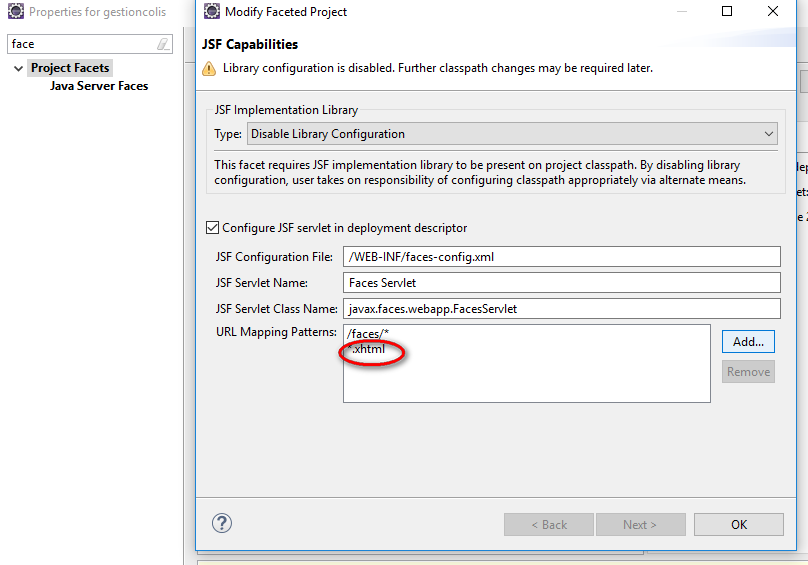
<scope>test</scope> ---> lors de la compilation du WAR (Web Archive) n'importe pas les librairies

### MyFaces

L'implementation Oracles JSF : Mojara

Le JAR des Drivers doit être posé dans Glassfish

Config altEntrée

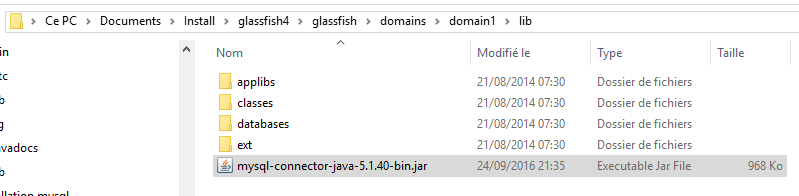
 Ajout de

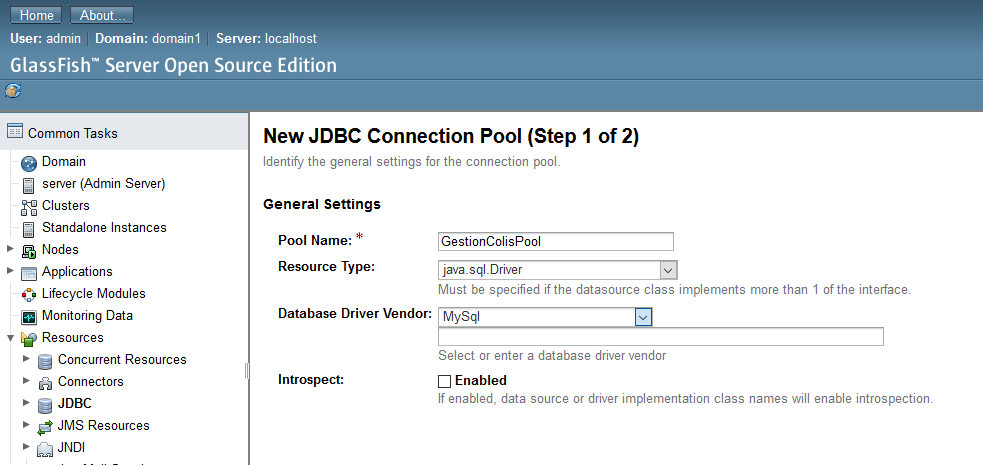
*Créé avec HelpNDoc Personal Edition:* [*Création d'aide CHM, PDF, DOC et HTML d'une même source*](http://www.helpndoc.com/fr/outils-creation-aide)

**Mysql pool**

## Copier le jar de mysql dans Glassfish :

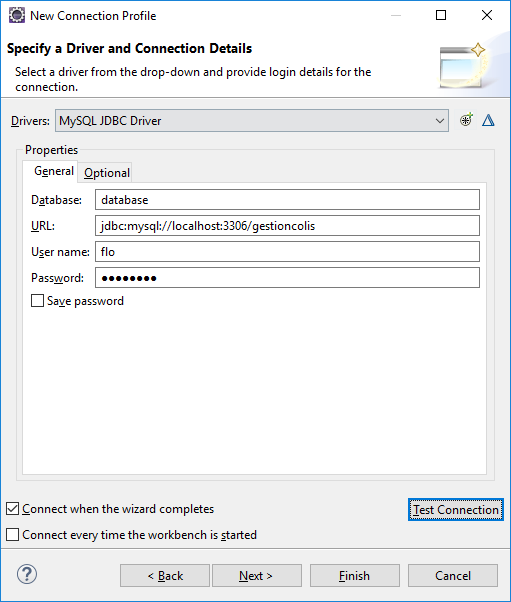
C:\Users\hb-asus\Documents\Install\glassfish4\glassfish\domains\domain1\lib





Créer un user dans la base de données avec un mot de passe.

Puis paramétrer la connexion dans eclipse :



*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation Qt Help facile*](http://www.helpndoc.com/fr)

**Dans eclipse**

# Création de la connection à la base de données

|  |  |  |
| --- | --- | --- |
| Aller dans les propriétés du projet : |  |  |
| Penser à sélectionner le drivers de la BDD |  |  |

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Schéma Appli**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| APPLICATION | | | ServeurGlassFish | BDDMysql |
| Présentation | Métier | Persistence |  |  |
| HTML | Commande Id=1 | Hibernate | Ressource JBBC | Table Commande |
| Render Response | Commande Id=null | JPA | 1 connexion |
| XHTML | Commande Id=2 | Entity Manager | Pool de connexion |
| Autres étape du cycle |  | Persistence Context |  |
| JSF | Managed Beans | DAO |  |

Pas de couche SERVICES =objet passe plat idée de passage dans la couche service

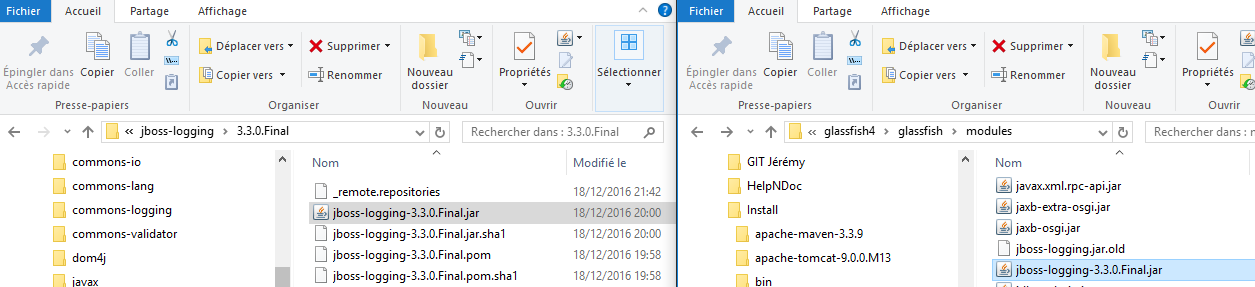
## MAJ dans GlassFish

Copier jboss-logging-3.3.0.Final.jar :

Qui se trouve sous C:\Users\hb-asus\.m2\repository\org\jboss\logging\jboss-logging\3.3.0.Final

Dans répertoire de GlassFish :

1. Renommer jboss-logging.jar en jboss-logging.jar.old
2. C:\Users\hb-asus\Documents\Install\glassfish4\glassfish\modules



*Créé avec HelpNDoc Personal Edition:* [*Générateur gratuit de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Remote debugging**

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Configuration Glassfish et éclipse**

|  |  |
| --- | --- |
| Sous Glassfish |  |
| Sous eclipse | Sous  Debug config  remote Java Appli  new  nom  lier appli  localhost  port 9009 |

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**Déclaration des Logs en java**

## Commande Maven pour connaître les dépendances créer dans le pom.xml

* + mvn dependency:tree

## Qd on déclarer un log :

On ne doit déclarer qu'un seul log par classe en static et final

* + **private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(ProductController.**class**);

postConstruct méthode

entité détachée c'est à dire entité non manager.

(à voir)

## Astuce:

Normalement, pour le readAll, il faut une transaction.

On ne va pas rajouter une transaction mais appeler le read() de la Dao et utiliser la même transaction.

Lors de l'écriture de la classe qui gère les erreurs, pas de constructeur par défaut cela évite de créer une erreur sans rien

*Créé avec HelpNDoc Personal Edition:* [*Générateur complet de livres électroniques ePub*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Classe EntityManager et UserTransaction**

EntityManager :

C'est une classe qui est chargée de mettre en musique les correspondances définies dans les entités,

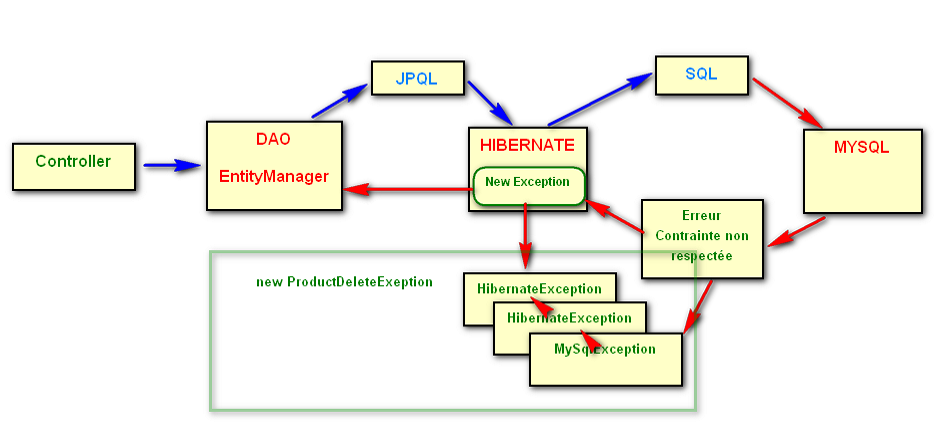
et qui réalise donc toutes les opérations CRUD (Create, Read, Update, Delete) sur la base de données.

## UserTransaction

C'est une classe

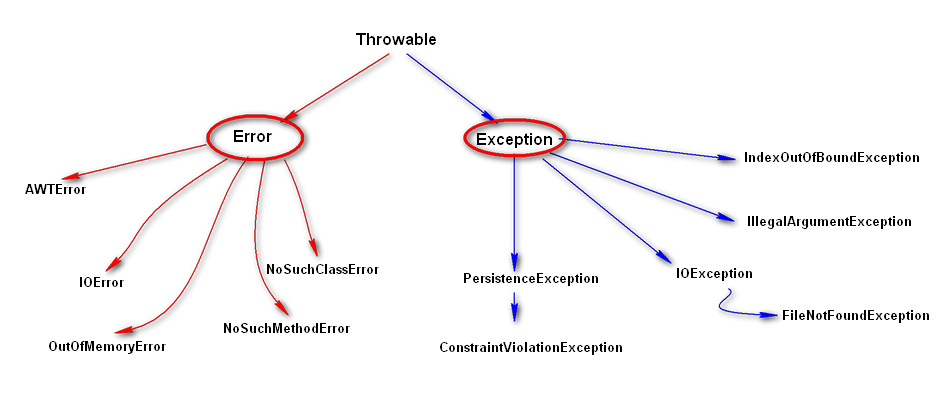
*Créé avec HelpNDoc Personal Edition:* [*Écrire des livres électronique Kindle*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/creer-ebooks-pour-kindle-amazon)

**Schéma de vie d'une exception**



*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**Schéma des ERROR / EXCEPTION**



*Créé avec HelpNDoc Personal Edition:* [*Générateur facile de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Lancer Glassfish par un Bat**

1. Créer un répertoire scripts sous le workspace d'éclipse
2. Créer un fichier gl.bat :

@echo off

Set GLASSFISH\_PATH=C:\Users\hb-asus\Documents\Install\glassfish4

**if** "%1" == "start" (

echo [INFO] Starting Glassfish

call %GLASSFISH\_PATH\bin\asadmin start-domain

**goto** end

)

**if** "%1" == "stop" (

echo [INFO] Stopping Glassfish

call %GLASSFISH\_PATH\bin\asadmin stop-domain

**goto** end

)

**if** "%1" == "" (

echo [INFO] Il manque un argument

) **else** (

echo [INFO] Il manque un argument

)

:end

@echo on

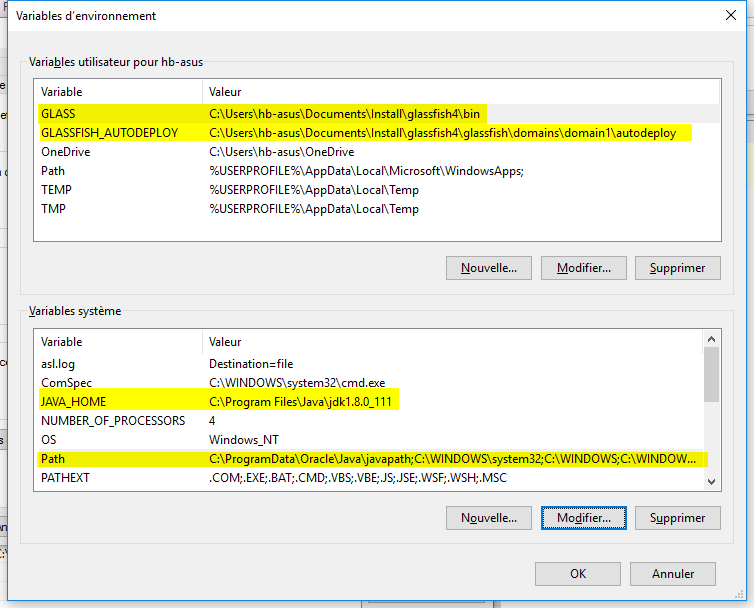
1. Créer une variable d'environnement

Sous windows :

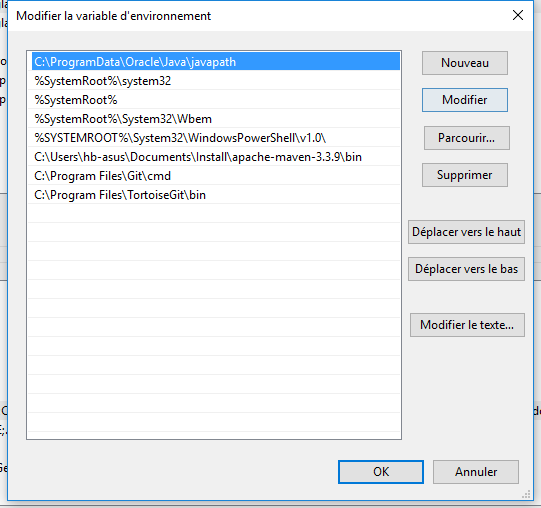
Win + Pause

Paramètres système avancés

Variables d'environnement



Détail supplémentaire pour la variable Path



*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide PDF facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Liste des tutos donnés par Jérémy**

1. JSF (liste) : <http://www.mkyong.com/tutorials/jsf-2-0-tutorials/>
   1. Navigation implicite : <http://www.mkyong.com/jsf2/implicit-navigation-in-jsf-2-0/>
      * Vous donnera un aperçu de la configuration de navigation dans faces-config.xml
   2. Navigation par action dans un form et méthode Java : <http://www.mkyong.com/jsf2/jsf-form-action-navigation-rule-example>
   3. Composant de liste déroulante : <http://www.mkyong.com/jsf2/jsf-2-dropdown-box-example/>
   4. Le tag 'f:param' pour envoyer des paramètres (sera utilisé pour l'édition) : <http://www.mkyong.com/jsf2/jsf-2-param-example/>
   5. Utilisation simple d'un bean managé (explique la configuration nécessaire pour les anciennes version de JSF) : <http://www.mkyong.com/jsf2/configure-managed-beans-in-jsf-2-0/>
2. Spring
   1. Hello World Spring dans un projet Java (pas de web) : <http://www.mkyong.com/spring3/spring-3-hello-world-example/>
   2. Exemple d'injection de dépendance (DI) sans l'annotation Autowired, avec de la configuration XML et le setter : <http://www.mkyong.com/spring/spring-di-via-setter-method/>
   3. Exemple d'injection de dépendances avec annotation (dans la 2eme partie) : <http://www.mkyong.com/spring/spring-auto-scanning-components/>
3. Hibernate
   1. Association one-to-many (XML) :   
      http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example/
   2. Association one-to-many (Annotations) :   
      http://www.mkyong.com/hibernate/hibernate-one-to-many-relationship-example-annotation/
4. Jackson (conversion Java<->JSON)
   1. http://www.mkyong.com/java/how-to-convert-java-object-to-from-json-jackson/

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation et EPub gratuit*](http://www.helpndoc.com/fr)

**JFS**

*Créé avec HelpNDoc Personal Edition:* [*Générer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Annotations**

Synthèse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | bean | | controller | dao |
|  | entity | login |  |  |
| implements | Serializable | Serializable | Serializable |  |
| **extends** |  |  |  | AbstractDao<Product> |
| Classe | @ManagedBean  @ViewScoped | @ManagedBean  @ApplicationScoped | @ManagedBean  @ViewScoped | rien |
| Attributs |  |  |  | @PersistanceContext  **protected** EntityManager em;  @Resource  **private** UserTransaction transaction; |
| Méthode |  |  |  |  |

|  |  |  |
| --- | --- | --- |
|  | entity | exception |
|  |  |  |
| implements | Serializable |  |
| **extends** |  |  |
| Classe | @Entity  @NamedQuery(name="Bordereau.findAll", query="SELECT b FROM Bordereau b")  **public** **class** Bordereau { |  |
| Attributs | @Id  @GeneratedValue(strategy=GenerationType.***IDENTITY***)  **private** **int** id;  @Temporal(TemporalType.***TIMESTAMP***)  @Column(name="DATE\_SIGNATURE")  **private** Date dateSignature;  **private** String detail;  //bi-directional many-to-one association to Commande  @ManyToOne  @JoinColumn(name="COMMANDE")  **private** Commande commandeBean; |  |
| Méthode |  |  |

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Annotations BEAN**

## Annotations nécessaires pour BEAN

### 1) Au dessus des classes en connexion avec la BDD

@ManagedBean

@ViewScoped

**public** **class** ProductBean **implements** Serializable {

### OU

### 1) Au dessus de la classe de login

@ManagedBean

@SessionScoped

**public** **class** LoginBean **implements** Serializable {

*Créé avec HelpNDoc Personal Edition:* [*Outil de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**Annotations CONTROLLER**

## Annotations nécessaires pour CONTROLLER

### 1) Au dessus de la classe

@ManagedBean

@ViewScoped

public class ProductController implements Serializable {

private static final Logger *LOGGER* = LoggerFactory.*getLogger*(ProductController.class);

### 2) Au dessus de l'attribut

@ManagedProperty("#{productBean}")

private ProductBean productBean;

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides CHM gratuit*](http://www.helpndoc.com/fr)

**Annotations DAO**

## Annotations nécessaires pour DAO

### 1) Au dessus de la classe

Rien

### 2) Attribut

@PersistenceContext

**protected** EntityManager em;

@Resource

**private** UserTransaction transaction;

*Créé avec HelpNDoc Personal Edition:* [*Générateur complet de livres électroniques ePub*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Annotations ENTITY**

## Annotations nécessaires pour ENTITY

### 1) Au dessus de la classe

*Créé avec HelpNDoc Personal Edition:* [*Créer facilement des fichiers Qt Help*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**@ManageBean**

Cette classe sera instancié :

au démarrage de l'application

par défaut le nom est basé sur le nom de la classe avec une minuscule sur la première lettre

Voir l'option lazy

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation iPhone gratuit*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**Ajax**

En ajax, la page n'est pas rafraîchit, il faut indiquer ce qui doit être mis à jour

le render("le rendu") appelle l'attribut à mettre à jour.

Selon l'API utilisant ajax, <h:ajax> <f:ajax> : L'écriture sera particulière à chaque API, les balises n'auront pas les même arguments.

## Pour <f:ajax >

Si on souhaite rafraîchir un élément d'un autre formulaire, il faut faire précéder le champ par le nom du formulaire

ex : @form updateModeleValue voir tout ce qui est cablé sur le formulaire.

D'autres attributs sont disponible pour ajax :

* + **Attribut disable**  
    Dès que l'on commence ajax, on est en ajax donc si l'on souhaite récupérer autre chose, il faut utilisé la balise **disable.**
  + **Attribut event**

celui-ci déclenche une requête ajax dans un événement inputtext onchange

Lorsque l'on teste un élément à null il faut que se soit un objet. Donc pour les int il faut convertir en Integer!

*Créé avec HelpNDoc Personal Edition:* [*Sites web iPhone faciles*](http://www.helpndoc.com/fr/tour-des-fonctionnalites/generation-de-site-web-iphone)

**Qq tags**

|  |  |
| --- | --- |
| <f:metadata> | Tag permettant de gérer des données contextuelles à la vue |
| <f:viewParam name=*"productId"*  value=*"#{productController.productId}"* /> | Ce tag permet de lier une propriété d'un bean sans en faire un champ dans la page.  C'est simplement contextuel à la vue. |
| <f:event type=*"preRenderView"* listener=*"#{productController.prepareEdit}"* />  </f:metadata> | Ce tag permet de déclencher une méthode Java lors d'un "process event" du cycle de vie JSF.  Le preRenderView est l'événement déclenché avant la dernière étape du cycle 'RenderView'. |
| <ui:fragment rendered=*"#{empty productController.productId}"*>  <h2>Créer un nouveau produit :</h2>  </ui:fragment> | Tag permettant d'encapsuler un autre tag sans générer d'HTML supplémentaire. La balise fragment disparaît lors de l'étape RenderView qui génère le code HTML. |
| <ui:fragment rendered=*"#{!empty productController.productId}"*>  <h2>Modifier un produit :</h2>  </ui:fragment> | L'attribut rendered permet de conditionner (true/false) la présence d'un tag dans le code HTML.  Cet attribut permet donc de conditionner l'affichage, mais il faut bien se rappeler qu'il gère  cette présence lors de l'étape RenderView, pas dans la page HTML. Ce n'est donc quelque chose  qu'on ne peut altérer côté client en CSS ou JS, car un élément dont le rendered est à false ne  sera pas du tout dans la page. |
| <b:form> |  |
| <b:inputText label=*"Id"* value=*"#{productController.productId}"*  rendered=*"#{!empty productController.productId}"*  readonly=*"true"* /> | Ce champ n'est visible qu'en édition est n'est pas modifiable (readonly).  Il existe en CSS un sélecteur ':read-only' qui permet de modifier facilement le style. |
| <b:inputText label=*"Intitulé"* value=*"#{productBean.intitule}"*> |  |
| <f:validateRequired />  </b:inputText>  <b:inputText type=*"number"* label=*"Poids"* value=*"#{productBean.poids}"*> | Rend la saisie du champ obligatoire. |
| <f:validateRequired />  <f:validateDoubleRange />  </b:inputText>  <b:inputText label=*"Référence (unique)"* value=*"#{productBean.reference}"*>  <f:validateRequired />  <f:validateLength minimum=*"3"* maximum=*"32"* />  </b:inputText>  <b:commandButton value=*"Valider"* action=*"#{productController.save}"* />  </b:form> | La saisie doit pouvoir être transformée en float Java. |

*Créé avec HelpNDoc Personal Edition:* [*Nouvelles et informations sur les outils de logiciels de création d'aide*](http://www.helpauthoringsoftware.com/fr)

**Web Services**

# Spring security

La sécurité est souvent posée au niveau réseau.

code pin généré par des calculettes, permet de se connecter àun environnement sécurisé machine virtuelle ou ssh.

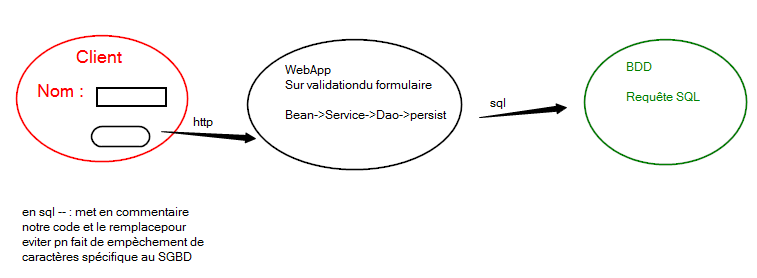
Niveau applicatif rare besoin de sécurité car géré par les parfeu

**Authentification** : savoir reconnaître un user de confiance

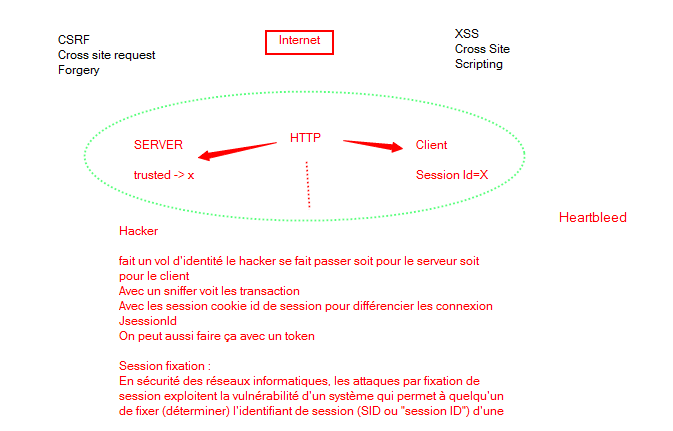
**Authorisation** : trier sur les droits de ce que l'on peut faire

Vérification sur le serveur

## Hackage possible



## Deux attaques Web possible :



Le navigateur n'envoie plus de requête http strict http : HSTS

kick jacking : fenêtre extérieure à l'appli qui renvoie sur une action

CSRF : client envoie une 1er requête le serveur renvoie un header de type CSRF et une clé qui ne peut pas être devinée.

*Créé avec HelpNDoc Personal Edition:* [*Créer des livres électroniques facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Spring security**

Voir la doc sur <https://projects.spring.io/spring-security/>

Spring Security est un framework qui met l'accent sur l'authentification et l'autorisation des applications Java.

Comme tous les projets Spring, le véritable pouvoir de Spring Security réside dans la facilité avec laquelle

il peut être étendu pour répondre aux exigences personnalisées

Caractéristiques

* + Prise en charge complète et extensible de l'authentification et de l'autorisation
  + Protection contre les attaques comme la fixation de session, le clickjacking, la falsification de requêtes sur site, etc.
  + Intégration de l'API Servlet
  + Intégration facultative avec Spring Web MVC
  + Beaucoup plus…

La mise en place de Spring security :

il supporte un 15ne de standard comme SSO avec NTLM (windows) les sessions Windows sont enregistrées : carberos et

OAUTH

XSS

openid

Pour JSF

|  |  |
| --- | --- |
| Spring Security | |
| Client  requete HTTP | Server  Filter->Filter->Filter->Filter->  les filtres choisissent cequ'il faut faire |

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**Config Java**

Dans applicationContext

dans notre class WebSecurityConfig

Class DataSource correspond à la connexion déclarer avec la Jdbc.

Ajouter @Bean

En java on appelle directement une méthode

et en xhtml on utilise ref=""

DataSource de **import** javax.sql.DataSource

Si invalideLocal voir les dependence dans le pom.xml

entityManager n'est plus déclarer par contre il faut une transaction

Dans les

@RestController

*Créé avec HelpNDoc Personal Edition:* [*Éditeur complet de livres électroniques ePub*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Nouveau chapitre**

**package** fr.formation.gestioncolis.config;

**import** javax.sql.DataSource;

**import** org.hibernate.jpa.HibernatePersistenceProvider;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.jdbc.datasource.lookup.JndiDataSourceLookup;

**import** org.springframework.orm.jpa.JpaTransactionManager;

**import** org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@EnableWebSecurity

**public** **class** WebSecurityConfig **extends** WebSecurityConfigurerAdapter {

@Autowired

**public** **void** configureGlobal(**final** AuthenticationManagerBuilder auth) **throws** Exception {

auth.jdbcAuthentication().rolePrefix("ROLE\_").dataSource(**this**.dataSource())

.usersByUsernameQuery(**this**.usersByUsername()).authoritiesByUsernameQuery(**this**.authentitiesByUsername());

}

**private** String authentitiesByUsername() {

**return** "SELECT username, upper(role.name) as authority FROM user, role WHERE user.username = ? AND user.roleId = role.id";

}

**private** String usersByUsername() {

**return** "SELECT USERNAME as username , PASSWORD as password , true FROM user WHERE username = ?";

}

// pas de role dans la BDD mais dans spring oui

// Configuration à la bbd pour Spring

@Bean

**public** DataSource dataSource() {

**final** JndiDataSourceLookup nslookup = **new** JndiDataSourceLookup();

nslookup.setResourceRef(**true**);

**return** nslookup.getDataSource("jdbc/gestioncolis"); // Par defaut

// java:comp/env/

// set resourceRef à

// true

}

@Bean

**public** LocalContainerEntityManagerFactoryBean entityManagerFactory() {

**final** LocalContainerEntityManagerFactoryBean factory = **new** LocalContainerEntityManagerFactoryBean();

// Notre provider est hibernet donc : ...

factory.setPersistenceProvider(**new** HibernatePersistenceProvider());

factory.setPackagesToScan("fr.formation.gestioncolis.entity");

**return** factory;

}

@Bean

**public** JpaTransactionManager transactionManager() {

**final** JpaTransactionManager transaction = **new** JpaTransactionManager();

transaction.setEntityManagerFactory(**this**.entityManagerFactory().getObject());

**return** transaction;

}

}

hocon

*Créé avec HelpNDoc Personal Edition:* [*Outils facile d'utilisation pour créer des aides HTML et des sites web*](http://www.helpndoc.com/fr/outils-creation-aide)

**Test unitaire**

*Créé avec HelpNDoc Personal Edition:* [*Outil de création d'aide complet*](http://www.helpndoc.com/fr/outils-creation-aide)

**Service rest**

## Quelques explications

Comme on a besoin d'une requête vers la BBD il faut ingredientService

Le but de REST est de faire une API

depuis internet envoi/reception depuis un serveur

les annotations de REST s'accorde avec Spring Security

Respect de http standard mais coté présentation

* + - ajouter (POST),
    - modifier (PUT),
    - Lire (GET)
    - Supprimer (DELETE)

ex : dataIngredient on peut faire les 4 actions par leur identifiant

Un service REST est un service WEB qui fait les mm actions que le CRUD

NIVEAU 1 : Bien définir API REST avec les noms logiques

NIVEAU 2 : Standardisé pour être utilisable avec une documentation

NIVEAU 3 : Pas de doc l'api fournit tte la doc (http option)

## Dans DataController :

@Controller

@RequestMapping("/data")

**public class** Datacontroller {

@Autowired

**private** IngredientService ingredientService;

// APPLICATION\_JSON\_UTF8\_VALUE ==> tjs prendre \_value

@GetMapping(value = "/ingredients" , produces = MediaType.***APPLICATION\_JSON\_UTF8\_VALUE***)

**public** List<Ingredient> listIngredients() {

**return** **this**.ingredientService.getAll();

}

}

*Créé avec HelpNDoc Personal Edition:* [*Générateur de documentation d'aide HTML gratuit*](http://www.helpndoc.com/fr)

**Ecriture d'un stream**

**package** fr.formation.gestioncolis.controller;

**import** java.io.Serializable;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** javax.annotation.PostConstruct;

**import** javax.faces.bean.ManagedBean;

**import** javax.faces.bean.ManagedProperty;

**import** javax.faces.view.ViewScoped;

**import** javax.persistence.Transient;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** fr.formation.gestioncolis.bean.CommandeBean;

**import** fr.formation.gestioncolis.bean.LiasseBean;

**import** fr.formation.gestioncolis.dao.CommandeDao;

**import** fr.formation.gestioncolis.dao.LiasseDao;

**import** fr.formation.gestioncolis.entity.Commande;

**import** fr.formation.gestioncolis.entity.Liasse;

**import** fr.formation.gestioncolis.exception.CreateEntityException;

**import** net.bootsfaces.utils.FacesMessages;

@ManagedBean

@ViewScoped

**public** **class** LiasseController **implements** Serializable {

**private** **static** **final** Logger ***LOGGER*** = LoggerFactory.*getLogger*(EtatController.**class**);

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** Integer liasseId;

// Injection de dépendance ;-) Merci Jérémy

@ManagedProperty("#{liasseBean}")

**private** LiasseBean liasseBean;

@ManagedProperty("#{liasseDao}")

**private** LiasseDao liasseDao;

**private** List<Liasse> liasses;

@ManagedProperty("#{commandeDao}")

**private** CommandeDao commandeDao;

@ManagedProperty("#{commadeBean}")

**private** CommandeBean commandeBean;

@Transient

// private Map<String, String> natures;

**private** List<String> natures;

**private** List<Commande> readAllCommande;

**private** List<Integer> readAllIdCommande;

// ------------------------------------------------------------------------------------------------------------

@PostConstruct

**public** **void** \_init() {

LiasseController.***LOGGER***.debug(

"\n[LOG LiasseController] \*\*\*\*\*\*\*\*\*\* Chargement de la liste des liasses et des natures \*\*\*\*\*\*\*\*\*\*");

**this**.liasses = **this**.liasseDao.readAll();

**this**.natures = **new** ArrayList<>();

**this**.natures.add("Cadeaux");

**this**.natures.add("Document");

**this**.natures.add("Echantillon Commercial");

**this**.natures.add("Retour marchandise");

**this**.natures.add("Autres");

**this**.readAllCommande = **this**.commandeDao.readAll();

|  |
| --- |
| **this**.readAllIdCommande = **this**.readAllCommande.stream().map(Commande::getId).collect(Collectors.*toList*()); |

LiasseController.***LOGGER***.debug("\n LiasseController Liste des commandes : " + **this**.readAllCommande);

}

// ------------------------------------------------------------------------------------------------------------

**public** **void** save() {

LiasseController.***LOGGER***.debug("\n\*\*\*\*\*\*\*\*\*\* [INFO-LiasseController] Sauvegarde de liasseBean en BDD. \*\*\*\*\*\*\*\*\*\*");

FacesMessages.*info*("\n\*\*\*\*\*\*\*\*\*\* [INFO-LiasseController] Ecrire la méthode save() servant à sauvegarder la liasse saisie dans vue create de liasse.");

**if** (**this**.liasseBean.getId() == **null**) {

**try** {

**final** Liasse liasse = **new** Liasse();

**this**.liasseDao.create(liasse);

FacesMessages.*info*("Nouvelle liasse sur la commande'" + **this**.liasseBean.getCommandeBean().getId()

+ "' créée avec succès.");

} **catch** (**final** CreateEntityException e) {

LiasseController.***LOGGER***.error(

"\n\*\*\*\*\*\*\*\*\*\* [INFO-LiasseController] Erreur pendant la création d'une nouvelle liasse", e);

}

} **else** {

FacesMessages.*info*(

"\n\*\*\*\*\*\*\*\*\*\* [INFO-LiasseController] Lancement de la création d'une liasse en Base de donnée");

// **TODO**: Modification.

}

}

// ------------------------------------------------------------------------------------------------------------

/\*\*

\*

\* LISTE DES GETTERS SETTERS

\*/

**public** LiasseBean getLiasseBean() {

**return** **this**.liasseBean;

}

**public** Integer getLiasseId() {

**return** **this**.liasseId;

}

**public** **void** setLiasseId(**final** Integer liasseId) {

**this**.liasseId = liasseId;

}

**public** **void** setLiasseBean(**final** LiasseBean liasseBean) {

**this**.liasseBean = liasseBean;

}

**public** LiasseDao getLiasseDao() {

**return** **this**.liasseDao;

}

**public** **void** setLiasseDao(**final** LiasseDao liasseDao) {

**this**.liasseDao = liasseDao;

}

**public** List<Liasse> getLiasses() {

**return** **this**.liasses;

}

**public** **void** setLiasses(**final** List<Liasse> liasses) {

**this**.liasses = liasses;

}

**public** List<String> getNatures() {

**return** **this**.natures;

}

**public** **void** setNatures(**final** List<String> natures) {

**this**.natures = natures;

}

**public** CommandeDao getCommandeDao() {

**return** **this**.commandeDao;

}

**public** **void** setCommandeDao(**final** CommandeDao commandeDao) {

**this**.commandeDao = commandeDao;

}

**public** List<Commande> getReadAllCommande() {

**return** **this**.readAllCommande;

}

**public** **void** setReadAllCommande(**final** List<Commande> readAllCommande) {

**this**.readAllCommande = readAllCommande;

}

**public** List<Integer> getReadAllIdCommande() {

**return** **this**.readAllIdCommande;

}

**public** **void** setReadAllIdCommande(**final** List<Integer> readAllIdCommande) {

**this**.readAllIdCommande = readAllIdCommande;

}

**public** CommandeBean getCommandeBean() {

**return** **this**.commandeBean;

}

**public** **void** setCommandeBean(**final** CommandeBean commandeBean) {

**this**.commandeBean = commandeBean;

}

}

*Créé avec HelpNDoc Personal Edition:* [*Produire des livres EPub gratuitement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Cycles de développement**

Il existe différents types de cycles de développement entrant dans la réalisation d'un logiciel.

Ces cycles prendront en compte toutes les étapes de la conception d'un logiciel.

# Cycle en cascade

® Analyse Préparation

® Conception Blueprint

® Développements / Implémentation

® Validation/ Test

® Evaluation / Publication

Test unitaire

## test-driven development

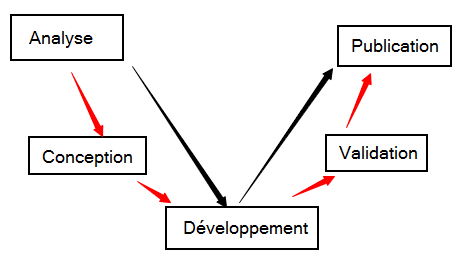
Le test-driven development (TDD) ou en français développement piloté par les tests est une technique de développement de logiciel qui préconise d'écrire les tests unitaires avant d'écrire le code source d'un logiciel.

1. Niveau de scénario de validation des tests
2. Outils de tidating Mentis trac jira Redmine
3. Mantis est un système de suivi d'anomalies logicielles (bugs) basé sur une interface web.
4. Redmine est une application web libre de gestion de projets presque complète en mode web, développée en Ruby sur la base du framework Ruby on Rails. La gestion des tests devra être faite avec un autre outil.
5. Trac est une application web libre de gestion complète de projet par Internet, développée en Python (à ne pas confondre avec un autre logiciel de gestion de projet, Track+)
6. Jira est un système de suivi de bugs, un système de gestion des incidents, et un système de gestion de projets développé par Atlassian.

*Créé avec HelpNDoc Personal Edition:* [*Éditeur de documentation CHM facile*](http://www.helpndoc.com/fr)

**Modèle en V**

# Le modèle en V



L'idée c'est de faire des choses en parallèle.

L'analyse est faite en amont.

Lorsqu'on est dans une méthode agile, on fait plus de chose.

L'origine d'Agile est un manifeste autour du code réutilisable.

# SCRUM

## 3 ROLES

Product owner : personne qui est en charge du dialogue avec le client

Scrum Master : très proche d'un chef de projet, à le rôle de rassembler tous le monde

Team Master : plus fonctionnel que

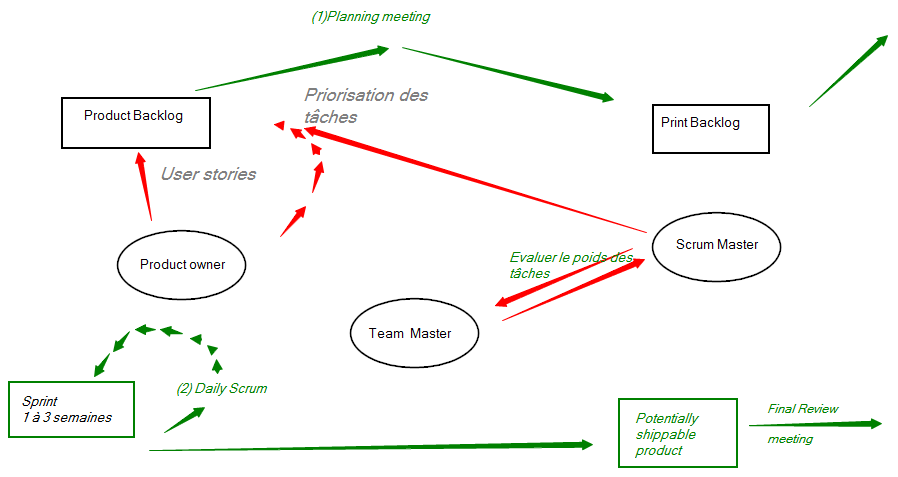
Product Backlog : idée de tableau de bord, user story : tout petit besoin de l'utilisateur

Print Backlog

Le travail en Scrum est représenté par des poids (les taches une quantité de travail et donc de temps) et non du temps.

Le poids définit le temps selon une action : simple / avec du code / nouvelle page / .....

Le poids est représenté de 1 à 8 mais souvent 1 à 5.



*Créé avec HelpNDoc Personal Edition:* [*Générer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**JUnit**

Dans éclipse,il est possible de lancer les tests soit par :

* maven : clean package
* run as

1. Découpage important si le 1er foire les autres "foires"
2. Besoin de préparation de données et libération des ressources

avec setUp et tearDown

Dans notre projet, Tests effectués sur résultat de la suite.

Documentation Junit : <http://junit.org/junit4/>

Junit s'appuie sur **CoreMatchers** , présent dans les imports.

On voit toutes les méthodes qui sont très parlantes.

La class Assert : vérifie la valeur retour du test OK ou NOK.

ex : assertNotNull, assertEquals / assertSame ,

Petite parenthèse :

API de réflexion (java.lang.reflect) permet un objet java quels sont les champs et méthode, méthode de classe

Les objets de proxy encapsule, ne connaît pas l'objet à appeler mais vérifie si l’existence de la méthode.

Pour faire des nouvelles classe sans connaître l'implémentation,

JVM => securityManager peut être actif ou non, renvoie le private ou public des méthodes.

Dp4j est un outil Java qui vous permet de tester des méthodes privées comme si elles n'étaient pas privées, car dp4j analysera votre code de test et génèrera le code Java nécessaire pour accéder aux méthodes privées et faire compiler votre test JUnit.

On peut implémenter des méthodes dans les interfaces.

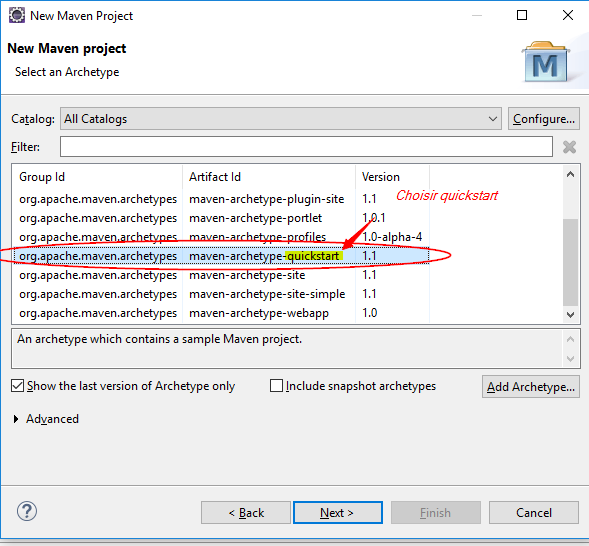
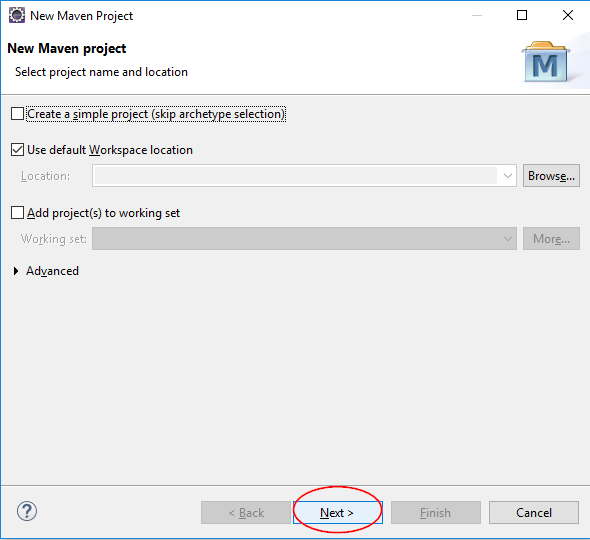
Si le test est lancer avec Maven :

mvn clean package

*Créé avec HelpNDoc Personal Edition:* [*Générer des livres électroniques EPub facilement*](http://www.helpndoc.com/fr/creer-des-livres-numeriques-epub)

**Création d'un projet Mavec via éclipse**

File > new > Maven projet



mvn jar:jar ===> crée un jar dans target

*Créé avec HelpNDoc Personal Edition:* [*Générateur d'aides Web gratuit*](http://www.helpndoc.com/fr)

**Test de Spring MVC**

Spring boot est le modele ....

*Créé avec HelpNDoc Personal Edition:* [*Créer des aides HTML, DOC, PDF et des manuels depuis une même source*](http://www.helpndoc.com/fr/outils-creation-aide)

**Export JAR**

|  |  |  |
| --- | --- | --- |
|  |  |  |

*Créé avec HelpNDoc Personal Edition:* [*Créer des documents d'aide HTML facilement*](http://www.helpndoc.com/fr/tour-des-fonctionnalites)

**Vérifier la qualité du code**

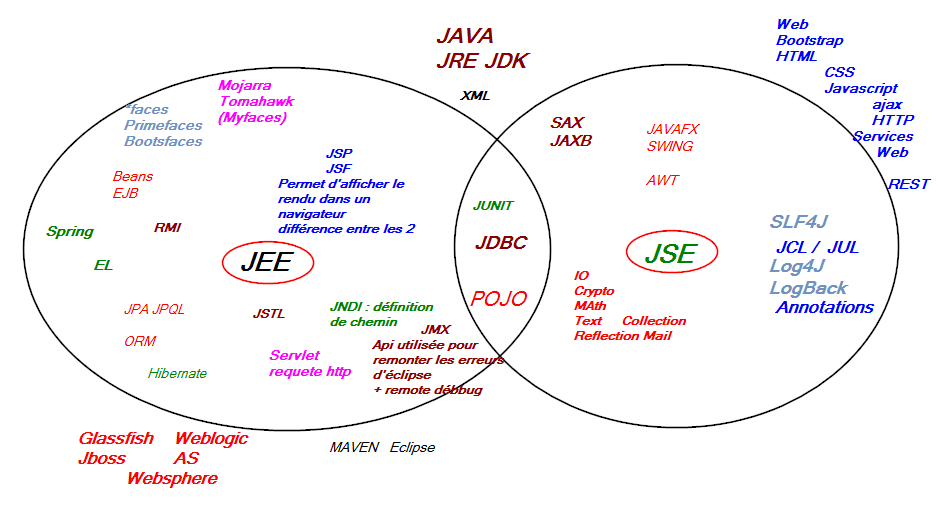
On peut utiliser le logiciel SonarQube.

Il est activé sur localhost:9000

Sous /workspace/ExoEmbauche > mvn sonar:sonar

*Créé avec HelpNDoc Personal Edition:* [*Générateur facile de livres électroniques et documentation*](http://www.helpndoc.com/fr)

**Résumé de la formation!**



*Créé avec HelpNDoc Personal Edition:* [*Avantages d'un outil de création d'aide*](http://www.helpauthoringsoftware.com/fr)