

# JAVA JEE

## Table des matières

---

DocPapier .....	4
Toutes les installations .....	4
NOTIONS GENERALES .....	5
A connaitre / Tutoriel .....	5
Raccourcis .....	5
Syntaxe dans un java .....	6
Syntaxe jstl .....	7
Paramétrage Eclipse .....	8
Config Markers .....	8
Cartographie .....	9
Solutions erreurs .....	9
MAVEN .....	10
Installation .....	10
1) Déclaration des variables d'environnement dans windows .....	10
2) Dans eclipse .....	12
3) Définition du pom.xml .....	12
4) Les liens .....	13
Nouveau chapitre .....	13
SPRING .....	14
Installation .....	14
Obtenir de l'aide .....	14
Fonctionnement du déploiement .....	14
@Annotation .....	15
Projet Cocktail avant BDD .....	15
pom.xml .....	15
web.xml .....	16
applicationContext.xml .....	17
menu.properties .....	17
IngredientController.java .....	18
Menu.java .....	19
IngredientService.java .....	19
IngredientDAO.java .....	20
Ingredient.java .....	20
MainController.java .....	21
Placer correctement les fichiers .....	22
ingredient.jsp .....	23
index.jsp .....	23
pom.properties .....	24
menu.properties .....	24
Structure .....	24
Projet Cocktail avec BDD .....	26
orm.xml .....	28
persistence.xml .....	28
pom.xml .....	29
applicationContext.xml .....	30
IngredientController.java .....	31
IngredientDAO.java .....	32

IngredientService.java .....	33
AddIngredient.jsp .....	33
Ingredient.jsp .....	34
Mysql .....	34
Config de départ .....	35
Config tjs JPA .....	35
Vue/Controller/ .....	36
GIT .....	37
Liste des commandes .....	37
Projet Cocktail de Jeremy .....	38
src/main/java .....	38
CONTROLLER .....	38
CocktailController .....	39
IngredientController .....	40
MainController .....	41
DAO .....	41
CocktailDAO .....	41
IngredientDao .....	42
ENTITY .....	42
Cocktail .....	42
Ingredient .....	42
MODEL .....	42
Menu .....	42
SERVICE .....	42
CocktailService .....	42
IngredientService .....	42
sr/main/resources .....	43
menu.properties .....	43
META-INF .....	43
orm.xml .....	43
persistence.xml .....	43
webapp .....	43
css .....	43
Nouveau chapitre .....	43
views .....	43
WEB-INF .....	43
Outils .....	43
STAN .....	43
GlassFish .....	44
Lancer la console Glassfish .....	44
Install .....	45
Changement port 8080 .....	45
Paramétrage des logs .....	46
Mysql pool .....	47
Dans eclipse .....	48
Schéma Appli .....	50
Remote debugging .....	50



## NOTIONS GENERALES

---

### 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](https://www.tutorialspoint.com/jsp/jsp_overview.htm)

Maven repository..... :

### Raccourcis

ctrl /T --> pour v  rifier les arborescences  
 alt F5 --> pour mettre    jour les d  pendances  
 ctrl 1 --> pour assigner une variable  
     ctrl 2 L --> locale  
     ctrl 2 F --> field

## Syntaxe dans un java

Ajouter les annotations de spring @ :

Instancier un objet ModelAndView en final

Lui ajouter les objets avec addObject("nomUtilis  DansLaJSP", 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;
    }
}
```

## 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>
```

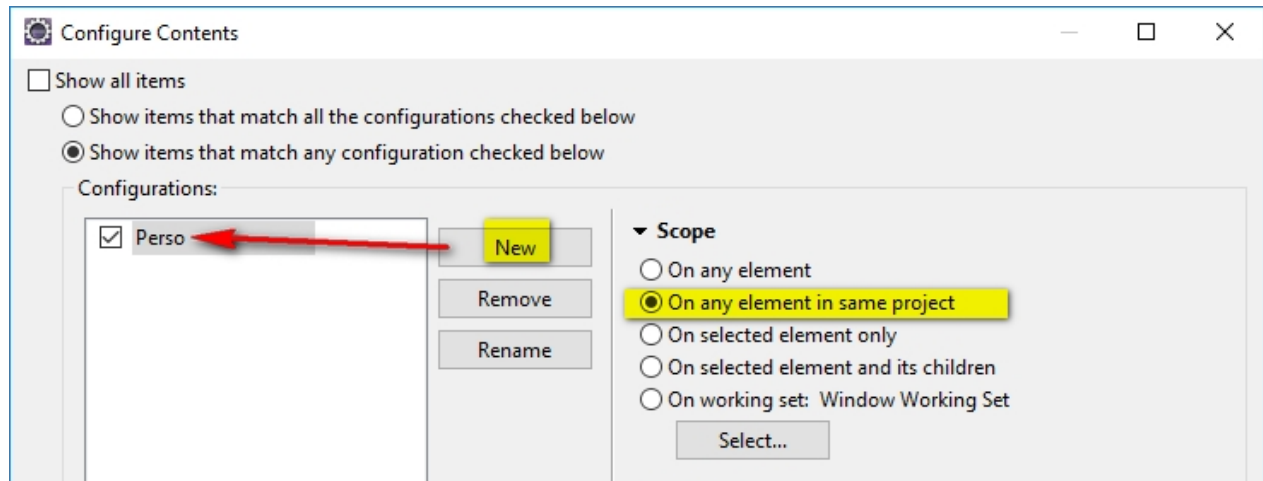
Utilisation de la librairie core :

```
<c:forEach items="" var="">                </c:forEach>
<c:if test="" >                            </c:if>
<c:out value=""></c:out>
```

## Paramétrage Eclipse

## Config Markers

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

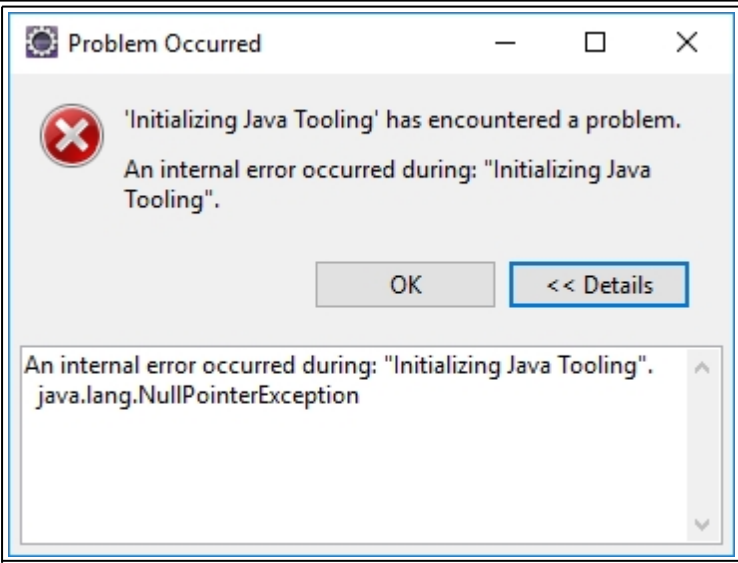




## Cartographie

	<ul style="list-style-type: none"> <li>Proj��tMaven           <ul style="list-style-type: none"> <li>Deployment Descriptor: Proj��tMaven</li> <li>JAX-WS Web Services</li> <li>Java Resources               <ul style="list-style-type: none"> <li>src/main/java</li> <li>src/main/resources                   <ul style="list-style-type: none"> <li>META-INF                       <ul style="list-style-type: none"> <li>orm.xml</li> <li>persistence.xml</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Proj��tMaven           <ul style="list-style-type: none"> <li>Deployment Descriptor: Proj��tMaven</li> <li>JAX-WS Web Services</li> <li>Java Resources</li> <li>JavaScript Resources</li> <li>Deployed Resources               <ul style="list-style-type: none"> <li>webapp                   <ul style="list-style-type: none"> <li>css                       <ul style="list-style-type: none"> <li>paper.css</li> </ul> </li> <li>inc                       <ul style="list-style-type: none"> <li>footer.jsp</li> <li>header.jsp</li> <li>menu.jsp</li> </ul> </li> <li>js                       <ul style="list-style-type: none"> <li>bootstrap.js</li> <li>bootstrap.min.js</li> <li>npm.js</li> </ul> </li> <li>views                       <ul style="list-style-type: none"> <li>index.jsp</li> </ul> </li> <li>WEB-INF                       <ul style="list-style-type: none"> <li>applicationContext.xml</li> <li>web.xml</li> </ul> </li> </ul> </li> </ul> </li> </ul> </li></ul>
--	---	--

## Solutions erreurs

	
<p>org.springframework.beans.factory.NoSuchBeanDefinitionException:</p> <p>No qualifying bean of type</p>	<p>Le probl���me ���tait que je n'avais pas d���clarer mes packages dao et services dans mon fichier servlet-dispatch.xml&gt;</p>

<p>'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)}</p>	<p>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 !</p>

Créé avec HelpNDoc Personal Edition: [Générateur de documentation iPhone gratuit](#)

## 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](#)

## Installation

MAVEN : Outil de gestion de projet

POM : Project Object Model

Créé avec HelpNDoc Personal Edition: [Créer facilement des fichiers Qt Help](#)

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

Après avoir téléchargé Maven

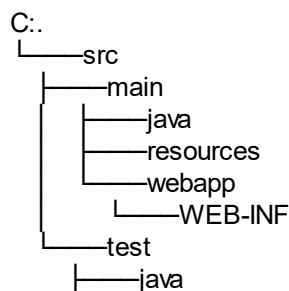
Ajouter JAVA\_HOME lien vers le répertoire Java/jdk1.8.0\_11

Ajouter dans Path le répertoire bin de Maven

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

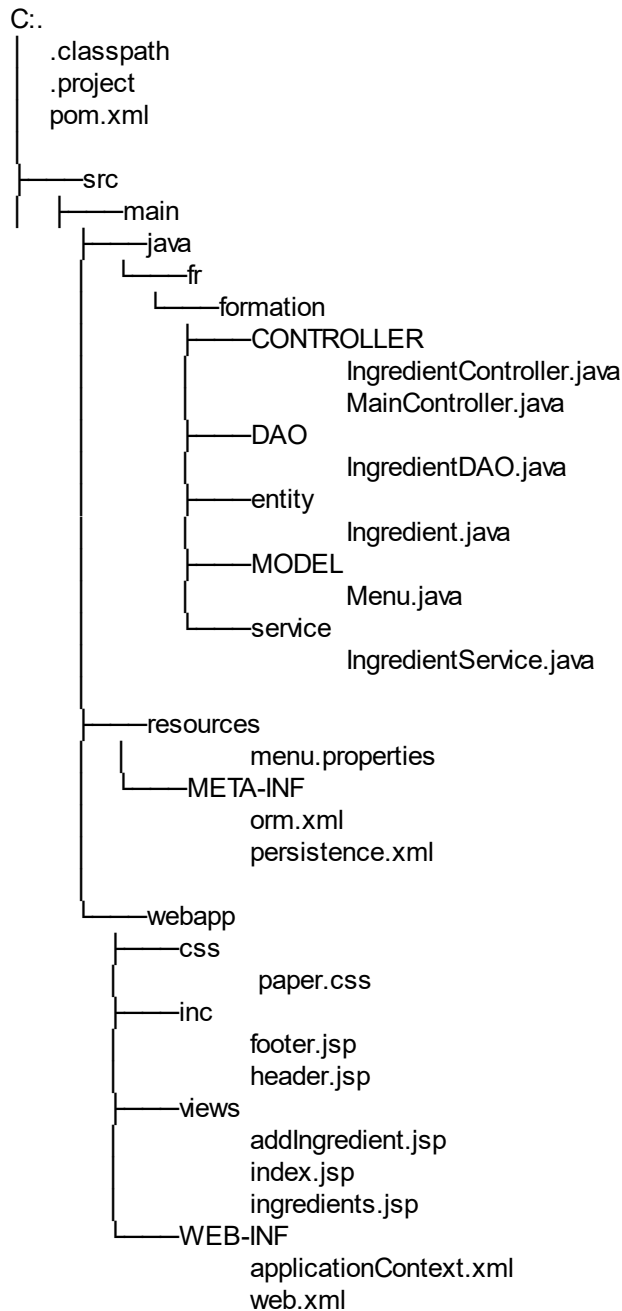
Structure du dossier

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



└─resources

## Par la suite avec la BDD



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
    
```

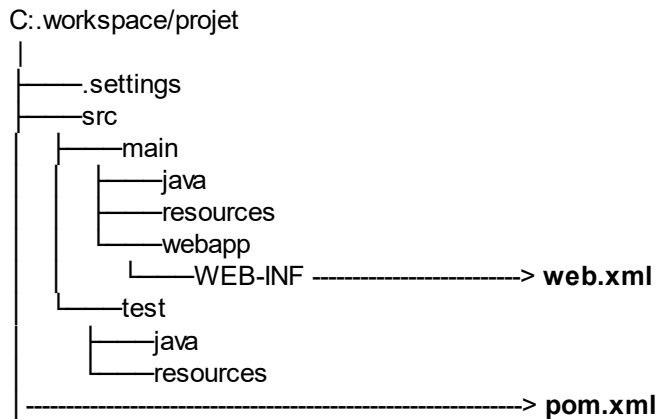
org.eclipse.wst.jsdt.ui.superType.name  
org.eclipse.wst.validation.prefs

Cr   avec HelpNDoc Personal Edition: G  n  rer des livres   lectroniques EPub facilement

## 2) Dans   clipse

Dans   clipse il faut cr  er deux fichiers :

**pom.xml et web.xml**



Cr   avec HelpNDoc Personal Edition:   crire des livres   lectronique Kindle

## 3) D  finition du pom.xml

Aller sous mvn repository pour copier les repository  
Faire alt+F5 pour mettre    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](#)

---

## 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](#)

---

## 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  
 -> Exemples -->Advanced initialisation  
 -> DOM/JQuery events

---

Créé avec HelpNDoc Personal Edition: [Création d'aide CHM, PDF, DOC et HTML d'une même source](#)

---

## SPRING

---



---

Créé avec HelpNDoc Personal Edition: [Avantages d'un outil de création d'aide](#)

---

## Installation

---



---

Créé avec HelpNDoc Personal Edition: [Générateur de documentation complet](#)

---

## Obtenir de l'aide

---



---

Créé avec HelpNDoc Personal Edition: [Générateur de documentation et EPub facile](#)

---

## Fonctionnement du déploiement

Désolé cette prise de note est en relation directe avec ma compréhension sur le moment... vous comprendrez en la lisant....  
 Merci Jeremy 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](#)

---

## @Annotation

### Les annotations

@Autowired : permet de voir qu'on surcharge la méthode  
final classe et methode empêche la surcharge ou extends

Créé avec HelpNDoc Personal Edition: [Créer des documents d'aide PDF facilement](#)

### 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](#)

### 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>

```

## web.xml

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```



```
<?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>
```

17 / 51

```

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

menu.cocktailList.title=Liste des cocktails
menu.cocktailList.url=/cocktails

menu.ingredientList.title=Liste des ingrédients
menu.ingredientList.url=/ingrédients

menu.addCocktail.title=Ajouter un cocktail
menu.addCocktail.url=/cocktail/add

menu.addIngredient.title=Ajouter un ingrédient
menu.addIngredient.url=/ingredient/add

menu.search.title=Recherche
menu.search.url=/search

```

persistant	Business	Presentation
menu.properties -->	<pre> --&gt; MessageSource           List&lt;Menu&gt;           ModelAndView --&gt; </pre>	-->index.jsp

Créé avec HelpNDoc Personal Edition: [Générateur d'aides Web gratuit](#)

## 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("/ingrédients")

public class IngredientController {

    @Autowired
    private IngredientService service;

    @RequestMapping
    public ModelAndView list() {
        final ModelAndView mav = new ModelAndView();
        mav.setViewName("ingrédients");

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

        return mav;
    }
}

```

```
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;
    }

}
```

Cr    avec HelpNDoc Personal Edition: Cr   ation d'aide CHM, PDF, DOC et HTML d'une m   me source

```
@Service
public class IngredientService {

    @Autowired
    private IngredientDAO dao;

    public List<Ingredient> getAll(){
        return this.dao.readAll();
    }
}
```

## 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"));
    }
}
```

## 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;
    }
}

```

---

Cr   avec HelpNDoc Personal Edition: [Nouvelles et informations sur les outils de logiciels de cr  ation d'aide](#)

---

## 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 m  thode
    @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;
    }
}

```

```

    }

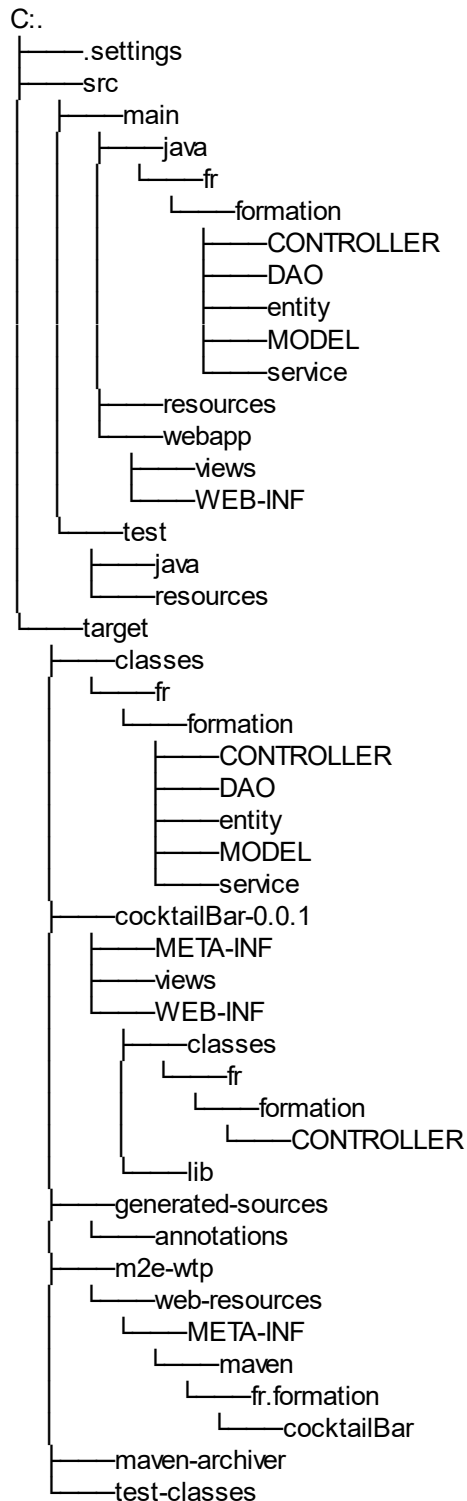
    private String getMessage(final String key) {
        return this.messages.getMessage(key, null,null);
    }

}

```

Cr    avec HelpNDoc Personal Edition: [Cr   er des documents d'aide facilement](#)

## Placer correctement les fichiers



```
<%@ 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>
```

## 23 / 51

```

        <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>

```

---

Cr    avec HelpNDoc Personal Edition: [Produire des aides en ligne pour les applications Qt](#)

---

## pom.properties

---

Cr    avec HelpNDoc Personal Edition: [Qu'est-ce qu'un outil de cr   ation d'aide ?](#)

---

## 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

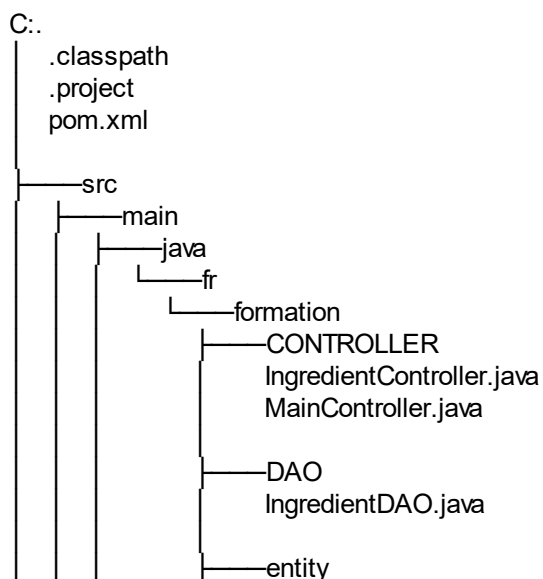
menu.search.title=Recherche  
menu.search.url=/search

---

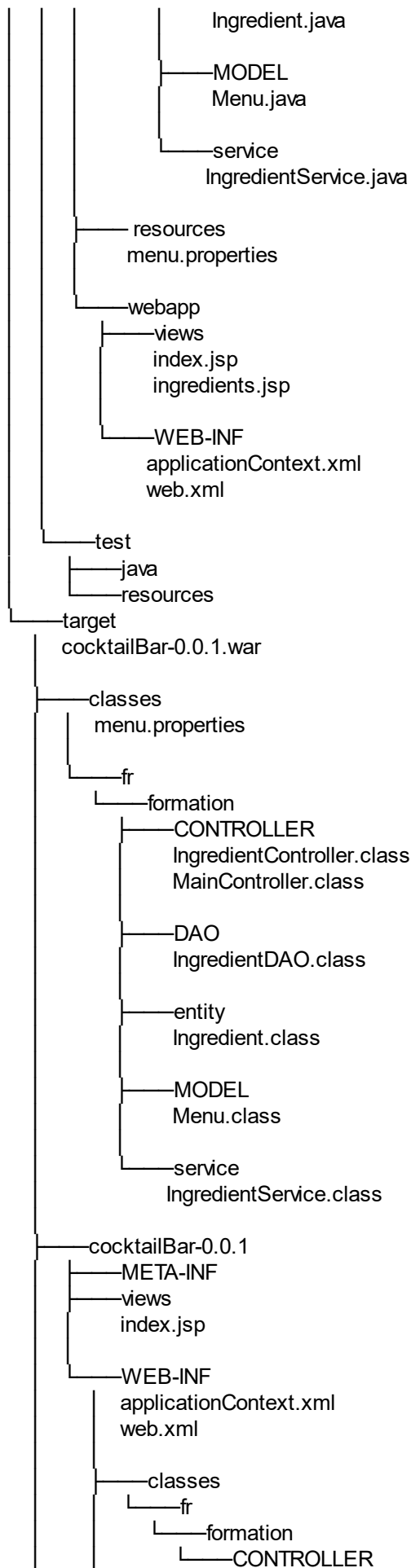
Cr    avec HelpNDoc Personal Edition: [G    rateur gratuit de livres   lectroniques et documentation](#)

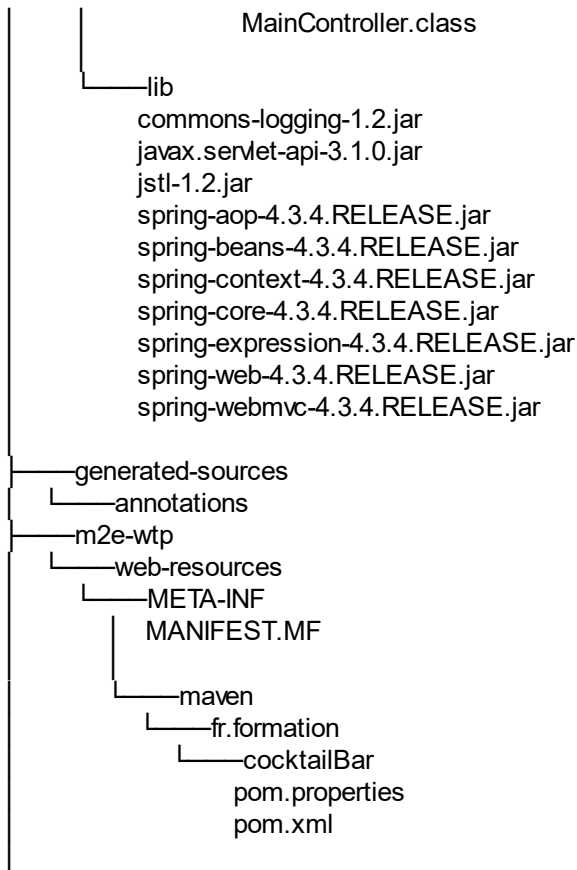
---

## Structure









Cr    avec HelpNDoc Personal Edition:   diteur de documentation CHM facile

## Projet Cocktail avec BDD

orm.xml	<p>D��claration de la structure des tables mysql</p> <pre> &lt;entity class="fr.formation.entity.Ingredient"&gt; &lt;table name="ingredient"&gt;&lt;/table&gt;   &lt;attributes&gt;     &lt;id name="id"&gt;       &lt;column name="id_ing" /&gt;.....-&gt; nom du champ       &lt;generated-value strategy="IDENTITY" /&gt;...-&gt; indique que   c'est une cl��     &lt;/id&gt;      &lt;basic name="etat"&gt;.....-&gt; nom dans Java       &lt;column name="state" /&gt;.....-&gt; nom dans mysql     &lt;/basic&gt;   &lt;/attributes&gt; &lt;/entity&gt;  &lt;entity class="chemin complet jusqu'�� la classe java"&gt;   &lt;table name="nom_table"&gt;&lt;/table&gt;    &lt;basic name="idJava"&gt;     &lt;column name="id_mysql" /&gt;   &lt;/basic&gt;    &lt;/attributes&gt; &lt;/entity&gt; </pre>
---------	---

persistence.xml	Définit le "DriverManager"
pom.xml	<p>Ajout des dépendences utiles pour la base :</p> <ol style="list-style-type: none"> <li>1) "mysql connector"</li> <li>2) "spring data jpa"</li> <li>3) "hibernate core" dernière version</li> <li>4) Copie "hibernate core" changer core en hibernate-entitymanager</li> </ol>
applicationContext.xml	<p>Spring attend entityManagerEntity Name pris dans persistence.xml --&gt;</p> <ol style="list-style-type: none"> <li>1. Ajout de 2 beans : <ul style="list-style-type: none"> <li>○ entityManagerFactory</li> <li>○ transactionManager</li> </ul> </li> <li>2. Ajout <code>&lt;jpa:repositories&gt;</code> <pre> base-package="fr.formation.DAO"  &lt;bean id="entityManagerFactory"  class ="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"&gt;     &lt;property name="persistenceUnitName" value="bar"&gt;&lt;/property&gt;  &lt;/bean&gt;  &lt;bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager"&gt;     &lt;property name="entityManagerFactory" ref="entityManagerFactory" /&gt; &lt;/bean&gt; </pre> </li> </ol>
IngredientController.java	Ajout de ModelAndView avec un appel à la jsp qui fait action
	<pre> 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&lt;Ingredient&gt; getAll(){         return this.dao.findAll();     }      @Transactional     public Ingredient create(final Ingredient ingredient){         return this.dao.save(ingredient);     }  } </pre>

Cr   avec HelpNDoc Personal Edition:   diteur de documentation CHM facile

## 28 / 51

&lt;/persistence&gt;

Cr    avec HelpNDoc Personal Edition: [Outil de cr   tion d'aide complet](#)**pom.xml**

```

<!-- _____ -->
<!-- Aller sous mvn repository pour copier les repository -->
<!-- Faire alt+F5 pour mettre   jour les libraries dans le projet -->
<!-- Pos     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>

```

&lt;!-- D   pendance   ajouter pour les acc   s   la base --&gt;

&lt;!-- 1) "mysql connector" Ajout BDD --&gt;

```

<!-- 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" -->

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>

```

```
<?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 attend 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

---

## 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";
}
}

```

---

Cr    avec HelpNDoc Personal Edition: [Cr   er des livres   lectroniques facilement](#)

---

## 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>{

}

```

---

Cr    avec HelpNDoc Personal Edition: [Produire des livres   lectroniques facilement](#)

---



## 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);
    }
}
```

---

Cr    avec HelpNDoc Personal Edition:   diteur complet de livres   lectroniques ePub

---

## AddIngredient.jsp

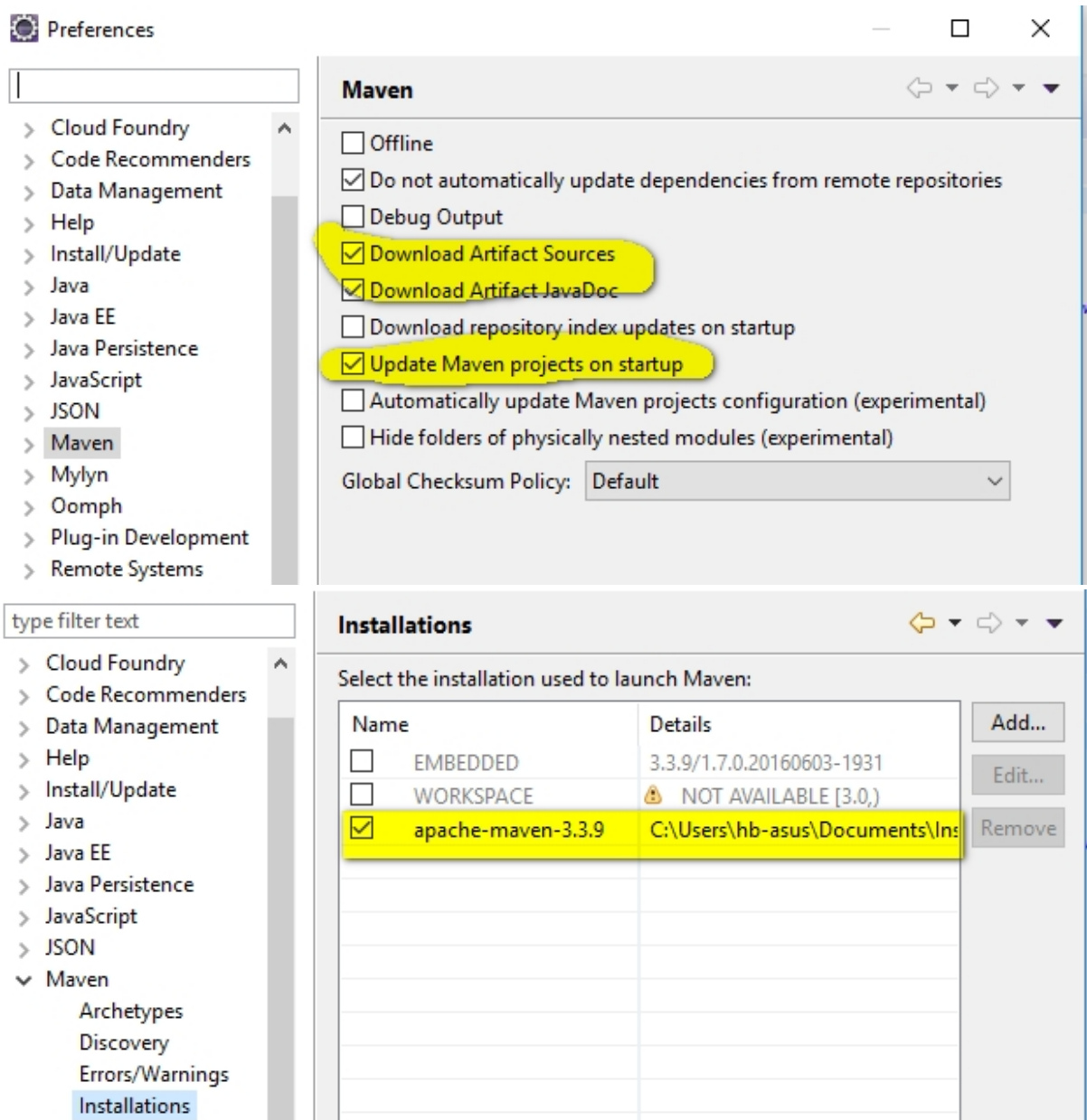
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!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 ingr  dient</title>
</head>
<body>

<h1>AJOUT D'UN INGR  DIENT</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>
```





Pour chaque entité il faut un **id**.

Créé avec HelpNDoc Personal Edition: [Nouvelles et informations sur les outils de logiciels de création d'aide](#)

## Config de départ

Créé avec HelpNDoc Personal Edition: [Créer des sites web d'aide facilement](#)

## Config tjs JPA

JPA	JAVA	SQL
PersistenceUnit "Cocktail" (*)	IngredientJava  --> new Ingredient("",0)	Table ingredient

		Ligne dans la table
PersistenceContext (**) --> contient les informations Unit	<b>EntityManagerFactory</b> : on lui configure le nom de notre entité = "Cocktail" (*) (**)  en JPA on a un persistence objet  <b>EntityManager</b> : 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  <b>TransactionManager</b>	
	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](#)

## Vue/Controller/

VUE	CONTROLLER		
ingr edient s.jsp< ----- -----	IngredientController RequestMapping ModelAndView view name model List<Ingredient> <-----	<b>@Transactional</b> IngredientService DAO<-----	IngredientDAO JPAREpository Spring  EntityMan ager BDD Requê tes 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">

```

GIT	REMOTES
Gestionnaire de source --> repository distant --> Histoire de toutes les modifications --> Commit --> nouvelle révision / Version	Github --> Origin  Branches <ul style="list-style-type: none"> <li>○ MASTER</li> <li>○ tags</li> </ul>
	<div style="text-align: center;"><b>Local (clone)</b></div>
	PC --> Staging --> Working Directory

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

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

## 2. 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**

## 3. Autres commandes

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

Pour voir la différence :

- **git diff**

---

Créé avec HelpNDoc Personal Edition: [Créer des documents d'aide CHM facilement](#)

---

## 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](#)

---

## src/main/java

---

Créé avec HelpNDoc Personal Edition: [Créer des documents d'aide PDF facilement](#)

---

## CONTROLLER

## 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());
    }
}
```

}

---

Cr    avec HelpNDoc Personal Edition: [Produire des aides en ligne pour les applications Qt](#)

---

## 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";
    }
}
```



## }

Cr   avec HelpNDoc Personal Edition: G  n  rateur complet de livres   lectroniques Kindle

## 41 / 51

```
import fr.formation.entity.Cocktail;

public interface CocktailDao extends JpaRepository<Cocktail, Integer> {

}
```

---

Cr    avec HelpNDoc Personal Edition: [Outils facile d'utilisation pour cr   er des aides HTML et des sites web](#)

---

## IngredientDao

```
package fr.formation.dao;

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

import fr.formation.entity.Ingredient;

public interface IngredientDao extends JpaRepository<Ingredient, Integer> {

}
```

---

Cr    avec HelpNDoc Personal Edition: [Sites web iPhone faciles](#)

---

## ENTITY

---

Cr    avec HelpNDoc Personal Edition: [G    rateur facile de livres   lectroniques et documentation](#)

---

## Cocktail

---

Cr    avec HelpNDoc Personal Edition: [Environnement de cr   ation d'aide complet](#)

---

## Ingredient

---

Cr    avec HelpNDoc Personal Edition: [Cr   er des livres   lectroniques facilement](#)

---

## MODEL

---

Cr    avec HelpNDoc Personal Edition: [Avantages d'un outil de cr   ation d'aide](#)

---

## Menu

---

Cr    avec HelpNDoc Personal Edition: [Cr   er des livres   lectroniques EPub facilement](#)

---

## SERVICE

---

Cr    avec HelpNDoc Personal Edition: [G    rateur de documentation Qt Help gratuit](#)

---

## CocktailService

---

Cr    avec HelpNDoc Personal Edition: [G    rateur complet de livres   lectroniques Kindle](#)

---

## IngredientService

**sr/main/resources**

---

Créé avec HelpNDoc Personal Edition: Générateur d'aide complet

## menu.properties

---

Cr    avec HelpNDoc Personal Edition: Produire des aides en ligne pour les applications Qt

**META-INF**

Cr    avec HelpNDoc Personal Edition: Cr   er des fichiers d'aide pour la plateforme Qt Help

orm.xml

Créé avec HelpNDoc Personal Edition: Créer des fichiers d'aide pour la plateforme Qt Help

persistence.xml

---

Créé avec HelpNDoc Personal Edition: Éditeur de documentation Qt Help facile

## webapp

---

Cr    avec HelpNDoc Personal Edition: [Avantages d'un outil de cr   ation d'aide](#)

CSS

---

Cr    avec HelpNDoc Personal Edition: [Cr   er des livres   lectroniques facilement](#)

## Nouveau chapitre

Créé avec HelpNDoc Personal Edition: Générateur complet de livres électroniques ePub

views

Cr    avec HelpNDoc Personal Edition: [Produire facilement des livres   lectroniques Kindle](#)

WEB-INF

Cr   avec HelpNDoc Personal Edition: G  n  rateur d'aides Web gratuit

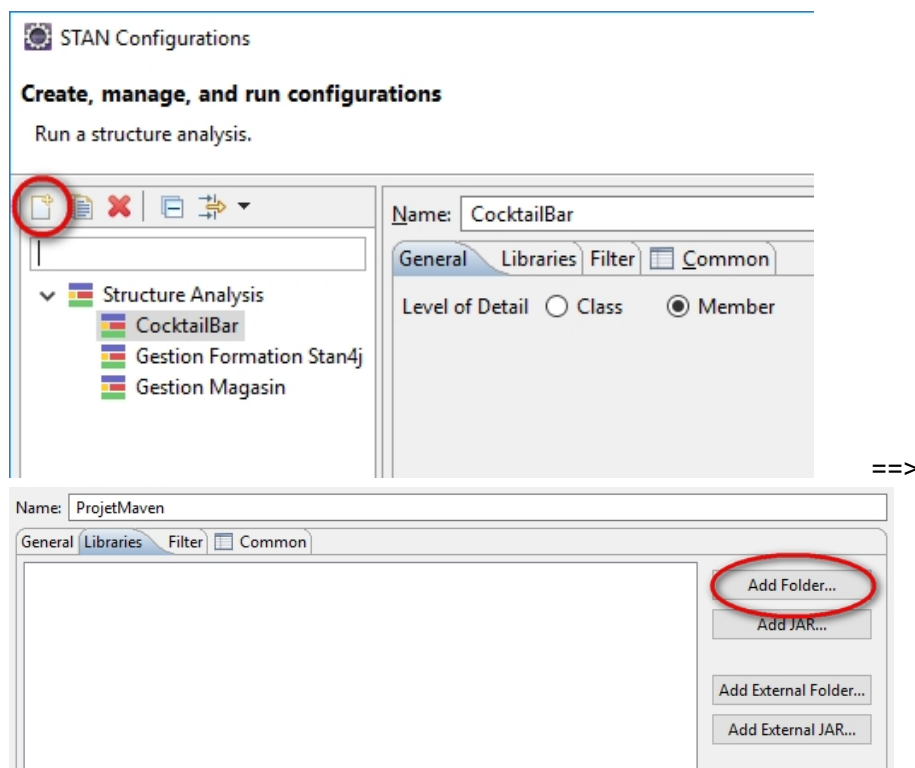
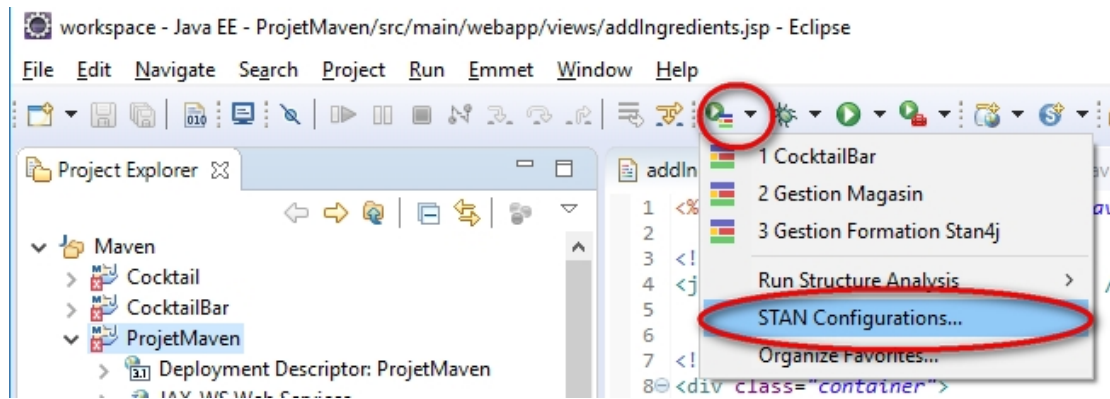
## Outils

---

Cr   avec HelpNDoc Personal Edition:  crire des livres  lectronique Kindle

**STAN**

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



## GlassFish

### Lancer la console Glassfish

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 ?](#)

## Install

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

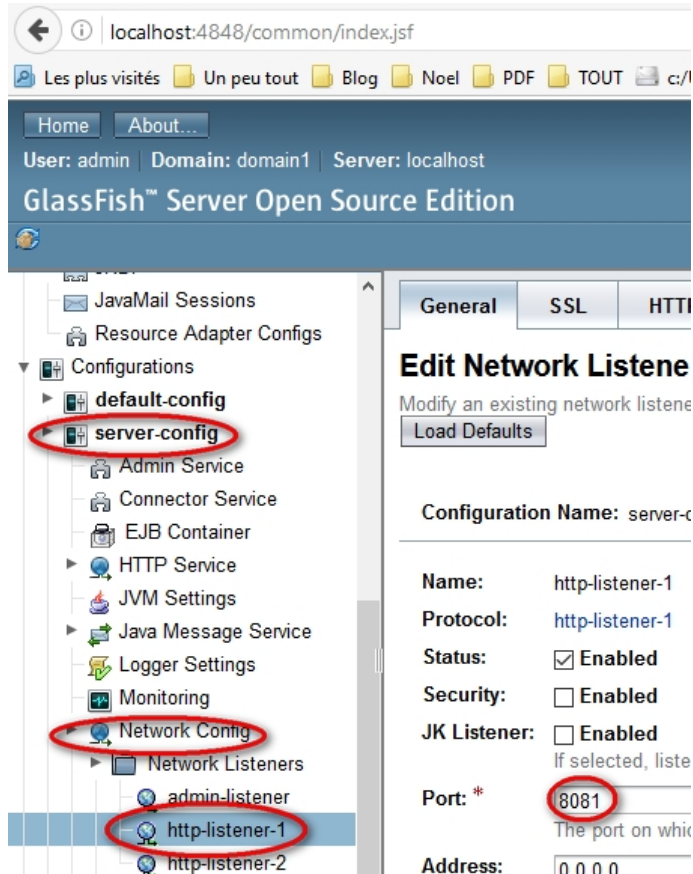
Sous unix : **`jar xvf glassfish-4.1.zip`**

Lire le fichier readme.txt :

Créé avec HelpNDoc Personal Edition: [Créer des documents d'aide facilement](#)

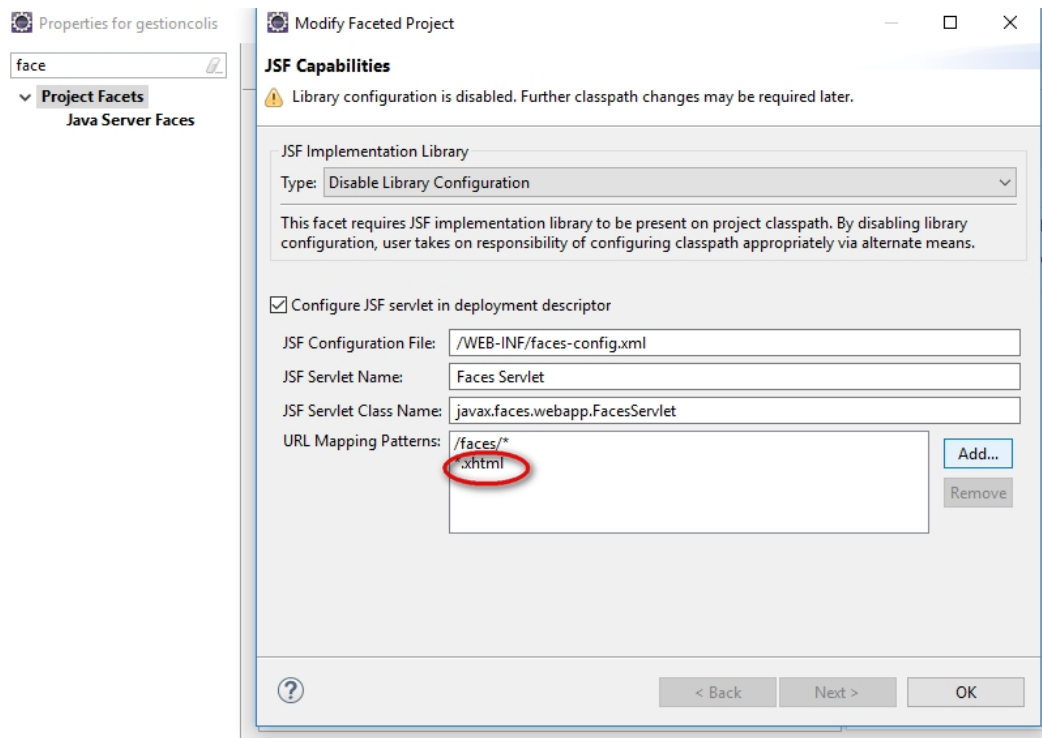
## Changement port 8080

Aller sous la console Glassfish  
changer



ajouter la variable d'environnement `GLASSFISH_AUTODEPLOY`





Ajout

de

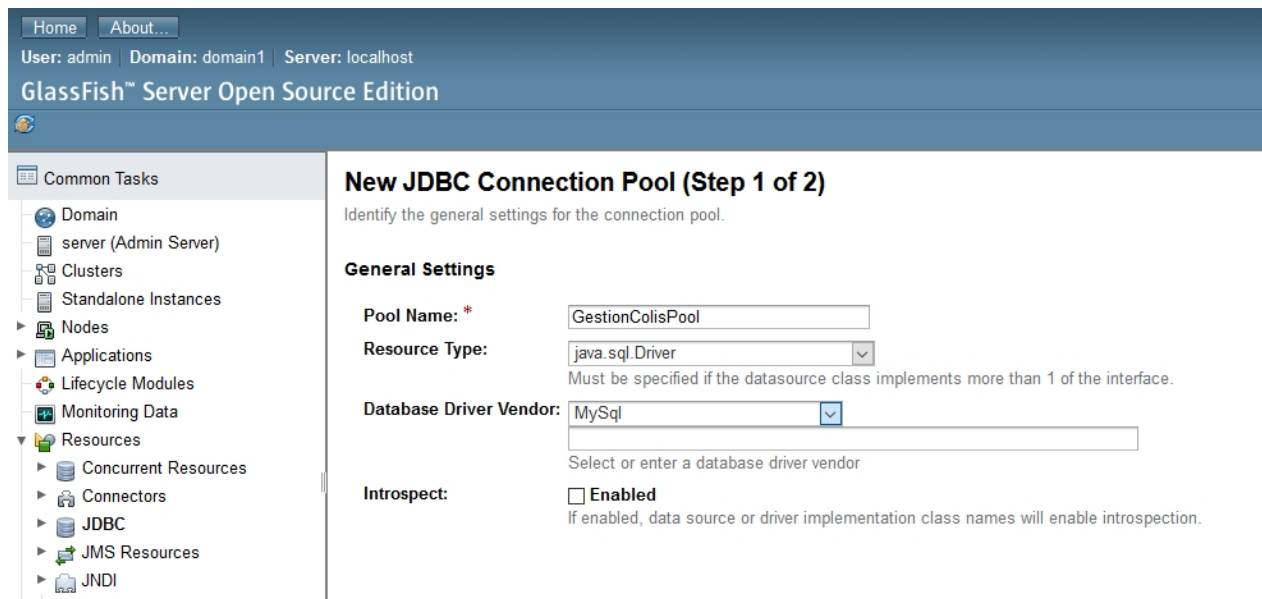
Cr    avec HelpNDoc Personal Edition: [Cr          CHM, PDF, DOC et HTML d'une m         ](#)

## Mysql pool

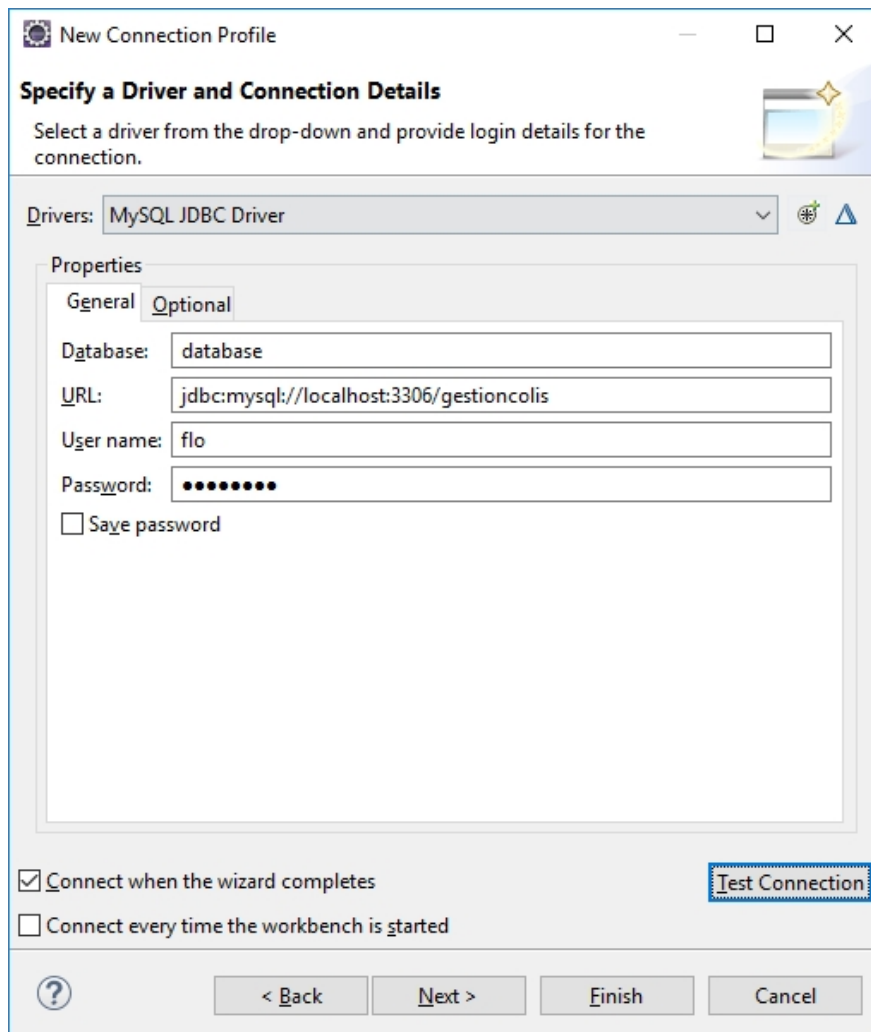
Copier le jar de mysql dans :

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

Ce PC > Documents > Install > glassfish4 > glassfish > domains > domain1 > lib				
	Nom	Modifi�� le	Type	Taille
	applibs	21/08/2014 07:30	Dossier de fichiers	
	classes	21/08/2014 07:30	Dossier de fichiers	
	databases	21/08/2014 07:30	Dossier de fichiers	
	ext	21/08/2014 07:30	Dossier de fichiers	
	mysql-connector-java-5.1.40-bin.jar	24/09/2016 21:35	Executable Jar File	968 Ko



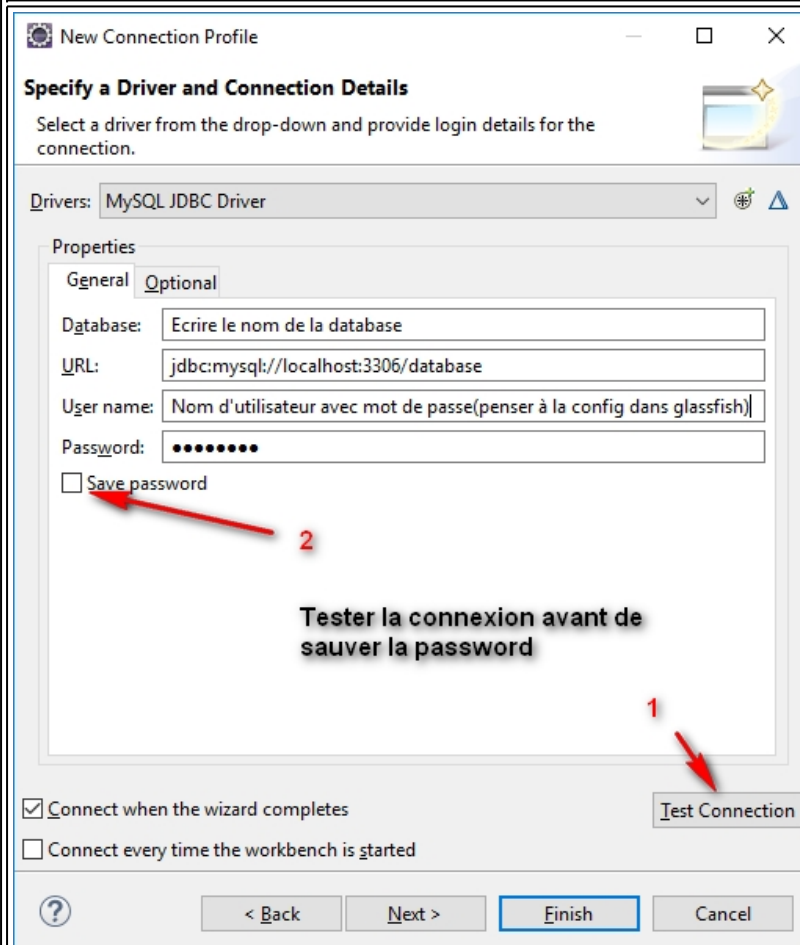
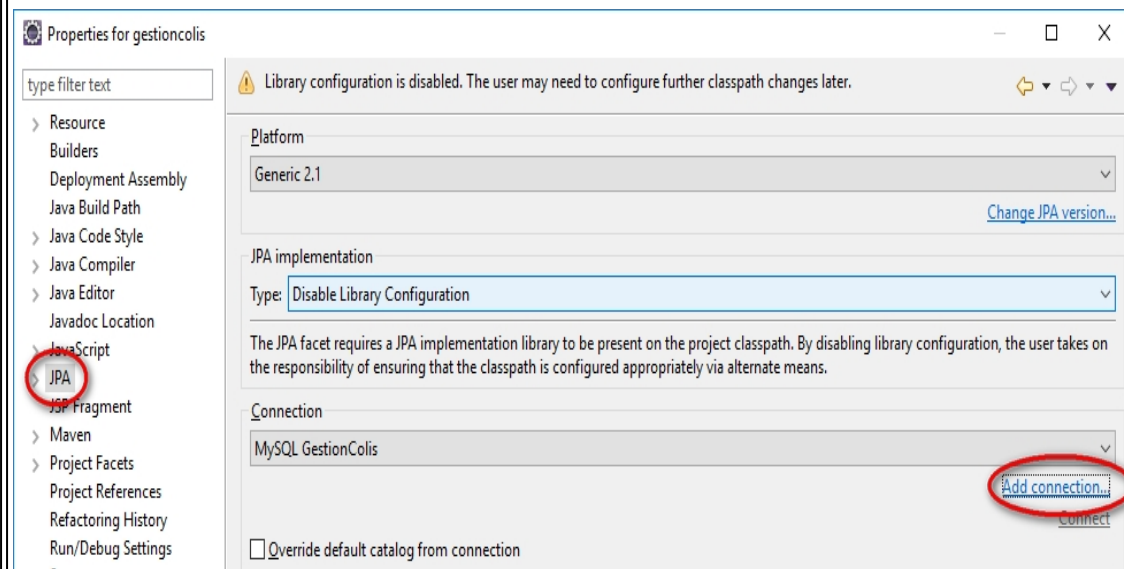
Créer un user dans la base de données avec un mot de passe.





## Création de la connexion à la base de données

**Aller dans les propriétés du projet :**



drivers de la

Penser à sélectionner le

## Schéma Appli

<b>APPLICATION</b>			<b>Serveur</b>	<b>BDD</b>
			<b>GlassFish</b>	<b>Mysql</b>
<b>Présentation</b>	<b>Métier</b>	<b>Persistence</b>		
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

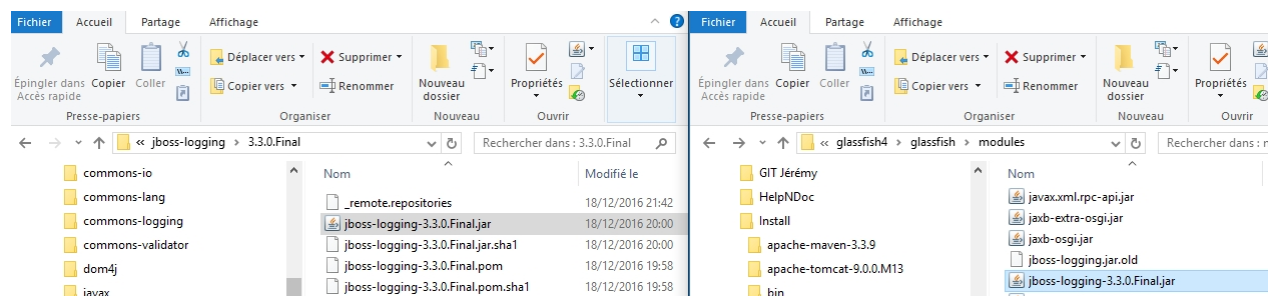
## MAJ dans GlassFish

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

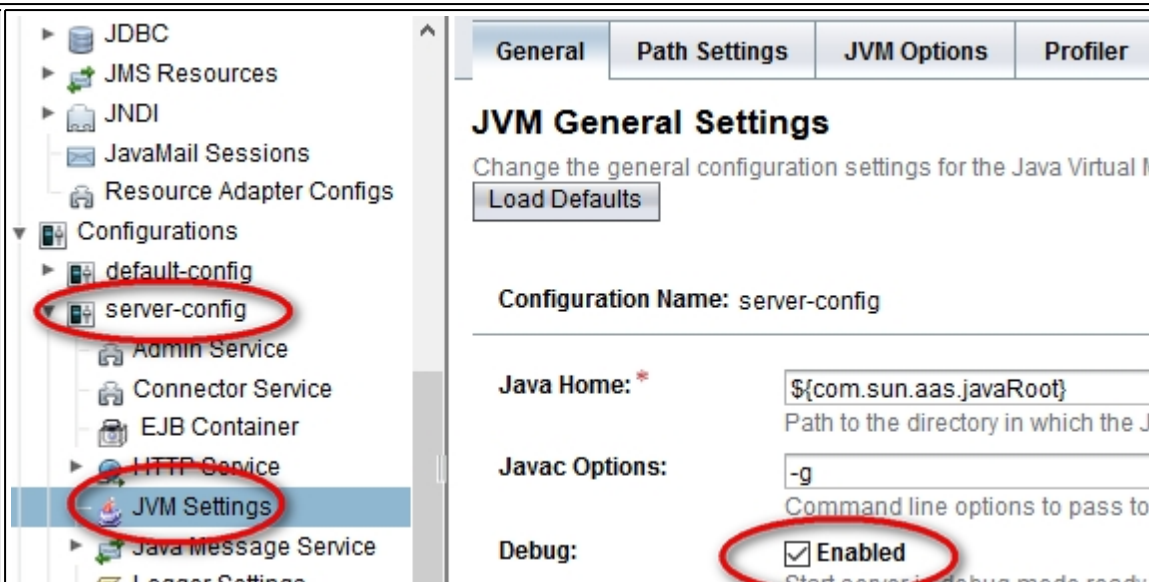
Dans répertoire de GlassFish :

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](#)

## Remote debugging

Sous Glass fish	
Sous eclipse	Debug config remote Java Appli new nom liar appli localhost port 9009