

JAVA JEE

Table des matières

DocPapier	5
Toutes les installations	5
Marques pages utilisés durant la formation	6
NOTIONS GENERALES	6
A connaître / Tutoriel	6
Raccourcis	6
Syntaxe dans un java	7
Syntaxe jstl	8
Paramétrage Eclipse	9
Config Markers	9
Cartographie	10
Solutions erreurs	10
MAVEN	11
Installation	11
1) Déclaration des variables d'environnement dans windows	11
2) Dans éclipse	13
3) Définition du pom.xml	13
4) Les liens	14
Nouveau chapitre	14
SPRING	15
Installation	15
Obtenir de l'aide	15
Fonctionnement du déploiement	15
@Annotation	16
Projet Cocktail avant BDD	16
pom.xml	16
web.xml	17
applicationContext.xml	18
menu.properties	18
IngredientController.java	19
Menu.java	20
IngredientService.java	20
IngredientDAO.java	21
Ingredient.java	21
MainController.java	22
Placer correctement les fichiers	23
ingredient.jsp	24
index.jsp	24
pom.properties	25
menu.properties	25
Structure	25
Projet Cocktail avec BDD	27
orm.xml	29
persistence.xml	29
pom.xml	30
applicationContext.xml	31
IngredientController.java	32

IngredientDAO.java	33
IngredientService.java	34
AddIngredient.jsp	34
Ingredient.jsp	35
Mysql	35
Config de départ	36
Config tjs JPA	36
Vue/Controller/	37
GIT	38
Liste des commandes	38
Création de mon Git pour la doc	39
Projet Cocktail de Jeremy	40
src/main/java	40
CONTROLLER	40
CocktailController	40
IngredientController	41
MainController	42
DAO	43
CocktailDAO	43
IngredientDao	43
ENTITY	43
Cocktail	44
Ingredient	44
MODEL	44
Menu	44
SERVICE	44
CocktailService	44
IngredientService	44
sr/main/resources	44
menu.properties	44
META-INF	44
orm.xml	44
persistence.xml	44
webapp	44
css	45
Nouveau chapitre	45
views	45
WEB-INF	45
Outils	45
STAN	45
Organigramme choix Objet Collection	46
GlassFish	47
Install	47
Lancer la console Glassfish	48
Changement port 8080	48
Paramétrage des logs	49
Mysql pool	50
Dans eclipse	51
Schéma Appli	53
Remote debugging	54

Configuration Glassfish et éclipse	54
Déclaration des Logs en java	55
Classe EntityManager et UserTransaction	55
Schéma de vie d'une exception	56
Schéma des ERROR / EXCEPTION	56
Liste des tutos donnés par JérémY	56
JFS	57
Annotations	57
Annotations BEAN	58
Annotations CONTROLLER	59
Annotations DAO	59
Annotations ENTITY	60
@ManageBean	60
Ajax	60
Qq tags	60
Web Services	62
Spring security	63
Config Java	64
Nouveau chapitre	64
Test unitaire	66
Service rest	66
Ecriture d'un stream	67
Cycles de développement	70

documentation pdf: <C:\Users\hb-asus\Documents\HelpNDoc\Projects\JAVA JEE.pdf>

Toutes les installations

5 / 71

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

Marques pages utilis   s durant la formation

<C:\Users\hb-asus\Documents\HelpNDoc\Projects\Java JEE\bookmarks.html>

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

NOTIONS GENERALES

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

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..... :

Cr     avec HelpNDoc Personal Edition: [G     rateur de documentation d'aide HTML gratuit](#)

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>
```

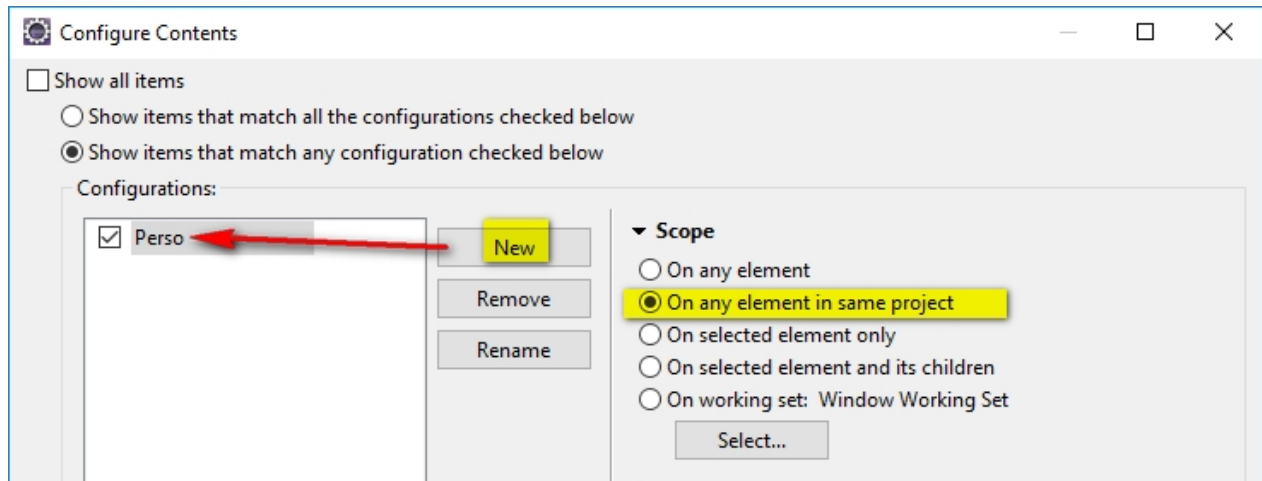
Utilisation de la librairie core :

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


Paramétrage Eclipse

Config Markers

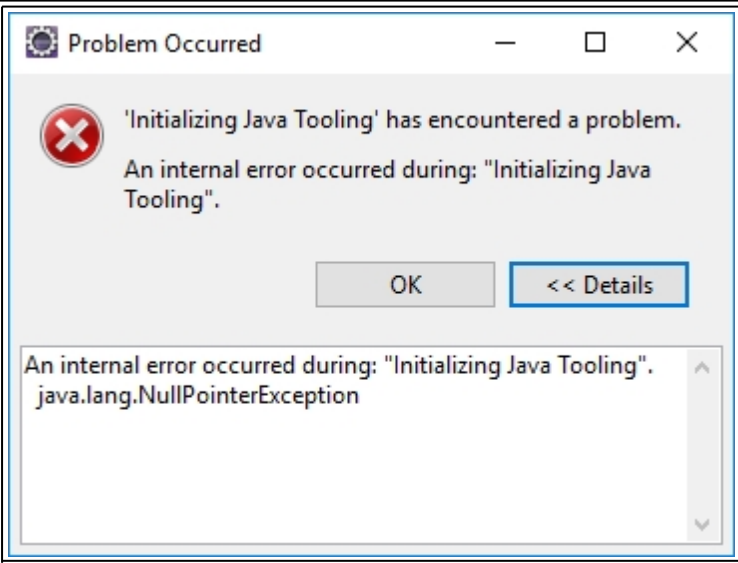
Pour ne voir dans la console **Markers** que les erreurs du projet en cour : (à partir de la petite flèche)



Cartographie

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

Solutions erreurs

 <p>The dialog box shows an error during 'Initializing Java Tooling'. The message is: 'Initializing Java Tooling' has encountered a problem. An internal error occurred during: "Initializing Java Tooling". The details section shows: An internal error occurred during: "Initializing Java Tooling". java.lang.NullPointerException</p>	
<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></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

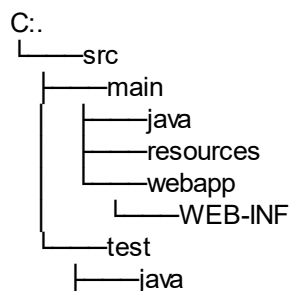
Ajouter JAVA_HOME lien vers le répertoire Java/jdk1.8.0_11

Ajouter dans Path le répertoire bin de Maven

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

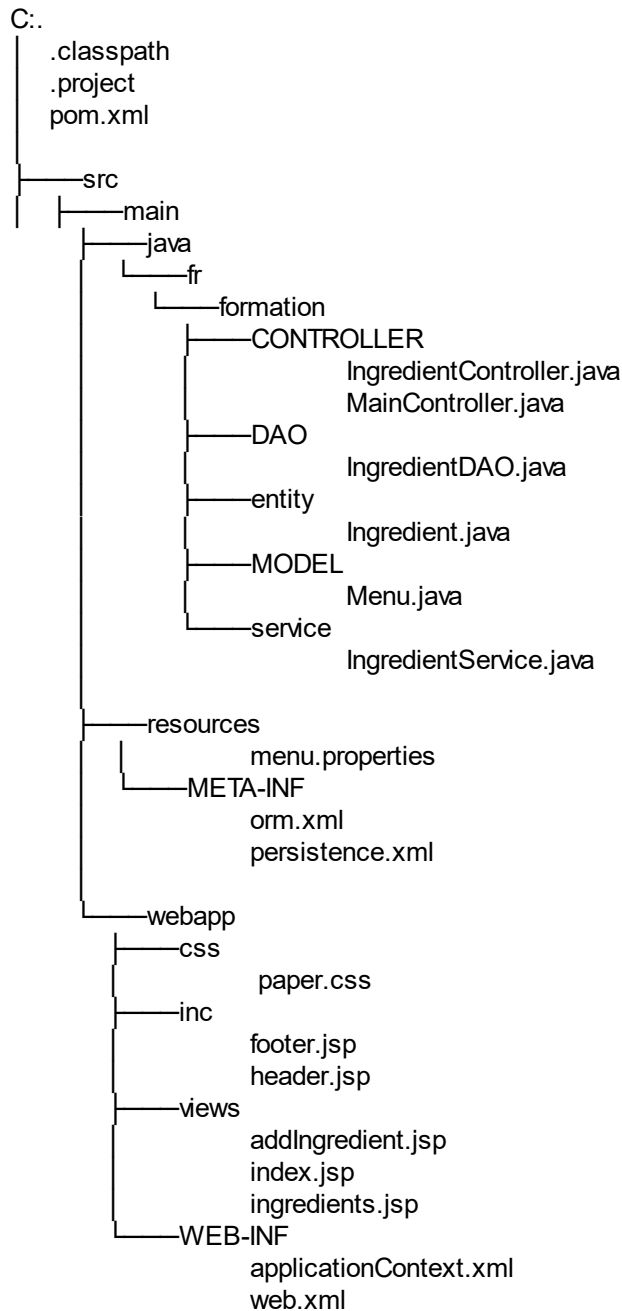
Structure du dossier

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
    
```

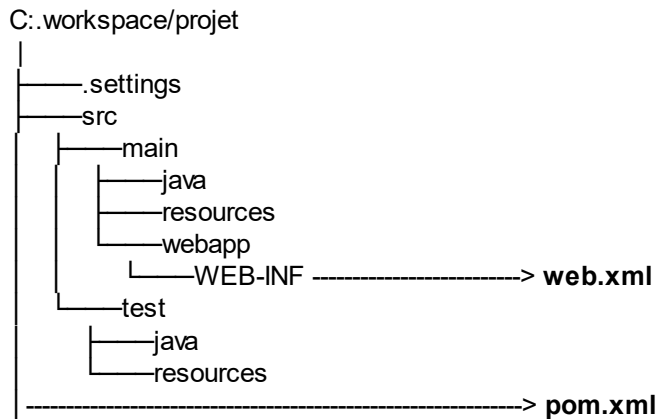
org.eclipse.wst.jsdt.ui.superType.name
org.eclipse.wst.validation.prefs

Créé avec HelpNDoc Personal Edition: [Générer des livres électroniques EPub facilement](#)

2) Dans eclipse

Dans eclipse il faut créer deux fichiers :

pom.xml et web.xml



Créé avec HelpNDoc Personal Edition: [Écrire des livres électronique Kindle](#)

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](#)

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) **properties 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>
```

18 / 71

```

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=/ingrédient/add

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

```

persistant	Business	Presentation
menu.properties -->	<pre> --> MessageSource List<Menu> ModelAndView --> </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  e 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;
    }
}

```

```

    }

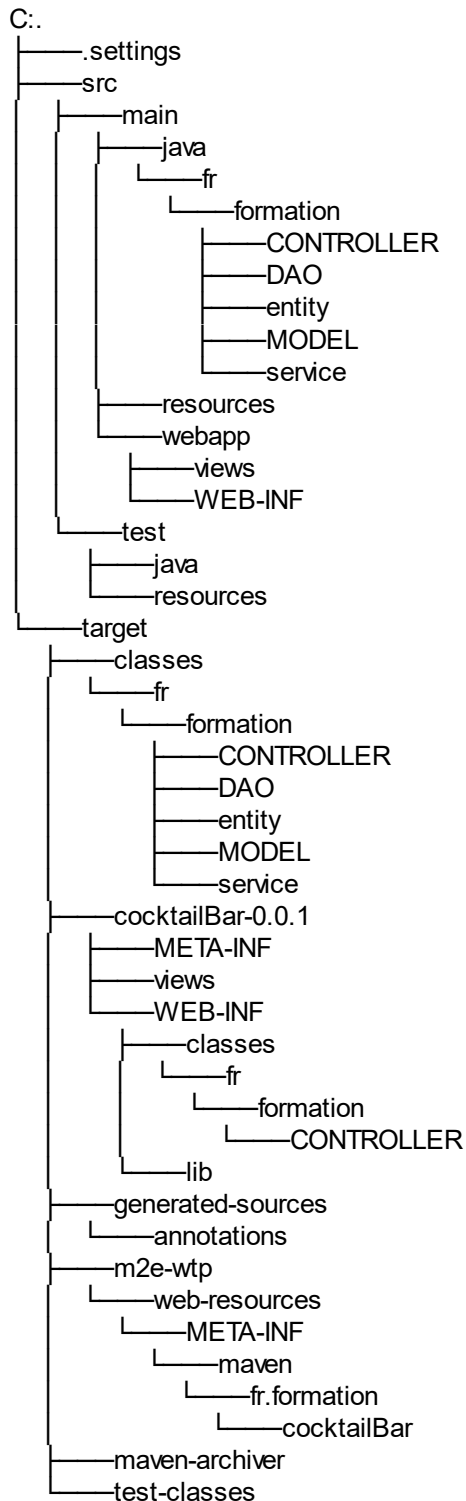
    private String getMessage(final String key) {
        return this.messages.getMessage(key, null,null);
    }

}

```

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>
```

24 / 71


```

        <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

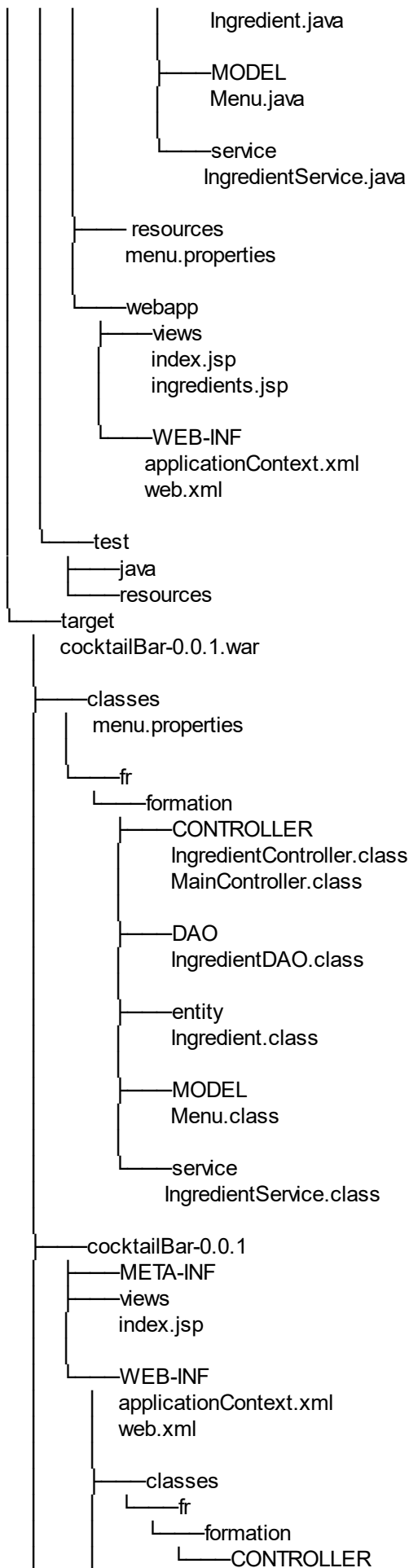
Cr    avec HelpNDoc Personal Edition: [G    rateur gratuit de livres   lectroniques et documentation](#)

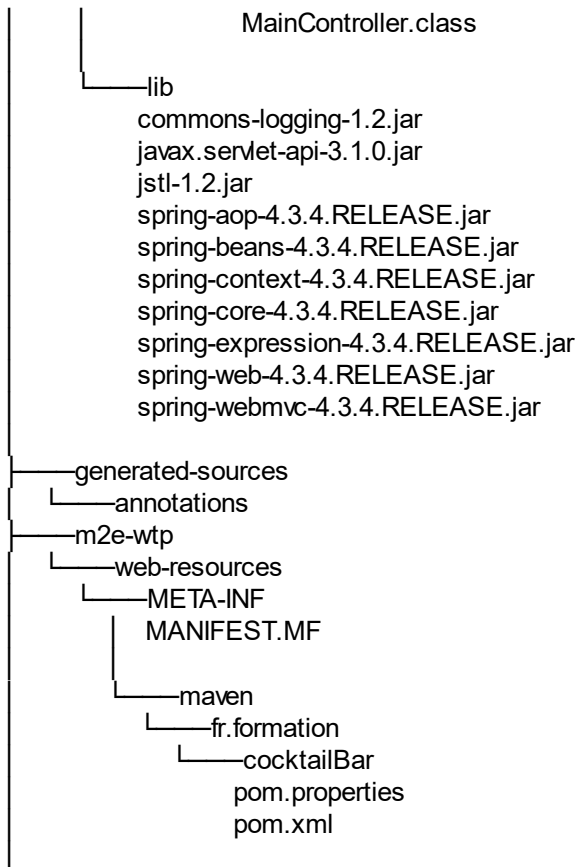
Structure

```

C:.
├── .classpath
├── .project
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── fr
│   │   │   │   ├── formation
│   │   │   │   │   ├── CONTROLLER
│   │   │   │   │   │   ├── IngredientController.java
│   │   │   │   │   │   └── MainController.java
│   │   │   │   │   ├── DAO
│   │   │   │   │   │   └── IngredientDAO.java
│   │   │   │   └── entity

```





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> <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 cl�� </id> <basic name="etat">.....-> nom dans Java <column name="state" />.....-> nom dans mysql </basic> </attributes> </entity> <entity class="chemin complet jusqu'�� la classe java"> <table name="nom_table"></table> <basic name="idJava"> <column name="id_mysql" /> </basic> </attributes> </entity> </pre>
---------	---

persistence.xml	Définit le "DriverManager"
pom.xml	<p>Ajout des dépendences utiles pour la base :</p> <ol style="list-style-type: none"> 1) "mysql connector" 2) "spring data jpa" 3) "hibernate core" dernière version 4) Copie "hibernate core" changer core en hibernate-entitymanager
applicationContext.xml	<p>Spring attend entityManagerEntity Name pris dans persistence.xml --></p> <ol style="list-style-type: none"> 1. Ajout de 2 beans : <ul style="list-style-type: none"> ○ entityManagerFactory ○ transactionManager 2. Ajout <code><jpa:repositories></code> <pre> 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> </pre>
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<Ingredient> getAll(){ return this.dao.findAll(); } @Transactional public Ingredient create(final Ingredient ingredient){ return this.dao.save(ingredient); } } </pre>

Créé avec HelpNDoc Personal Edition: Éditeur de documentation CHM facile

29 / 71

</persistence>

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>
    </dependencies>

```

<!-- D   pendance   ajouter pour les acc   s   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" -->

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   r 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

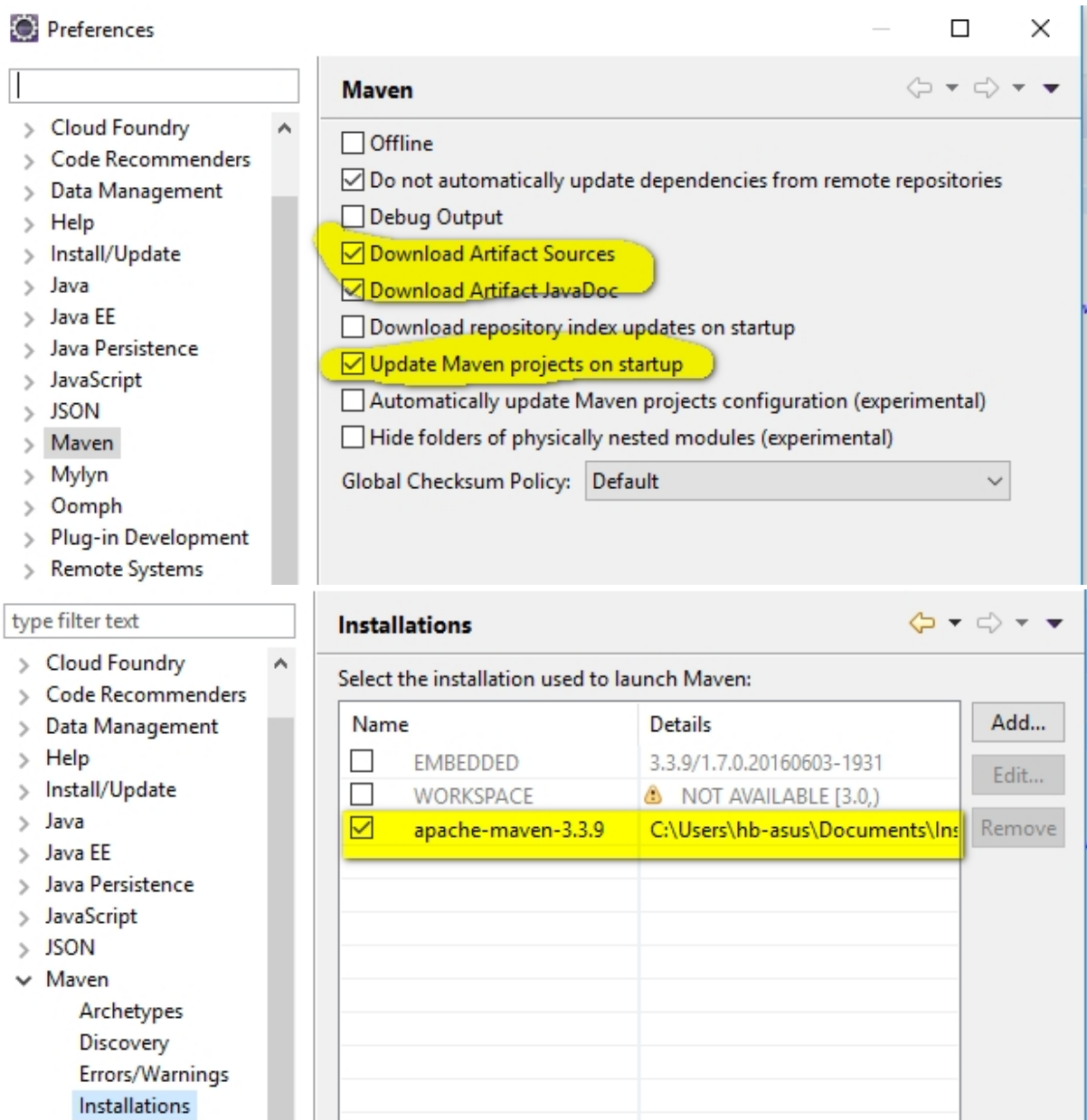
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!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>
```

35 / 71



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	EntityManagerFactory : on lui configure le nom de notre entité = "Cocktail" (*) (**) <p>en JPA on a un persistence objet</p> EntityManager : gère les objetsJava qui sont des entités (0, "Whiskey"), (0, "Tequila"), (1, "Ice cubes"), (1, "Sugar") (2, "CO2")); <p>EntityManaget soccupe de créer les requêtes sql</p> 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](#)

Vue/Controller/

VUE	CONTROLLER		
ingr edient s.jsp< ----- -----	IngredientController RequestMapping ModelAndView view name model List<Ingredient> <-----	@Transactional 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"> ○ MASTER ○ tags
	Local (clone)
	PC --> Staging --> Working Directory

tag : copie figée des sources

3 couches en local			Distant
Workspace	staging	Repository local	Repository distant (remote)
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

4. Avec plusieurs branches

-
- **git branch**
- **git checkout security**
- **git status**
- **git branch**

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

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

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

Projet Cocktail de Jeremy

org.webjars

- o jquery-ui 1.12.1
- o datatables 1.10.12-1
- o datatables-colreorder 1.2.0
- o bootstrap 3.1.0

jquery datatables (jquery est un framework de javascript)

Cr   avec HelpNDoc Personal Edition: [Cr  er des aides HTML, DOC, PDF et des manuels depuis une m  me source](#)

src/main/java

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

CONTROLLER

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

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";
}
}

```

Créé avec HelpNDoc Personal Edition: [Générateur de documentation d'aide HTML gratuit](#)

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);
    }
}

```

DAO

CocktailDAO

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

```
package fr.formation.dao;

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

import fr.formation.entity.Ingredient;

public interface IngredientDao extends JpaRepository<Ingredient, Integer> {

}
```

ENTITY

43 / 71

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  n  rateur de documentation Qt Help gratuit](#)

CocktailService

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

IngredientService

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

sr/main/resources

Cr    avec HelpNDoc Personal Edition: [G  n  rateur 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

Cr    avec HelpNDoc Personal Edition: [Cr   er des fichiers d'aide pour la plateforme Qt Help](#)

persistence.xml

Cr    avec HelpNDoc Personal Edition: [  diteur 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

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

T  l  charger Plugins pour Eclipse

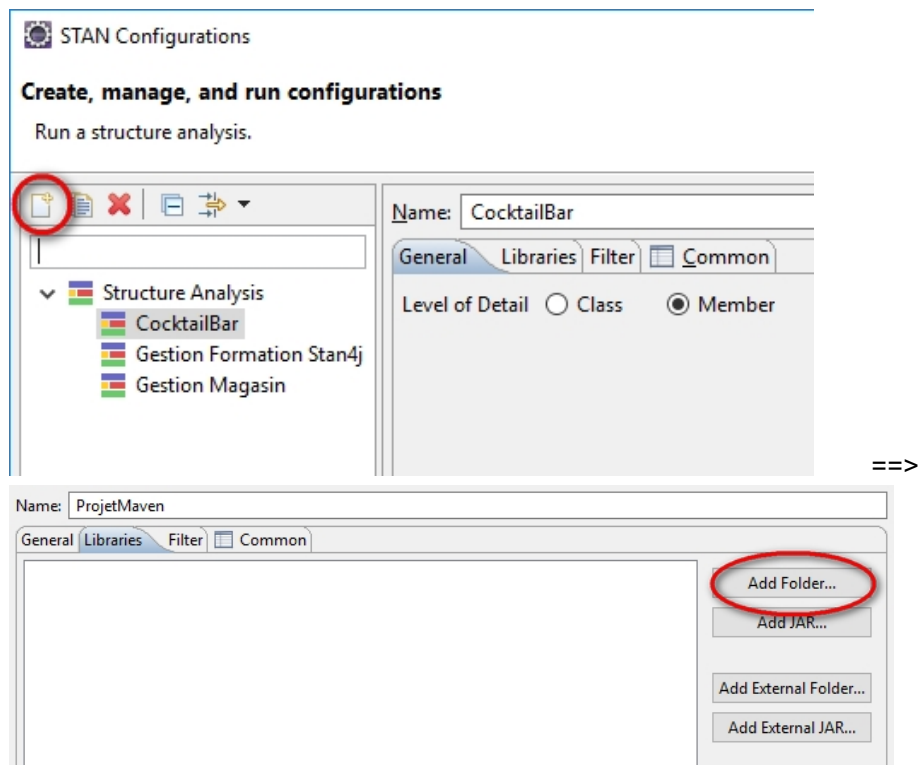
Installation de la fonction Eclipse

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

- Depuis Eclipse, allez    Aide - Installer un nouveau logiciel ...
- Ajouter un nouveau site pour STAN avec l'URL <http://update.stan4j.com/ide>
- S  lectionnez la fonction STAN IDE et appuyez sur "Suivant"
- Suivez l'assistant, lisez et acceptez les termes de la licence
- Red  marrer Eclipse

G  n  rer un projet STAN    partir d'un projet   clipse :





Organigramme choix Objet Collection



GlassFish

Install

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

47 / 71

Lancer la console Glassfish

La commande "asadmin" est utilis   pour contr  ler et manager GlassFish

- o start,
- o stop,
- o configure,
- o deploy applications,
- o etc...

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

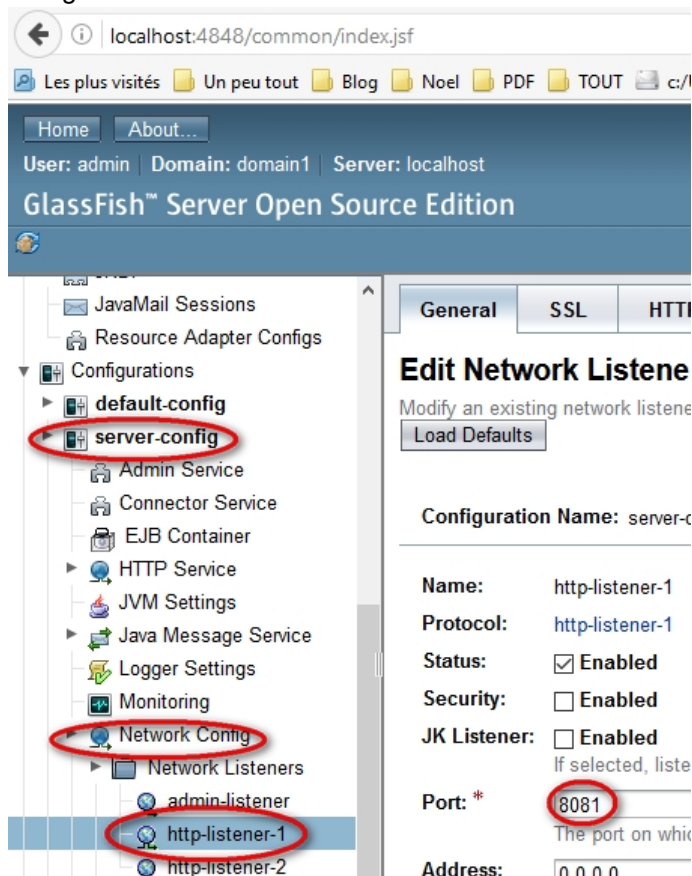
Pour d  marrer : asadmin start-domain

Pour arr  ter : asadmin stop-domain

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

Changement port 8080

Aller sous la console Glassfish
changer

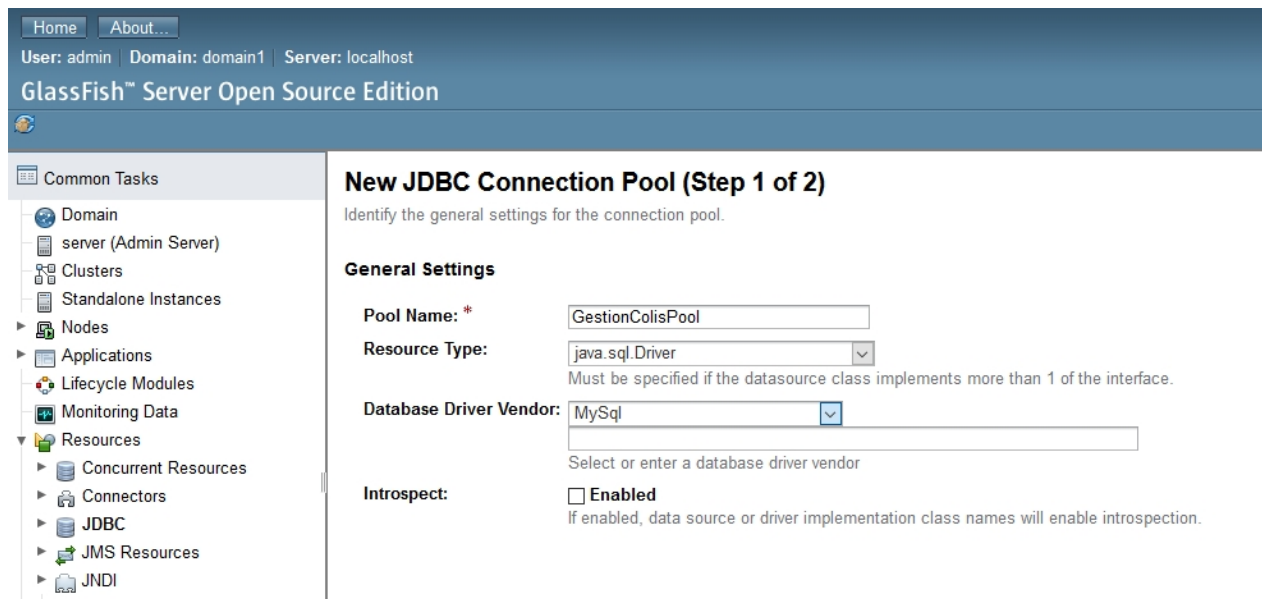


ajouter la variable d'environnement **GLASSFISH_AUTODEPLOY**
qui pointe vers

```
C:\Users\hb-asus>echo %GLASSFISH_AUTODEPLOY%
```

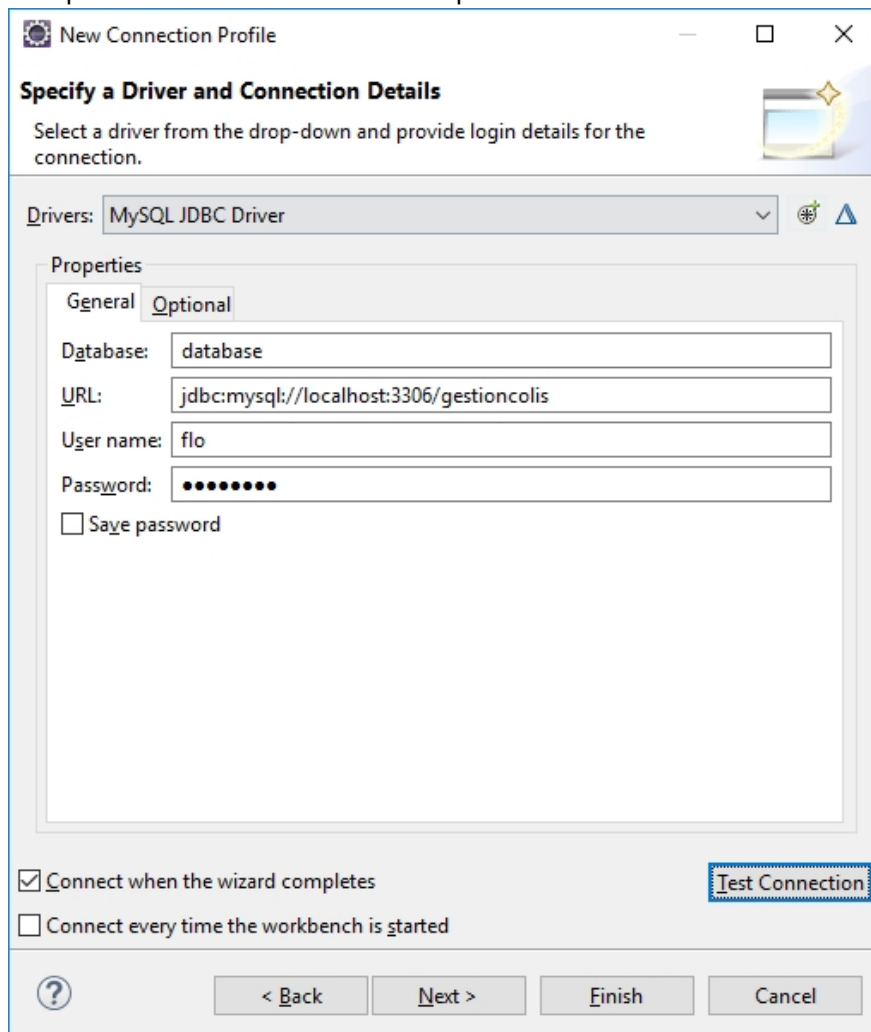

Cr   avec HelpNDoc Personal Edition: Cr  ation d'aide CHM, PDF, DOC et HTML d'une m  me source

Copier le jar de mysql dans Glassfish :



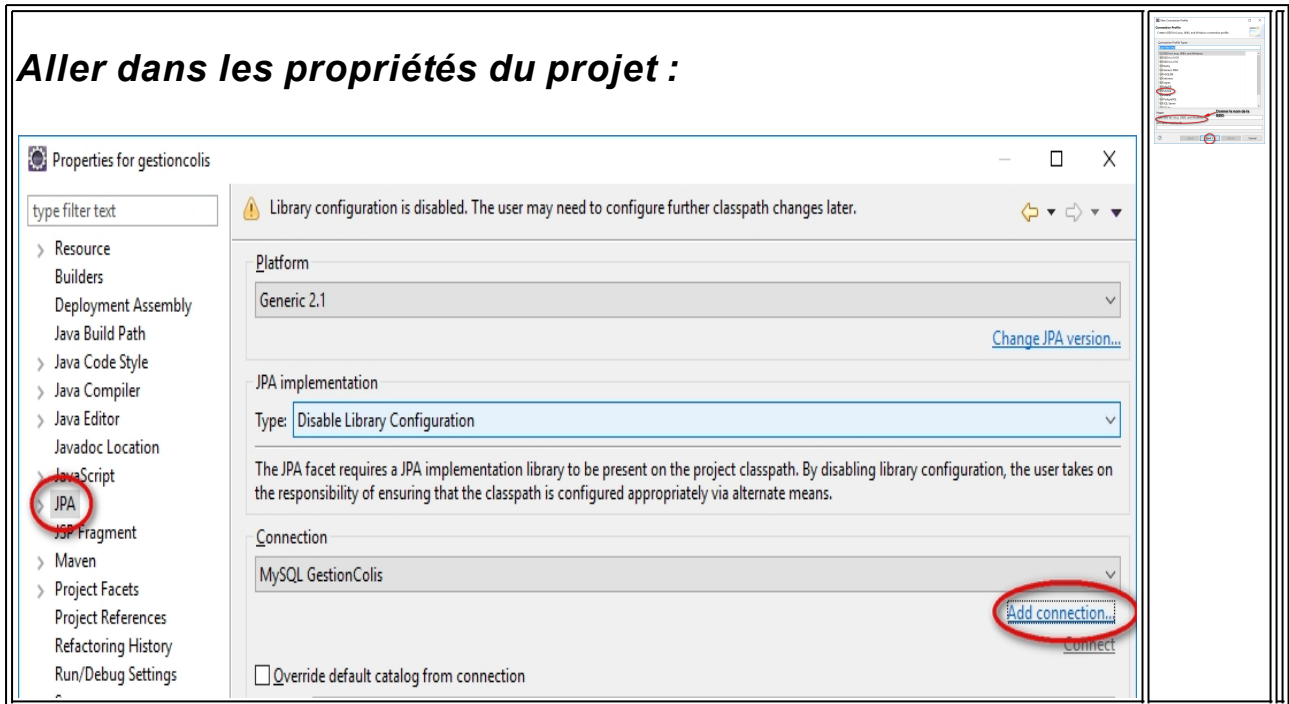
Créer un user dans la base de données avec un mot de passe.

Puis paramétrer la connexion 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

New Connection Profile

Specify a Driver and Connection Details

Select a driver from the drop-down and provide login details for the connection.

Drivers: MySQL JDBC Driver

Properties

General Optional

Database: Ecrire le nom de la database

URL: jdbc:mysql://localhost:3306/database

User name: Nom d'utilisateur avec mot de passe(penser à la config dans glassfish)

Password:

☐ Save password

Tester la connexion avant de sauver la password

☒ Connect when the wizard completes

☐ Connect every time the workbench is started

Test Connection

< Back Next > Finish Cancel

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

Sch  ma Appli

APPLICATION			Serveur GlassFish	BDD Mysql
Pr��sentation	M��tier	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 ��tape du cycle		Persistence Context		

JSF	Managed Beans	DAO		
-----	---------------	-----	--	--

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

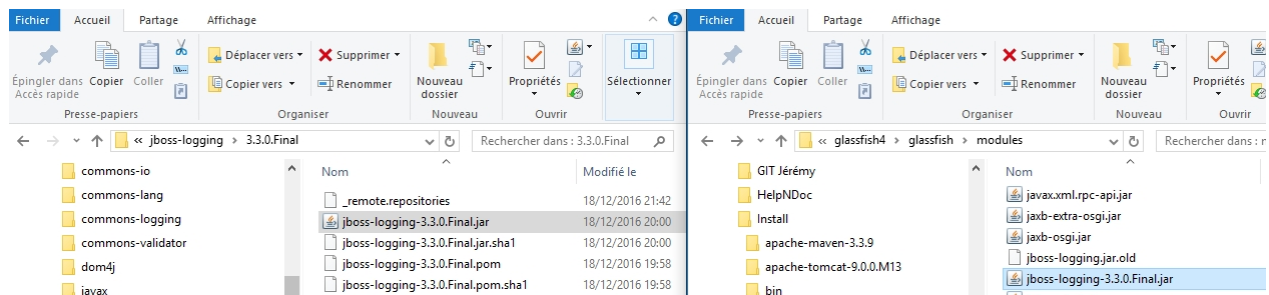
MAJ dans GlassFish

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



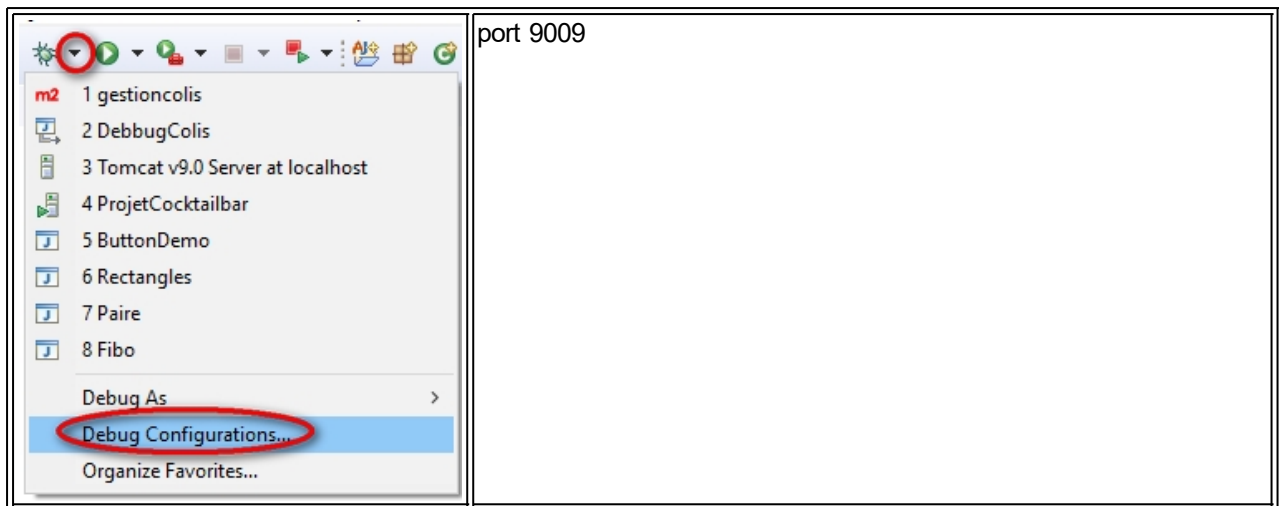
Créé avec HelpNDoc Personal Edition: [Générateur gratuit de livres électroniques et documentation](#)

Remote debugging

Créé avec HelpNDoc Personal Edition: [Créer des livres électroniques EPub facilement](#)

Configuration Glassfish et eclipse

Sous Glassfish	
Sous eclipse	<p>Sous Debug config remote Java Appli new nom liier appli localhost</p>



Cr   avec HelpNDoc Personal Edition: [G  n  rateur d'aides Web gratuit](#)

D  claration des Logs en java

Commande Maven pour conna  tre les d  pendances cr  er dans le pom.xml

- o mvn dependency:tree

Qd on d  clarer un log :

On ne doit d  clarer qu'un seul log par classe en static et final

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

postConstruct m  thode

entit   d  tach  e c'est    dire entit   non manager.
(   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 m  me transaction.

Lors de l'  criture de la classe qui g  re les erreurs, pas de constructeur par d  faut cela   vite de cr  er une erreur sans rien

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

Classe EntityManager et UserTransaction

EntityManager :

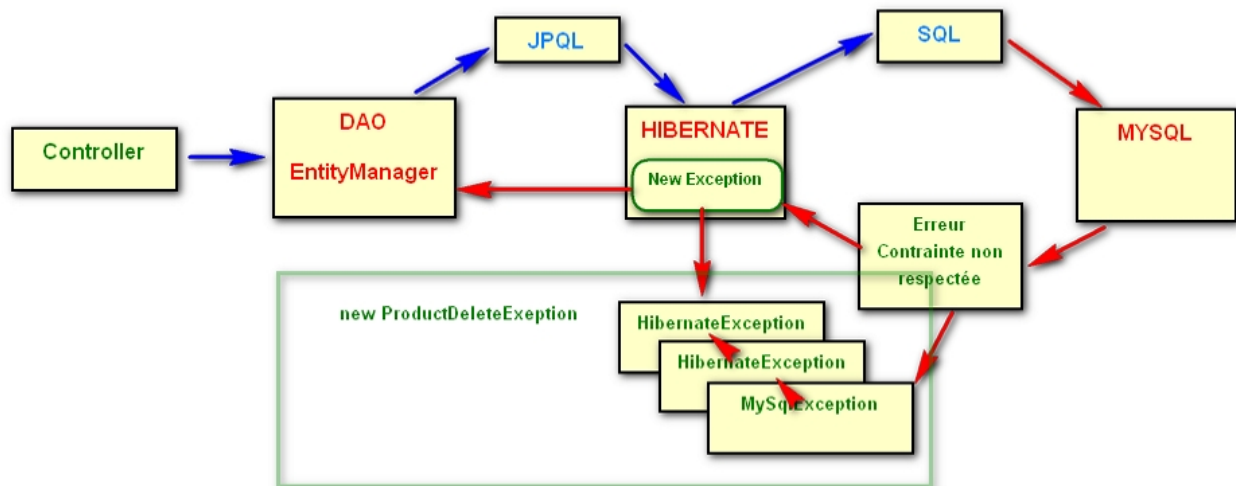
C'est une classe qui est charg  e de mettre en musique les correspondances d  finies dans les entit  s, et qui r  alise donc toutes les op  rations CRUD (Create, Read, Update, Delete) sur la base de donn  es.

UserTransaction

C'est une classe

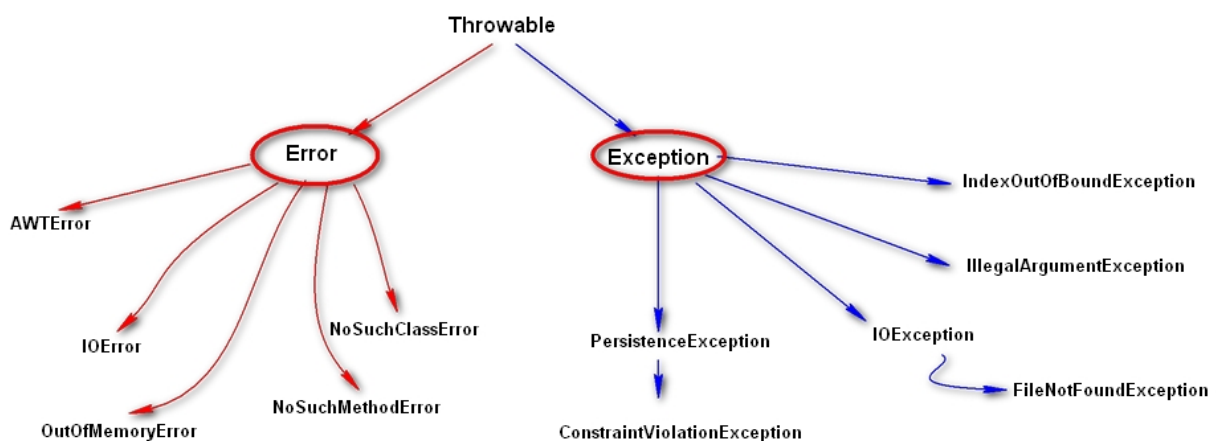
Cr   avec HelpNDoc Personal Edition:   crire des livres   lectronique Kindle

Sch  ma de vie d'une exception



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

Sch  ma des ERROR / EXCEPTION



Cr   avec HelpNDoc Personal Edition: G  n  rateur facile de livres   lectroniques et documentation

Liste des tutos donn  s par J  r  my

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](#)

JFS

Créé avec HelpNDoc Personal Edition: [Générer des livres électroniques EPub facilement](#)

Annotations

Synthèse

	bean		controller	dao
	entity	login		

implements	Serializable	Serializable	Serializable	
extends				AbstractDao<Product>
Classe	@ManagedBean @ViewScoped	@ManagedBean @ApplicationScoped	@ManagedBean @ViewScoped	rien
Attributs				@PersistenceContext 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 <u>Commande</u> @ManyToOne @JoinColumn(name="COMMANDE") private Commande commandeBean ;	
Méthode		

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 {
```

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

Annotations CONTROLLER

Annotations n  cessaires 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;
```

Cr    avec HelpNDoc Personal Edition: [G  n  rateur d'aides CHM gratuit](#)

Annotations DAO

Annotations n  cessaires pour DAO

1) Au dessus de la classe

Rien

2) Attribut

```
@PersistenceContext
protected EntityManager em;

@Resource
private UserTransaction transaction;
```

Cr    avec HelpNDoc Personal Edition: Cr   er facilement des fichiers Qt Help

Cr   avec HelpNDoc Personal Edition: G  n  rateur de documentation iPhone gratuit

<pre><f:viewParam name="productId" value="#{productController.productId}" /></pre>	<p>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.</p>
<pre><f:event type="preRenderView" listener="#{productController.prepareEdit}" /> </f:metadata></pre>	<p>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'.</p>
<pre><ui:fragment rendered="#{empty productController.productId}"> <h2>Créer un nouveau produit :</h2> </ui:fragment></pre>	<p>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.</p>
<pre><ui:fragment rendered="#{!empty productController.productId}"> <h2>Modifier un produit :</ h2> </ui:fragment></pre>	<p>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.</p>
<pre><b:form></pre>	
<pre><b:inputText label="Id" value="#{productController.productId}" rendered="#{!empty productController.productId}" readonly="true" /></pre>	<p>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.</p>
<pre><b:inputText label="Intitulé" value="#{productBean.intitule}"></pre>	
<pre><f:validateRequired /> </b:inputText> <b:inputText type="number" label="Poids" value="#{productBean.poids}"></pre>	<p>Rend la saisie du champ obligatoire.</p>
<pre><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"</pre>	<p>La saisie doit pouvoir être transformée en float Java.</p>

```
action="#{productController.save}" />
</b:form>
```

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

Web Services

Spring security

La s  curit   est souvent pos  e au niveau r  seau.

code pin g  n  r   par des calcul  ttes, permet de se connecter    un environnement s  curis   machine virtuelle ou ssh.

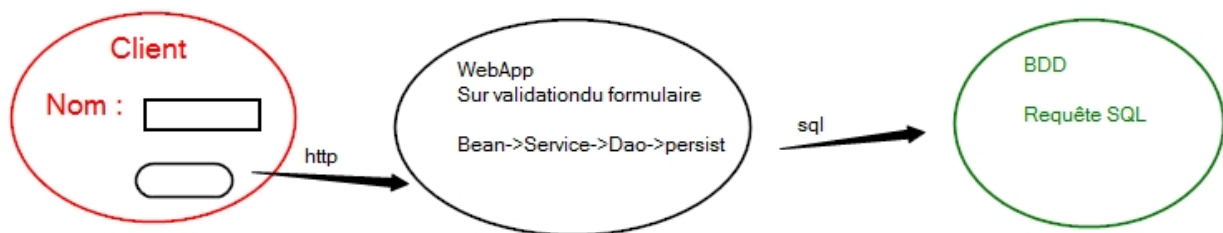
Niveau applicatif rare besoin de s  curit   car g  r   par les par  feu

Authentification : savoir reconn  tre un user de confiance

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

V  rification sur le serveur

Hackage possible



en sql -- : met en commentaire
notre code et le remplace pour
  viter un fait de emp  chement de
caract  res sp  cifiques au SGBD

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](#)

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

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

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.AuthenticationMana
```



```

gerBuilder;
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.authoritiesBy
Username());
    }

    private String authoritiesByUsername() {

        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 bdd pour Spring

    @Bean
    public DataSource dataSource() {

        final JndiDataSourceLookup nslookup = new JndiDataSourceLookup();
        nslookup.setResourceRef(true);

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

        // java:comp/env/
        // set resourceRef à
        // true
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory() {
        final LocalContainerEntityManagerFactoryBean factory = new
LocalContainerEntityManagerFactoryBean();

        // Notre provider est hibernate 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

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

Test unitaire

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

Service rest

Quelques explications

Comme on a besoin d'une requ   te vers la BDD 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 cot    pr   sentation

- 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 d   finir API REST avec les noms logiques
 NIVEAU 2 : Standardis    pour    tre 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();
    }
}

```

67 / 71

```

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> getLiasse() {
    return this.liasses;
}

public void setLiasse(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;
    }
}

```

Cr   avec HelpNDoc Personal Edition: [Produire des livres EPub gratuitement](#)

Cycles de d  veloppement

Il existe diff  rents types de cycles de d  veloppement entrant dans la r  alisation d'un logiciel.

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

1. Cycle en cascade

→ Analyse Pr  paration

→ Conception Blueprint

→ D  veloppements / Impl  mentation

→ Validation/ Test

→ Evaluation / Publication

Test unitaire

2. *test-driven development*

Le test-driven development (TDD) ou en fran  ais d  veloppement pilot   par les tests est une technique de d  veloppement de logiciel qui pr  conise d'  crire les tests unitaires avant d'  crire le code source d'un logiciel.

3. Niveau de sc  nario de validation des tests

4. Outils de tidating Mentis trac jira Redmine

5. Mantis est un syst  me de suivi d'anomalies logicielles (bugs) bas   sur une interface web.

6. Redmine est une application web libre de gestion de projets presque compl  te en mode web, d  velopp  e en Ruby sur la base du framework Ruby on Rails. La gestion des tests devra   tre faite avec un autre outil.

7. 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+)
8. Jira est un système de suivi de bugs, un système de gestion des incidents, et un système de gestion de projets développé par Atlassian.