

Développement Web JEE

Spring MVC

Mohamed Youssfi

Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)

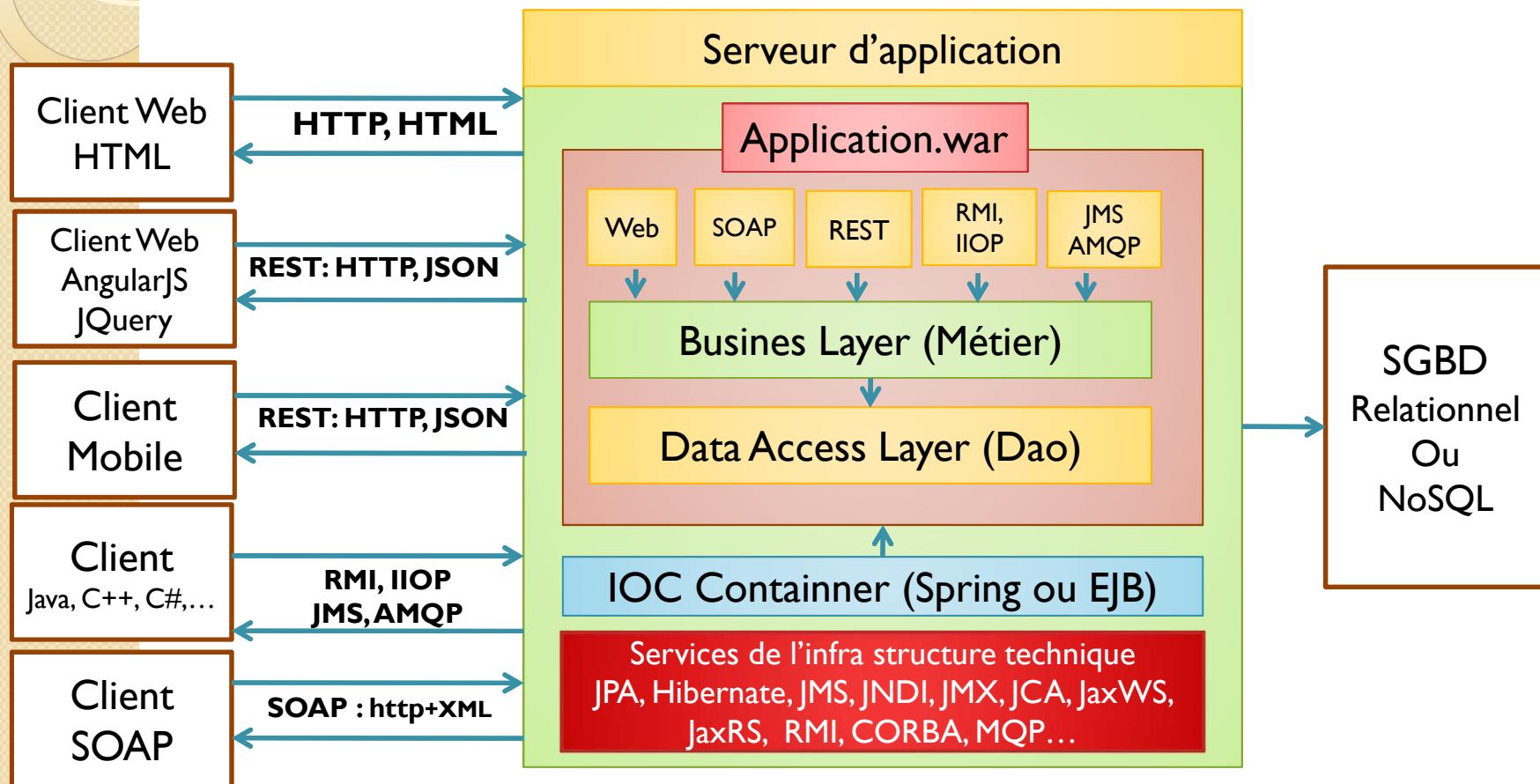
ENSET, Université Hassan II Casablanca, Maroc

Email : med@youssfi.net

Supports de cours : <http://fr.slideshare.net/mohamedyoussfi9>

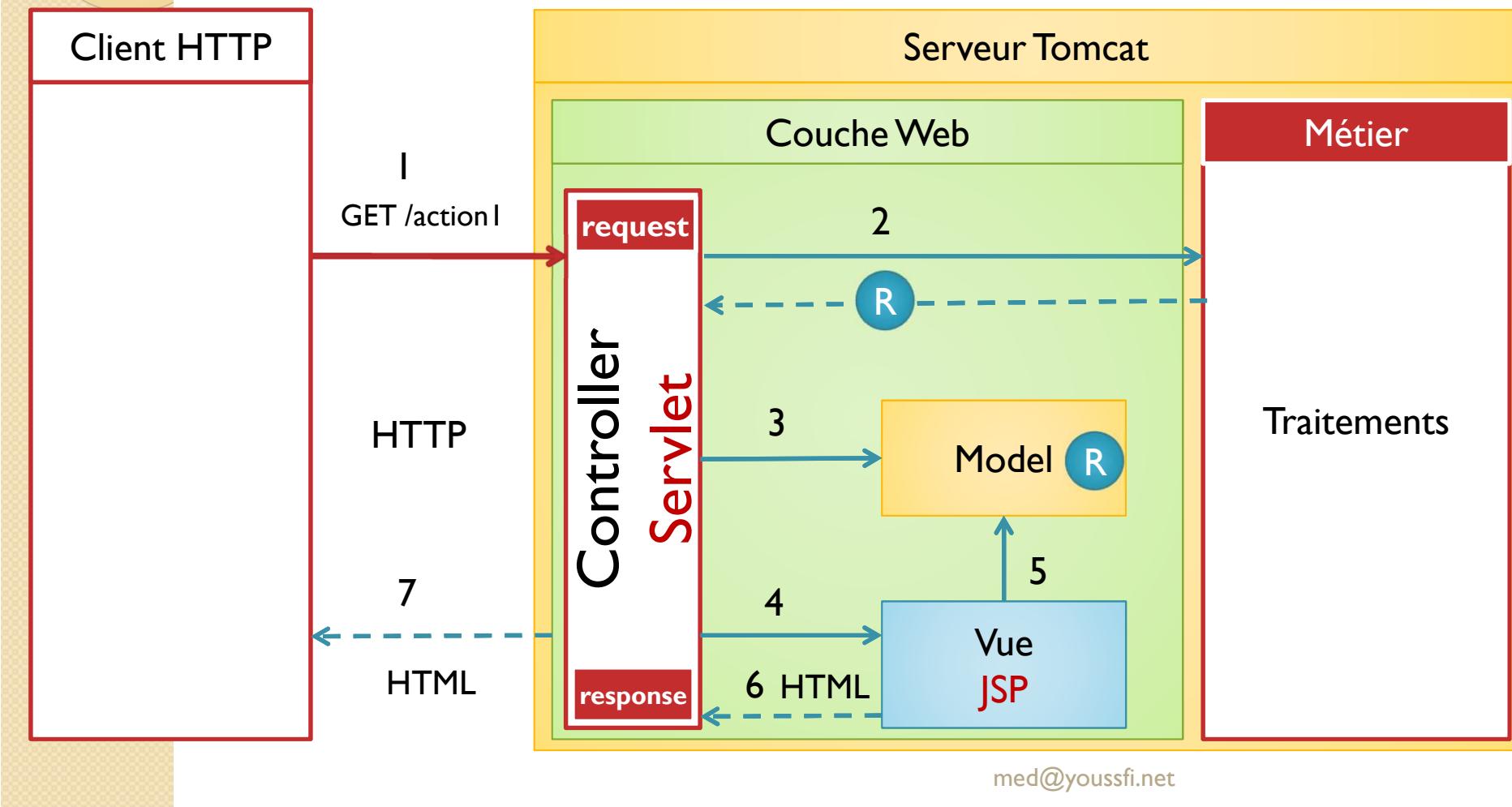
Chaîne vidéo : <http://youtube.com/mohamedYoussfi>

Architecture JEE



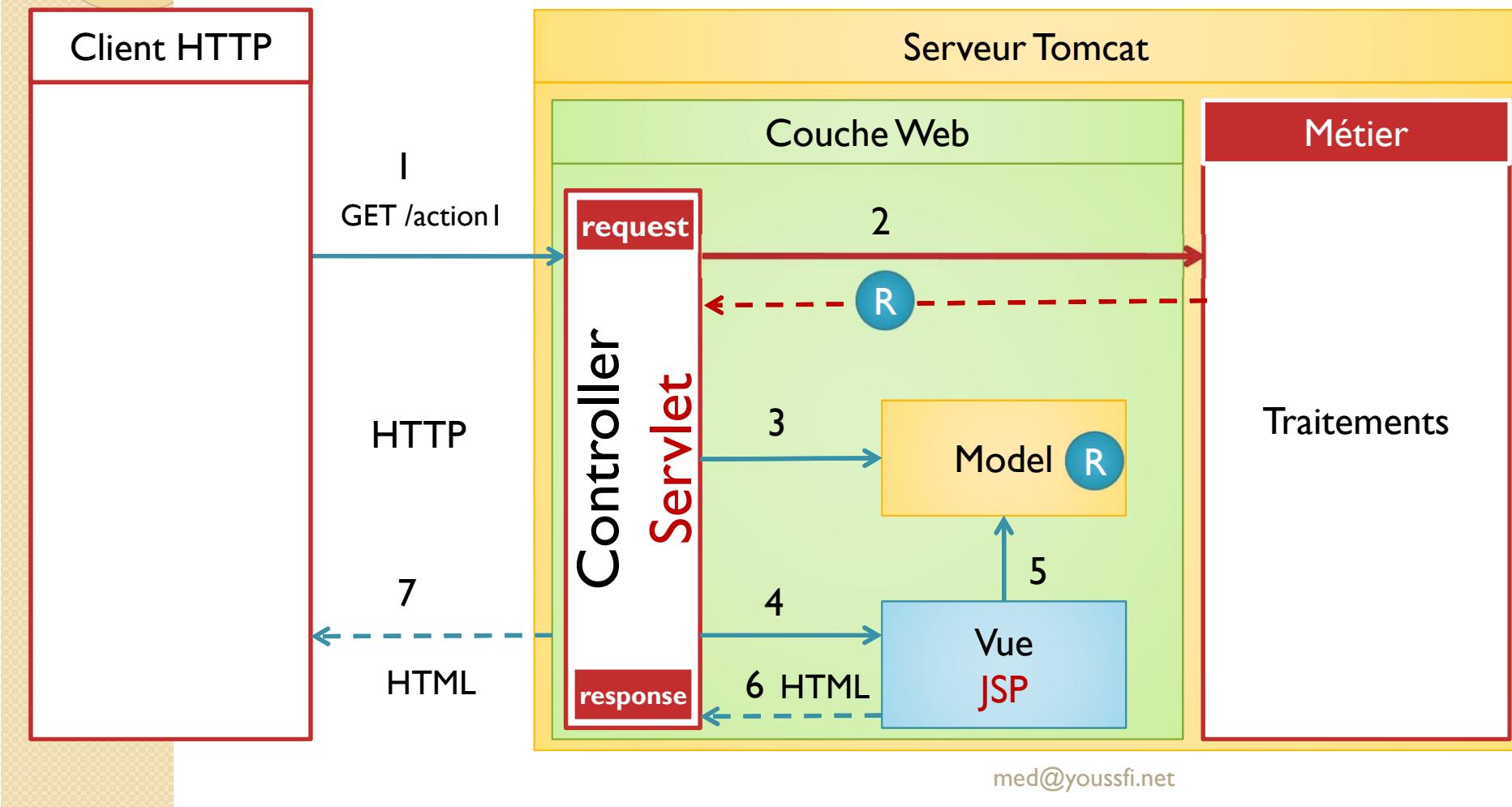
Architecture Web JEE

I – Le client envoie une requête HTTP de type GET ou POST vers le contrôleur représenté par un composant Web JEE : **SERVLET**. Pour lire les données de la requête HTTP le contrôleur utilise l'objet **request** de type **HttpServletRequest**



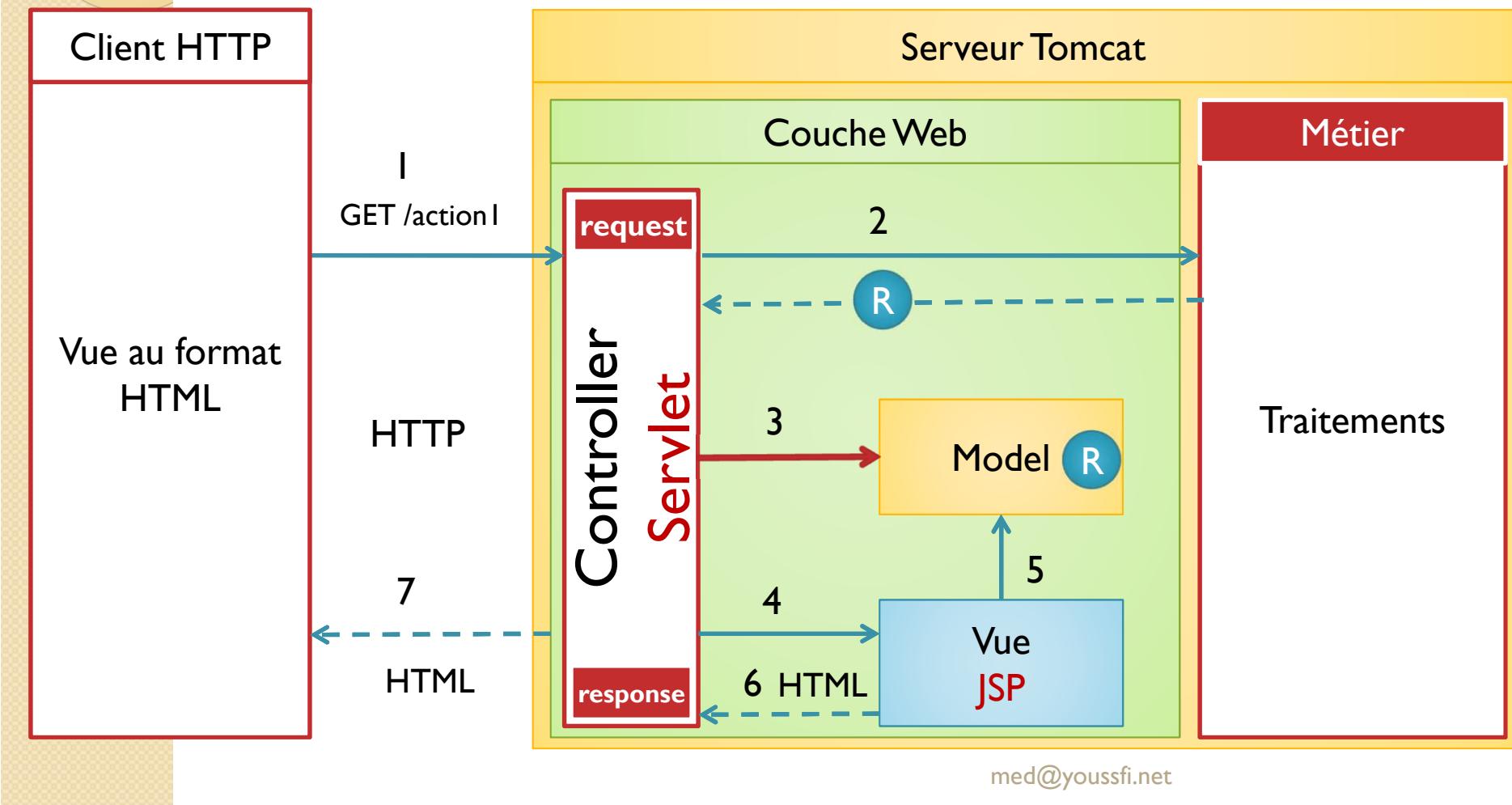
Architecture Web JEE

2 – Le contrôleur fait appel à la couche métier pour effectuer les traitements et récupère les résultats R



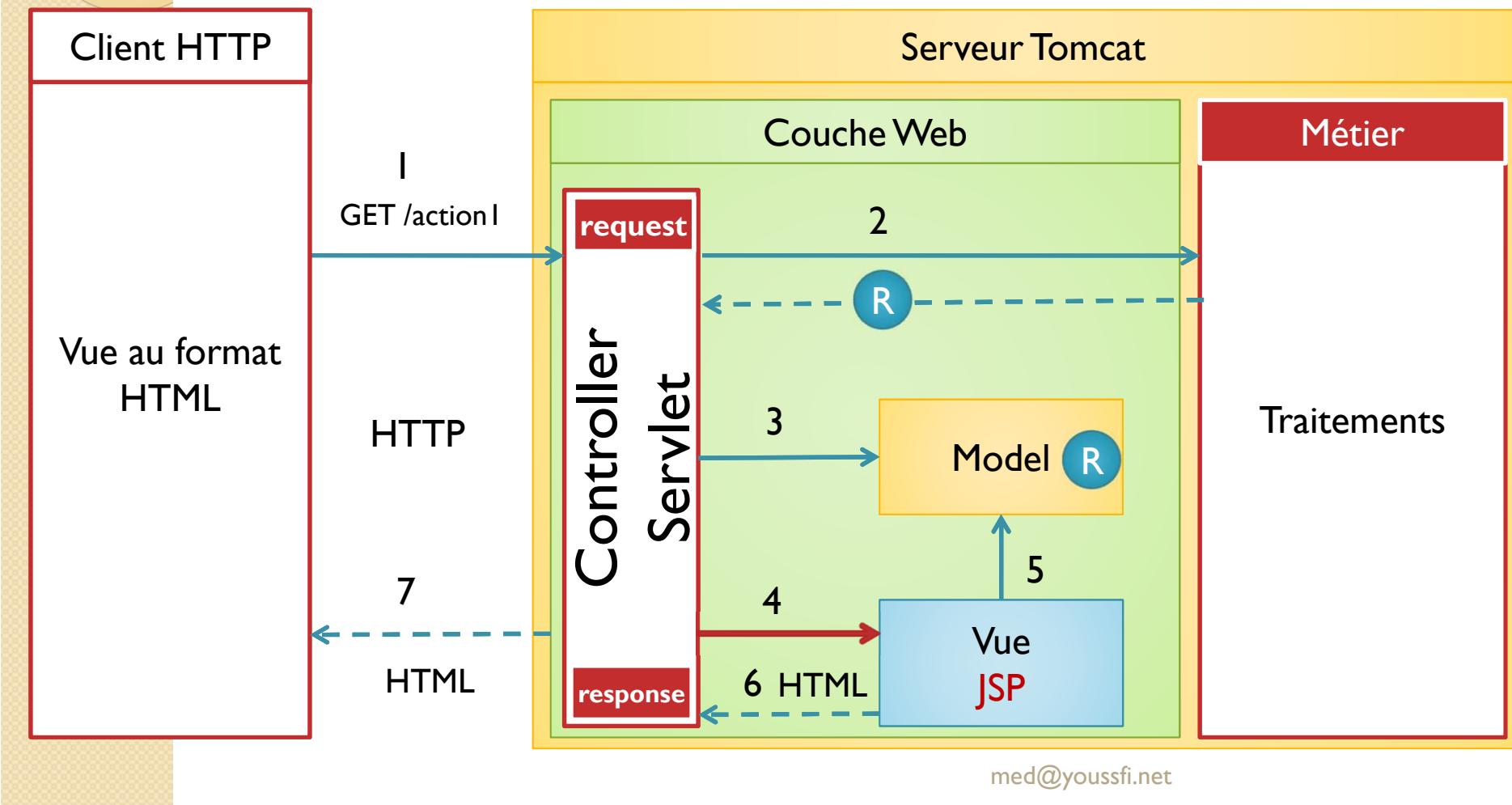
Architecture Web JEE

3 – Le contrôleur Stocke le résultat R dans le modèle M. Le modèle est généralement une objet qui permet de stocker toutes les données qui seront affichées dans la vue. Généralement, le contrôleur stocke le modèle dans l'objet request ou session.



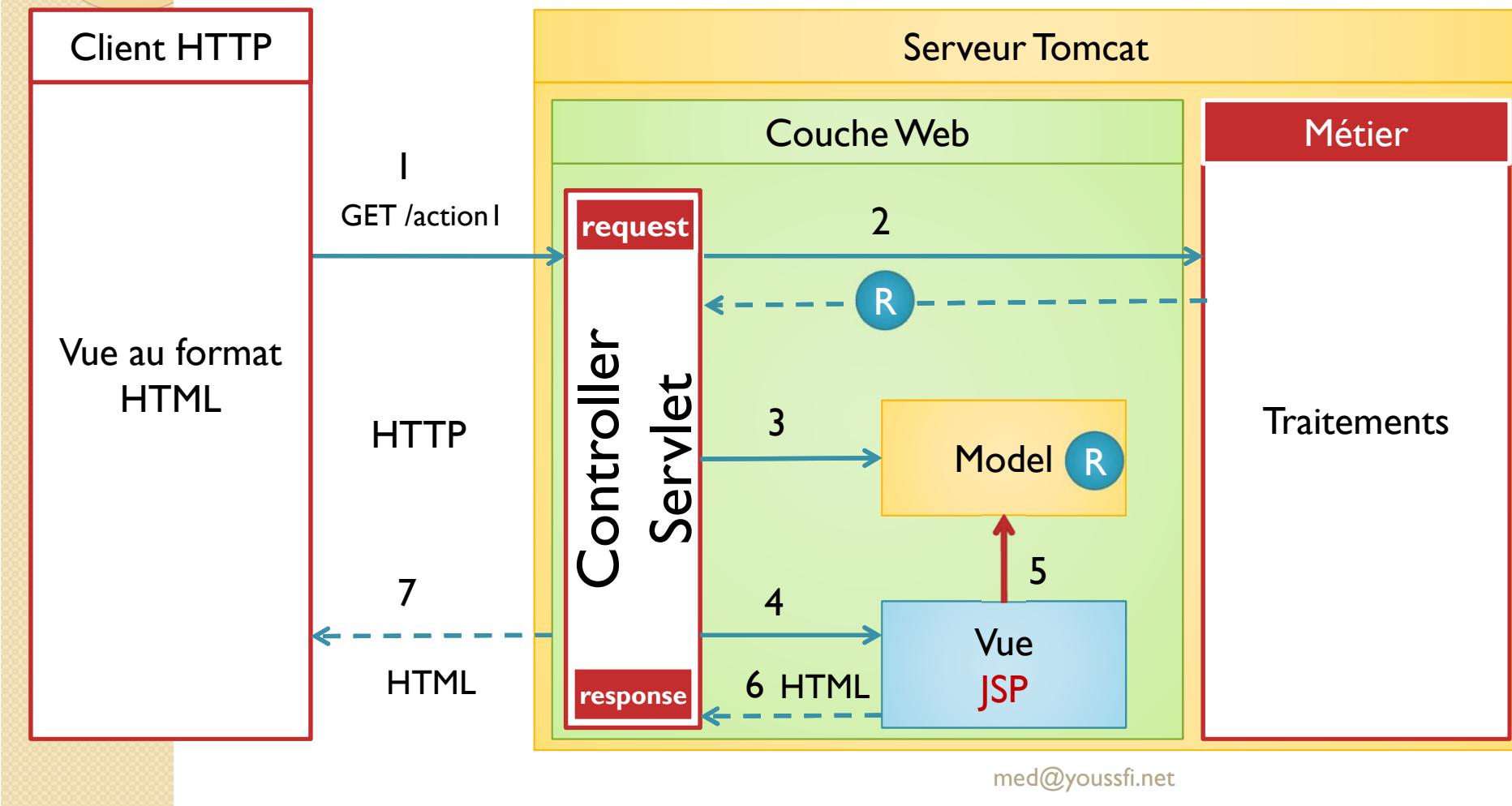
Architecture Web JEE

4 – Le contrôleur fait appel à la vue JSP (Java Server Pages) en lui transmettant les mêmes objets request et response. Cette opération s'appelle Forwarding.



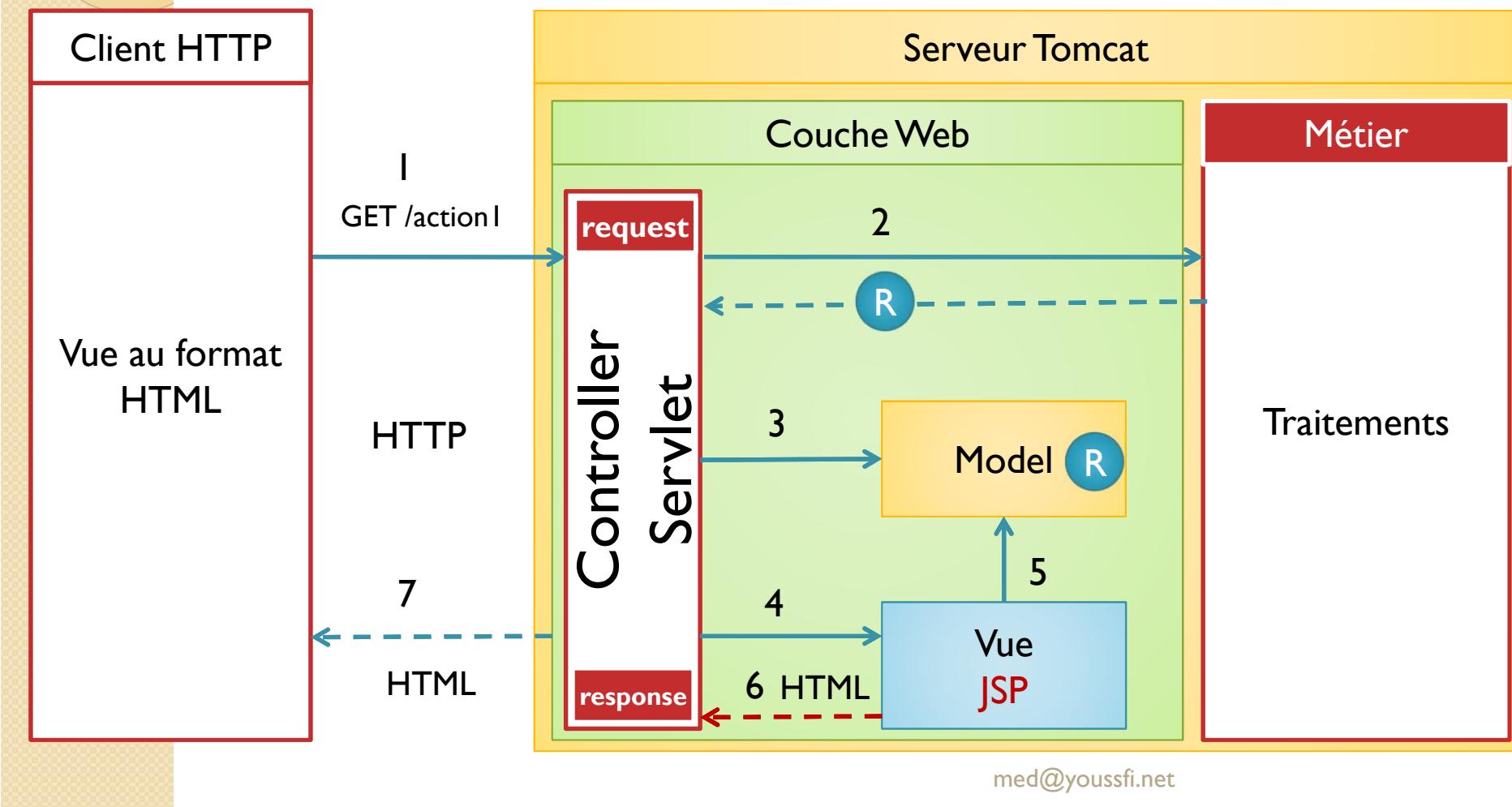
Architecture Web JEE

5 – La vue JSP récupère le résultat à partir du modèle. La vue retrouve le modèle dans l'objet request ou session.



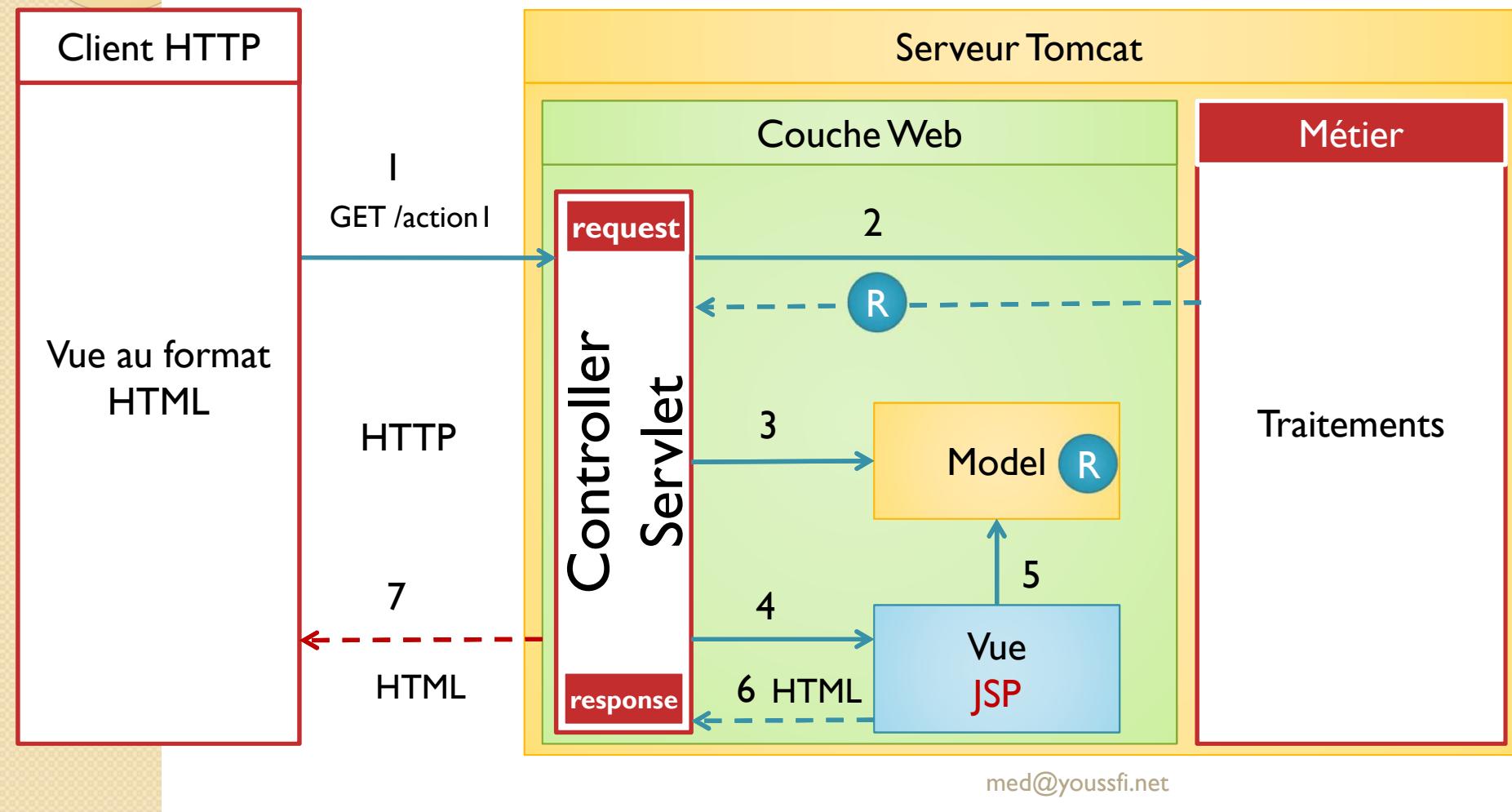
Architecture Web JEE

6 – Le vue JSP génère dynamiquement une page HTML qui contient les résultats du modèle en utilisant l'objet response.



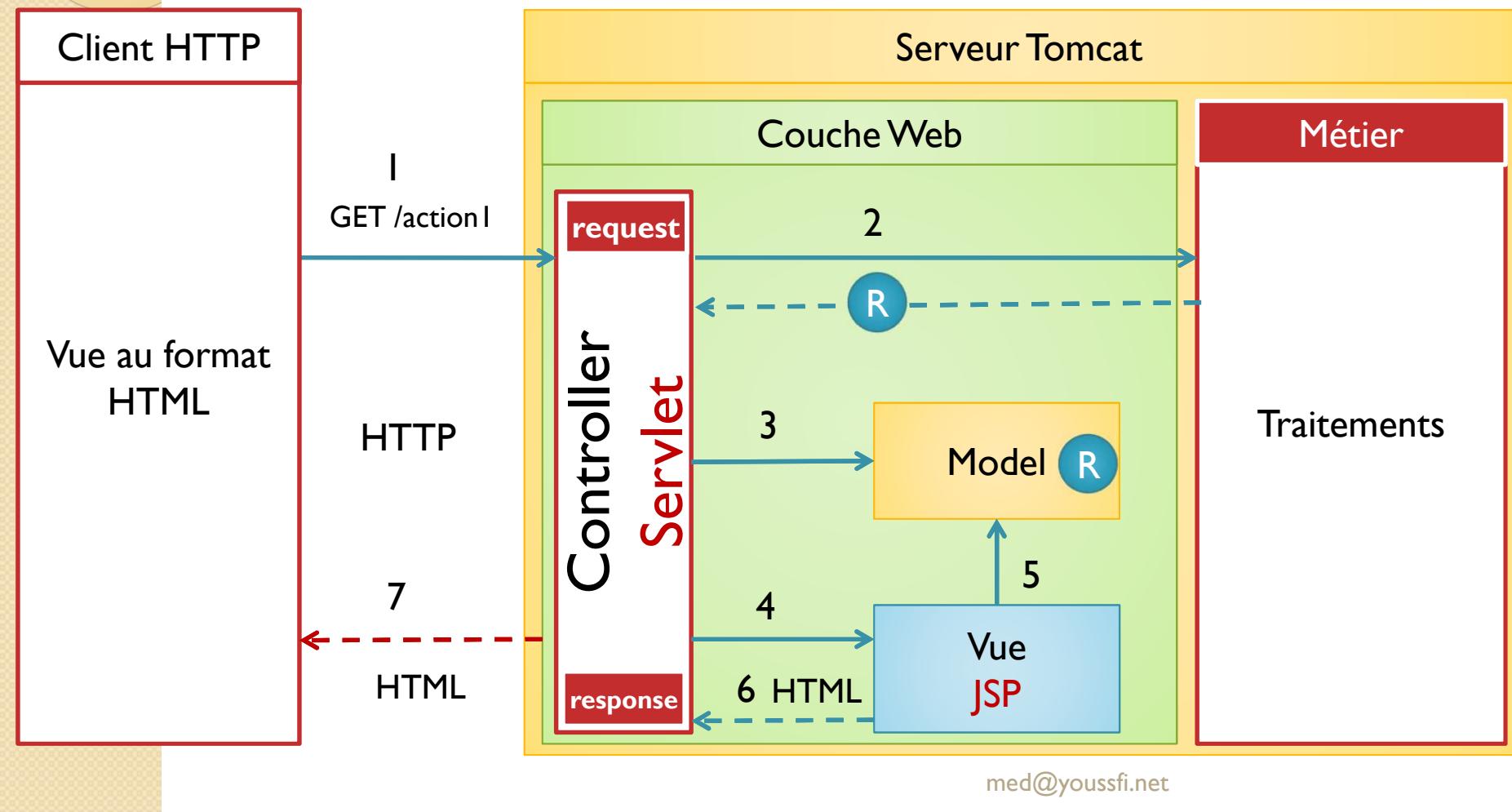
Architecture Web JEE

7 – La page HTML générée est envoyée dans le corps de la réponse HTTP.



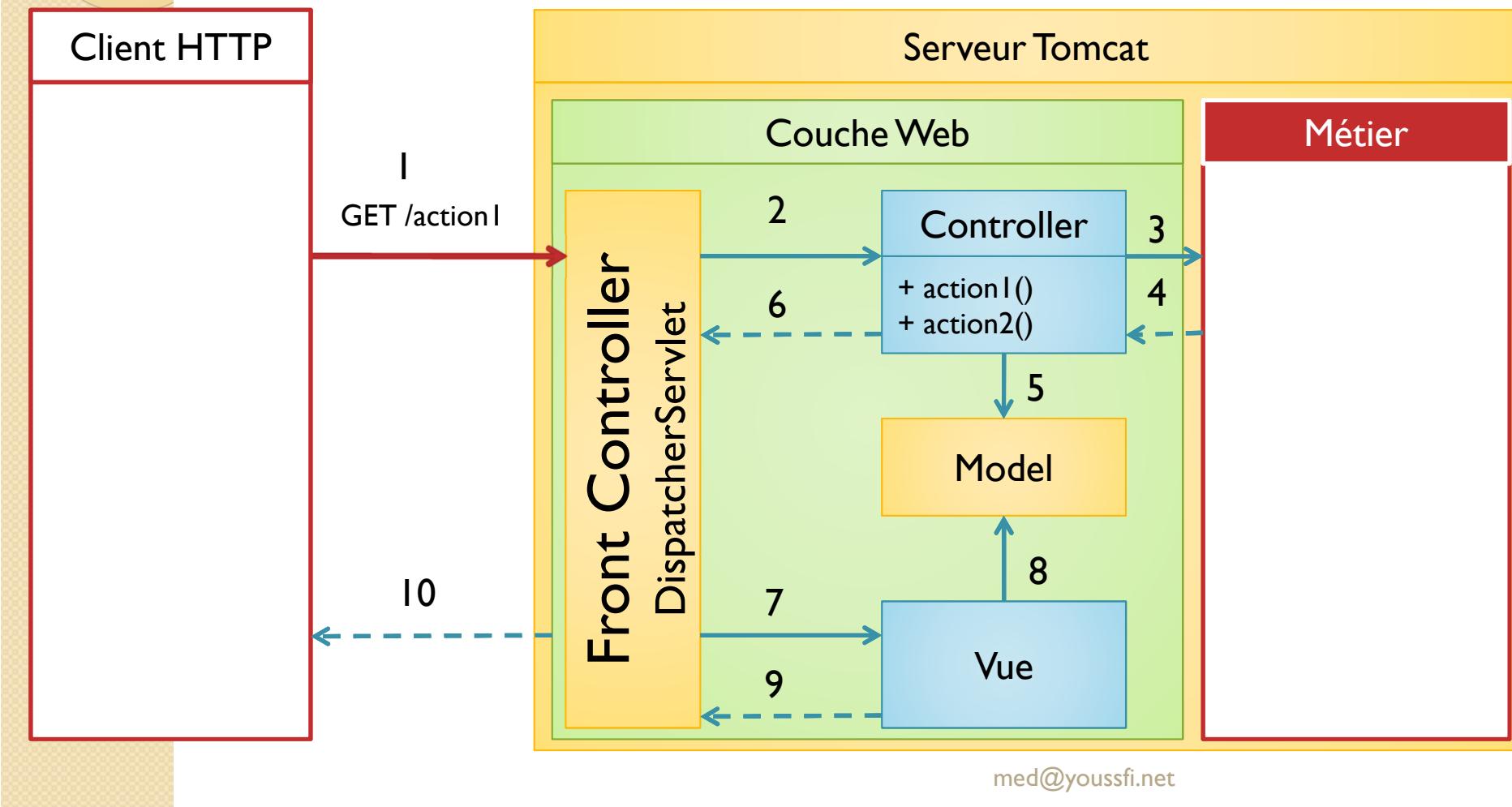
Architecture Web JEE

8 – Le Browser Web affiche le rendu de la page HTML reçue.



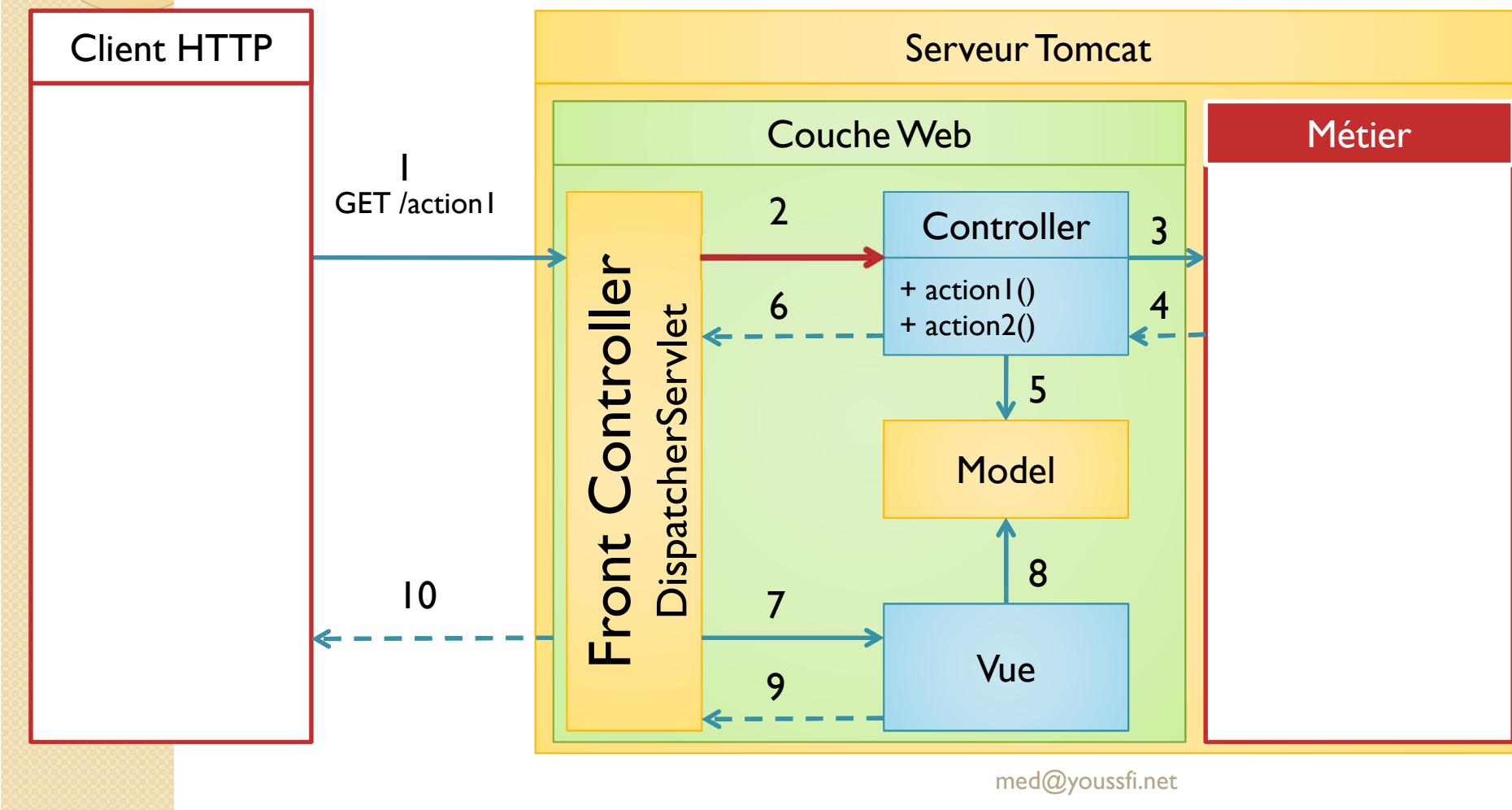
Architecture Spring MVC

I – Le client envoie une requête HTTP de type GET ou POST



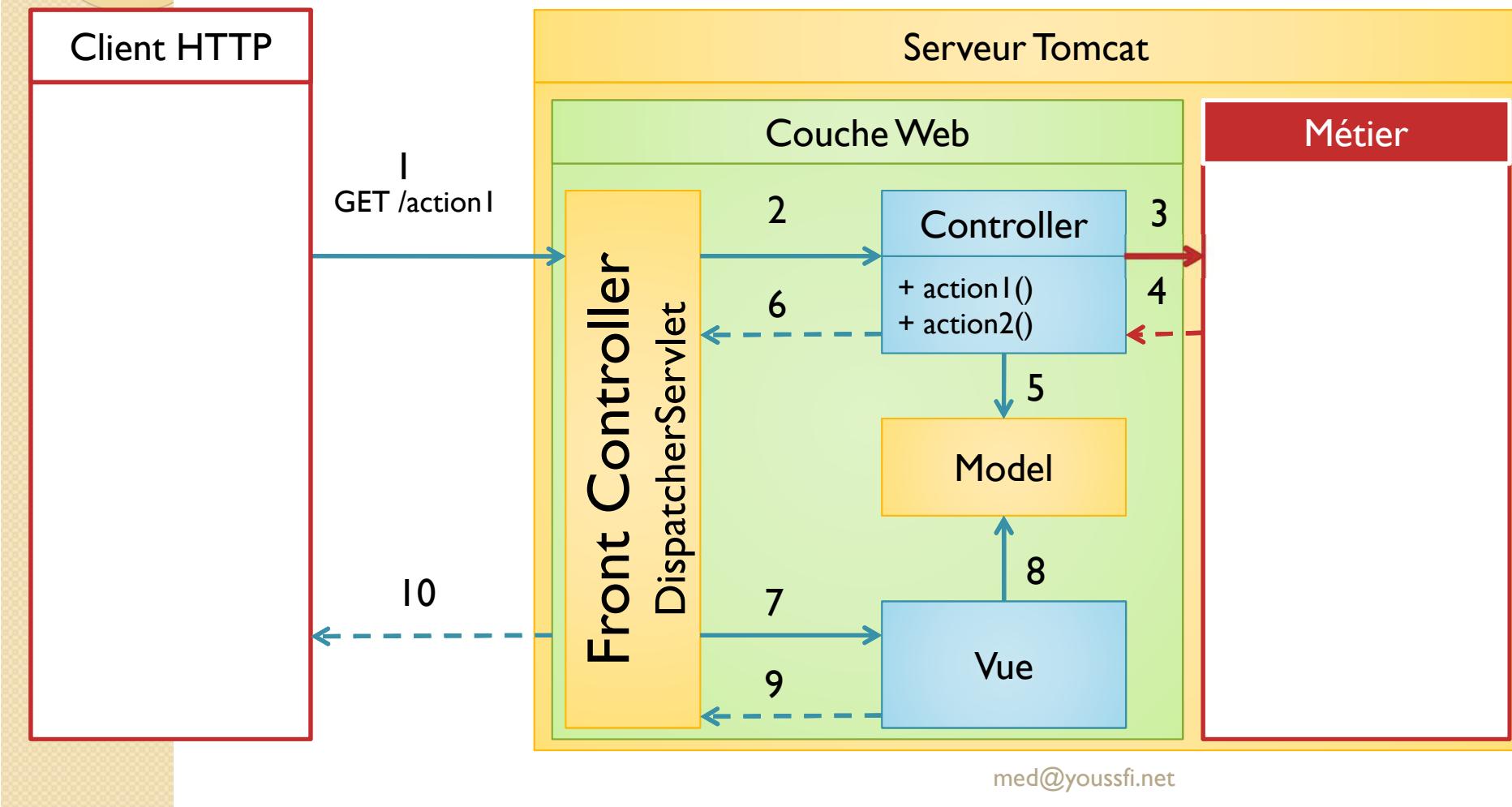
Architecture Spring MVC

2– Toutes les requêtes HTTP sont traitées par un contrôleur frontal fourni par Spring. C'est une servlet nommée DispatcherServlet . Chaque action de l'URL, DispatcherServlet devrait exécuter une opération associée à cette action. Cette opération est implémentée dans une classe appelée Controller qui représente un sous contrôleur ou un contrôleur secondaire.



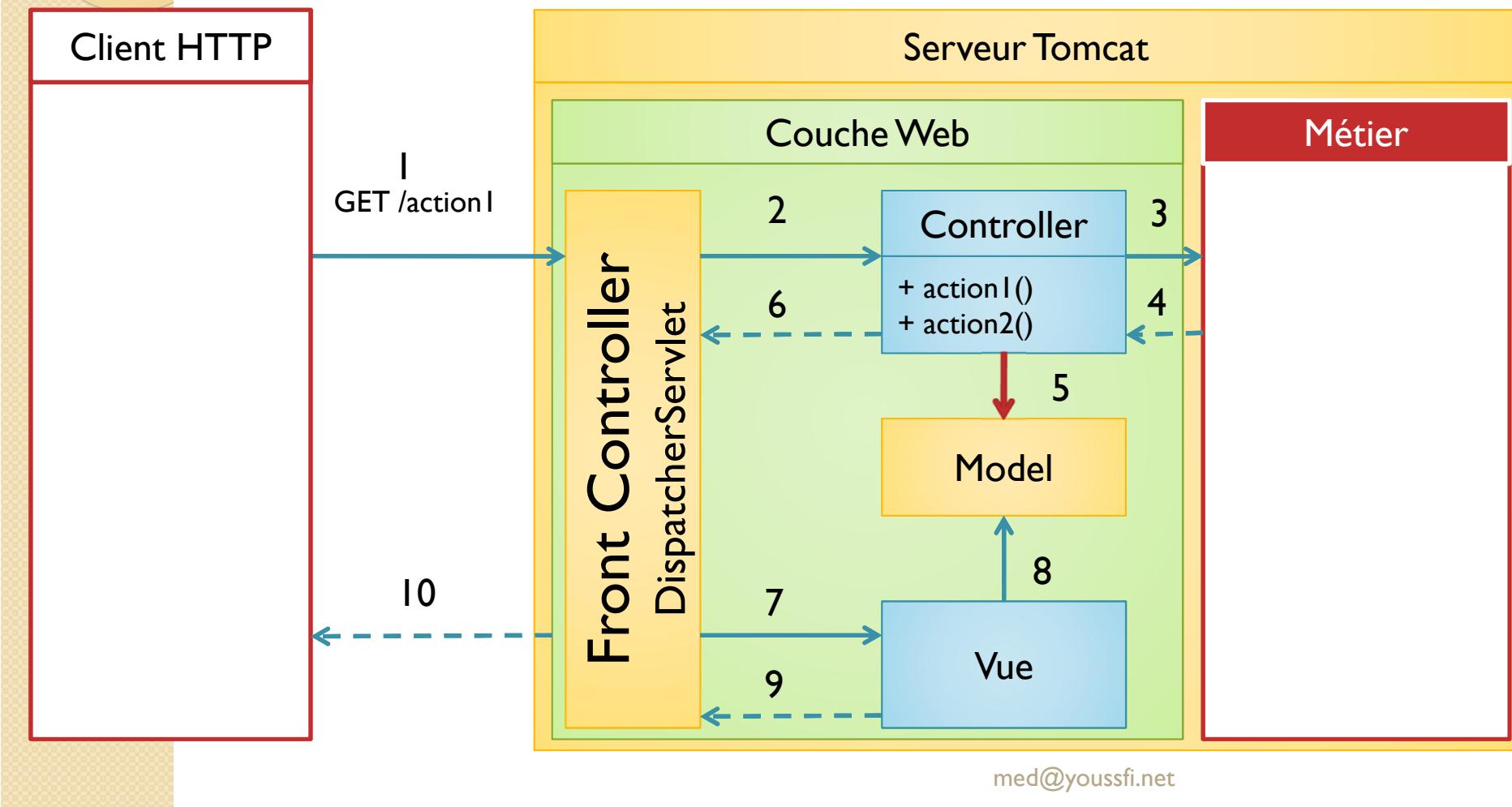
Architecture Spring MVC

3 et 4 – Le sous contrôleur exécute le traitement associé à l'action en faisant appel à la couche métier et récupère le résultat .



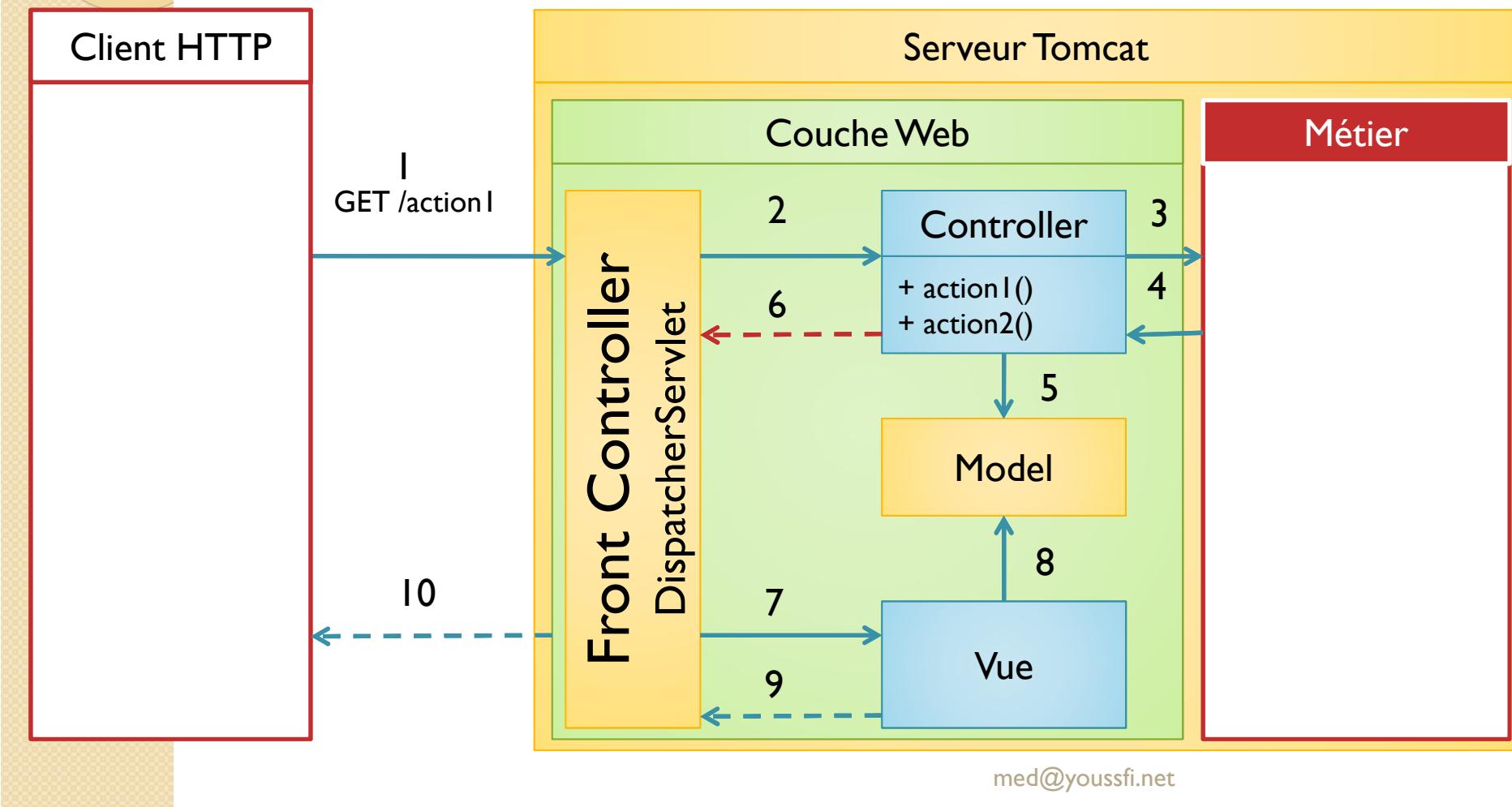
Architecture Spring MVC

5– Le sous contrôleur stocke le résultat dans le modèle fourni par Spring MVC.



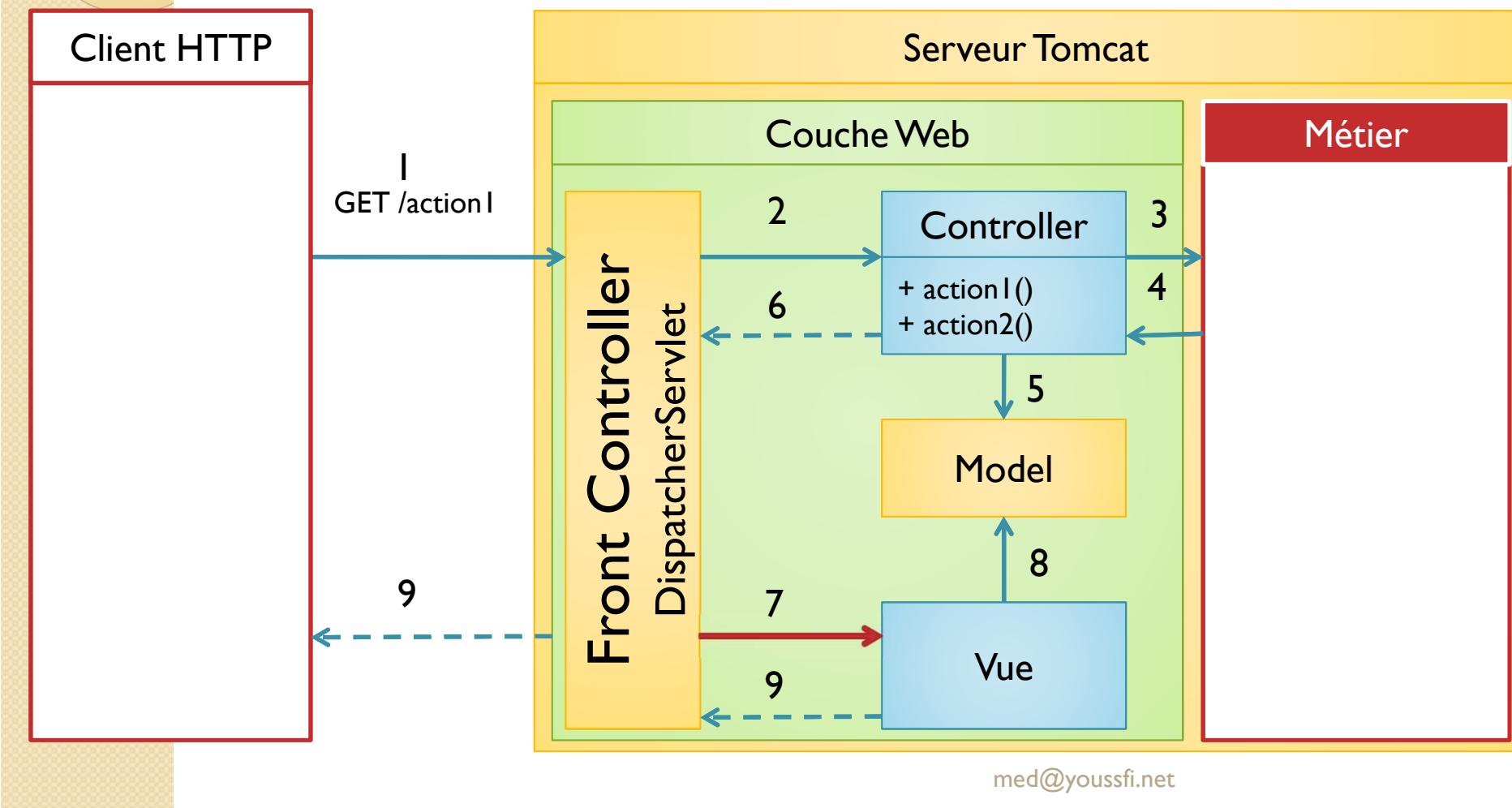
Architecture Spring MVC

6– Le sous contrôleur retourne le nom de la vue et le modèle à DispatcherServlet



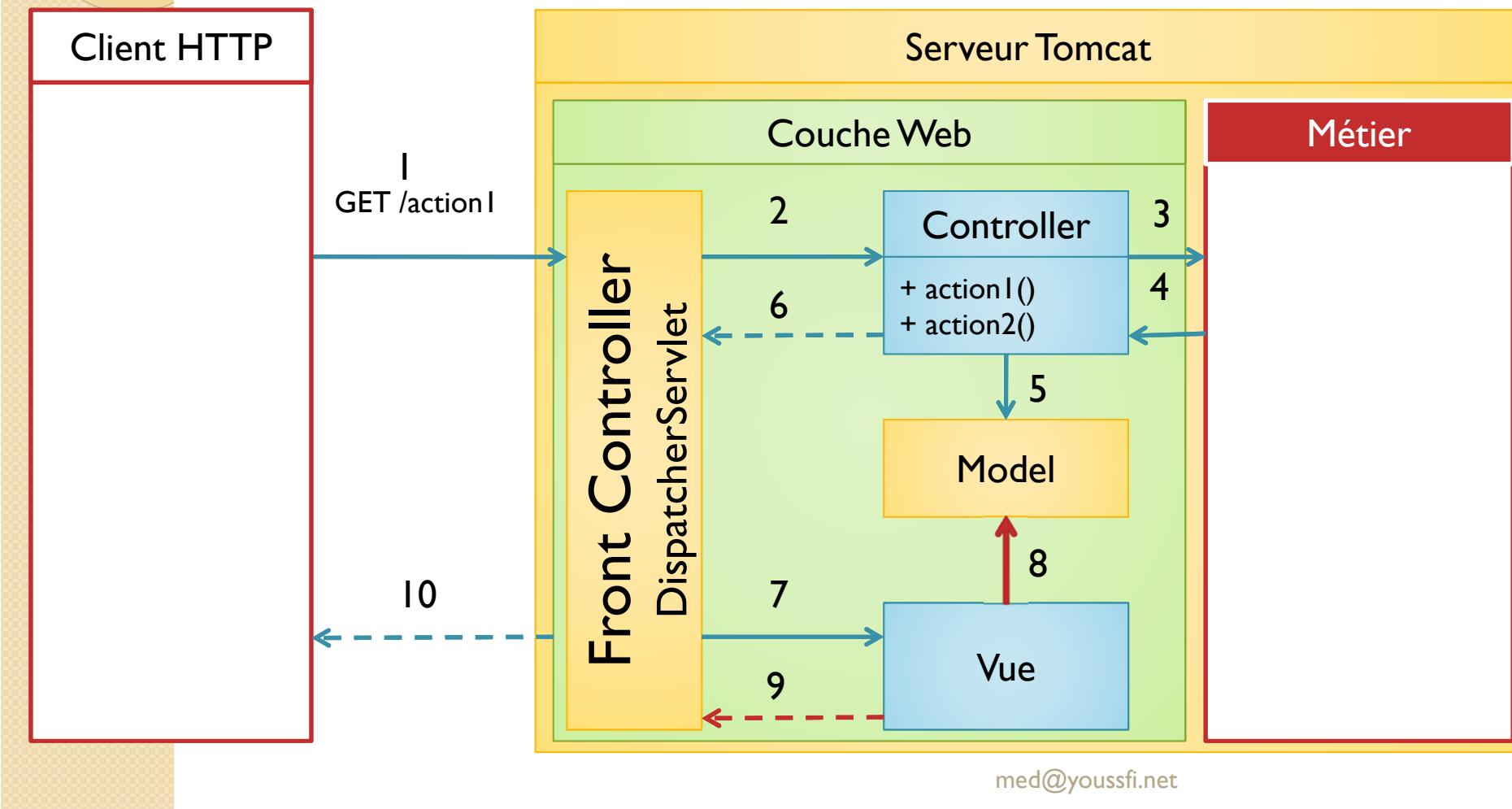
Architecture Spring MVC

7– Le contrôleur frontal DispatcherServlet fait appel à la vue et lui transmet le modèle



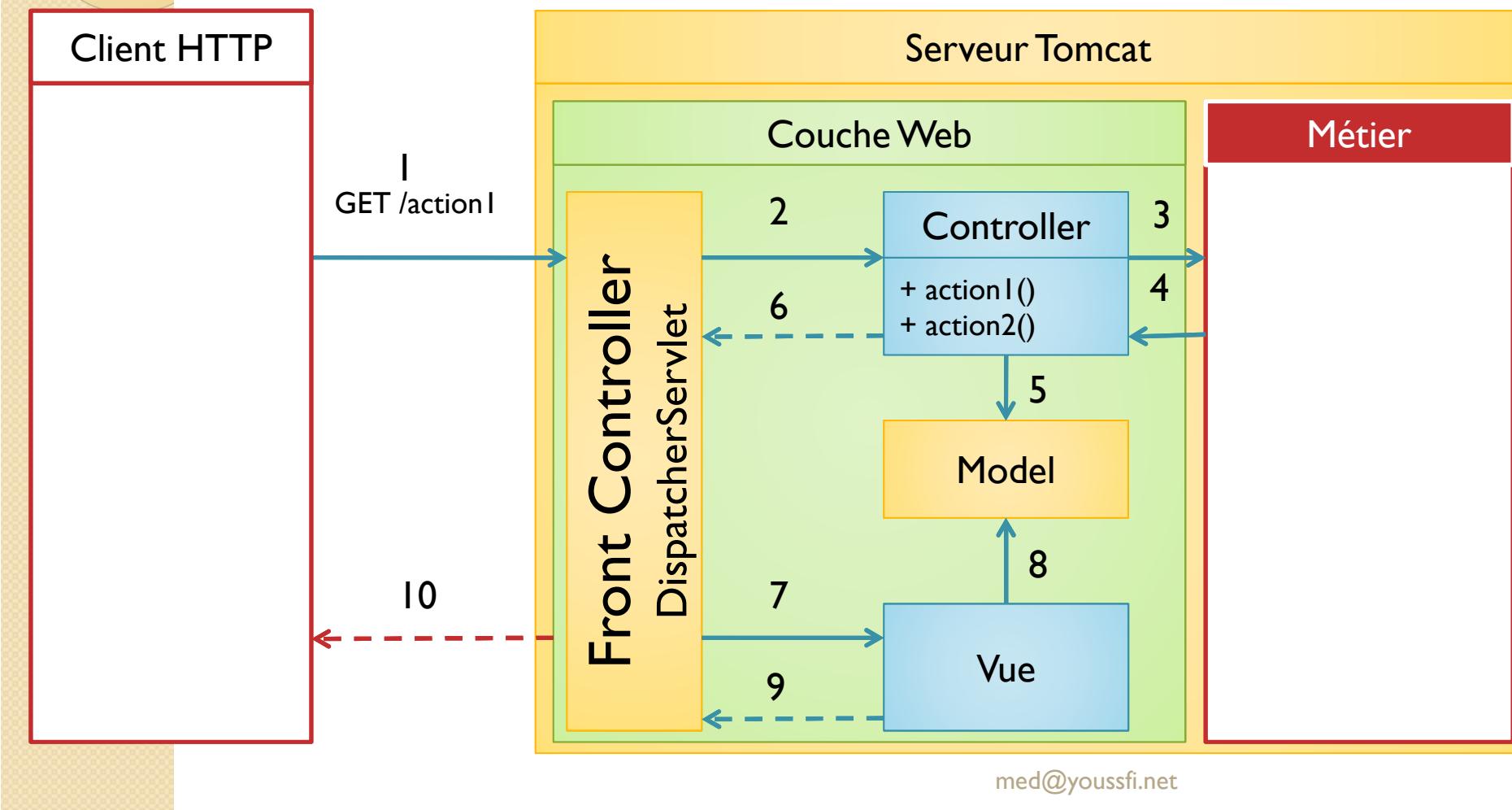
Architecture Spring MVC

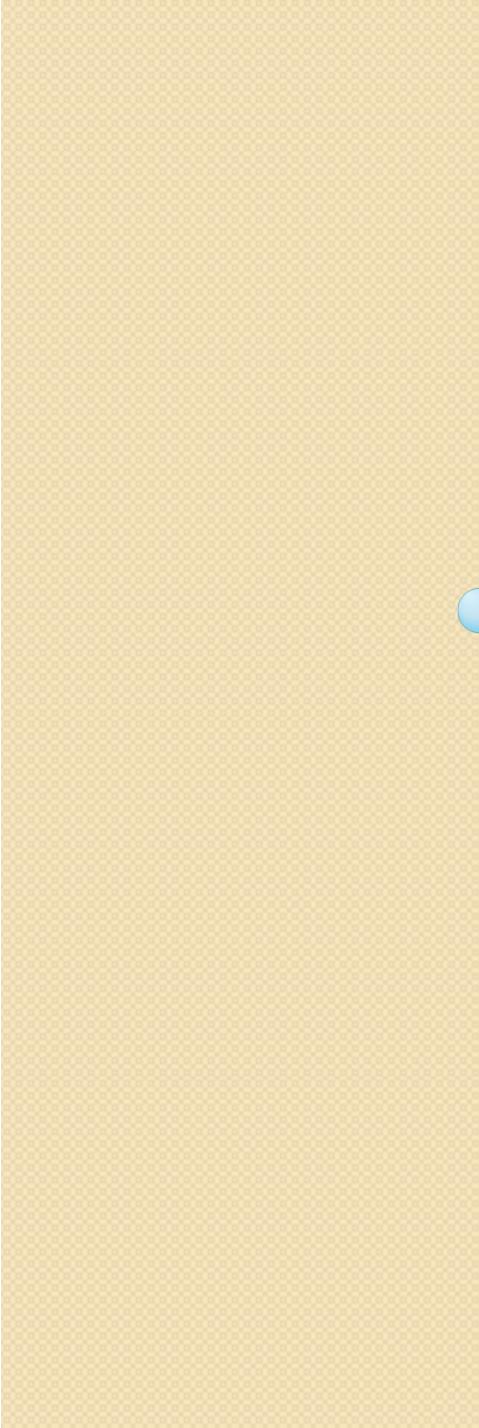
8 et 9 – La vue récupère les résultats à partir du modèle et génère un rendu HTML qui est retourné à DispatcherServlet



Architecture Spring MVC

10– DispatcherServlet envoie la réponse HTTP au client. Cette réponse http contient le code HTML générée par la vue.





- **EXEMPLE
D'APPLICATION
SPRING BOOT**

Premier Exemple d'application

- On souhaite créer une application qui permet de gérer des produits.
- Chaque produit est défini par :
 - Sa référence de type Long
 - Sa désignation de type String
 - Son prix
- L'application permet de :
 - Ajouter de nouveaux produits
 - Consulter les produits
 - Chercher les produits par mot clé
 - Consulter un produit
 - Mettre à jour un produit
 - Supprimer un produit
- Les données sont stockées dans une base de données MySQL
- La couche web respecte MVC côté serveur.

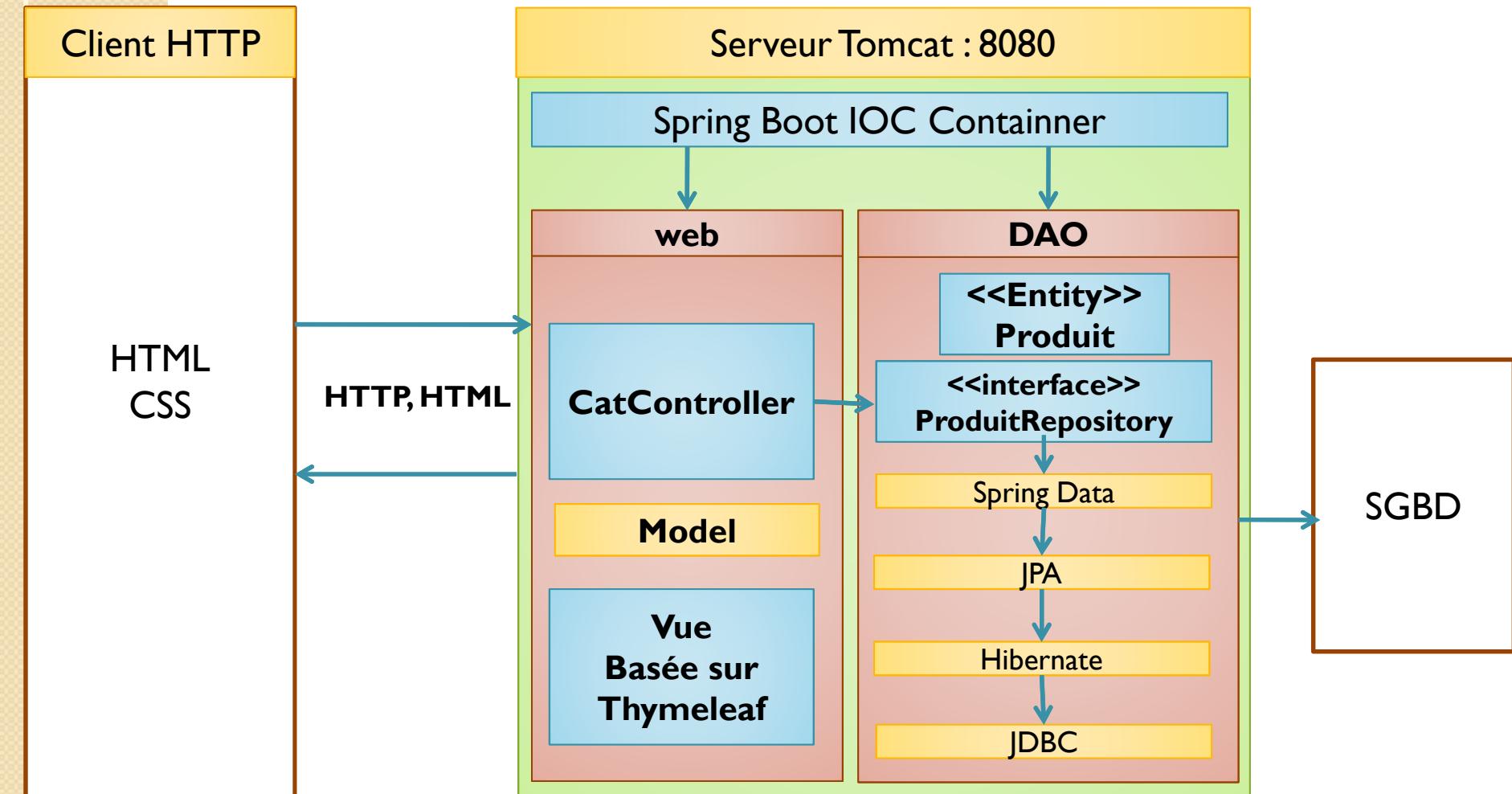
Application

The screenshot shows a web browser window with a blue header bar. The address bar displays "localhost:8080/index.html". The main content area is titled "Catalogue". At the top left, there is a search bar labeled "Mot Clé:" with an input field and a "Chercher" button. Below the search bar is a table with three columns: "REF", "DES", and "PRIX". The table contains five rows of data:

REF	DES	PRIX
7	MPL32x32	2300
8	XRSL32x32	2300
9	XRSL32x32	2300
10	XRSL32x32	2300
11	XRSL32x32	2300

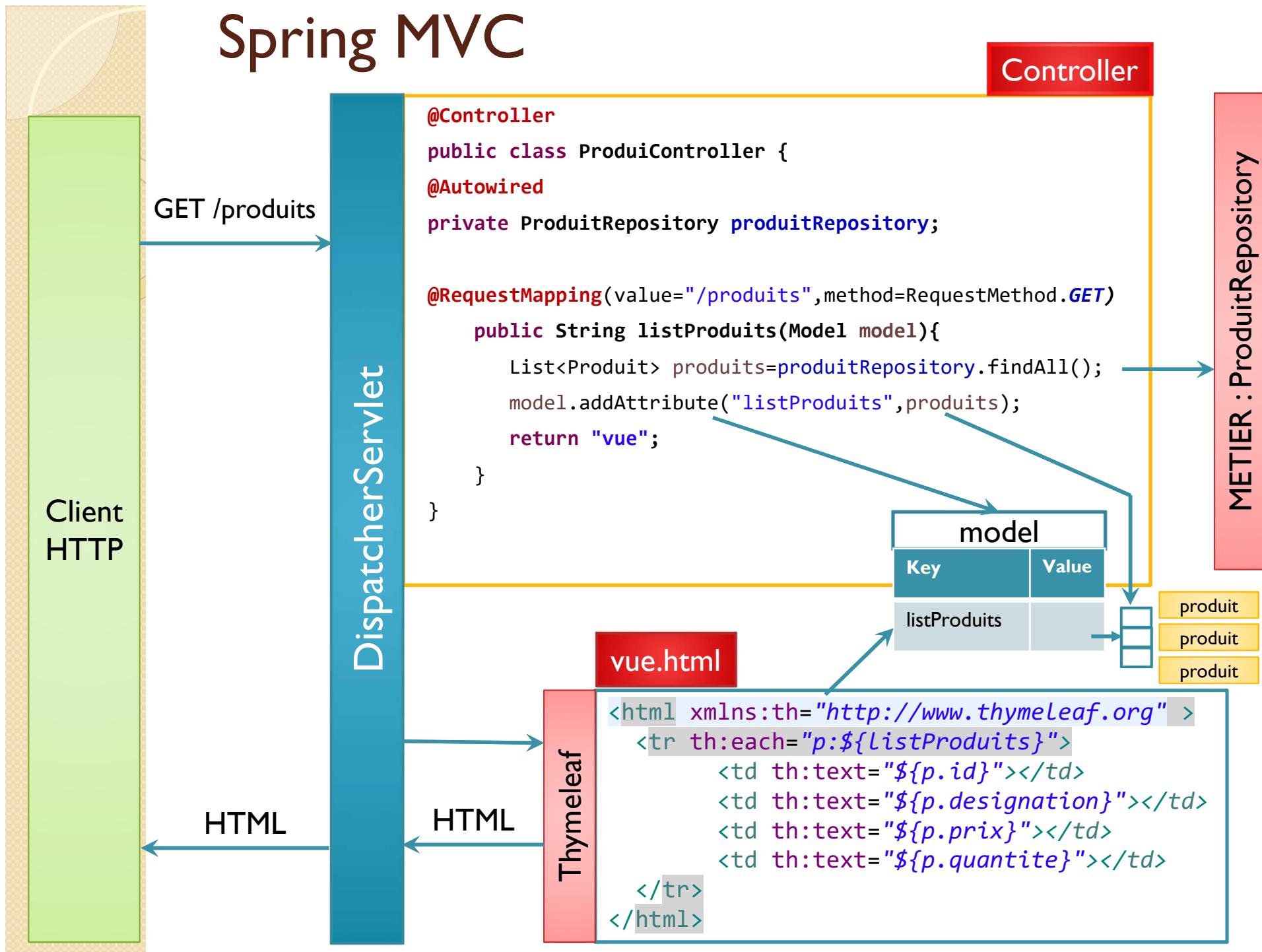
At the bottom of the table, there is a navigation bar with page numbers: 0, 1, 2, 3, 4, 5. The number "1" is highlighted with a blue background.

Spring MVC avec Thymeleaf

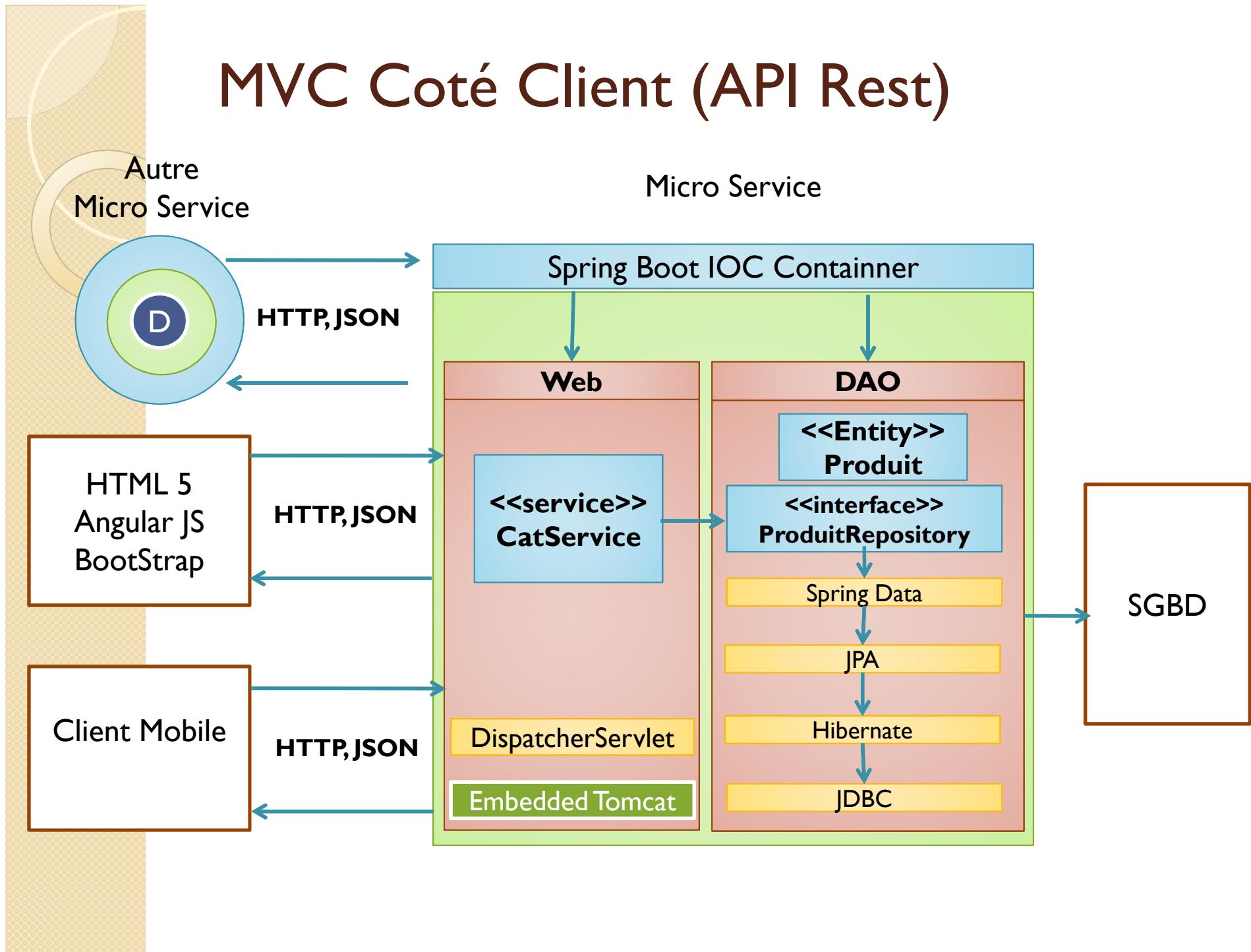


- Mohamed Youssfi
- Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)
- ENSET, Université Hassan II Casablanca, Maroc

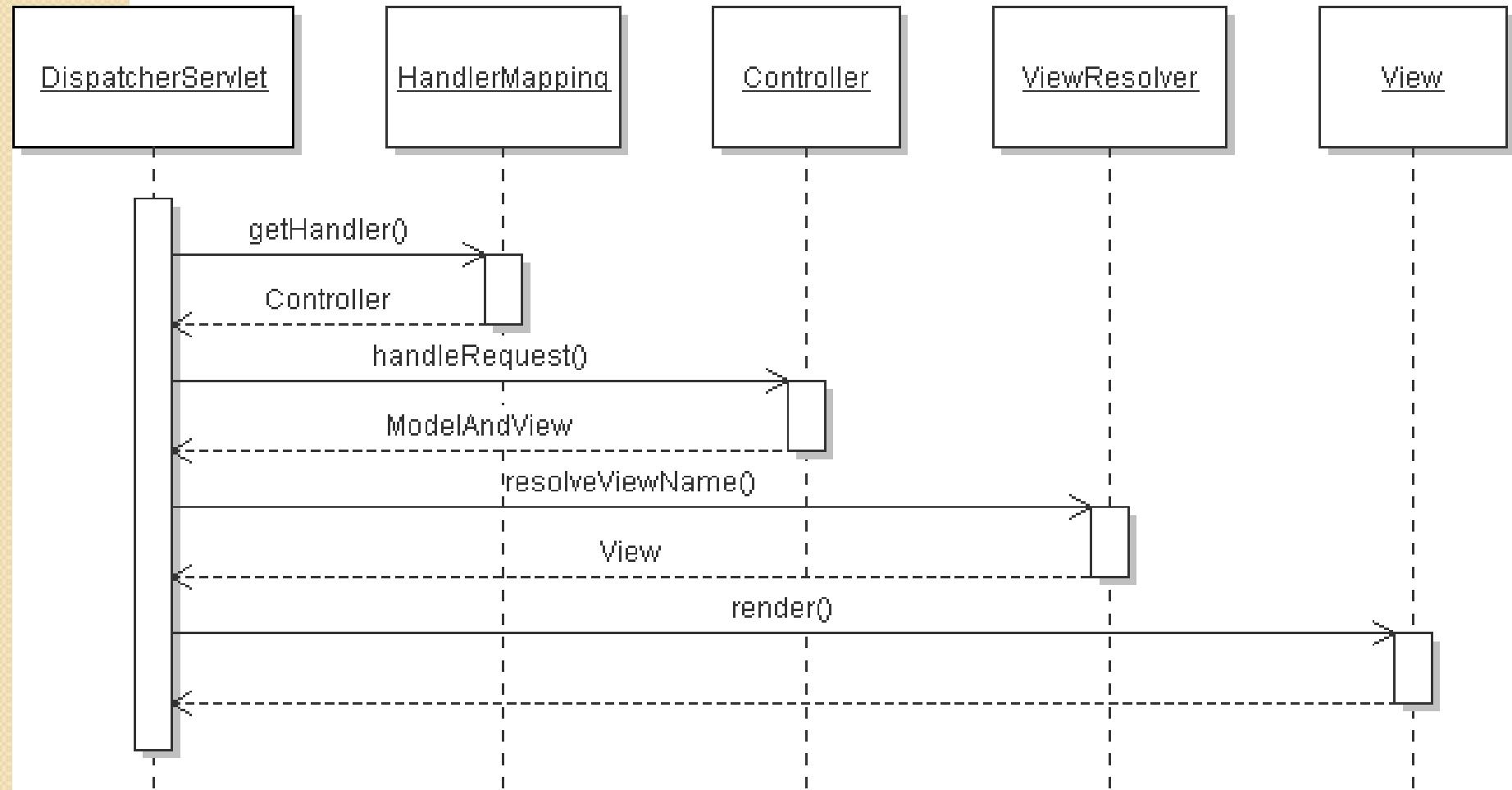
Spring MVC



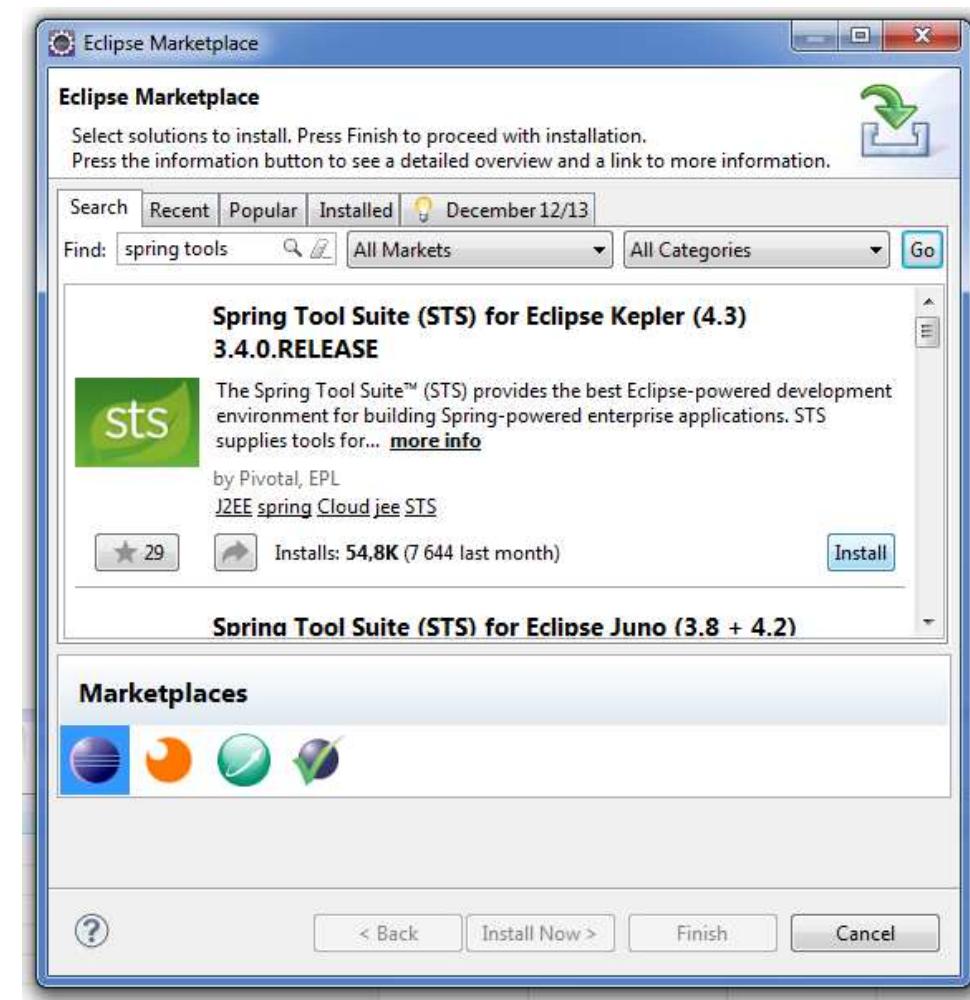
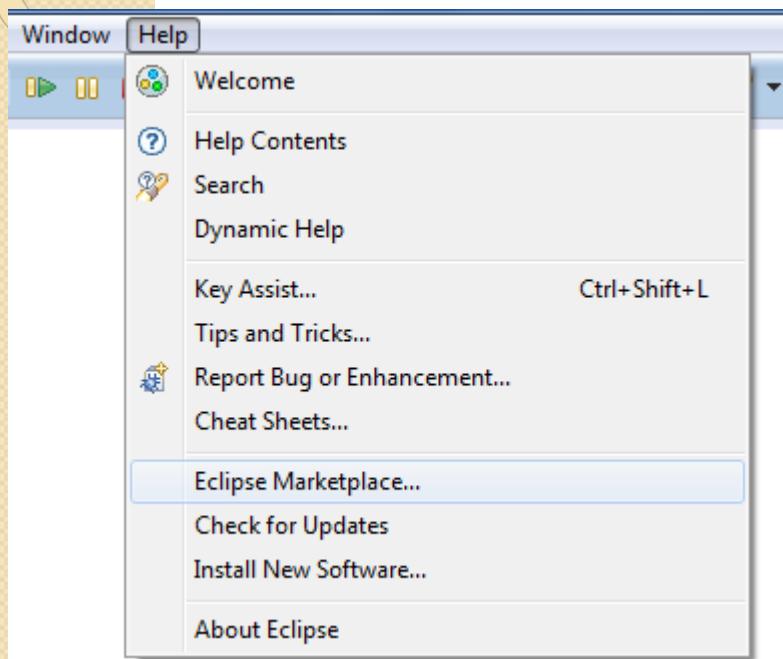
MVC Coté Client (API Rest)



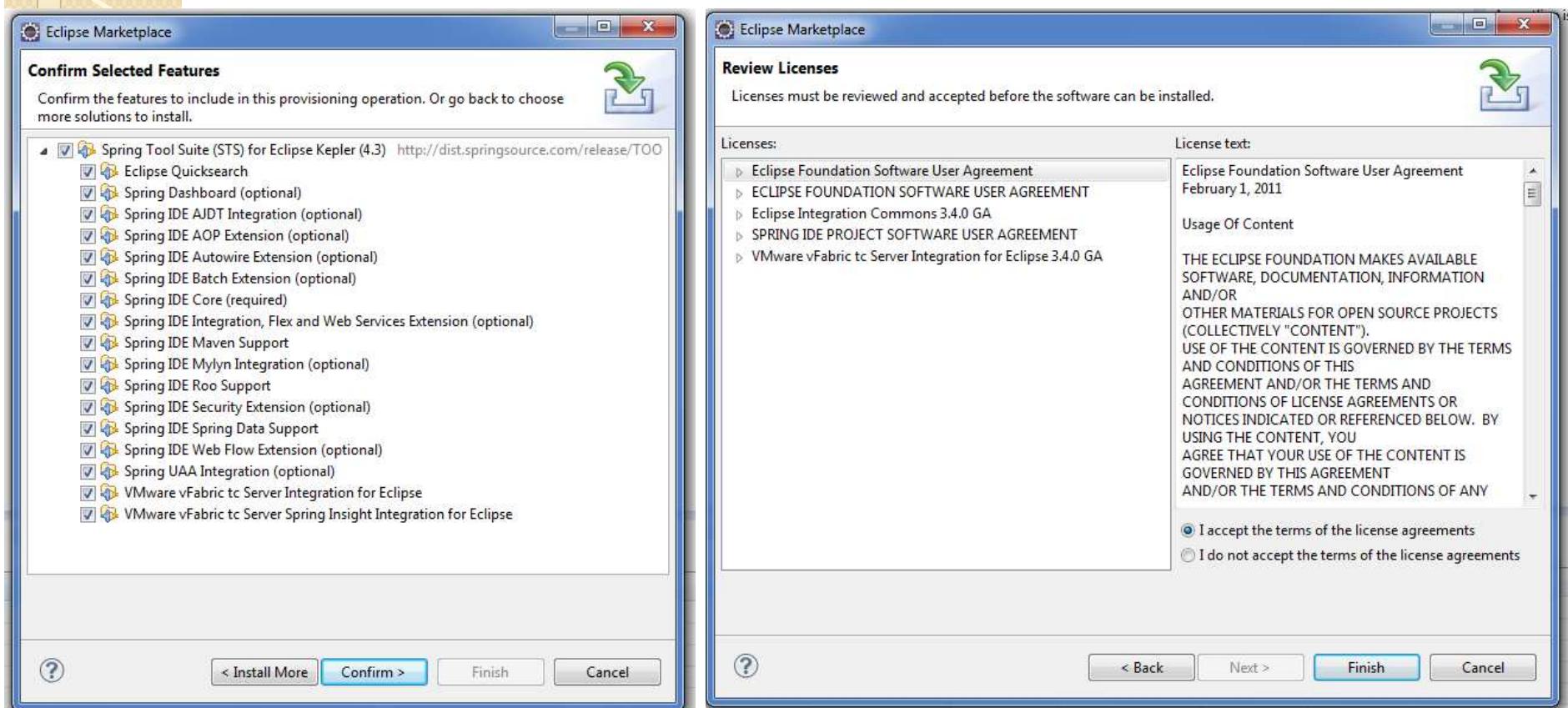
Spring MVC



Installation du plugin : spring tools pour eclipse



Installation du plugin : spring tools pour eclipse

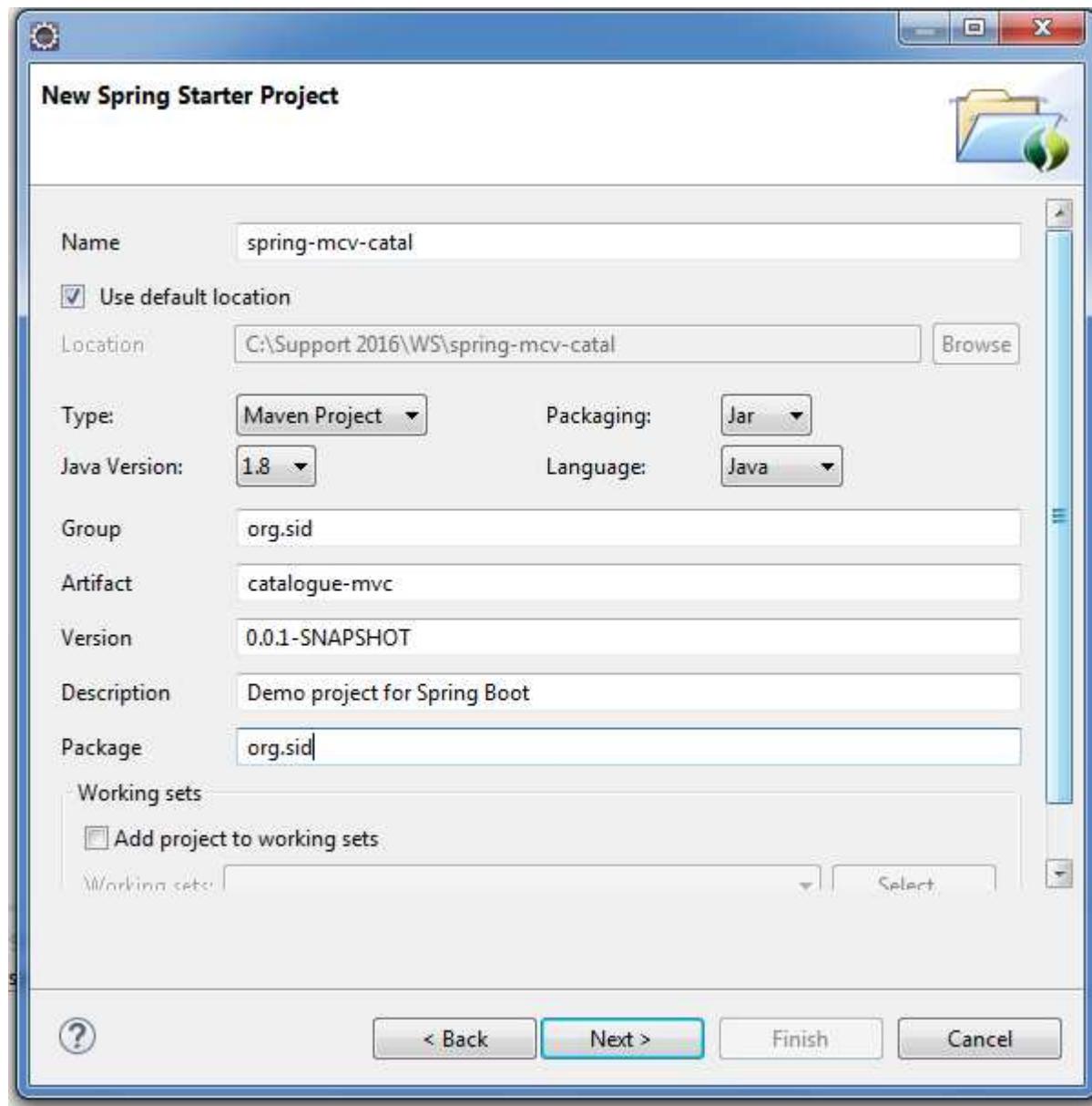




Spring Boot

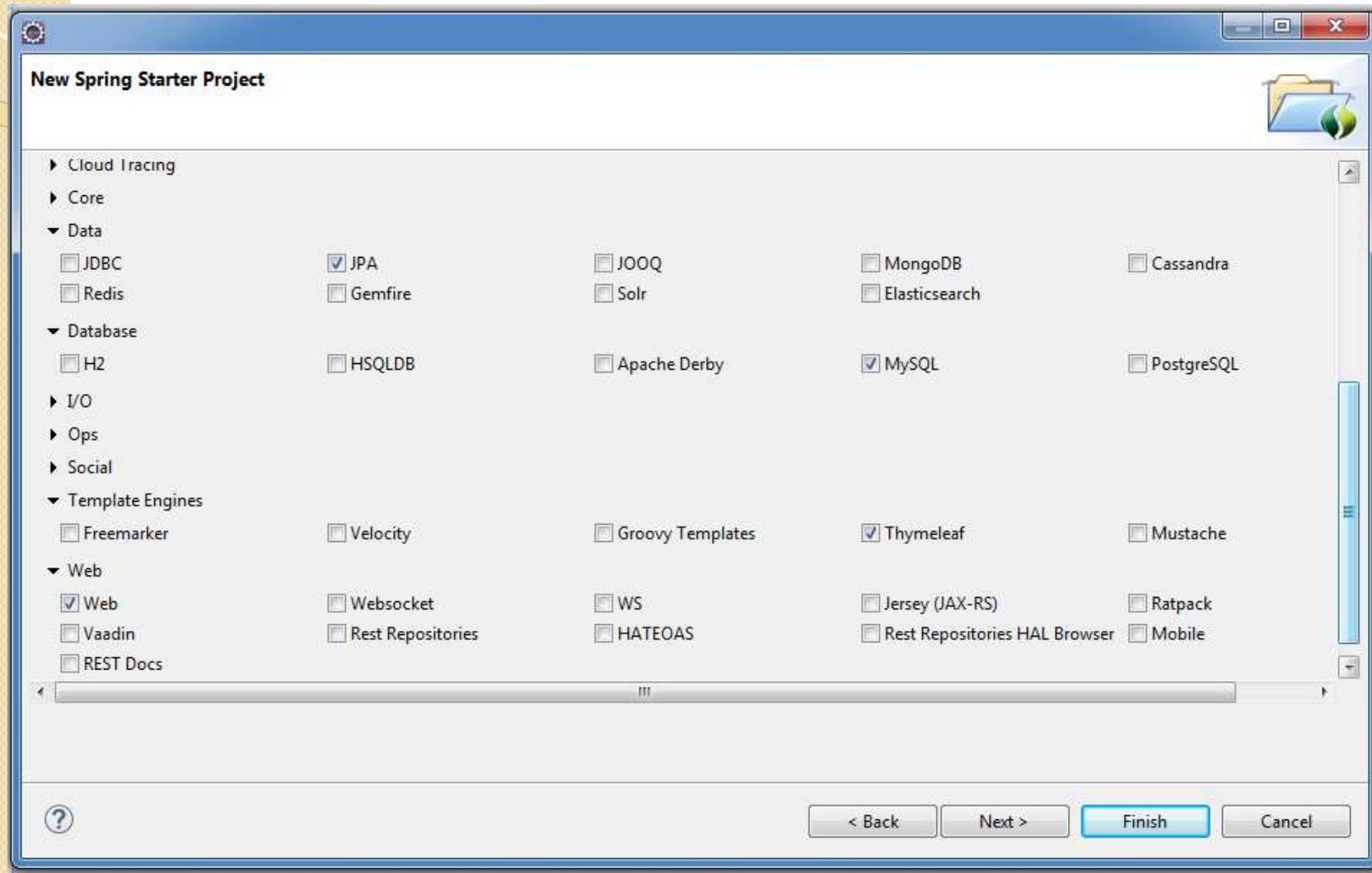
- Spring Boot est un Framework qui permet de créer des applications basées sur des micro services.
- Atouts de Spring Boot :
 - Faciliter le développement d'applications complexes.
 - Faciliter à l'extrême l'injection des dépendances
 - Réduire à l'extrême les fichier de configurations
 - Faciliter la gestion des dépendances Maven.
 - Auto Configuration : la plupart des beans sont créés si le ou les jar(s) adéquats sont dans le classpath.
 - Fournir un conteneur de servlet embarqué (Tomcat, Jetty)
 - Créer une application autonome (jar ou war)

Création d'un projet Spring Starter



Sélectionner les dépendances maven

Web, JPA, MySQL et Thymeleaf

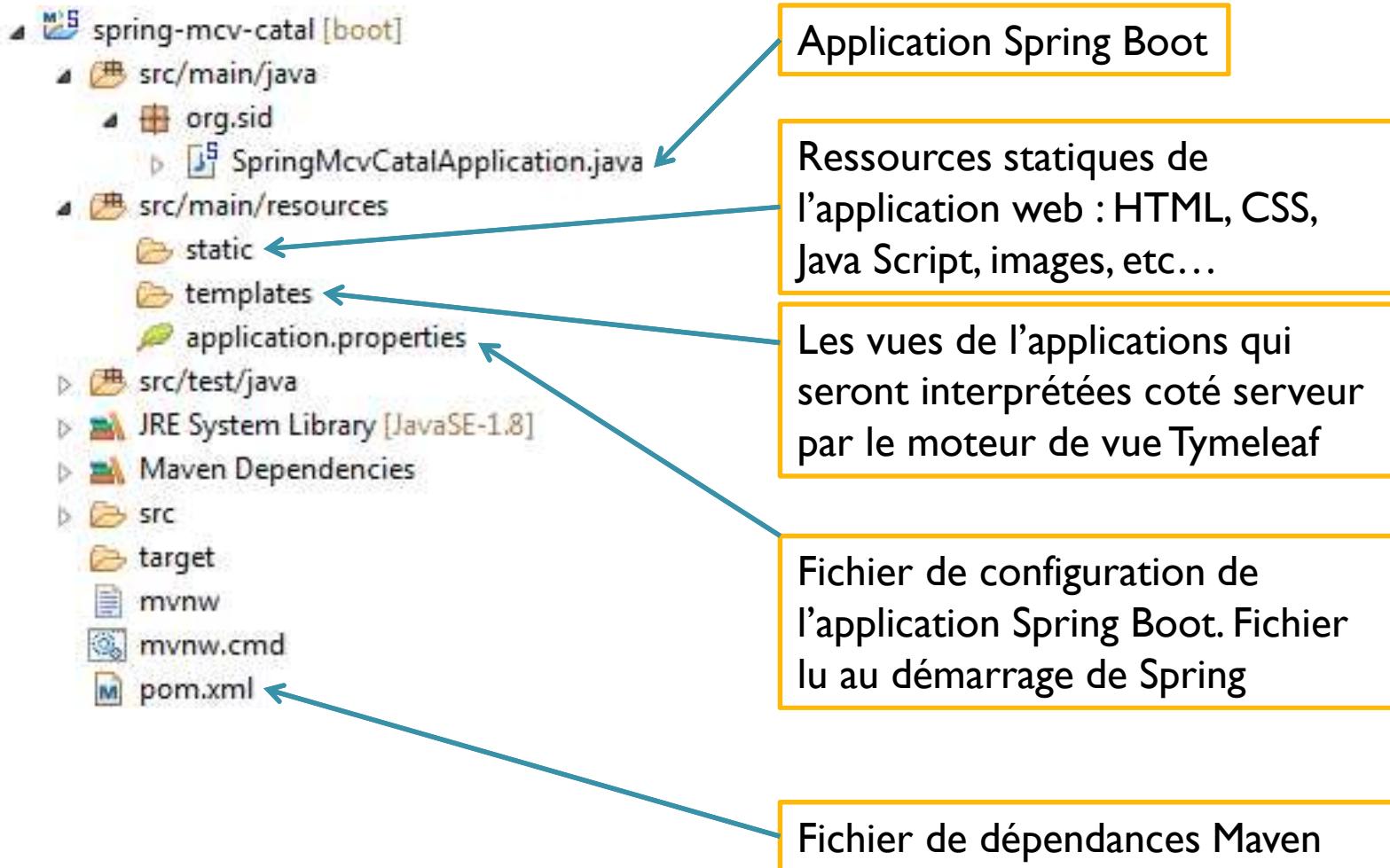




Dépendances maven Spring boot

- La dépendance Web
 - Permet d'ajouter au classpath du projet une version de tomcat embarquée comme conteneur web de l'application.
- La dépendance JPA :
 - Ajouter au classpath du projet :
 - Les dépendances JPA
 - Les dépendances Hibernate
 - Les dépendance Spring Data qui permet de faciliter à l'extrême l'accès aux données de l'application.
- La dépendance MySQL :
 - permet d'ajouter au classpath du projet le pilote JDBC MySQL
- La dépendance Tymeleaf :
 - permet d'ajouter au class path du projet une moteur de template thymeleaf qui permet de faciliter la création des vues et d'éviter de travailler avec JSP et JSTL.

Structure du projet



Application Spring Boot

```
package org.sid;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class SpringMcvCatalApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringMcvCatalApplication.class, args);
    }
}
```

Contenu de pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>org.sid</groupId>
  <artifactId>catalogue-mvc</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>spring-mcv-catal</name>
  <description>Demo project for Spring Boot</description>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.3.0.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>1.8</java.version>
  </properties>
```

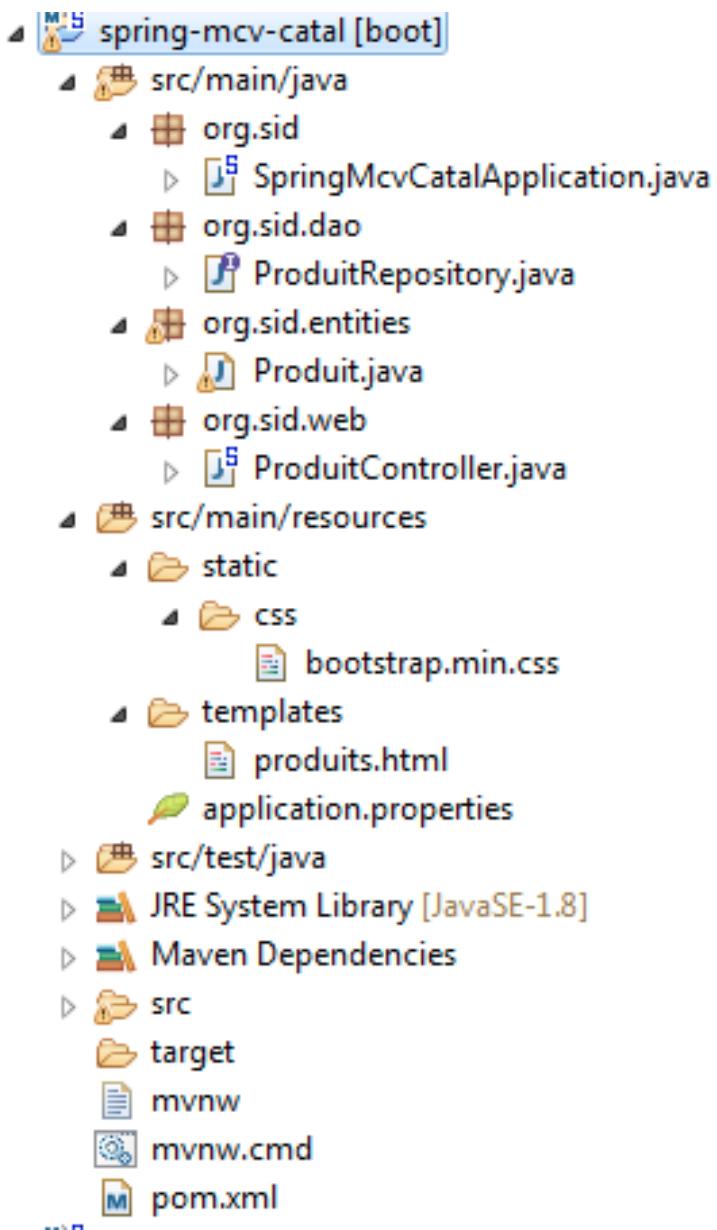
```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Aperçu des dépendances maven

Maven Dependencies	
spring-boot-starter-data-jpa-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-data-jpa\1.3.0.RELEASE\spring-boot-starter-data-jpa-1.3.0.RELEASE.jar	spring-data-commons-1.11.1.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\data\spring-data-commons\1.11.1.RELEASE\spring-data-commons-1.11.1.RELEASE.jar
spring-boot-starter-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter\1.3.0.RELEASE\spring-boot-starter-1.3.0.RELEASE.jar	spring-orm-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\orm\spring-orm\4.2.3.RELEASE\spring-orm-4.2.3.RELEASE.jar
spring-boot-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot\1.3.0.RELEASE\spring-boot-1.3.0.RELEASE.jar	spring-context-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\context\spring-context\4.2.3.RELEASE\spring-context-4.2.3.RELEASE.jar
spring-boot-autoconfigure-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-autoconfigure\1.3.0.RELEASE\spring-boot-autoconfigure-1.3.0.RELEASE.jar	spring-tx-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\tx\spring-tx\4.2.3.RELEASE\spring-tx-4.2.3.RELEASE.jar
spring-boot-starter-logging-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-logging\1.3.0.RELEASE\spring-boot-starter-logging-1.3.0.RELEASE.jar	spring-beans-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\beans\spring-beans\4.2.3.RELEASE\spring-beans-4.2.3.RELEASE.jar
logback-classic-1.1.3.jar - C:\Users\youssf\m2\repository\ch\qos\logback\logback-classic\1.1.3\logback-classic-1.1.3.jar	slf4j-api-1.7.13.jar - C:\Users\youssf\m2\repository\org\slf4j\slf4j-api\1.7.13\slf4j-api-1.7.13.jar
logback-core-1.1.3.jar - C:\Users\youssf\m2\repository\ch\qos\logback\logback-core\1.1.3\logback-core-1.1.3.jar	jcl-over-slf4j-1.7.13.jar - C:\Users\youssf\m2\repository\org\.slf4j\jcl-over-slf4j\1.7.13\jcl-over-slf4j-1.7.13.jar
jul-to-slf4j-1.7.13.jar - C:\Users\youssf\m2\repository\org\apachelog\jul-to-slf4j\1.7.13\jul-to-slf4j-1.7.13.jar	spring-aspects-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\aspects\spring-aspects\4.2.3.RELEASE\spring-aspects-4.2.3.RELEASE.jar
log4j-over-slf4j-1.7.13.jar - C:\Users\youssf\m2\repository\org\log4j\log4j-over-slf4j\1.7.13\log4j-over-slf4j-1.7.13.jar	spring-boot-starter-thymeleaf-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-thymeleaf\1.3.0.RELEASE\spring-boot-starter-thymeleaf-1.3.0.RELEASE.jar
snakeyaml-1.16.jar - C:\Users\youssf\m2\repository\org\yaml\snakeyaml\1.16\snakeyaml-1.16.jar	thymeleaf-spring4-2.1.4.RELEASE.jar - C:\Users\youssf\m2\repository\org\thymeleaf\thymeleaf-spring4\2.1.4.RELEASE\thymeleaf-spring4-2.1.4.RELEASE.jar
spring-boot-starter-aop-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-aop\1.3.0.RELEASE\spring-boot-starter-aop-1.3.0.RELEASE.jar	thymeleaf-2.1.4.RELEASE.jar - C:\Users\youssf\m2\repository\org\thymeleaf\thymeleaf\2.1.4.RELEASE\thymeleaf-2.1.4.RELEASE.jar
spring-aop-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\aop\spring-aop\4.2.3.RELEASE\spring-aop-4.2.3.RELEASE.jar	ognl-3.0.8.jar - C:\Users\youssf\m2\repository\com\fasterxml\ognl\ognl\3.0.8\ognl-3.0.8.jar
aopalliance-1.0.jar - C:\Users\youssf\m2\repository\com\caucho\java\javassist\3.18.1-GA\javassist-3.18.1-GA.jar	unescape-1.1.0.RELEASE.jar - C:\Users\youssf\m2\repository\com\caucho\java\javassist\3.18.1-GA\javassist-3.18.1-GA.jar
aspectjweaver-1.8.7.jar - C:\Users\youssf\m2\repository\org\aspectj\aspectjweaver\1.8.7\aspectjweaver-1.8.7.jar	thymeleaf-layout-dialect-1.3.1.jar - C:\Users\youssf\m2\repository\org\thymeleaf\thymeleaf-layout-dialect\1.3.1\thymeleaf-layout-dialect-1.3.1.jar
spring-boot-starter-jdbc-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-jdbc\1.3.0.RELEASE\spring-boot-starter-jdbc-1.3.0.RELEASE.jar	groovy-2.4.4.jar - C:\Users\youssf\m2\repository\org\groovy\groovy\2.4.4\groovy-2.4.4.jar
tomcat-jdbc-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-jdbc-8.0.28.jar	spring-boot-starter-web-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-web\1.3.0.RELEASE\spring-boot-starter-web-1.3.0.RELEASE.jar
tomcat-juli-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-juli-8.0.28.jar	spring-boot-starter-tomcat-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-tomcat\1.3.0.RELEASE\spring-boot-starter-tomcat-1.3.0.RELEASE.jar
spring-jdbc-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\jdbc\spring-jdbc\4.2.3.RELEASE\spring-jdbc-4.2.3.RELEASE.jar	tomcat-embed-core-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-embed-core-8.0.28.jar
hibernate-entitymanager-4.3.11.Final.jar - C:\Users\youssf\m2\repository\org\hibernate\hibernate-entitymanager\4.3.11.Final\hibernate-entitymanager-4.3.11.Final.jar	tomcat-embed-el-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-embed-el-8.0.28.jar
jboss-logging-3.3.0.Final.jar - C:\Users\youssf\m2\repository\org\jboss\jboss-logging\3.3.0.Final\jboss-logging-3.3.0.Final.jar	tomcat-embed-logging-juli-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-embed-logging-juli-8.0.28.jar
jboss-logging-annotations-1.2.0.Beta1.jar - C:\Users\youssf\m2\repository\org\jboss\jboss-logging\1.2.0.Beta1\jboss-logging-annotations-1.2.0.Beta1.jar	tomcat-embed-websocket-8.0.28.jar - C:\Users\youssf\m2\repository\org\apache\tomcat\javaee\8.0.28\tomcat-embed-websocket-8.0.28.jar
hibernate-core-4.3.11.Final.jar - C:\Users\youssf\m2\repository\org\hibernate\hibernate-core\4.3.11.Final\hibernate-core-4.3.11.Final.jar	spring-boot-starter-validation-1.3.0.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\boot\spring-boot-starter-validation\1.3.0.RELEASE\spring-boot-starter-validation-1.3.0.RELEASE.jar
antlr-2.7.7.jar - C:\Users\youssf\m2\repository\org\antlr\antlr\2.7.7\antlr-2.7.7.jar	hibernate-validator-5.2.2.Final.jar - C:\Users\youssf\m2\repository\org\hibernate\validator\5.2.2.Final\hibernate-validator-5.2.2.Final.jar
jandex-1.1.0.Final.jar - C:\Users\youssf\m2\repository\org\javassist\jandex\1.1.0.Final\jandex-1.1.0.Final.jar	validation-api-1.1.0.Final.jar - C:\Users\youssf\m2\repository\org\javax\validation\validation-api\1.1.0.Final\validation-api-1.1.0.Final.jar
dom4j-1.6.1.jar - C:\Users\youssf\m2\repository\org\dom4j\dom4j\1.6.1\dom4j-1.6.1.jar	classmate-1.1.0.jar - C:\Users\youssf\m2\repository\org\pickett\classmate\1.1.0\classmate-1.1.0.jar
xml-apis-1.0.b2.jar - C:\Users\youssf\m2\repository\org\xerces\xml-apis\1.0.b2\xml-apis-1.0.b2.jar	jackson-databind-2.6.3.jar - C:\Users\youssf\m2\repository\com\fasterxml\jackson\databind\jackson-databind\2.6.3\jackson-databind-2.6.3.jar
hibernate-commons-annotations-4.0.5.Final.jar - C:\Users\youssf\m2\repository\org\hibernate\hibernate-commons-annotations\4.0.5.Final\hibernate-commons-annotations-4.0.5.Final.jar	jackson-annotations-2.6.3.jar - C:\Users\youssf\m2\repository\com\fasterxml\jackson\annotation\jackson-annotations\2.6.3\jackson-annotations-2.6.3.jar
hibernate-jpa-2.1-api-1.0.0.Final.jar - C:\Users\youssf\m2\repository\org\hibernate\hibernate-jpa-2.1-api\1.0.0.Final\hibernate-jpa-2.1-api-1.0.0.Final.jar	jackson-core-2.6.3.jar - C:\Users\youssf\m2\repository\com\fasterxml\jackson\core\jackson-core\2.6.3\jackson-core-2.6.3.jar
javassist-3.18.1-GA.jar - C:\Users\youssf\m2\repository\org\javassist\javassist\3.18.1-GA\javassist-3.18.1-GA.jar	spring-web-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\web\spring-web\4.2.3.RELEASE\spring-web-4.2.3.RELEASE.jar
javax.transaction-api-1.2.jar - C:\Users\youssf\m2\repository\javax\transaction\javax.transaction-api\1.2\javax.transaction-api-1.2.jar	spring-webmvc-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\webmvc\spring-webmvc\4.2.3.RELEASE\spring-webmvc-4.2.3.RELEASE.jar
spring-data-jpa-1.9.1.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\data\spring-data-jpa\1.9.1.RELEASE\spring-data-jpa-1.9.1.RELEASE.jar	spring-expression-4.2.3.RELEASE.jar - C:\Users\youssf\m2\repository\org\springframework\expression\spring-expression\4.2.3.RELEASE\spring-expression-4.2.3.RELEASE.jar

Structure du projet



med@youssf.net

Entité Produit

```
package org.sid.entities;

import java.io.Serializable; import javax.persistence.Entity;
import javax.persistence.GeneratedValue; import javax.persistence.Id;
import javax.validation.constraints.DecimalMin;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
public class Produit implements Serializable {

    @Id @GeneratedValue
    private Long idProduit;
    @NotNull
    @Size(min=5,max=12)
    private String designation;
    @DecimalMin(value="100")
    private double prix;
    // getters et setters
    // Constructeur par défaut
    // Constructeur avec params
}
```

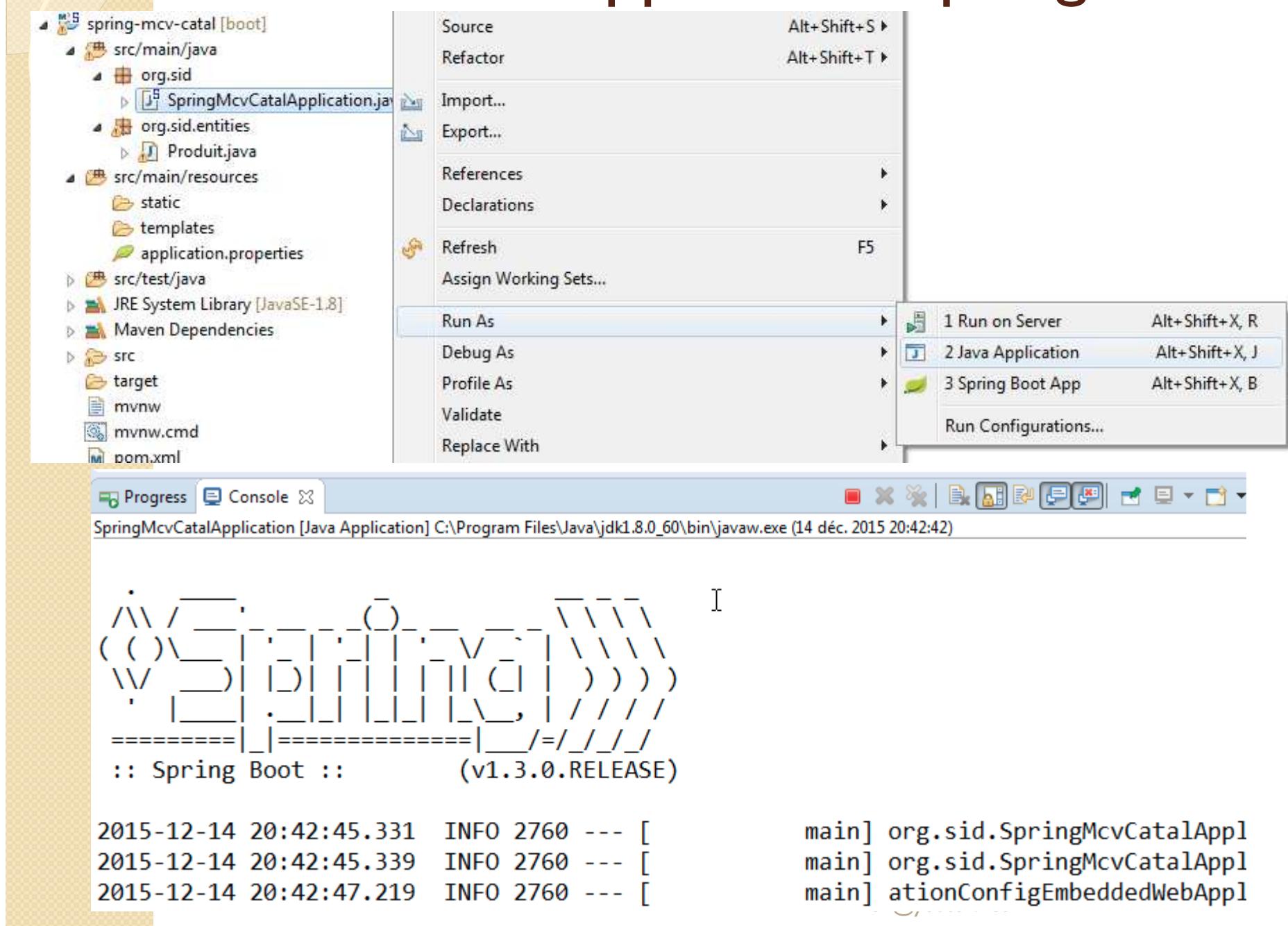
Déployer le data source : application.properties

```
spring.datasource.url = jdbc:mysql://localhost:3306/db_cat_mvc  
spring.datasource.username = root  
spring.datasource.password =  
spring.datasource.driverClassName = com.mysql.jdbc.Driver  
spring.jpa.show-sql = true  
spring.jpa.hibernate.ddl-auto = update  
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
```

Créer la base de données



Démarrer l'application Spring Boot



Vérifier si la table produits a été bien générée

```
Progress Console ×
SpringMvcCatApplication [Java Application] C:\Program Files\Java\jdk1.8.0_60\bin\javaw.exe (14 déc. 2015 20:42:42)
ansiatorfactory : HHH000397: Using ASIQueryTranslatorFactory
2ddl.SchemaUpdate : HHH000228: Running hbm2ddl schema update
2ddl.SchemaUpdate : HHH000102: Fetching database metadata
2ddl.SchemaUpdate : HHH000396: Updating schema
ata : HHH000262: Table not found: produit
ata : HHH000262: Table not found: produit
ata : HHH000262: Table not found: produit
2ddl.SchemaUpdate : HHH000232: Schema update complete
```

The screenshot shows the phpMyAdmin interface for the 'produit' table in the 'db_cat_mvc' database. The table has three columns: 'id_produit' (bigint(20)), 'designation' (varchar(255)), and 'prix' (double). The 'id_produit' column is set as the primary key (PRIMARY, BTREE index). The table currently contains no data.

Colonne	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
id_produit	bigint(20)			Non	Aucun	AUTO_INCREMENT	
designation	varchar(255)	latin1_swedish_ci		Oui	NULL		
prix	double			Non	Aucun		

Index:

Action	Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
	PRIMARY	BTREE	Oui	Non	id_produit	0	A		

Couche DAO avec Spring data

- Une seule interface à créer:

```
package org.sid.dao;
import org.sid.entities.Produit;
import org.springframework.data.jpa.repository.JpaRepository;
public interface ProduitRepository extends JpaRepository<Produit, Long> {

}
```

Tester la couche DAO

```
package org.sid;

import java.util.List; import org.sid.dao.ProduitRepository;
import org.sid.entities.Produit; import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class SpringMcvcatalApplication {

    public static void main(String[] args) {
        ApplicationContext ctx=SpringApplication.run(SpringMcvcatalApplication.class, args);
        ProduitRepository dao=ctx.getBean(ProduitRepository.class);
        // Ajouter quelques produits
        dao.save(new Produit("TR342",540));           dao.save(new Produit("HR 4378",540));
        dao.save(new Produit("AXL 123",540));
        // Consulter tous les produits
        System.out.println("-----");
        List<Produit> prods=dao.findAll();
        for(Produit p:prods){  System.out.println(p.getDesignation()+"--"+p.getPrix());   }
        // Consulter un produit
        System.out.println("-----");
        Produit p=dao.findOne(2L);
        System.out.println(p.getDesignation()+"--"+p.getPrix());
    }
}
```

Exécution

```
Hibernate: insert into produit (designation, prix) values (?, ?)
Hibernate: insert into produit (designation, prix) values (?, ?)
Hibernate: insert into produit (designation, prix) values (?, ?)
-----
Hibernate: select produit0_.id_produit as id_produ1_0_, produit0_.de
TR342--540.0
HR 4378--540.0
AXL 123--540.0
-----
Hibernate: select produit0_.id_produit as id_produ1_0_0_, produit0_
HR 4378--540.0
```

	←↑→	id_produit	designation	prix
		1	TR342	540
		2	HR 4378	540
		3	AXL 123	540

Ajouter des méthodes à l'interface JPARepository

- Une méthode pour consulter les produits sachant la désignation
- Une méthode qui permet de retourner une page de produits ont la désignation contient un mot clé:

```
package org.sid.dao;
import java.util.List;
import org.sid.entities.Produit;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface ProduitRepository extends JpaRepository<Produit, Long> {
    public List<Produit> findByDesignation(String designation);
    @Query("select p from Produit p where p.designation like :x")
    public Page<Produit> chercherProduits(@Param("x")String mc,Pageable pageable);
}
```

Tester les méthodes

```
System.out.println("-----");
Page<Produit> pageProduits=dao.chercherProduits("%R%", new PageRequest(0, 2));
System.out.println("Page numéro:"+pageProduits.getNumber());
System.out.println("Nombre de produits:"+pageProduits.getNumberOfElements());
System.out.println("Page numéro:"+pageProduits.getTotalPages());
System.out.println("Total produits:"+pageProduits.getTotalElements());
for(Produit pr:pageProduits.getContent()){
System.out.println(pr.getDesignation());
}
```

Page numéro:0
Nombre de produits:2
Page numéro:3
Total produits:6
TR342
HR 4378

Couche Web

A screenshot of a web browser window titled "Insert title here". The address bar shows "localhost:8080/chercher?mc=H". The main content area displays a search result table with columns: ID, DES, and PRIX. The results are:

ID	DES	PRIX
2	HLX	54300.0
4	HP870	3400.0
6	HLX	54300.0
8	HP870	3400.0
10	HLX	54300.0

Below the table are navigation links: 0, 1, 2.

A screenshot of a web browser window titled "Saisie d'un produit". The address bar shows "localhost:8080/form". The form has fields for "Désignation" and "Prix", both currently empty. A "Save" button is at the bottom.

A screenshot of a web browser window titled "Saisie d'un produit". The address bar shows "localhost:8080/saveProduit". The "Désignation" field contains "aaaaaa" and the "Prix" field contains "0.0". Error messages are displayed: "la taille doit être entre 5 et 12" for the designation and "doit être supérieur ou égal à 100" for the price. A "Save" button is at the bottom.

A screenshot of a web browser window titled "Saisie d'un produit". The address bar shows "localhost:8080/saveProduit". The message "Confirmation" is displayed, showing the saved data: "ID: 50", "Désignation: prod78", and "Prix: 400.0".

Contrôleur

```
package org.sid.web;
import javax.validation.Valid;
import org.sid.dao.ProduitRepository;
import org.sid.entities.Produit;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class ProduitController {
    @Autowired
    private ProduitRepository produitRepository;
```

Contrôleur

```
@RequestMapping(value="/chercher",method=RequestMethod.GET)
public String chercher(Model model,
    @RequestParam(name="mc",defaultValue "")String motCle,
    @RequestParam(name="page",defaultValue="0")int page){

    Page<Produit> pageProduits=
        produitRepository.chercherProduits("%"+motCle+"%", new PageRequest(page, 5));
    model.addAttribute("pageProduit", pageProduits);
    model.addAttribute("pageCourante", page);
    model.addAttribute("mc",motCle);
    int[] pages=new int[pageProduits.getTotalPages()];
    for(int i=0;i<pages.length;i++)
        pages[i]=i;
    model.addAttribute("pages", pages);
    return "produits";
}
```

Contrôleur

```
@RequestMapping(value="/form")
public String formProduit(Model model){
    model.addAttribute("produit", new Produit());
    return "formProduit";
}

@RequestMapping(value="/saveProduit",method=RequestMethod.POST)
public String save(Model model,@Valid Produit p, BindingResult
bindingResult){
    if(bindingResult.hasErrors()){
        return "formProduit";
    }
    produitRepository.save(p);
    model.addAttribute("produit", p);
    return "confirmation";
}
}
```

Vue : produits.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Insert title here</title>
<link rel="stylesheet" type="text/css" href="../static/css/bootstrap.min.css"
th:href="@{css/bootstrap.min.css}"/>
<link rel="stylesheet" type="text/css" href="../static/css/myStyle.css"
th:href="@{css/myStyle.css}"/>
</head>
<body>
<div class="container spacer">
<form action="chercher?page=0" method="GET">
<label>Mot Clé:</label>
<input type="text" name="mc" th:value="${mc}"/>
<input type="submit" value="Chercher"/>
</form>
</div>
```

Vue : produits.html

```
<div class="container">
  <table class="table table-striped">
    <thead>
      <tr>
        <th>ID</th><th>DES</th><th>PRIX</th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="p:${pageProduit.content}">
        <td th:text="${p.idProduit}"></td>
        <td th:text="${p.designation}"></td>
        <td th:text="${p.prix}"></td>
      </tr>
    </tbody>
  </table>
</div>
<div class="container">
  <ul class="nav nav-pills">
    <li th:each="p:${pages}" th:class="${pageCourante==p}? 'active': ''">
      <a th:href="@{chercher(page=${p},mc=${mc})}" th:text="${p}"></a>
    </li>
  </ul>
</div>
</body>
</html>
```

Vue : formProduit.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Saisie d'un produit</title>
<link rel="stylesheet" type="text/css"
      href="../static/css/bootstrap.min.css"
      th:href="@{css/bootstrap.min.css}"/>
<link rel="stylesheet" type="text/css" href="../static/css/myStyle.css"
      th:href="@{css/myStyle.css}"/>
</head>
```

Vue : formProduit.html

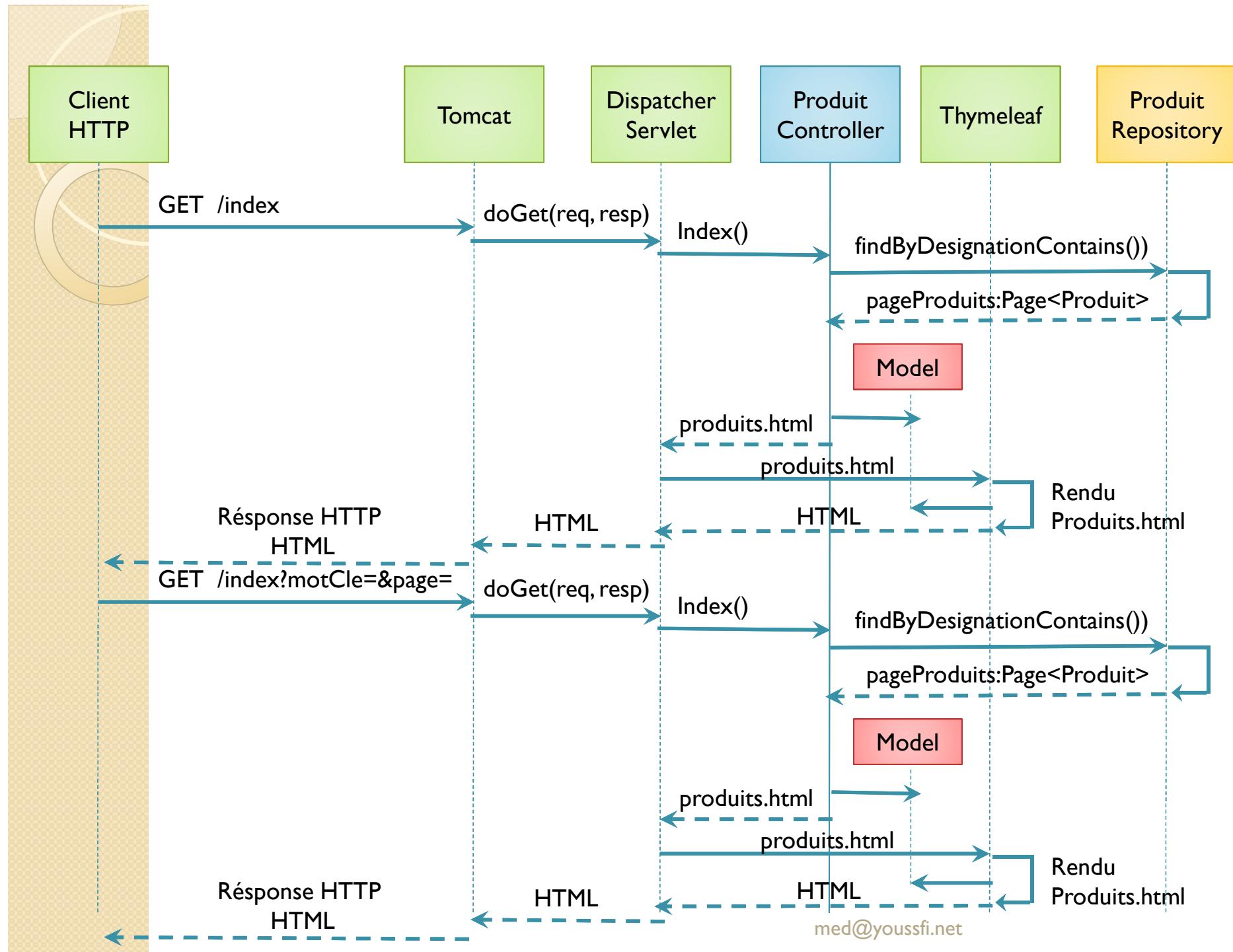
```
<body>
  <div class="col-md-6 col-sm-6 col-xs-12">
    <form th:action="saveProduit" method="post">
      <div th:object="${produit}">
        <div class="form-group">
          <label class="control-label">Désignation:</label>
          <input class="form-control" type="text" name="designation"
            th:value="*{designation}"/>
          <span class="error" th:errors="*{designation}"></span>
        </div>
        <div class="form-group">
          <label class="control-label">PrixA:</label>
          <input class="form-control" type="text" name="prix" th:value="*{prix}"/>
          <span class="error" th:errors="*{prix}"></span>
        </div>
        <div>
          <input class="btn btn-primary" type="submit" value="Save"/>
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

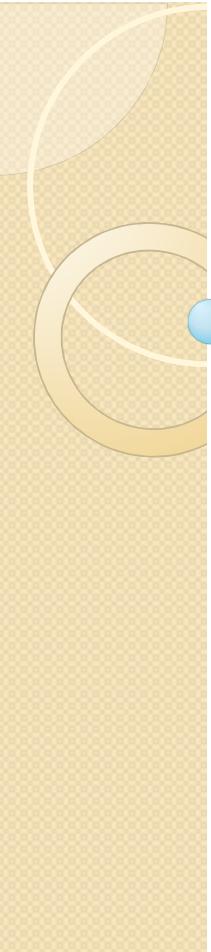
Vue : confirmation.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8"/>
<title>Saisie d'un produit</title>
<link rel="stylesheet" type="text/css" href="../static/css/bootstrap.min.css"
th:href="@{css/bootstrap.min.css}"/>
<link rel="stylesheet" type="text/css" href="../static/css/myStyle.css"
th:href="@{css/myStyle.css}"/>
</head>
```

Vue : confirmation.html

```
<body>
<div class="col-md-6 col-sm-6 col-xs-12">
<div class="panel panel-info spacer">
    <div class="panel-heading">Confirmation </div>
    <div class="panel-body">
        <div class="form-group">
            <label class="control-label">ID:</label>
            <label class="control-label" th:text="${produit.idProduit}"></label>
        </div>
        <div class="form-group">
            <label class="control-label">Désignation:</label>
            <label class="control-label" th:text="${produit.designation}"></label>
        </div>
        <div class="form-group">
            <label class="control-label">Prix:</label>
            <label class="control-label" th:text="${produit.prix}"></label>
        </div>
    </div>
</div>
</div>
</body>
</html>
```





Développement Web JEE Spring MVC (Suite)

Mohamed Youssfi

Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)
ENSET, Université Hassan II Casablanca, Maroc

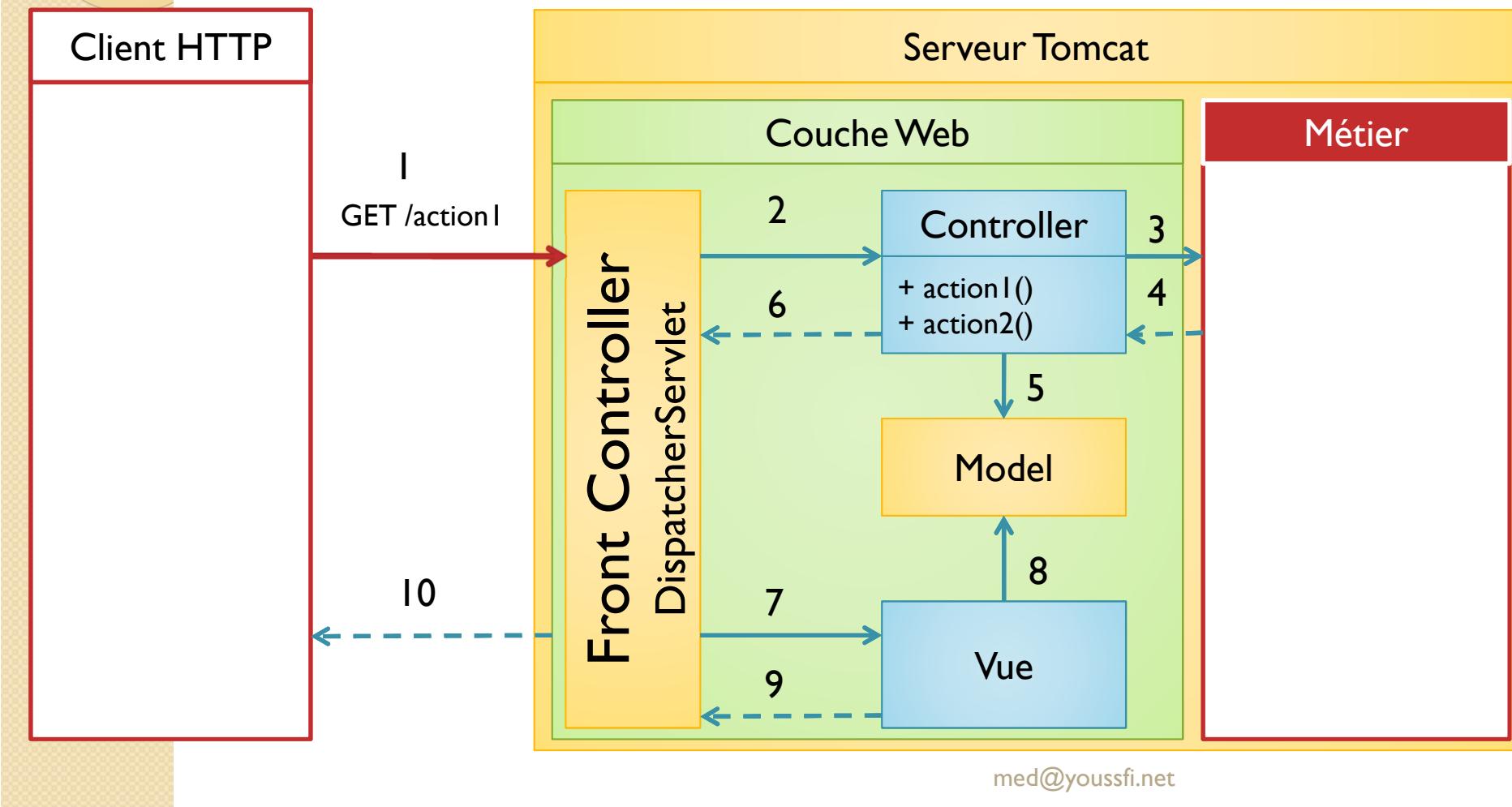
Email : med@youssfi.net

Supports de cours : <http://fr.slideshare.net/mohamedyoussfi9>

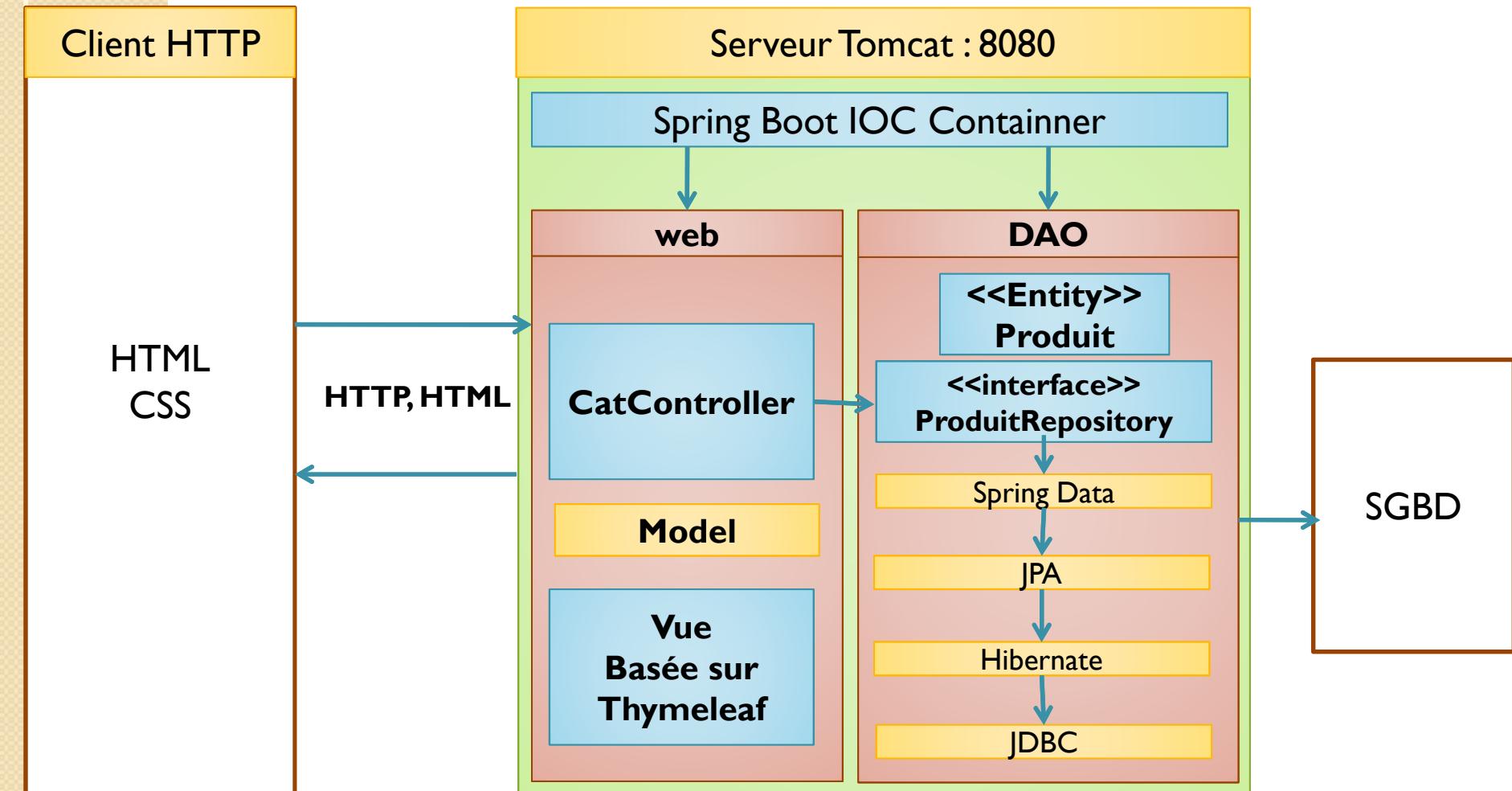
Chaîne vidéo : <http://youtube.com/mohamedYoussfi>

Architecture Spring MVC

I – Le client envoie une requête HTTP de type GET ou POST

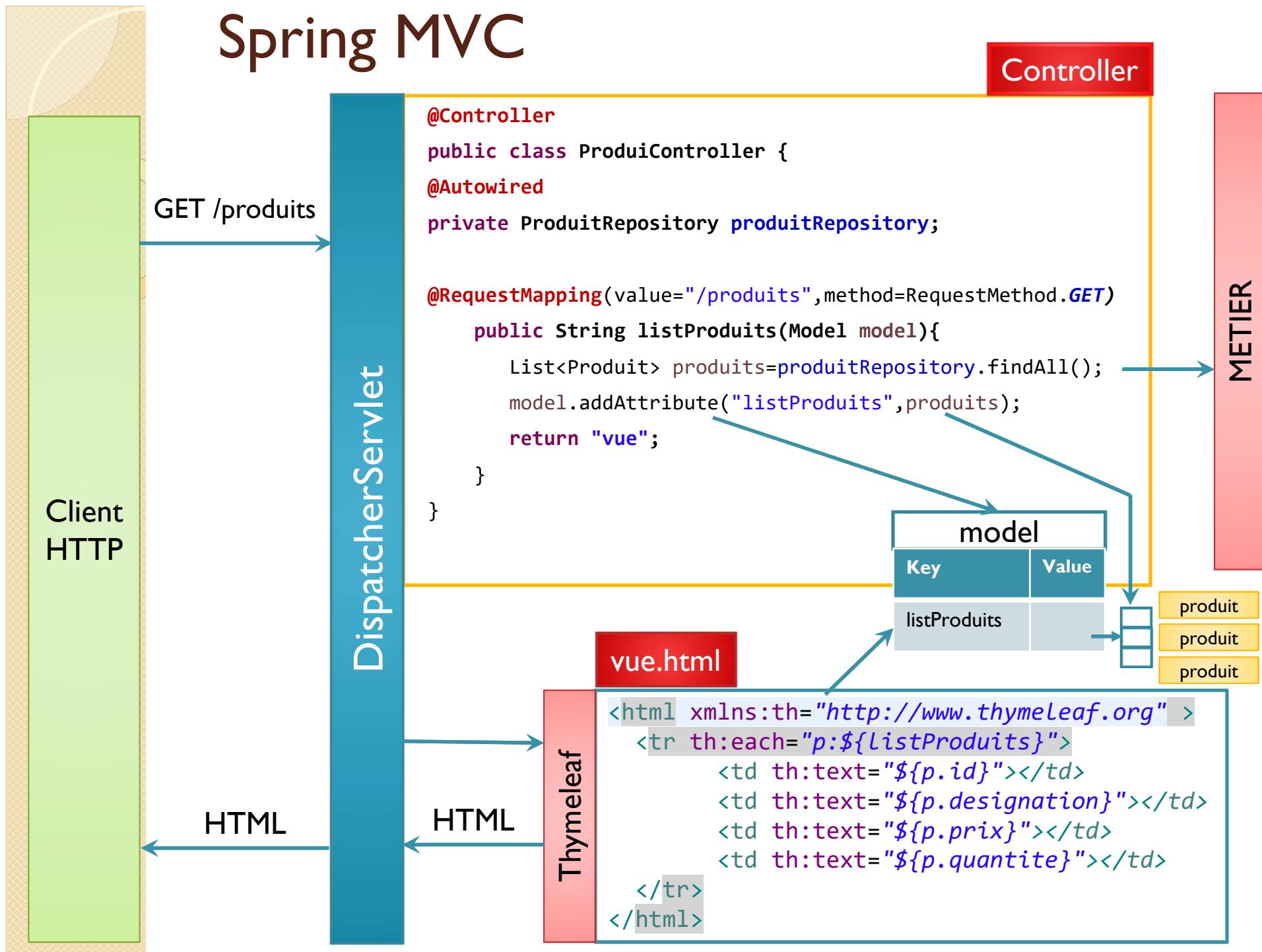


Spring MVC avec Thymeleaf



- Mohamed Youssfi
- Laboratoire Signaux Systèmes Distribués et Intelligence Artificielle (SSDIA)
- ENSET, Université Hassan II Casablanca, Maroc

Spring MVC





VUES DE L'APPLICATION BASÉES SUR LE MOTEUR DE TEMPLATE TYMELEAF



Utilisation des Layouts : Templates

- Généralement toutes les pages d'une application web partagent le même contenu html (Header, footer, menus, etc..)
- Pour éviter de faire des copies coller dans toutes les pages, il est important de définir une page Template qui définit
 - toutes les parties fixes de toutes les pages (header, footer, menus, etc...)
 - et déclarer les sections qui changeront de contenu en fonction de chaque page.

Exemple de template: layout.html

```
<!DOCTYPE html>
<html
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/Layout">

    <head>
        <meta charset="utf-8"/>
        <title>Gestion des produits</title>
        <link rel="stylesheet" type="text/css" href="../../static/css/bootstrap.min.css"
              th:href="@{/css/bootstrap.min.css}" />
        <link rel="stylesheet" type="text/css" href="../../static/css/style.css"
              th:href="@{/css/style.css}" />
    </head>
    <body>
        <header> Header FIXE </header>
        <section layout:fragment="content">
            </section>
        <footer class="navbar-fixed-bottom">
            </footer>FOOTER FIXE
    </body>
</html>
```

HEADER FIXE

CONTENT VARIABLE

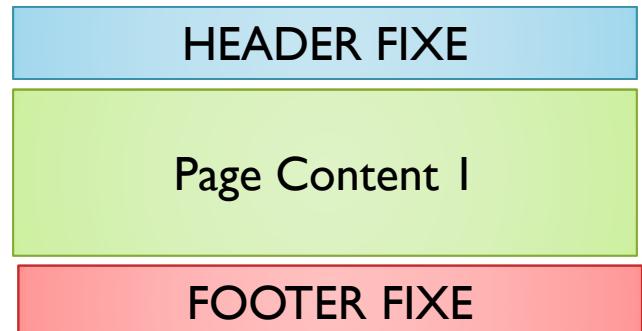
FOOTER FIXE

Utilisation du layout dans une page

page1.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/Layout"
      layout:decorator="Layout" >
<body>
    <div layout:fragment="content">
        Page 1 Content
    </div>
</body>
</html>
```

→ layout.html



Suite de l'application

```
cata_mvc_1 [boot] [devtools]
├── Spring Elements
│   └── src/main/java
│       ├── org.sid
│       │   ├── CataMvc1Application.java
│       ├── org.sid.dao
│       │   └── ProduitRepository.java
│       ├── org.sid.entities
│       │   └── Produit.java
│       ├── org.sid.security
│       │   └── SecurityConfig.java
│       └── org.sid.web
│           └── ProduitController.java
└── src/main/resources
    ├── static
    │   └── css
    │       ├── bootstrap.min.css
    │       └── style.css
    └── templates
        ├── Confirmation.html
        ├── EditProduit.html
        ├── FormProduit.html
        ├── layout.html
        ├── login.html
        └── produits.html
    └── application.properties
└── pom.xml
```

Authentification

User Name:

Pass Word:

Login

localhost:8080/produits/index?mc=HP

Chercher Produits admin

Mot Clé: HP

Liste des produits

ID	Désignation	Prix	Quantité	Edit	Delete
2	Imprimante HP 548	300.0	14	Edit	Delete
3	Ordinateur HP 87	200.0	23	Edit	Delete
6	Ordinateur HP 87 9000	800.0	23	Edit	Delete
8	Imprimante HP 548	300.0	14	Edit	Delete

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47		
48	49	50	51	52	53	54	55	56	57	58														

Suite de l'application

```
cata_mvc_1 [boot] [devtools]
  Spring Elements
    src/main/java
      org.sid
        CataMvc1Application.java
      org.sid.dao
        ProduitRepository.java
      org.sid.entities
        Produit.java
      org.sid.security
        SecurityConfig.java
      org.sid.web
        ProduitController.java
    src/main/resources
      static
        css
          bootstrap.min.css
          style.css
      templates
        Confirmation.html
        EditProduit.html
        FormProduit.html
        layout.html
        login.html
        produits.html
      application.properties
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
```

localhost:8080/admin/form

Chercher Produits admin

Saisie d'un produit

Désignation:

Prix:

Quantité:

 ← → C localhost:8080/admin/save
Chercher Produits admin

Saisie d'un produit

Désignation:

la taille doit être entre 4 et 70

Prix:

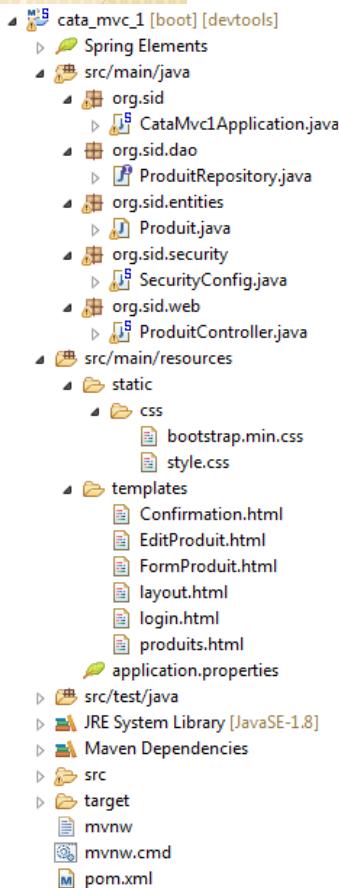
doit être supérieur ou égal à 100

Quantité:

 ← → C localhost:8080/admin/save
Chercher Produits admin

med@youssfi.net

Suite de l'application



localhost:8080/admin/edit?id=6&page=0&mc=

Chercher Produits admin

Edition d'un produit

ID: 6

Désignation: Ordinateur HP 87 9000

Prix: 800.0

Quantité: 23

← → C localhost:8080/admin/save

Chercher Produits admin

Save

Conformation

ID: 375

Désignation: LP FD 453

Prix: 780.0

Quantité: 34

med@youssfi.net

Entité Produit

```
package org.sid.entities;

import java.io.Serializable; import javax.persistence.Entity;
import javax.persistence.GeneratedValue; import javax.persistence.Id;
import javax.validation.constraints.DecimalMin;import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

@Entity
public class Produit implements Serializable {
    @Id @GeneratedValue
    private Long id;
    @NotNull
    @Size(min=4,max=70)
    private String designation;
    @DecimalMin(value="100")
    private double prix;
    private int quantite;

    // Constructeurs
    public Produit() {}
    public Produit(String designation, double prix, int quantite) {
        this.designation = designation; this.prix = prix; this.quantite = quantite;
    }
    // Getters et Setters
}
```

```
graph LR
    A["@NotNull  
@Size(min=4,max=70)  
@DecimalMin(value="100")"] --> B[Validation]
```

Interface ProduitRepository

```
package org.sid.dao;

import org.sid.entities.Produit;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface ProduitRepository extends JpaRepository<Produit, Long>{
    @Query("select p from Produit p where p.designation like :x")
    public Page<Produit> chercher(@Param("x")String mc,Pageable pageable);
}
```

Déployer le data source : application.properties

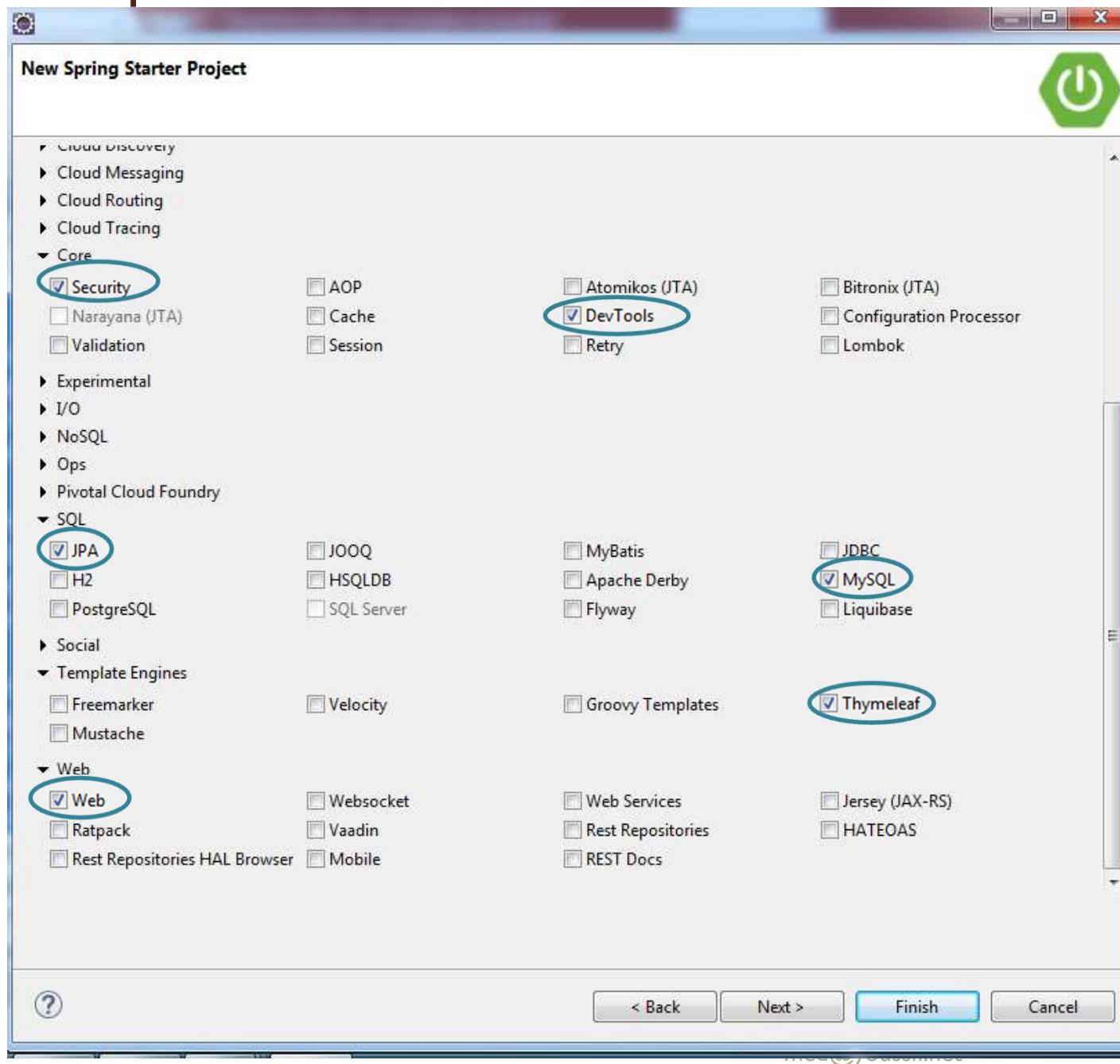
```
#security.user.name=admin
#security.user.password=123
spring.datasource.url = jdbc:mysql://localhost:3306/db_cat_mvc_1
spring.datasource.username = root
spring.datasource.password =
spring.datasource.driverClassName = com.mysql.jdbc.Driver
spring.jpa.show-sql = true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
spring.thymeleaf.cache=false
```

Application Spring Boot

```
package org.sid;
import org.sid.dao.ProduitRepository; import org.sid.entities.Produit;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

@SpringBootApplication
public class CataMvc1Application {
    public static void main(String[] args) {
        ApplicationContext ctx=SpringApplication.run(CataMvc1Application.class, args);
        ProduitRepository produitRepository=ctx.getBean(ProduitRepository.class);
        produitRepository.save(new Produit("AX 548",900, 4));
        produitRepository.save(new Produit("Imprimante HP 548",300, 14));
        produitRepository.save(new Produit("Ordinateur HP 87",200,23));
        produitRepository.findAll()
            .forEach(p->System.out.println(p.getDesignation()));
    }
}
```

Dépendances Maven



Dépendances Maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

ProduitController

```
package org.sid.web;
import javax.validation.Valid;
import org.sid.dao.ProduitRepository;
import org.sid.entities.Produit;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
```

ProduitController : Chercher les produits

```
@Controller
public class ProduitController {
    @Autowired
    private ProduitRepository produitRepository;
    @RequestMapping(value="/produits/index")
    public String produits(Model model,
        @RequestParam(name="page",defaultValue="0")int page,
        @RequestParam(name="size",defaultValue="4")int size,
        @RequestParam(name="mc",defaultValue "")String mc){
        //List<Produit> produits=produitRepository.findAll();
        Page<Produit> pageProduits=
        produitRepository.chercher("%"+mc+"%",new PageRequest(page, size));
        model.addAttribute("listProduits", pageProduits.getContent());
        model.addAttribute("pageCourante", page);
        int[] pages=new int[pageProduits.getTotalPages()];
        model.addAttribute("pages",pages);
        model.addAttribute("mc", mc);
        return "produits";
    }
}
```

Vue : produits.html

```
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/Layout"
      layout:decorator="template1">
<div layout:fragment="content">
    <div class="container">
        <form th:action="@{/user/index}" method="get">
            <label>Mot Clé:</label> <input type="text" name="motCle" th:value="${motCle}"/>
            <button class="btn btn-primary">Chercher</button>
        </form>
    </div>
    <div class="container">
        <table class="table">
            <tbody>
                <tr th:each="p:${listProduits}">
                    <td th:text="${p.id}"></td> <td th:text="${p.designation}"></td><td th:text="${p.prix}"> </td>
                    <td th:text="${p.quantite}"></td> <td><a th:href="@{/admin/edit(id=${p.id})}">Edit</a></td>
                    <td><a th:href="@{/admin/delete(id=${p.id},page=${pageCourante},size=${size},motCle=${motCle})}">
                        Delete</a></td>
                </tr>
            </tbody>
        </table>
    </div>
    <div class="container">
        <ul class="nav nav-pills">
            <li th:class="${pageCourante}==${status.index}? 'active': ''"
                th:each="pa,status:${pages}">
                <a th:href="@{/user/index(page=${status.index},size=${size},motCle=${motCle})}"
                   th:text="${status.index}"></a>
            </li>
        </ul>
    </div>

```

Formulaire de recherche

Tableau des produits

Pagination

Suppression d'un produit

ProduitController.java

```
@RequestMapping(value="/admin/delete",method=RequestMethod.GET)
public String delete(Long id,String mc,int page){
    produitRepository.delete(id);
    return "redirect:/produits/index?page="+page+"&mc="+mc;
}
```

produits.html

```
<a th:href="@{/admin/delete(id=${p.id})}">
    Delete
</a>
```

Saisie et Ajout d'un produit avec Validation

```
@RequestMapping(value="/admin/form",method=RequestMethod.GET)           ProduitController.java
public String form(Model model){
    model.addAttribute("produit", new Produit());
    return "FormProduit";
}

@RequestMapping(value="/admin/save",method=RequestMethod.POST)
public String save(Model model,@Valid Produit p,BindingResult bindingResult){
    if(bindingResult.hasErrors()) { return "FormProduit"; }
    produitRepository.save(p);
    return "Confirmation";
}
```

```
@Entity
public class Produit implements
    @Id @GeneratedValue
private Long id;
    @NotNull
    @Size(min=4,max=70)
private String designation;
    @DecimalMin(value="100")
private double prix;
private int quantite;
```

```
<form th:action="@{/admin/save}" method="post">
    <label class="control-label">Désignation:</label>
    <input type="text" name="designation" th:value="${produit.designation}"/>
    <span th:errors="${produit.designation}"></span>

    <label>Prix:</label>
    <input type="text" name="prix" th:value="${produit.prix}"/>
    <span th:errors="${produit.prix}"></span>

    <label>Quantité:</label>
    <input type="text" name="quantite" th:value="${produit.quantite}"/>
    <span th:errors="${produit.quantite}"></span>

    <button type="submit">Save</button>
</form>
```

FormProduit.html

Confirmation.html

```
<div layout:fragment="content">
  <div class="col-md-6 col-md-offset-3 col-xs-12">
    <div class="panel panel-primary">
      <div class="panel-heading">Confirmation</div>
      <div class="panel-body">
        <div class="form-group">
          <label class="control-label">ID:</label>
          <label class="control-label" th:inline="text">[${produit.id}]</label>
        </div>

        <div class="form-group">
          <label class="control-label">Désignation:</label>
          <label class="control-label" th:inline="text">[${produit.designation}]</label>
        </div>

        <div class="form-group">
          <label class="control-label">Pix:</label>
          <label class="control-label" th:inline="text">[${produit.prix}]</label>
        </div>

        <div class="form-group">
          <label class="control-label">Quantité:</label>
          <label class="control-label" th:inline="text">[${produit.quantite}]</label>
        </div>
      </div>
    </div>
  </div>
</div>
```

FormProduit.html

Edition d'un produit

ProduitController.java

```
@RequestMapping(value="/admin/edit",method=RequestMethod.GET)  
public String edit(Model model,Long id){  
    Produit p=produitRepository.findOne(id);  
    model.addAttribute("produit",p);  
    return "EditProduit";  
}
```

EditProduit.html

```
<form th:action="@{/admin/save}" method="post">  
    <label class="control-label">ID:</label>  
    <input type="text" name="id" th:value="${produit.id}"/>  
    <span th:errors="${produit.id}" ></span>  
  
    <label class="control-label">Désignation:</label>  
    <input type="text" name="designation" th:value="${produit.designation}"/>  
    <span th:errors="${produit.designation}" ></span>  
  
    <label>Prix:</label>  
    <input type="text" name="prix" th:value="${produit.quantite}"/>  
    <span th:errors="${produit.prix}" ></span>  
  
    <label>Quantité:</label>  
    <input type="text" name="quantite" th:value="${produit.quantite}"/>  
    <span th:errors="${produit.quantite}" ></span>  
  
    <button type="submit">Save</button>  
</form>
```

Action Par défaut

```
@RequestMapping(value="/")
public String index(){
    return "redirect:/produits/index";
}
```

Action Pour le formulaire d'authentification

```
@RequestMapping(value="/login")  
public String login(){  
    return "login";  
}
```

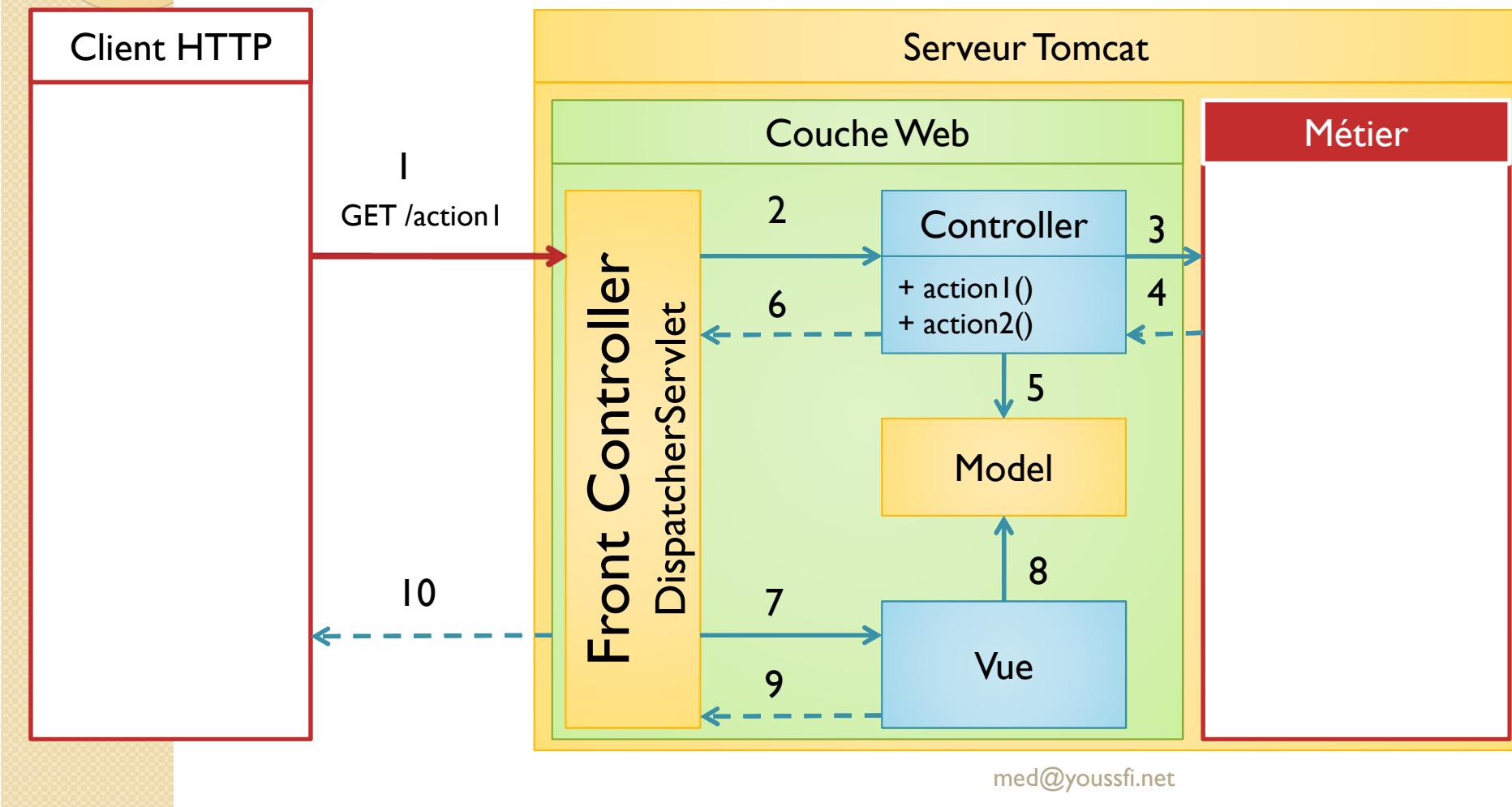


- **SPRING
SECURITY**

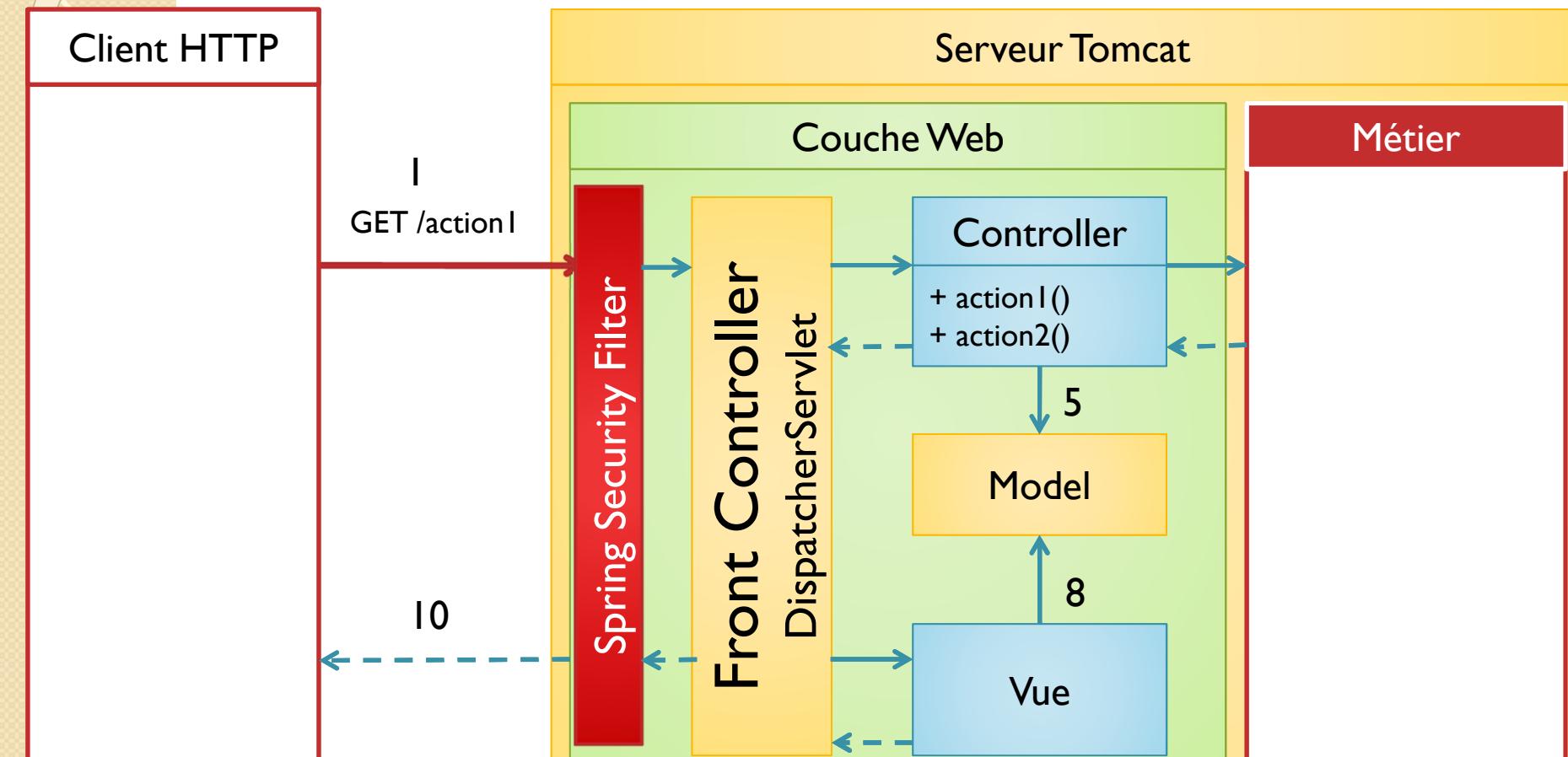
med@youssf.net

Architecture Spring MVC

I – Le client envoie une requête HTTP de type GET ou POST



Spring Security





Configuration de Spring Security

- Spring Security est un module de Spring qui permet de sécuriser les applications Web.
- Spring Security configure des filtres (`springSecurityFilterChain`) qui permet d'intercepter les requêtes HTTP et de vérifier si l'utilisateur authentifié dispose des droits d'accès à la ressource demandée.
- Les actions du contrôleur ne seront invoquées que si l'utilisateur authentifié dispose de l'un des rôles attribués à l'action.
- Spring Security peut configurer les rôles associés aux utilisateurs en utilisant différentes solution (Mémoire, Base de Données SQL, LDAP, etc..)

Configuration de Spring Security

- Dans cet exemple , nous supposons que nous avons trois tables dans la base de données : users, roles et users_roles et que les mot de passes sont cryptés en format MD5.

roles

role
USER
ADMIN

users

username	password	active
user	ee11cbb19052e40b07aac0ca060c23ee	1
admin	21232f297a57a5a743894a0e4a801fc3	1

users_roles

username	role
admin	ADMIN
admin	USER
user	USER

Dépendance Maven

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

SecurityConfig.java : inMemoryAuthentication

```
package org.sid.sec;

import javax.sql.DataSource; import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.encoding.Md5PasswordEncoder;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
/* Pour le cas ou les utilisateurs sont connues et fixes d'une manière statique */
auth.inMemoryAuthentication().withUser("admin").password("1234").roles("ADMIN", "USER");
auth.inMemoryAuthentication().withUser("user").password("1234").roles("USER");
}
}
```

SecurityConfig.java : JDBC Authentication

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private DataSource dataSource;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        /* Pour le cas où les utilisateurs sont stockés dans une base la même base de
        données de l'application */
        auth.jdbcAuthentication()
            .dataSource(dataSource) // Data source de l'application
            .usersByUsernameQuery("select username as principal, pass as credentials, active
from users where username=?") // Requête SQL pour chercher si l'Utilisateur existe
            .authoritiesByUsernameQuery("select username as principal, role as role from
users_roles where username=?") // Requête SQL pour connaître les rôles de l'utilisateur
            .rolePrefix("ROLE_") // Préfixe ajouté au Rôle par EX ROLE_ADMIN
            .passwordEncoder(new Md5PasswordEncoder());
    }
}
```

SecurityConfig.java

```
@Override  
protected void configure(HttpSecurity http) throws Exception {  
    /* Indiquer à SpringSecurity que l'authentification passe par un formulaire  
    d'authentification avec username et password */  
    http.formLogin().loginPage("/login");  
    /* Toutes les requêtes HTTP avec URL /user/* nécessitent d'être authentifié avec  
    un utilisateur ayant le rôle USER */  
    http.authorizeRequests().antMatchers("/user/*").hasRole("USER");  
    /* Toutes les requêtes HTTP avec URL /admin/* nécessitent d'être authentifié  
    avec un Utilisateur ayant le rôle ADMIN*/  
    http.authorizeRequests().antMatchers("/admin/*").hasRole("ADMIN");  
    /* Si un Utilisateur tente d'accéder à une resource dont il n'a pas le droit,  
    il sera redirigué vers la page associée à l'action /403 */  
    http.exceptionHandling().accessDeniedPage("/403");  
}  
}
```

login.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" >
<head>
<meta charset="utf-8"/>
<title>Produits</title>
    <link rel="stylesheet" type="text/css"
        href="../static/css/bootstrap.min.css"
        th:href="@{/css/bootstrap.min.css}" />
    <link rel="stylesheet" type="text/css"
        href="../static/css/style.css"
        th:href="@{/css/style.css}" />
</head>
<body >
    <div class="col-md-4 col-md-offset-4 spacer40">
        <div class="container red">
            <div th:if="${param.error}">
                Invalid username and password.
            </div>
            <div th:if="${param.Logout}">
                You have been logged out.
            </div>
        </div>
    </div>
```

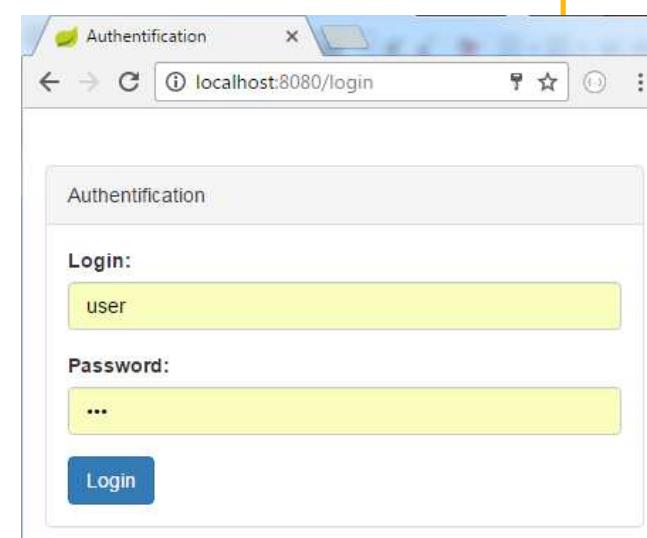
The figure displays three screenshots of a web browser window titled "Authentication" illustrating the login process:

- Screenshot 1:** The browser shows the URL `localhost:8080/login`. The login form contains the "Login:" field with "user" and the "Password:" field with "...".
- Screenshot 2:** The browser shows the URL `localhost:8080/login?error`. The message "Invalid Username or Password" is displayed above the login form.
- Screenshot 3:** The browser shows the URL `localhost:8080/login?logout`. The message "You Have been Logged out" is displayed above the login form.

Below the bottom screenshot, the email address `med@youssfi.net` is written.

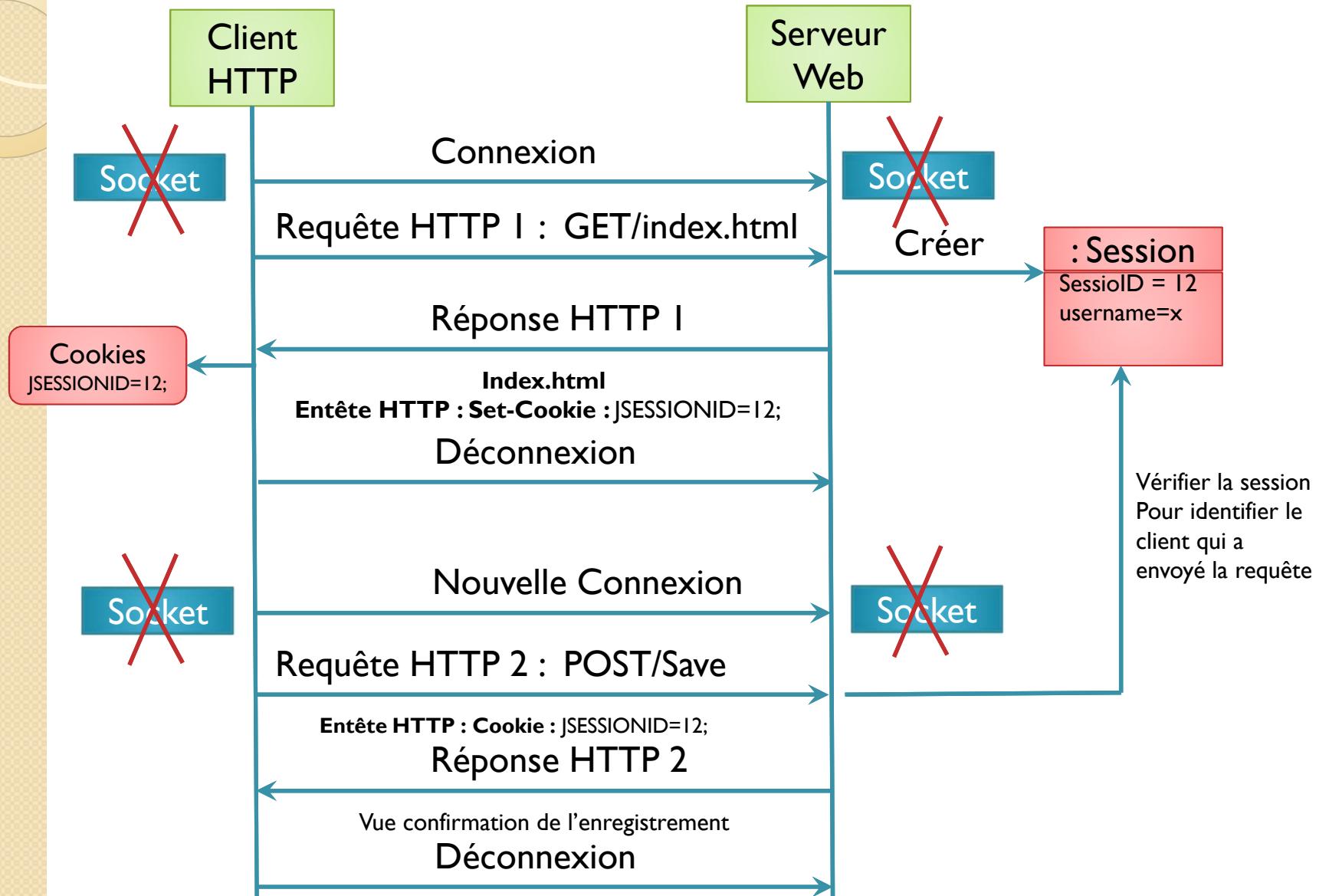
login.html

```
<div class="panel panel-default">
    <div class="panel-heading">Authentification</div>
    <div class="panel-body">
        <form th:action="@{Login}" method="post">
            <div class="form-group">
                <label class="control-label">User Name:</label>
                <input class="form-control" type="text" name="username"/>
            </div>
            <div class="form-group">
                <label class="control-label">Pass Word:</label>
                <input class="form-control" type="password" name="password"/>
            </div>
            <div>
                <button type="submit" class="btn btn-primary">Login</button>
            </div>
        </form>
    </div>
</div>
</div>
</body>
</html>
```



med@youssfi.net

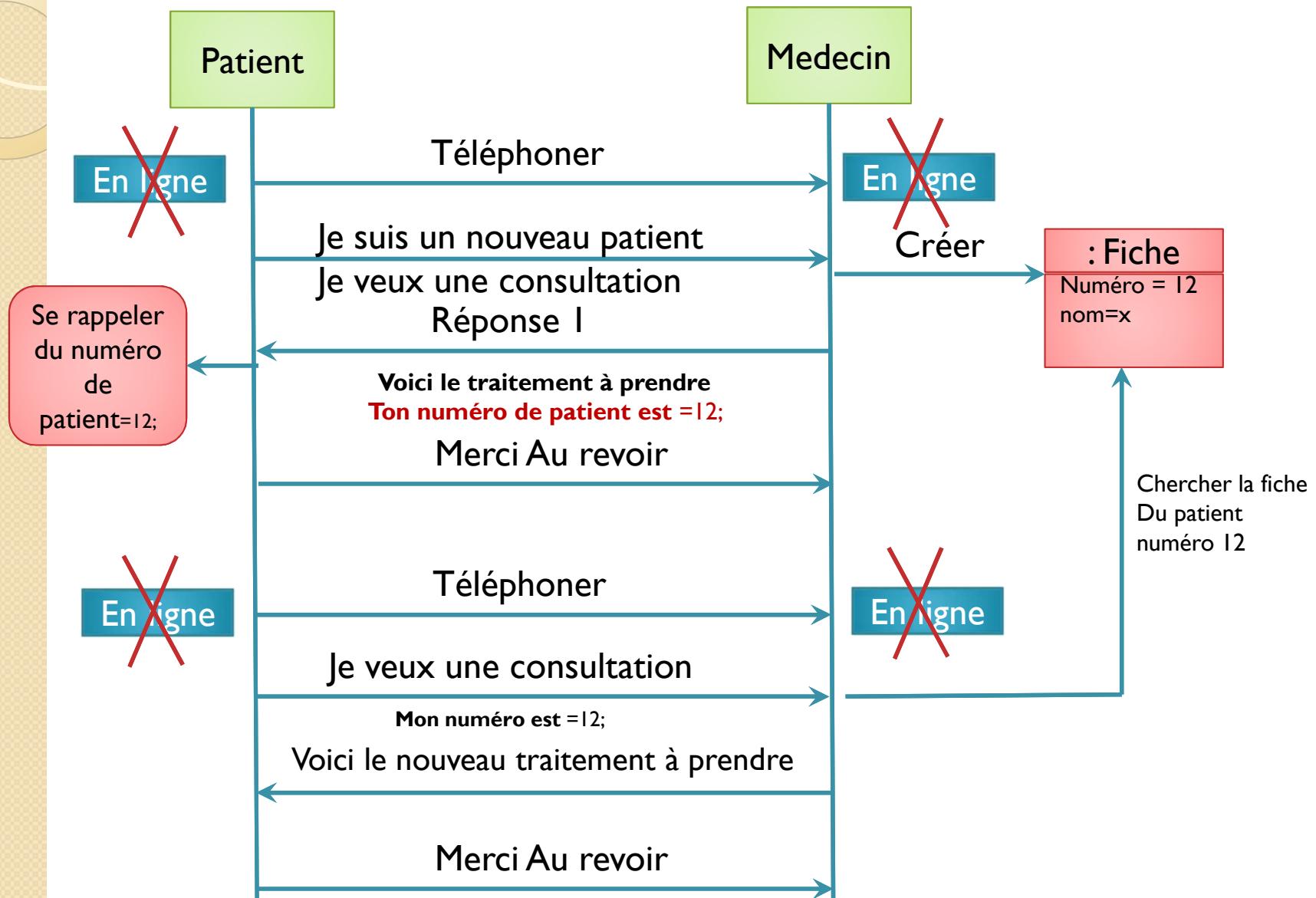
Session et Cookies



Utilisation des sessions et des cookies

- Généralement quant un client HTTP envoie sa première requête, le serveur web crée une session pour ce client.
- Une session est un objet stocké dans la mémoire du serveur qui peut servir pour stocker des informations relatives au client.
- Le serveur attribut un SessionID unique à chaque session.
- Ce SessionID est ensuite envoyé dans la réponse http en sousforme d'un cookie en utilisant l'entête de la réponse HTTP :
 - **Set-Cookie :** JSESSIONID=F84DB7B959F76B183DBF05F999FAEE11;
- Ce qui signifie que le serveur demande au client d'enregistrer ce SESSIONID dans un fichier stocké dans la machine du client appelé COOKIE.
- Une fois que le client reçoive la réponse HTTP, la connexion est fermée.
- A chaque fois que le client envoie une requête HTTP vers le serveur, il envoie toujours les données des cookies dont le SESSIONID.
- Les cookies sont envoyés dans la requête HTTP en utilisant une entête COOKIE:
 - **Cookie:** JSESSIONID=F84DB7B959F76B183DBF05F999FAEE11
- Grace à cette information, le serveur peut savoir de quel client s'agit-il même s'il s'agit d'une nouvelle connexion.

Session et Cookies > Patient et médecin





CSRF : Cross Site Request Forgery

- C'est une attaque par falsification de requête intersites
- Force le navigateur d'une victime authentifiée à envoyer une requête http, comprenant le cookie de session de la victime ainsi que toute autre information automatiquement inclue, à une application web vulnérable.
- Ceci permet à l'attaquant de forcer le navigateur de la victime à générer des requêtes,
- l'application vulnérable considérant alors qu'elles émanent légitimement de la victime.
- Une attaque CSRF va exécuter du code malveillant dans une application Web au travers de la session d'un utilisateur connecté.

CSRF

- Par défaut Spring Security permet de protéger le site contre ce genre d'attaque.
- Dans chaque formulaire de votre application Spring Security ajoute un champ caché contenant un jeton :

```
<input type="hidden" name="_csrf"  
value="61390c32-08f8-4c85-b55a-fe20bc7f0913" />
```
- Pour chaque requête envoyé, Spring Sec vérifie si ce jeton existe dans la requête client.
- Avec cette procédure, on est sûre que la requête vient vraiment du client authentifié et non pas d'une requête falsifiée.



Code Complet

med@youssf.net

layout.html

```
<!DOCTYPE html >
<html
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/Layout">
<head>
    <meta charset="utf-8"/>
    <title>Insert title here</title>
    <link rel="stylesheet" type="text/css" href="../static/css/bootstrap.min.css"
        th:href="@{/css/bootstrap.min.css}" />
    <link rel="stylesheet" type="text/css" href="../static/css/style.css"
        th:href="@{/css/style.css}" />
</head>
<body>
    <header>
        <div class="navbar navbar-default">
            <div class="container-fluid">
                <ul class="nav navbar-nav">
                    <li><a th:href="@{/produits/index}">Chercher</a></li>
                    <li><a th:href="@{/admin/form}">Produits</a></li>
                    <li>
                        <a th:href="@{/Login?Logout}" th:inline="text">[$[#httpServletRequest.remoteUser]]</a>
                    </li>
                </ul>
            </div>
        </div>
    </body>
```

layout.html

```
</header>

<section layout:fragment="content">

</section>

<footer class="navbar-fixed-bottom">
  <hr/>
  <div class="container">
    <small>
      Copyright @2016
    </small>
  </div>
</footer>
</body>
</html>
```

produits.html

```
<!DOCTYPE html>
<html
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/Layout"
    layout:decorator="Layout" >
<head>
    <meta charset="utf-8"/>
    <title>Produits</title>
    <link rel="stylesheet" type="text/css" href="../static/css/bootstrap.min.css"
        th:href="@{/css/bootstrap.min.css}" />
    <link rel="stylesheet" type="text/css" href="../static/css/style.css"
        th:href="@{/css/style.css}" />
</head>
<body >
    <section layout:fragment="content">
        <div class="container">
            <form th:action="@{index}" method="get">
                <label>Mot Clé:</label>
                <input type="text" name="mc" th:value="${mc}"/>
                <button type="submit" class="btn btn-primary">Chercher</button>
            </form>
        </div>
    </section>
</body>
```

produits.html

```
<div class="container">
    <h3>Liste des produits</h3>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>ID</th><th>Désignation</th><th>Prix</th><th>Quantité</th>
            </tr>
        </thead>
        <tbody>
            <tr th:each="p:${listProduits}">
                <td th:text="${p.id}"></td>
                <td th:text="${p.designation}"></td>
                <td th:text="${p.prix}"></td>
                <td th:text="${p.quantite}"></td>
                <td><a href="@{/admin/edit(id=${p.id},page=${pageCourante},mc=${mc})}">Edit</a></td>
                    <td><a onclick="return alert('Etes vous sûre?');" href="@{/admin/delete(id=${p.id},page=${pageCourante},mc=${mc})}">Delete</a></td>
            </tr>
        </tbody>
    </table>
</div>
```

produits.html

```
<div class="container">
    <ul class="nav nav-pills">
        <li th:class="${pageCourante}==${status.index}? 'active' : ''"
            th:each="p, status:${pages}" >
            <a th:href="@{index(page=${status.index}, mc=${mc})}" th:text="${status.index}"></a>
        </li>
    </ul>
</div>
</section>
</body>
</html>
```

FormProduit.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"
      xmlns:Layout="http://www.ultraq.net.nz/thymeleaf/Layout"
      layout:decorator="Layout" >

    <head>
        <meta charset="utf-8"/>
        <title>Produits</title>
        <link rel="stylesheet" type="text/css" href="../static/css/bootstrap.min.css"
              th:href="@{/css/bootstrap.min.css}" />
        <link rel="stylesheet" type="text/css" href="../static/css/style.css"
              th:href="@{/css/style.css}" />
    </head>
    <body>
        <div class="col-md-6" Layout:fragment="content">
            <div class="panel panel-info">
                <div class="panel-heading">Saisie d'un produit</div>
                <div class="panel-body">
                    <form th:action="@{/admin/save}" method="post">
                        <div class="form-group">
                            <label class="control-label">Désignation:</label>
                            <input class="form-control" type="text" name="designation"
                                   th:value="${produit.designation}"/>
                            <span th:errors="${produit.designation}" class="red"></span>
                        </div>
                </div>
        </div>
```

FormProduit.html

```
<div class="form-group">
    <label class="control-label">Prix:</label>
    <input class="form-control" type="text" name="prix" th:value="${produit.prix}"/>
    <span th:errors="${produit.prix}" class="red"></span>
</div>
<div class="form-group">
    <label class="control-label">Quantité:</label>
    <input class="form-control" type="text" name="quantite" th:value="${produit.quantite}"/>
    <span th:errors="${produit.quantite}" class="red"></span>
</div>
<div>
    <button type="submit" class="btn btn-primary">Save</button>
</div>
</form>
</div>
</div>
</body>
</html>
```

confirmation.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"
      xmlns:Layout="http://www.ultraq.net.nz/thymeleaf/Layout"
      layout:decorator="Layout" >

    <head>
        <meta charset="utf-8"/>
        <title>Produits</title>
        <link rel="stylesheet" type="text/css"
              href="../static/css/bootstrap.min.css"
              th:href="@{/css/bootstrap.min.css}" />
    </head>
    <body>
        <div class="col-md-6" Layout:fragment="content">
            <div class="panel panel-info">
                <div class="panel-heading">Conformation</div>
                <div class="panel-body">
                    <div class="form-group">
                        <label class="control-label">ID:</Label>
                        <label class="control-label" th:text="${produit.id}"></Label>
                    </div>
                    <div class="form-group">
                        <label class="control-label">Désignation:</Label>
                        <label class="control-label" th:text="${produit.designation}"></Label>
                    </div>
                </div>
            </div>
        </div>
    </body>

```

confirmation.html

```
<div class="form-group">
    <label class="control-label">PrixB:</Label>
    <label class="control-label" th:text="${produit.prix}"></Label>
</div>
<div class="form-group">
    <label class="control-label">Quantité:</Label>
    <label class="control-label" th:text="${produit.quantite}"></Label>
</div>
</div>
</div>
</body>
</html>
```

EditProduit.html

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"
      xmlns:Layout="http://www.ultraq.net.nz/thymeleaf/Layout"
      layout:decorator="Layout" >

    <head>
        <meta charset="utf-8"/>
        <title>Produits</title>
        <link rel="stylesheet" type="text/css"
              href="../static/css/bootstrap.min.css"
              th:href="@{/css/bootstrap.min.css}" />
    </head>
    <body>
        <div class="col-md-6" Layout:fragment="content">
            <div class="panel panel-info">
                <div class="panel-heading">Edition d'un produit</div>
                <div class="panel-body">
                    <form th:action="@{/admin/save}" method="post">
                        <div class="form-group">
                            <label class="control-label">ID:</Label>
                            <input class="form-control" type="text" name="id" th:value="${produit.id}"/>
                            <span th:errors="${produit.id}"></span>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </body>
</html>
```

EditProduit.html

```
<div class="form-group">
    <label class="control-label">Désignation:</label>
    <input class="form-control" type="text" name="designation"
th:value="${produit.designation}" />
    <span th:errors="${produit.designation}"></span>
</div>
<div class="form-group">
    <label class="control-label">Prix:</label>
    <input class="form-control" type="text" name="prix" th:value="${produit.prix}" />
    <span th:errors="${produit.prix}"></span>
</div>
<div class="form-group">
    <label class="control-label">Quantité:</label>
    <input class="form-control" type="text" name="quantite" th:value="${produit.quantite}" />
    <span th:errors="${produit.quantite}"></span>
</div>
<div>
    <button type="submit" class="btn btn-primary">Save</button>
</div>
</form>
</div>
</div>
</body>
</html>
```



Projet Développement Web JEE

- On souhaite créer une application qui permet de gérer des comptes bancaire.
 - Chaque compte est défini un code, un solde et une date de création
 - Un compte courant est un compte qui possède en plus un découvert
 - Un compte épargne est un compte qui possède en plus un taux d'intérêt.
 - Chaque compte appartient à un client.
 - Chaque client est défini par son code et son nom
 - Chaque compte peut subir plusieurs opérations.
 - Il existe deux types d'opérations : Versement et Retrait
 - Une opération est définie par un numéro, une date et un montant.



Exigences fonctionnelles

- L'application doit permettre de :
 - Gérer des clients :
 - Ajouter un client
 - Consulter tous les clients
 - Consulter les clients dont le nom contient un mot clé.
 - Gérer les comptes :
 - Ajouter un compte
 - Consulter un compte
 - Gérer les opérations :
 - Effectuer un versement d'un montant dans un compte
 - Effectuer un retrait d'un montant dans un compte
 - Effectuer un virement d'un montant d'un compte vers un autre
 - Consulter les opérations d'un compte page par page
- Les opérations nécessitent une opération d'authentification



Exigences Techniques

- Les données sont stockées dans une base de données MySQL
- L'application se compose de trois couches :
 - La couche DAO qui est basée sur Spring Data, JPA, Hibernate et JDBC.
 - La couche Métier
 - La couche Web basée sur MVC côté Serveur en utilisant Thymeleaf.
- La sécurité est basée sur Spring Security

Travail demandé :

- Etablir une architecture technique du projet
- Etablir un diagramme de classes qui montre les entités, la couche DAO et la couche métier.
- Créer un projet SpringBoot qui contient les éléments suivants :
 - Les entités
 - La couche DAO (Interfaces Spring data)
 - La couche métier (Interfaces et implémentations)
 - La couche web :
 - Les contrôleurs Spring MVC
 - Les Vue basée sur Thymeleaf
- Sécuriser l'application en utilisant un système d'authentification basé sur Spring Security

Architecture Technique

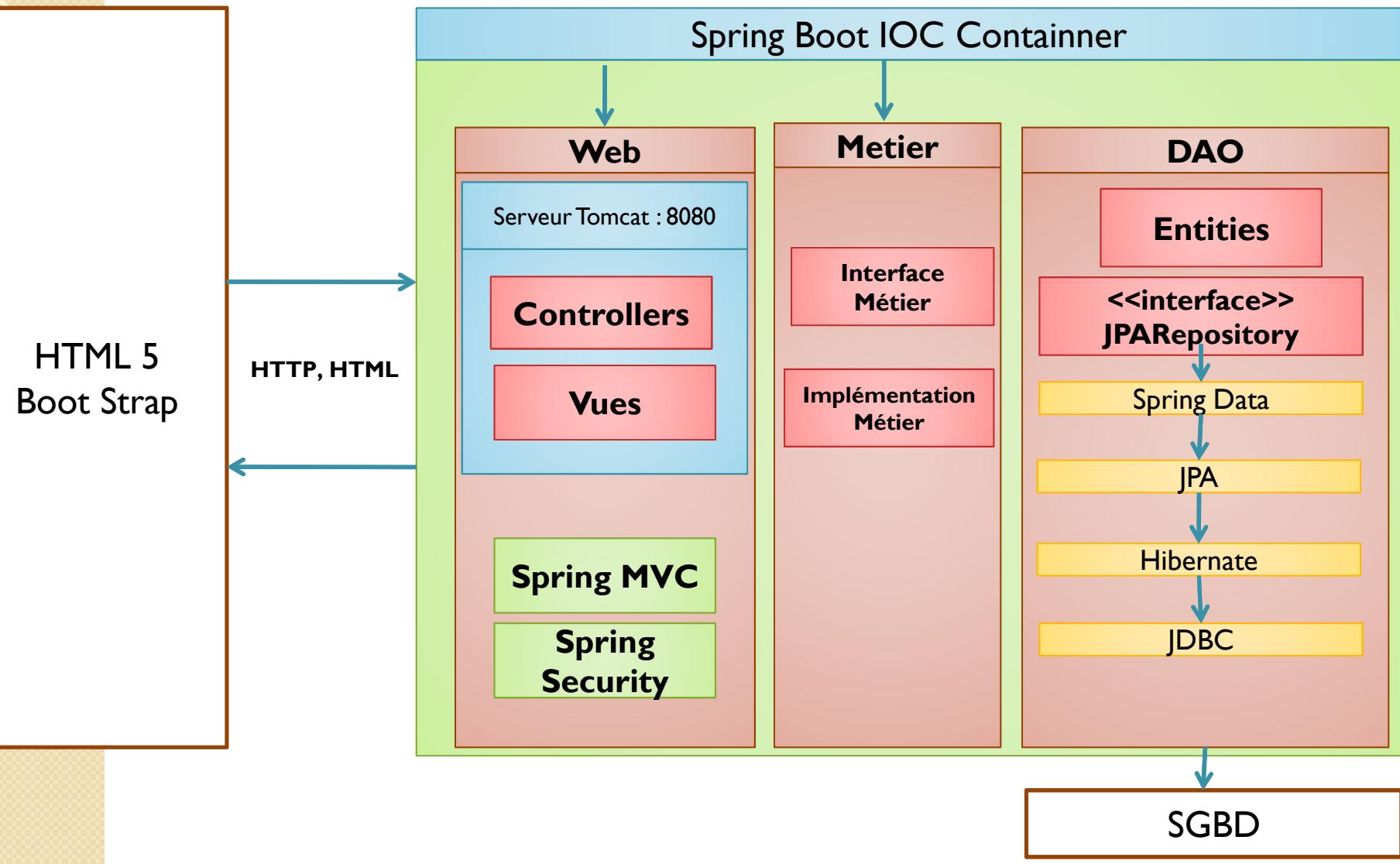
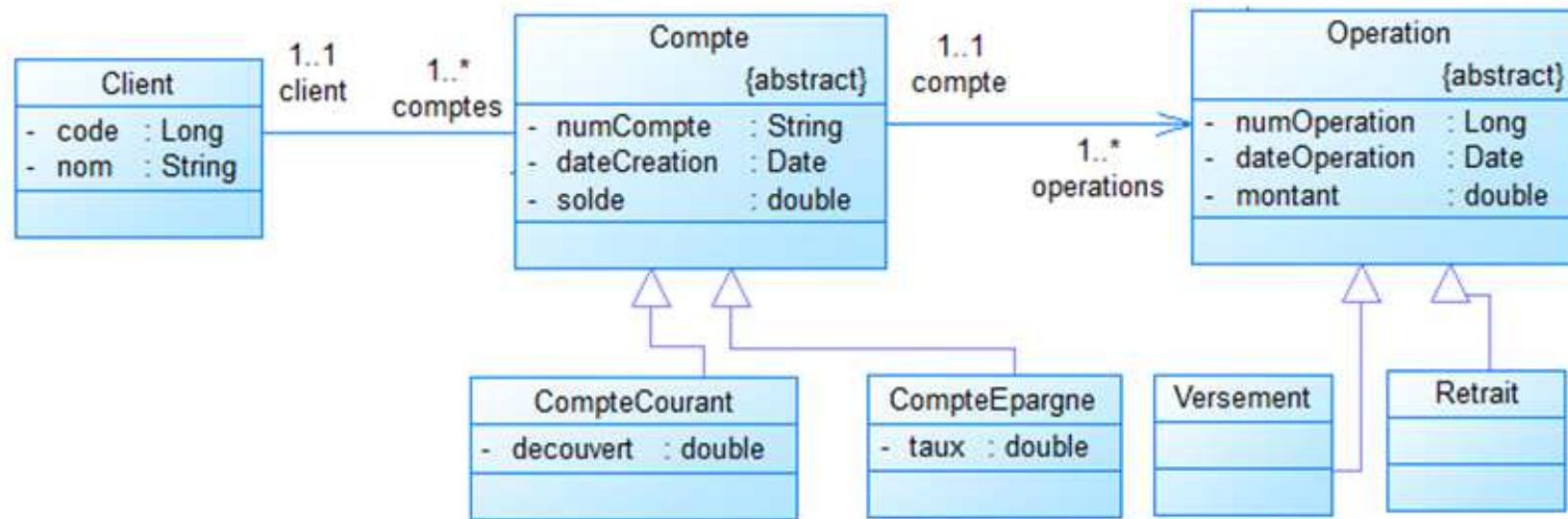


Diagramme de classes des entités et MLDR



- **MLRD : En utilisant la stratégie Single Table pour l'héritage**
 - **T_CLIENTS** (CODE_CLI, NOM_CLI)
 - **T_COMPTES** (NUM_CPT, TYPE_PTE, DATE_CR, SOLDE, DEC, TAUX, #CODE_CLI)
 - **T_OPERATIONS** (NUM_OP, TYPE_OP, DATE_OP, MONTANT, #NUM_CPT)

Aperçu d'une vue de l'application web :

The screenshot shows a web application interface titled "Gestion des comptes".

Header: Personne 1, admin, Logout.

Navigation: Operations, Comptes, Client.

Consultation d'un compte: Code Cpte: c1, OK.

Informations sur le compte:

- Client: Hassan
- Code Cpte: c1
- Solde: 90333.0
- Date création: 2016-12-28 20:37:20.0
- Type: CompteCourant
- Découvert: 800.0

Opérations sur le compte: Compte :c1
Versement, Retrait, Virement.
Montant : 0, Save.

Liste des opérations:

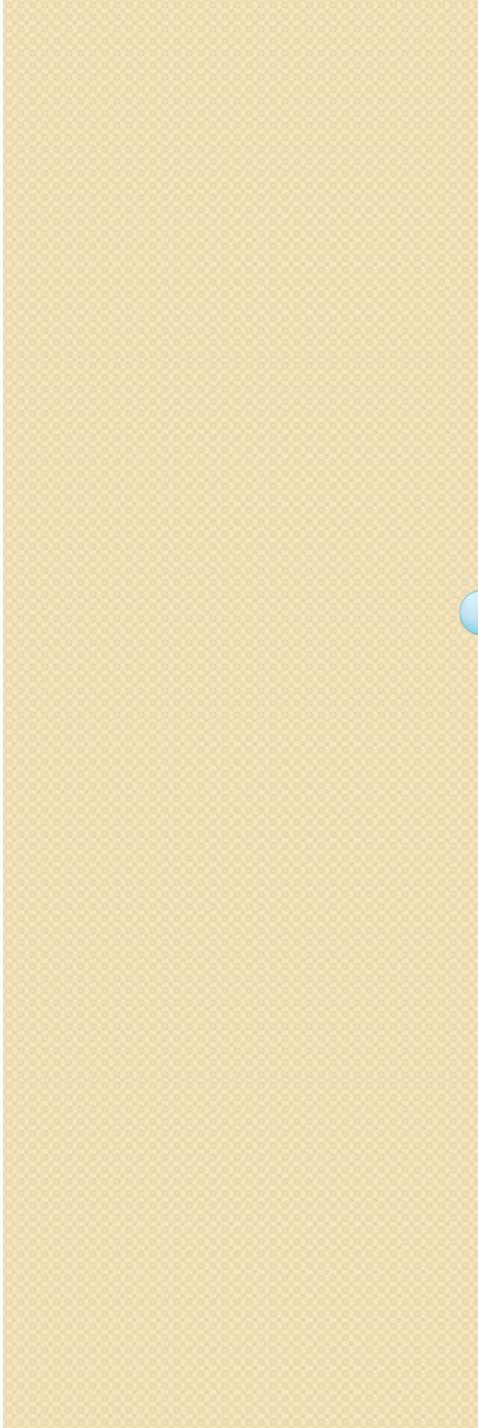
Num	Type	Date	Montant
33	Retrait	2016-12-31 14:09:15.0	8000.0
32	Versement	2016-12-31 14:09:08.0	9000.0
31	Versement	2016-12-31 14:09:01.0	0.0
29	Retrait	2016-12-31 12:55:21.0	800.0
27	Retrait	2016-12-31 12:54:57.0	90.0

Page navigation: 0, 1, 2, 3.

Copyright @2016

System tray icons: Windows, Internet Explorer, FileZilla, VLC, Google Chrome, Java EE IDE, Microsoft Word, Microsoft PowerPoint.

System status: FR, 14:10, 31/12/2016.



• **COUCHE DAO**

med@youssf.net

Entité JPA Client

```
@Entity
public class Client implements Serializable{
    @Id @GeneratedValue
    private Long code;
    private String nom;

    @OneToMany(mappedBy="client",fetch=FetchType.LAZY)
    private Collection<Compte> comptes;

    // Générer un constructeur sans param
    public Client() { super(); }

    // Générer un constructeur avec params
    public Client(String nom) { super();
        this.nom = nom;
    }

    //Générer les getters et setters
}
```

Entité JPA Compte

```
@Entity
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn( name="TYPE_CPTE",
discriminatorType=DiscriminatorType.STRING,length=2)
public abstract class Compte implements Serializable {

    @Id
    private String code;
    private double solde;
    private Date dateCreation;

    @ManyToOne
    @JoinColumn(name="CODE_CLI")
    private Client client;
    @OneToMany(mappedBy="compte")
    private Collection<Operation> operations;

    // Générer un constructeur sans param
    public Compte() { super(); }

    // Générer un constructeur avec params
    public Compte(String code, double solde, Date dateCreation, Client client) {
        super(); this.code = code; this.solde = solde;
        this.dateCreation = dateCreation; this.client = client;
    }

    //Générer les getters et setters
}
```

Entité JPA CompteCourant

```
@Entity  
@DiscriminatorValue("CC")  
public class CompteCourant extends Compte {  
    private double découvert;  
  
    // Constructeur sans param  
    public CompteCourant() { super(); }  
    // Constructeur avec params  
    public CompteCourant(String code, double  
        solde, Date dateCreation, Client client,  
        double découvert ) {  
        super(code, solde, dateCreation, client);  
        this.découvert = découvert;  
    }  
  
    // Getters et Setters  
  
}
```

Entité JPA CompteEpargne

```
@Entity  
@DiscriminatorValue("CE")  
public class CompteEpargne extends Compte {  
    private double taux;  
  
    // Constructeur sans param  
    public CompteEpargne() { super(); }  
    // Constructeur avec params  
    public CompteEpargne (String code, double solde, Date  
        dateCreation, Client client, double taux) {  
        super(code, solde, dateCreation, client);  
        this.taux = taux;  
    }  
  
    // Getters et Setters  
}
```

Entité Operation

```
@Entity  
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)  
@DiscriminatorColumn(name="TYPE",length=2)  
public abstract class Operation {  
    @Id @GeneratedValue  
    private Long numero;  
    private Date dateOperation;  
    private double montant;  
    @ManyToOne  
    @JoinColumn(name="CODE_CPTE")  
    private Compte compte;  
    // Constructeur sans param  
    public Operation() { super(); }  
    // Constructeur avec params  
    public Operation(Date dateOperation, double montant,  
    Compte compte) {  
        super();  
        this.dateOperation = dateOperation;  
        this.montant = montant;  
        this.compte = compte;  
    }  
    // Getters et Setters  
}
```

Entité Versement

```
@Entity  
@DiscriminatorValue("V")  
public class Versement extends Operation{  
    public Versement() { super(); }  
    public Versement(Date dateOperation,  
    double montant, Compte compte) {  
        super(dateOperation, montant, compte);  
    }  
}
```

Entité Retrait

```
@Entity  
@DiscriminatorValue("R")  
public class Retrait extends Operation{  
    public Retrait() { super(); }  
    public Retrait(Date dateOperation, double  
    montant, Compte compte) {  
        super(dateOperation, montant, compte);  
    }  
}
```

Interfaces DAO basées sur Spring Data

```
import org.springframework.data.jpa.repository.JpaRepository;
import ma.emsi.entities.Client;
public interface ClientRepository extends JpaRepository<Client, Long> {
}
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import ma.emsi.entities.Compte;
public interface CompteRepository extends JpaRepository<Compte, String>{
}
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import ma.emsi.entities.Operation;
public interface OperationRepository extends JpaRepository<Operation, Long>{
}
```

Test de la couche DAO

```
@SpringBootApplication
public class TpJpaApplication {
    public static void main(String[] args) {
        ApplicationContext ctx= SpringApplication.run(TpJpaApplication.class, args);
        ClientRepository clientRepository= ctx.getBean(ClientRepository.class);
        /* * Ajouter 3 clients */
        clientRepository.save(new Client("Hassan")); clientRepository.save(new Client("AAAA"));
        clientRepository.save(new Client("CCCCC"));
    }
}
```

code	nom
1	Hassan
2	AAAA
3	CCCCC

```
CompteRepository compteRepository= ctx.getBean(CompteRepository.class);
/* * consulter un client sachant son code */
Client c1=clientRepository.findOne(1L);
/* * Créer 2 comptes : un courant et l'autre épargne */
compteRepository.save( new CompteCourant("c1", 9000, new Date(), c1,800.0));
compteRepository.save( new CompteEpargne("c2", 7800, new Date(), c1,5.0));
```

type_cpte	code	date_creation	solde	découvert	taux	code_cli
CC	c1	2016-11-25 15:44:09	9000	800	NULL	1
CE	c2	2016-11-25 15:44:09	7800	NULL	5	1

```
OperationRepository operationRepository=ctx.getBean(OperationRepository.class);
/* * Consulter un compte sachant son code */
Compte cp=compteRepository.findOne("c1");
/* * Effectuer des versements et des retraits sur le compte c1 */
operationRepository.save(new Versement(new Date(),6000,cp));
operationRepository.save(new Versement(new Date(),7000,cp));
operationRepository.save(new Retrait(new Date(),5400,cp));
}}
```

type	numero	date_operation	montant	code_cpte
V	1	2016-11-25 15:44:09	6000	c1
V	2	2016-11-25 15:44:09	7000	c1
R	3	2016-11-25 15:44:09	5400	c1

TP Spring MVC, JPA, Hibernate et Spring Data

- On souhaite créer une application Web JEE qui permet de gérer les taxes relatives à des sociétés. Chaque Taxe concerne une entreprise. Dans l'application on traite deux types des taxes : TVA (Taxe sur la Valeur Ajoutée) ou IR (Impôt sur le Revenu). Une taxe est définie par son numéro (auto-incrémenté), le titre de la taxe, sa date, le montant de la taxe. Une entreprise est définie par son code, son nom, sa raison sociale et son email.
- L'application doit permettre de :
 - Gérer les entreprises (ajouter, éditer, mettre à jour, consulter, supprimer, chercher) :
 - Gérer les taxes (ajouter, éditer, mettre à jour, consulter, supprimer, chercher)
- Les exigences techniques de l'application sont :
- Les données sont stockées dans une base de données MySQL
- L'application se compose de deux couches:
 - La couche DAO qui est basée sur Spring Data, JPA, Hibernate et JDBC.
 - La couche Web basée sur MVC coté Serveur en utilisant Thymeleaf.
- L'inversion de contrôle est basée sur Spring IOC.



Travail demandé

1. Créer un projet Spring Starter avec les dépendances JPA, MySQL, Web et Thymeleaf.
2. Couche DAO
 - a) Créer les entités JPA
 - b) Créer les interfaces JPA Repository basées sur Spring Data
 - c) Tester quelques opérations de la couche DAO
3. Créer une application Web JEE basée sur Spring MVC qui permet de chercher les entreprises dont le nom contient un mot clé saisi dans une zone de texte avec un lien qui permet de consulter les taxes d'une entreprise.
 - a) Créer le contrôleur
 - b) Créer les vues basée sur Thymeleaf

Architecture Technique

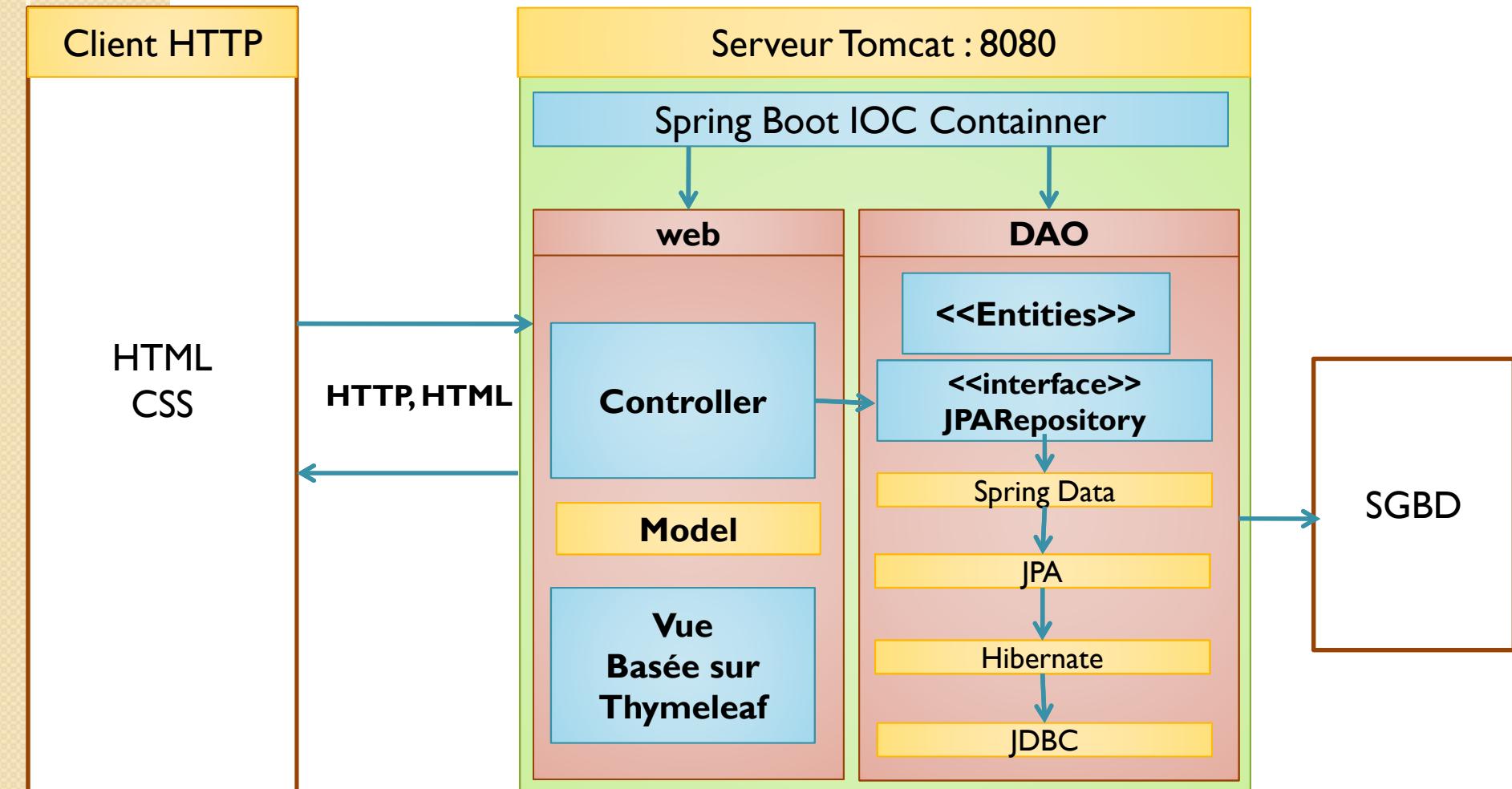


Diagramme de classes

