



## *Développement JEE*

### *Le Framework Spring Boot*

## *Chapitre 3 : Introduction à Spring Boot*

**Sabeur ELKOSANTINI**

Maitre de conférences en informatique

Sabeur.Elkosantini@fsegn.ucar.tn

<https://sabeur.elkosantini.me>

1

## Plan

- Chapitre 1 : Introduction aux architecture JEE
- Chapitre 2 : Introduction à Spring boot
- Chapitre 7 : Framework SpringBoot : Spring Data JPA
- Chapitre 8 : Framework SpringBoot : Formulaire
- Chapitre 9 : Framework SpringBoot : Rest et services
- Chapitre 10 : Framework SpringBoot : Session et sécurité

2

## Le modèle MVC

### Introduction

- Objectif :
  - organiser une application interactive en séparant :
    - ✓ les données (Modèle)
    - ✓ la représentation des données (Vues)
    - ✓ le comportement de l'application (Contrôle)

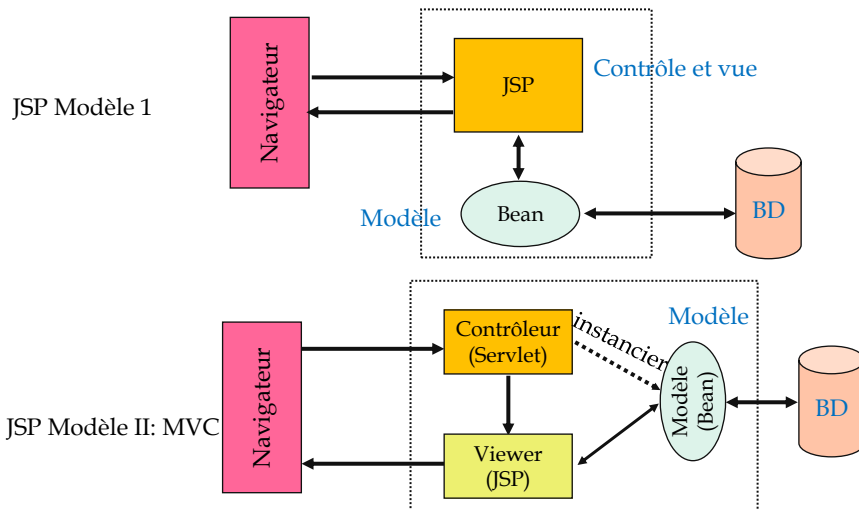
## Le modèle MVC

### Introduction

- Le modèle (M) représente la structure des données dans l'application
- La vue (V) présente les données sous une certaine forme à l'utilisateur.
- Le contrôleur (C) traduit les interactions utilisateur par des appels de méthodes sur le modèle et sélectionne la vue appropriée.

## Le modèle MVC

### 👉 MVC pour les applications web



S. Elkosantini

5

5

## Frameworks MVC

### 👉 Les frameworks

- Framework = cadre de développement.
- Un Framework peut inclure :
  - ✓ Un ensemble de classes généralement regroupées sous la forme de bibliothèques pour proposer des services plus ou moins sophistiqués
  - ✓ Un cadre de conception reposant sur les design patterns pour proposer tout ou partie d'un squelette d'application
  - ✓ Des recommandations sur la mise en œuvre et des exemples d'utilisation
  - ✓ Des normes de développement
  - ✓ Des outils facilitant la mise en œuvre

S. Elkosantini

6

6

## Frameworks MVC

### 👉 Les frameworks

#### Mais pourquoi utiliser les frameworks



- L'objectif d'un framework est de faciliter la mise en œuvre des fonctionnalités de son domaine d'activité
- Il doit permettre au développeur de se concentrer sur les tâches spécifiques de l'application à développer plutôt que sur les tâches techniques récurrentes.

## Frameworks MVC

### 👉 Les frameworks

- Exemples de tâches techniques récurrentes telles que :
  - ✓ l'architecture de base de l'application
  - ✓ l'accès aux données
  - ✓ l'internationalisation
  - ✓ la journalisation des événements (logging)
  - ✓ la sécurité (authentification et gestion des rôles)
  - ✓ le paramétrage de l'application
  - ✓ ...

## Frameworks MVC

### 👉 Les frameworks

- La mise en œuvre d'un framework permet ainsi de :
  - ✓ Capitaliser le savoir-faire sans "réinventer la roue"
  - ✓ Accroître la productivité des développeurs une fois le framework pris en main
  - ✓ Homogénéiser les développements des applications en assurant la réutilisation de composants fiables
  - ✓ Faciliter la maintenance notamment évolutive des applications

## Frameworks MVC

### 👉 Les frameworks

- Types de frameworks:
  - ✓ Frameworks MVC: Struts, spring, JavaServer Faces, Stripes, etc.
  - ✓ Framework persistances: Hibernate, JAXB, iBATIS, etc.

Le framework MVC Spring Boot

## Frameworks MVC

### 👉 Le framework Spring

- Basé sur l' API Servlet de Java JEE
- Permettant de simplifier le développement d'applications web en respectant le patron de conception MVC 2

#### Problèmes

- trop de dépendance à gérer (ce qui pose souvent un problème d'incompatibilité entre les versions)
- beaucoup de configuration (JPA, sécurité, contrôleurs, vues...)

11

## Frameworks MVC

### 👉 Le framework Spring boot

- Architecture Spring Boot

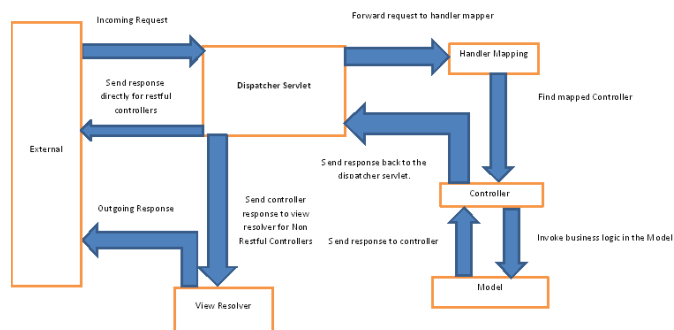


Fig 1 MVC Architecture Flow

12

# Frameworks MVC

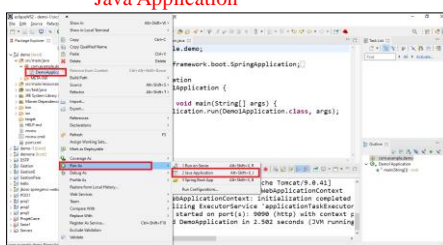
## Le framework Spring boot

- Pour éviter les problèmes de Spring, **Spring Boot** propose :
  - ✓ Les **démarrers** (starter) : un démarrage est une dépendance, contenant un paquet de dépendance, permettant de réaliser un type de projet (Web, Rest...). Ainsi, le développeur n'a plus à gérer, lui-même le problème d'incompatibilité entre les versions.
  - ✓ l'**auto-configuration** : c'est-à-dire laisser **Spring Boot** configurer le projet à partir de dépendances ajoutées par le développeur

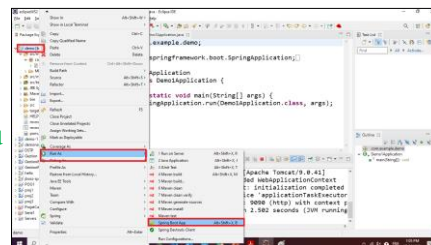
# Frameworks MVC

## Le framework Spring boot

- Pour exécuter le projet sous Eclipse:
  - Ajouter le plugin **Spring tools** (ou utiliser <https://start.spring.io/>) pour créer le projet
  - Faire un clic droit sur le projet et aller dans **Run As** et cliquer sur **Spring Boot App** ou faire un clic droit sur la classe **xxxApplication** dans **Package Explorer**, aller dans **Run As** et cliquer sur **Java Application**



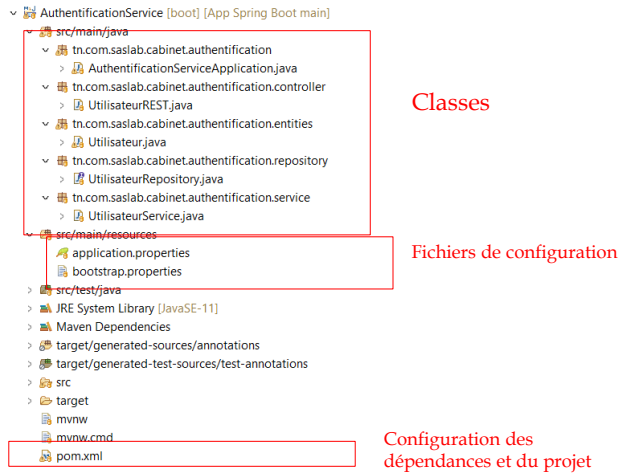
OU



## Frameworks MVC

### Le framework Spring boot

#### Structure d'un projet Spring Boot (Sous eclipse)



Classes

Fichiers de configuration

Configuration des dépendances et du projet

S. Elkosantini

15

15

## Frameworks MVC

### Le framework Spring boot

#### Configuration des dépendances (pom.xml)

- ✓ Exemple: pour créer un projet web, il faut ajouter la dépendance suivante

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

- La dépendance **spring-boot-starter-web** permet donc de créer un projet web contenant :
  - ✓ un serveur Apache Tomcat
  - ✓ Spring Framework et Spring MVC
  - ✓ les validateurs d'Hibernate
  - ✓ jackson pour les données au format JSON
  - ✓ ...

S. Elkosantini

16

16



# Frameworks MVC

## Le framework Spring boot

### Configuration des dépendances (pom.xml)

```
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.7.0</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>tn.com.saslab.cabinet</groupId>
<artifactId>authentication</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<name>AuthenticationService</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>11</java.version>
  <start-class>tn.com.saslab.cabinet.authentication.AuthenticatationServiceApplication</start-class>
  <spring-cloud.version>2021.0.3</spring-cloud.version>
</properties>
```

La version de Maven utilisé

Le projet utilise ici le parent "spring-boot-starter-parent" de Spring Boot en version 2.7.0

Le projet hérite des configurations de dépendances et des plugins Maven définis dans ce parent

Des informations sur le projet

Des informations complémentaires du projet

# Frameworks MVC

## Le framework Spring boot

### Configuration des dépendances (pom.xml)

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```

Liste des dépendances

Le groupId et l'artifactID de la dépendance

## Frameworks MVC

### 👉 Le framework Spring boot

#### ■ Configuration des dépendances (pom.xml)

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-dependencies</artifactId>
      <version>${spring-cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Spécifie la configuration du processus de construction (build) du projet.

Le groupId et l'artifactID de la dépendance

Configuration la gestion des dépendances

## Frameworks MVC

### 👉 Le framework Spring boot

Le point de démarrage de l'application: classe de bootstrapping

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Annotation indiquant que c'est la classe principale

Démarre le conteneur Spring et renvoi l'objet *ApplicationContext*

Définition d'un Bean

## Frameworks MVC

### 👉 Le framework Spring boot

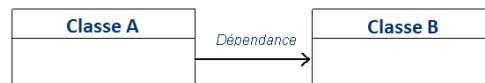
- Les annotations pour définir les beans :
  - ✓ @Component, @Service, or @Repository (chapters suivants)
  - ✓ @Configuration et utiliser @Bean pour chaque méthode de la classe

21

## Frameworks MVC

### 👉 Inversion de dépendances

- Toute application Java est une composition d'objets qui collaborent pour rendre le service attendu



Implémentation

```
public class A {
    public static void main(String[] args) {
        B b = new B();
        b.someMethod();
    }
}
```

Problèmes ??

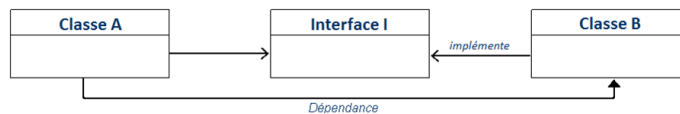
22

## Frameworks MVC

### 👉 Inversion de dépendances

L'objectif est de diminuer le couplage entre deux ou plusieurs objets

- Solution: Injection de dépendances par les interfaces



```
public class A {
    public static void main(String[] args) {
        I b = new B();
        b.someMethod();
    }
}
```

23

## Frameworks MVC

### 👉 Inversion de dépendances

- Exemple

Qui va se charger de créer de cette instance ?

```
public class ReservationSalleService {
    private ReservationSalleDao reservationSalleDao;

    public void reserver(ReservationSalle reservationSalle) {
        // faire un traitement nécessaire
        // (par exemple la validation de la réservation)

        // sauvegarder la réservation
        reservationSalleDao.sauver(reservationSalle);
    }
}
```

24

## Frameworks MVC

### 👉 Inversion de dépendances

#### ■ Exemple : solution 1

```
public class ReservationSalleService {  
    private ReservationSalleDao reservationSalleDao;  
    public ReservationSalleService() {  
        reservationSalleDao = new ReservationSalleDao();  
    }  
    public void reserver(ReservationSalle reservationSalle) {  
        // faire un traitement nécessaire  
        // (par exemple La validation de la réservation)  
  
        // sauvegarder la réservation  
        reservationSalleDao.sauver(reservationSalle);  
    }  
}
```

Initialisation dans un constructeur

Problème de couplage fort

## Frameworks MVC

### 👉 Inversion de dépendances

#### ■ Exemple : solution 2

*Injection de la dépendance par le constructeur*

```
public class ReservationSalleService {  
    private ReservationSalleDao reservationSalleDao;  
    public ReservationSalleService(ReservationSalleDao reservationSalleDao) {  
        this.reservationSalleDao = reservationSalleDao;  
    }  
    public void reserver(ReservationSalle reservationSalle) {  
        // faire un traitement nécessaire  
        // (par exemple La validation de la réservation)  
  
        // sauvegarder la réservation  
        reservationSalleDao.sauver(reservationSalle);  
    }  
}
```

## Frameworks MVC

### 👉 Inversion de dépendances dans spring boot : Spring IoC

- Inversion of Control (IoC): Injection par mutateur
  - ✓ Le flux de contrôle est orienté du code tiers vers le code de l'application
- Retour à l'exemple: Besoin d'un composant qui :
  - ✓ Créé une instance des classes ReservationSalleDao et ReservationSalleService
  - ✓ Réaliser l'injection de l'objet de type ReservationSalleDao

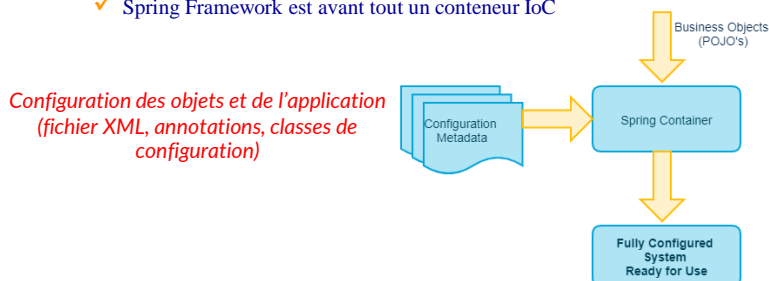
Conteneur IoC (IoC container).

27

## Frameworks MVC

### 👉 Inversion de dépendances dans spring boot : Spring IoC

- Inversion of Control (IoC): Injection par mutateur
- Accueille un ensemble d'objets dont il a la responsabilité de gérer le cycle de vie:
  - ✓ Spring Framework est avant tout un conteneur IoC



28

## Frameworks MVC

### 👉 Le framework Spring boot

- Pour exécuter le projet sous Eclipse:

```
2021-01-17 13:00:50.328 INFO 10368 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9090 (http) with context path "
```

- Pour changer le port par défaut :

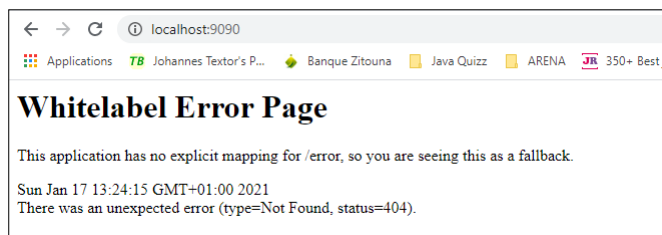
Ajouter la ligne suivante dans le fichier `resources/application.properties`

```
server.port=9090
```

## Frameworks MVC

### 👉 Le framework Spring boot

- Pour exécuter le projet sous Eclipse: Aller à <http://localhost:9090/>



#### Résultat : message d'erreur

- On a créé un projet web, mais on n'a aucune page **HTML** ni **JSP**.
- **Spring Boot**, comme **Spring MVC**, implémente le patron de conception **MVC 2**, donc il nous faut au moins un contrôleur et une vue.

## Frameworks MVC

### 👉 Le framework Spring boot

- Le contrôleur:
  - ✓ un des composants du modèle MVC
  - ✓ une classe Java annotée par **Controller** (ou **RestController**)
  - ✓ Il reçoit une requête du contrôleur frontal et communique avec le modèle pour préparer et retourner une réponse

31

## Frameworks MVC

### 👉 Le framework Spring boot

- Le contrôleur:

```
package tn.itbs.app.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller ← Pour indiquer qu'il s'agit d'un contrôleur
public class appControl {
    @GetMapping(value = "/index") ← La méthode appelé par GET et
        public void sayHello() { ← avec la route « index »
            System.out.println("Hello World!");
        }
}

@GetMapping(value = {"/index", "/"}) ← Plusieurs routes pour la même
                                     méthode
```

32



## Frameworks MVC

### 👉 Le framework Spring boot

- Le contrôleur:

Pensez à ajouter DevTools permettant de redémarrer le projet après chaque changement

Aller à Spring > Add DevTools

➔ Une nouvelle dépendance est ajoutée dans pom.xml



```
<dependency>  
<groupId>org.springframework.boot</groupId> pom.xml  
<artifactId>spring-boot-devtools</artifactId>  
</dependency>
```

## Frameworks MVC

### 👉 Le framework Spring boot

- Les vues

- ✓ Permettent d'afficher des données
- ✓ Communiquent avec le contrôleur pour récupérer ces données
- ✓ Doivent être créés dans le répertoire views dans src/main/webapp/ (qu'il faut le créer).

Ajouter les dépendances : spring-boot-starter-web et tomcat-embed-jasper

```
<dependency>  
  <groupId>org.apache.tomcat.embed</groupId>  
  <artifactId>tomcat-embed-jasper</artifactId>  
</dependency>
```

Faire la mise à jour du projet après l'ajout des nouvelles dépendances  
Maven > Update project

## Frameworks MVC

### 👉 Le framework Spring boot

#### ■ Les vues

Ajouter les dépendances à Apache Tomcat et JSTL dans pom.xml  
(voir lien <https://mvnrepository.com/>)

Par exemple JSTL :

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```

## Frameworks MVC

### 👉 Le framework Spring boot

#### ■ Les vues

##### Remarques

- On peut préciser un autre répertoire pour les vues (à placer aussi dans webapp)
- Pour éviter de préciser chaque fois l'extension et l'emplacement de la vue, on peut l'indiquer dans `application.properties` situé dans `src/main/resources`

Nouveau contenu d'`application.properties`

```
spring.mvc.view.prefix=/views/
spring.mvc.view.suffix=.jsp
```

Toutes les propriétés possibles de `application.properties` sont disponibles ici :  
<https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>

## Frameworks MVC

### 👉 Le framework Spring boot

- Les vues

Nouveau contenu du contrôleur

```
package tn.itbs.app.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class appControl {
    @GetMapping(value = "/index")
    public String sayHello() {
        return "index";
    }
}
```

Enlever l'extension .jsp

S. Elkosantini

37

37

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs

Coté contrôleur

```
package tn.itbs.app.controller;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AppControl {
    @GetMapping(value = "/index")
    public String sayHello(Model model) {
        model.addAttribute("nomP", "PC");
        return "index";
    }
}

↑
redirect:/
```

Une interface qui permet d'envoyer des attributs à la vue

Redirection vers la racine

S. Elkosantini

38

38

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs  
Côté Vue

```
<body>
<h1>Gestion de stock</h1>
Le nom du produit est ${ nomP }

</body>
```

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs  
Côté contrôleur (ou en utilisant l'interface ModelAndView)

```
public ModelAndView sayHello() {
    ModelAndView mv= new ModelAndView("index");
    mv.addObject("nomP", "PC");
    return mv;
}
```

Une classe qui permet  
d'indiquer à la fois le modèle  
et la vue

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs

Lecture de paramètres des requêtes (/chemin?param1=value1)

```
public ModelAndView sayHello(@RequestParam(value="nom") String
nom) {
    ModelAndView mv= new ModelAndView("index");
    mv.addObject("nomP", nom);
    return mv;
}
```

permet de récupérer la valeur du paramètre de la requête HTTP est de l'affecter au paramètre nom de la méthode

41

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs

Lecture de paramètres des requêtes (/chemin?param1=value1)

Et si on ne passe pas de paramètres ?

```
public ModelAndView sayHello(@RequestParam(value="nom", required =
false, defaultValue = "PC") String nom) {
    ModelAndView mv= new ModelAndView("index");
    mv.addObject("nomP", nom);
    return mv;
}
```

Valeur par défaut

Paramètre requis

42

## Frameworks MVC

### 👉 Le framework Spring boot

- Echange les beans entre les vues et les contrôleurs

Lecture de variables de chemins (/chemin/value)

```
@GetMapping(value = "/afficheProduit/{nom}")
public ModelAndView afficheProduit(@PathVariable String nom) {
    ModelAndView mv= new ModelAndView("index");
    mv.addObject("nomP", nom);
    return mv;
}
```

↑  
variables de chemins

Tester avec <http://localhost:9090/afficheProduit/PCC>

### Gestion de stock

Le nom du produit est PCC