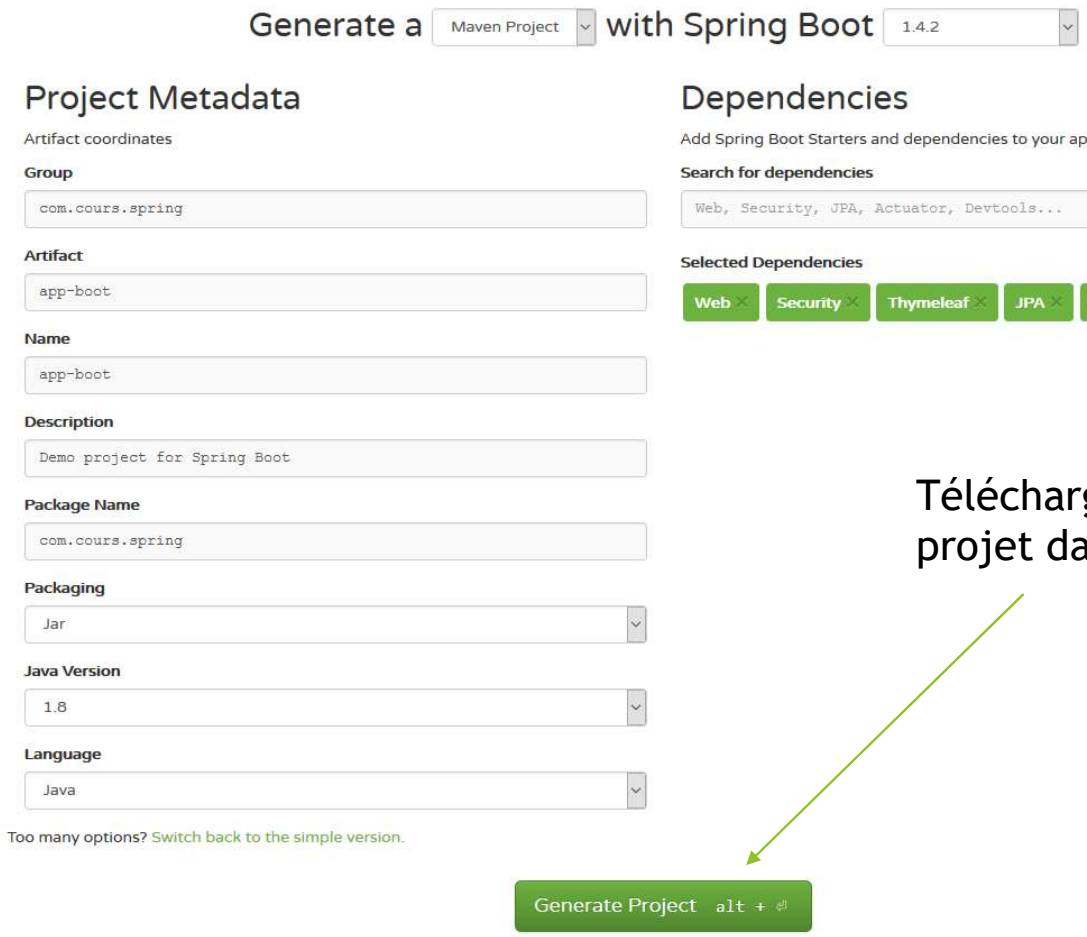
Spring MVC

Spring Boot

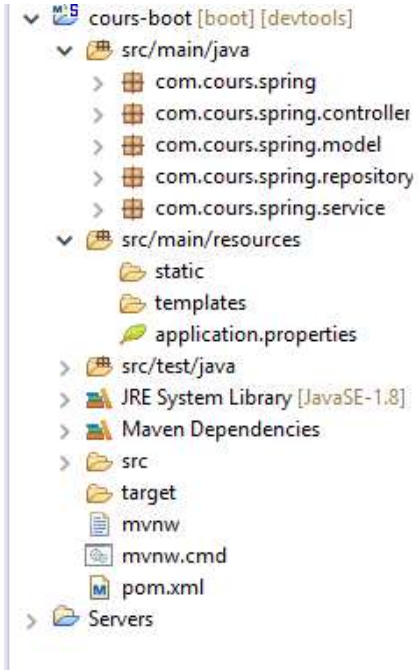
- CoC (Convention Over Configuration)

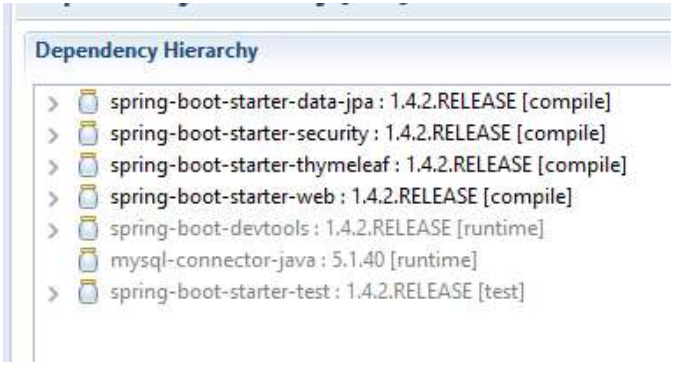
Spring Initializr



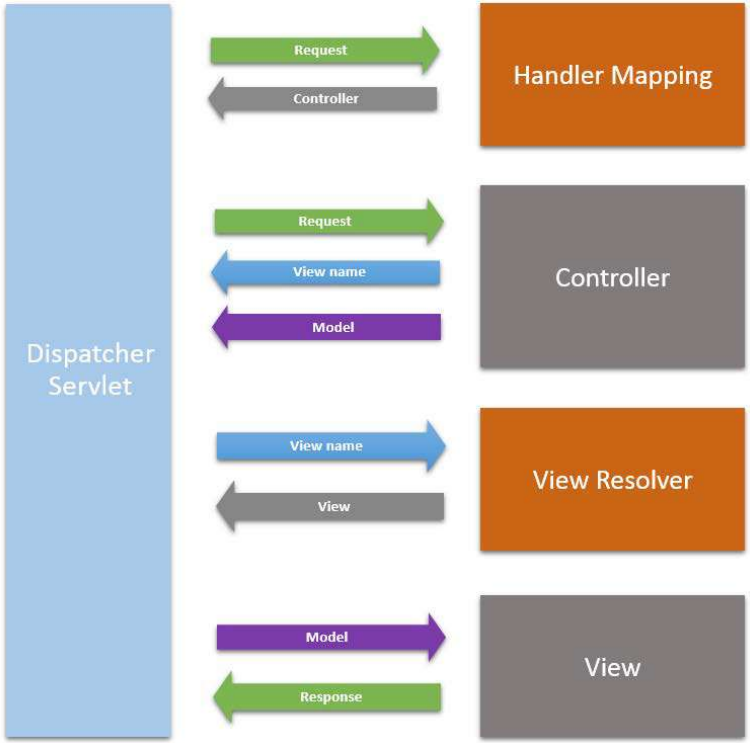
Télécharger et importer le projet dans STS

Structure du projet





SPRING MVC



- DispatcherServlet: Front Controller, c'est le point d'entrée d'une application web MVC.
- HandlerMapping gère les associations entre les urls et les contrôleurs.
- La méthode avec le @RequestMapping qui correspond à l'url est appelée.
- La méthode appelée doit définir le modèle et sélectionner une vue.
- DispatcherServlet Interroge l'interface ViewResolver pour trouver l'mplémentation qui correspond à la vue.
- Par exemple pour thymeleaf, si le nom de la vue est « index », alors DispatcherServlet recherche dans le dossier templates la page index.html.

Les contrôleurs

- Un contrôleur doit être annoté par @Controller
- @RequestMapping: Définit la correspondance entre le contrôleur ou une action du contrôleur et une url.
  - Attributs de @RequestParam
    - value:url
    - method: post,get,delete,put,head
    - params: sélection de la méthode selon l'existence ou non de paramètres (valeurs possibles, p1=v1 ou p1!=v1 ou !p1
  - Spring 4.3 introduit aussi les annotations GetMapping, PostMapping, PutMapping, DeleteMapping

Contrôleur

```java
import org.springframework.stereotype.Controller;

@Controller
public class RessourceController {

    @RequestMapping(value = "/", method = RequestMethod.GET)
    @ResponseBody
    public String ressource() {
        return "OOO";
    }

}
```

- Arguments supportés par l'action
  - @RequestParam(name="p1",defaultvalue="v1",value="v1")
  - @ModelAttribute: lier un argument à un objet du modèle
  - @CookieValue: récupérer une variable de type Cookie
  - @PathVariable

- Types de retour supportés:
  - ModelAndView: contient le modèle et le nom de la vue à afficher.
  - Model
  - View
  - String : nom de la vue

@ModelAttribute

```
@PostMapping("/users/ajouter")
public String ajouterUser(@ModelAttribute user) {
return null;
}
```

Si l'objet user est présent dans le modèle, ses champs seront remplis à partir des champs de la requête http ayant les mêmes noms, sinon un nouvel objet user sera créé.

Un second argument de type BindingResult peut être utilisé pour détecter éventuellement les erreurs de conversion (la méthode hasErrors() à cet effet)

@PathVariable

```
@GetMapping("/users/{userId}/commentaires/{postId}")
public User getCommentaires(@PathVariable Long userId,
@PathVariable Long postId) {


…;
}
```

Le contrôleur

```java
package com.cours.spring.controller;

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

@Controller
public class HomeController {
    @Autowired
    private RessourceService ressourceService;

    @RequestMapping(name = "/add", method = RequestMethod.GET)
    @ResponseBody
    public void ajouter(@RequestParam String n) {
        Ressource r = new Ressource();
        r.setNom(n);
        ressourceService.ajouterRessource(r);

    }
}
```

Exemple contrôleur sans vue

```
@Controller
@RequestMapping("/users")
public class UserController {

@Autowired
UserRepository userRepository;

@GetMapping("/")
@ResponseBody
public List<User> afficher() {

return userRepository.findAll();
}

//Ajout d'un utilisateur

// localhost:9911/users/ajouter?nom=u&mail=aa@users.com
@GetMapping("/ajouter")
@ResponseBody
public void ajouter(@RequestParam String
nom,@RequestParam("mail") String email) {
User u=new User(nom,email);
userRepository.save(u);
}

//ModelAttribute
//localhost:9911/users/ajouter2?nom=u&email=aa@users.com
@GetMapping("/ajouter2")
@ResponseBody
public void ajouter(@ModelAttribute User u) {

userRepository.save(u);
}
//localhost:9911/users/ajouter3/u1/aaa@users.com
@GetMapping("/ajouter3/{user}/{email}")
@ResponseBody
public void ajouter3(@PathVariable("user") String
nom,@PathVariable String email) {
User u=new User(nom,email);
userRepository.save(u);
}


}
```

Thymeleaf

- Moteur de vue réalisé en 2014
- Écrit en Java et supporte XML/XHTML/HTML5
- Plusieurs extensions sont disponibles depuis Spring Boot.
- Pour intégrer Thymeleaf à l'aide Spring Boot il faut ajouter la dépendance:

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

- Pour intégrer Thymleaf dans une page html:

```
<html lang="fr" xmlns:th="http://www.thymeleaf.org">
```

- Afficher du texte: `<p th:text="'Hello, ' + ${name} + '!'" />`
- *Une boucle:* `<li th:each="ressource : ${ressources}“ th:text="${ressource.nom}"/>`

Exemple 1: Liste des utilisateurs

```
<!DOCTYPE html>
<html lang="fr" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<title>Ressources</title>
</head>
<body>
<h2>Ressources</h2>
<table border="1">
<tr>
<th>Id</th>
<th>Nom</th>
</tr>
<tr th:each="r: ${users}">
<td th:text="${r.id}“/>
<td th:text="${r.nom}“/>
</tr>
</table>
</body>
</html>
```

Exemple 2: Ajout

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8"/>

<title>Ajout d'un intervenant</title>

</head>

<body>

<form action="/users/add" method="post">

<div><span>Id:</span><input type="text" name="id" /></div>

<div><span>Nom:</span><input type="text" name="nom" /></div>

<input type="submit" value="Enregistrer" />

</form>

</body>

</html>html>>
```

Le contrôleur

```
@Controller
@RequestMapping("/")
public class HomeContorller {

@Autowired
UserService userService;

/* Affichage du formulaire d'ajout */
@RequestMapping(value = "/add", method = RequestMethod.GET)
public String ajouter() {
return « users/ajout";
}

/* Enregistrement */
@RequestMapping(value = "/add", method = RequestMethod.POST)
public String ajouter(@ModelAttribute User u, BindingResult result) {
if (result.hasErrors())
return « users/ajout";

userService.ajoutere(u);

return "redirect:users/liste";
}

/*
 * Action index qui retourne la liste des utilisateurs
 */
@RequestMapping(value = "/liste", method = RequestMethod.GET)
public String index(Model m) {
m.addAttribute(« users", ressourceService.getUsers());

return « users/index";
}
}
```

SpEL

- http://docs.spring.io/spring/docs/current/spring-framework-reference/html/expressions.html

Accès à un élément à partir d'une liste: `list[0]`
Accès à un élément à partir de sa clé: `map[key]`
Opérateur ternaire: `condition ? 'yes' :'no'`
Opérateur Elvis: `pers ?: default` Retourne `pers si pers est non null sinon retourne default`
`pers?.nom` Returns `null` if pers ou pers.nom est null
Evaluation différée: `Nom:#{pers.nom}'`
Injection de valeurs dans une chaine: `${pers.![nom]}` extrait les nom d'une collection pers et les affiche dans chaîne
Sélection: `pers.?[nom =='Bob']`
Appel de méthode: `pers.methode()`