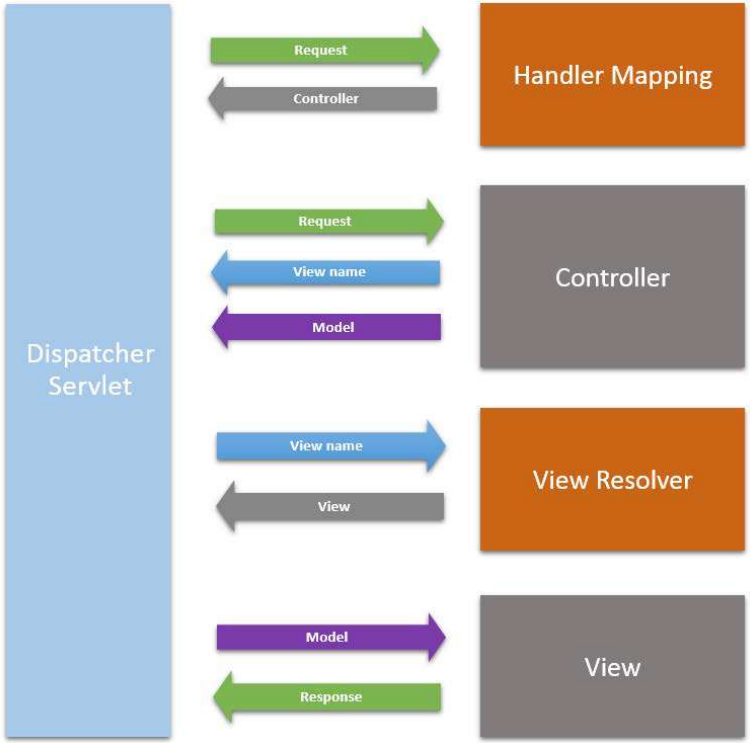
Spring MVC

Thymeleaf

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.

Contrôleur

```
4
5  @Controller("mvcProjet")
6  @RequestMapping("/mvcprojets")
7  public class ProjetController {
8
9      @Autowired
0      private ProjetService projetService;
1
2      @GetMapping
3      public String liste(Model m) {
4          m.addAttribute("projets", projetService.lesProjets());
5          return "projets";
6      }
7
```

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

- Un contrôleur doit être annoté par @Controller
- @RequestMapping: Définit la correspondance entre le contrôleur ou une action du contrôleur et une url.
- Attributs de @RequestMapping
  - 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)
- On peut utiliser raccourcis @GetMapping,PostMapping, PutMapping, DeleteMapping au lieu de RequestMapping

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

ModelAndView

- ModelAndView

```
*/
@GetMapping
public ModelAndView liste() {
    ModelAndView m=new ModelAndView("projets");
    m.addObject("projets", projetService.lesProjets());
    return m;
}
```

- action(@ModelAttribute Projet projet): si un projet existe dans le modèle, alors il sera récupéré sinon il sera instancié et ajouté dans le modèle.

Thymeleaf

Moteur de vue réalisé en 2014

Écrit en Java et supporte XML/XHTML/HTML5

- 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>
```

Eléments Thymeleaf

Thymeleaf supporte les expressions suivantes:

- *Expressions SpEL (Spring Expression language : ${expression}*
- *Expressions de sélection : * {…}*
- *Expression de liens : @{…}*
- *messages d'internationalisation: #{}*

Expressions SpEL

- Afficher du texte:
  - `<p th:text="'Hello, ' + ${nom} + '!'" />`
  - `<p th:text="${tache.projet.description}" />`

- *Une boucle:* `<li th:each="p : ${projets}" th:text="${p.description}"/>`

Expressions de sélection

Une expression de sélection est similaire à une expression SPEL, à l'exception qu'elle est exécutée sur une variable déjà sélectionnée dans la page, plutôt qu'une variable du modèle

```
<div th:object="${projet}">
    <span th:text="*{description}">...</span>
</div>
```

Expression de liens

@{} génère une url en prenant en compte le contexte de l'application web (le chemin racine), et éventuellement les informations de session (jsessionid=).

Par défaut le chemin racine est vide, la propriété server.servlet.context-path permet de définir le chemin racine:

```
server.servlet.context-path=/appmvc
```

code HTML généré

```
<a th:href="@{/projets}">Chemin avec th:href</a>
<a href="/projets">Sans th:href</a>
```

```
</div>
<a href="/appmvc/projets">Chemin avec th:href</a>
<a href="/projets">Sans th:href</a>
```

URL avec paramètres

```
<a th:href="@{https://www.google.com/search(q='thymeleaf',tbs='qdr:y',lr=lang_fr)}">Recherche: Thymeleaf</a>
```

code HTML généré

```
<a href="https://www.google.com/search?q=thymeleaf&amp;tbs=qdr:y&amp;lr=lang_fr">Recherche: Thymeleaf</a>
```

Expression de liens

- Variable de chemin dans une url

```
<a th:href="@{/projets/{idP}/tache/{idT} (idP=${projet.idProjet},idT=${tache.idTache})}">Ajouter Tache</a>
```

code HTML généré.

```
<div>
    <a href="/appmvc/projets/55/tache/0">Ajouter Tache</a>
</div>
```

Liste des projets

- Action

```
@GetMapping
public String liste(Model m) {
    m.addAttribute("projets", projetService.lesProjets());
    return "projets";
}
```

Ou bien

```
@GetMapping
public ModelAndView liste() {
    ModelAndView m = new ModelAndView("projets");
    m.addObject("projets", projetService.lesProjets());
    return m;
}
```

↳ nom de la vue projets.html

- Vue

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

<head>
    <title>Les projets</title>
</head>

<body>
    <h1>Les projets</h1>
    <a th:href="@{/projets/add}">Ajouter</a>
    <table>
        <thead>
            <tr>
                <th>Id</th>
                <th>Description</th>
                <th></th>
                <th></th>
            </tr>
        </thead>
        <tbody>

            <tr th:each="p:${projets}">
                <th th:text="${p.idProjet}">Id</th>
                <td th:text="${p.description}">Description</td>
                <td> <a th:href="@{/projets/edit/{id}(id=${p.idProjet})}" />Editer
                </td>
                <td> <a th:href="@{/projets/delete/{id}(id=${p.idProjet})}" />Supprimer</td>
            </tr>
        </tbody>
        </table>
</body>

</html>
```

Ajout/modification d'un projet

- Actions

```
@GetMapping("/add")
public String ajout(Model model) {
    Projet p = new Projet();
    model.addAttribute("projet", p);
    return "projets/ajout";

}

@PostMapping("/add")
public String enregistrer(@ModelAttribute("projet") Projet projet) {
    Projet p = projetService.getProjet(projet.getIdProjet());
    if (p != null)
        projetService.modifier(projet);
    else
        projetService.ajouter(projet);
    return "redirect:/mvcprojets";
}

@GetMapping("/edit/{id}")
public String modifier(@PathVariable long id, Model model) {
    Projet p = projetService.getProjet(id);
    if (p != null) {
        model.addAttribute("projet", p);
        return "projets/ajout";
    }

    return "redirect:/mvcprojets";
}
```

← affiche la vue pour ajouter un projet

enregistre un nouveau projet ou un projet modifié.

affiche la vue ajout pour modifier un projet

- Vue ajout.html

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

<head>
    <title>Ajout d'un projet</title>
</head>

<body>
    <form th:action="@{/projets/add}" th:object="${projet}" method="post">
        Id: <input readonly type="text" name="idProjet" th:field="*{idProjet}" />
        Description: <input type="text" name="description" th:field="*{description}" />
        <button type="submit">Enregistrer</button>
    </form>
</body>

</html>
```

```
@GetMapping("/delete/{id}")
public String suppr(@PathVariable int id) {
    projetService.supprimer(id);
    return "redirect:/mvcprojets";
}
```

Suppression d’un projet