Développement serveur Java

Moteur de rendu de document - Thymeleaf

Généralités

Architecture et fonctionnement

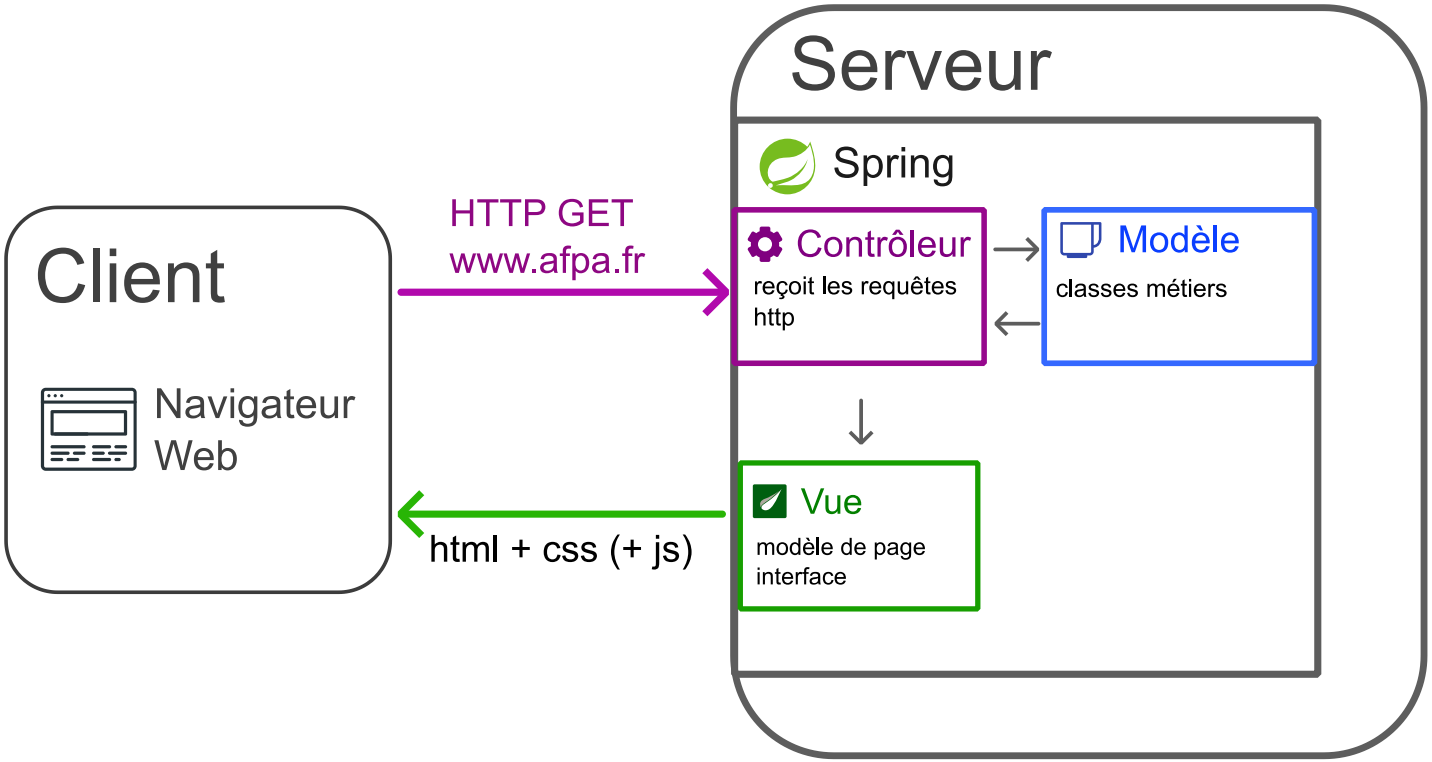
- Moteur de rendu de document :

→ Permet de construire un **document** sous un certain **format** à partir d'un **modèle** préconçu.

- Cas du développement Web :
- Document = page Web
- Format = vues html + css
- Modèle = propre au moteur de rendu

- Moteurs de rendu Java :

Architecture client-serveur "Server Side Rendering"



Le serveur génère le document

Exemple de rendu de vue (1/2)

Afpa

Modèle

```java
public class User {
    private String name;

    public User(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Contrôleur

```java
@Controller
public class UserController {
    @GetMapping("/users")
    public String home(Model model) {
        ArrayList<User> users = new ArrayList<>();

        User user1 = new User("Ada Lovelace");
        User user2 = new User("John von Neumann");

        users.add(user1);
        users.add(user2);

        model.addAttribute("users", users);

        return "users/index.html";
    }
}
```

Préparation des données disponibles pour la vue

Retourne le nom de la vue à générer

Vue

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Liste des utilisateurs</title>

    <link th:href="@{/styles/app.css}" rel="stylesheet" />
  </head>
  <body>
    <h1>Liste des utilisateurs</h1>
    <ul>
        <li th:each="user : ${users}" th:text="${user.name}"></li>
    </ul>
  </body>
</html>
```

Rendu dynamique de HTML en fonction de ce qui a été préparé par le contrôleur

Moteur de rendu Thymeleaf

```html
<html>
  <head>
    <meta charset="UTF-8">
    <title>Liste des utilisateurs</title>

    <link th:href="@{/styles/app.css}" rel="stylesheet" />
  </head>
  <body>
    <h1>Liste des utilisateurs</h1>
    <ul>
        <li th:each="user : ${users}" th:text="${user.name}"></li>
    </ul>
  </body>
</html>
```

Vue

```html
<html>
    <head>
        <meta charset="UTF-8">
        <title>Liste des utilisateurs</title>

        <link href="/styles/app.css" rel="stylesheet">
    </head>

    <body>
        <h1>Liste des utilisateurs</h1>
        <ul>
            <li>Ada Lovelace</li>
            <li>John von Neumann</li>
        </ul>
    </body>
</html>
```

HTML rendu

Template Thymeleaf

Attributs et SpEL

- # Template Thymeleaf
- Combinaison de code HTML et **d'expressions Thymeleaf**
- Inclut dynamiquement du contenu
- Fait le lien entre du HTML et du code Java (objets, beans, variables…)

- # Exemple :

```
<input type="text" name="userName" value="Ada Lovelace" th:value="${user.name}" />
```

Tag HTML

Attributs HTML

Expression Thymeleaf

Lien avec un objet

- # Lors du développement :
- L'extension de templates Thymeleaf est « .html »
- Mettre les fichiers de template dans `src/main/resources/templates`

- Attributs communs :

- `th:text` : permet d'injecter le texte de la balise HTML concernée
- `th:each` : fait une boucle sur l'élément HTML ciblé. Compatible avec plusieurs structures de données Java → `List`, `Array`, `Set`, `Map`
- `th:if` : permet de rendre l'élément HTML si une la condition est vraie
- `th:switch/th:case` : équivalent à un « switch/case » Java pour rendre un élément HTML en fonction d'une valeur

Exemples d'utilisation

- Exemple de balise HTML

```
<li th:each="user : ${users}" th:text="${user.name}"></li>
```

- Expression d'évaluation `${ … }`
- expression qui prendra sa valeur à **faisant référence à une variable Java**

```
${ user.name }
```

Expression SpEL (Spring Expression Language)

- SpEL :
- Permet de référencer un **objet par son nom** et d'accéder à ses attributs

- Expression d'accès à des objets communs:
  - `#ctx` : contexte de Spring, contient notamment des informations sur la **session**, les requête HTTP reçues et renvoyée…
  - `#vars` : contient les variables du contexte
  - `#locale` : la « locale » (information sur le pays) de l'application

```
Established locale country: <span th:text="${#locale.country}"></span>.
```

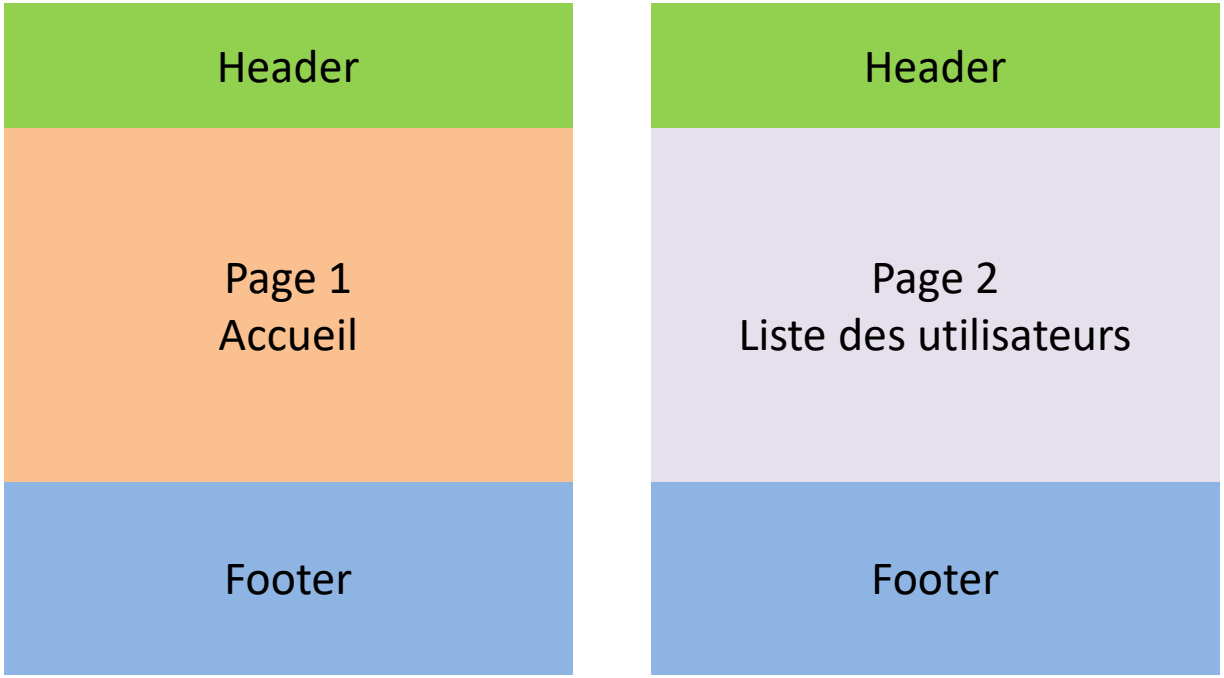
- Expression d'accès à des objets utilitaires :
  - `#temporals` :permet de manipuler des dates
  - `#strings` : permet de manipuler des chaînes de caractères

→ Liste exhaustive disponible ici

Système de layout

Comment limiter la redondance de code ?

- Zoning :



- Header + Footer identiques
- Seul le `<main>` change

Comment éviter de dupliquer le code d'une page à l'autre ?

Utilisation de fragments :

- L'attribut `th:insert` permet d'insérer le contenu d'un autre fichier HTML
- Pratique pour l'insertion de `header` et `footer`

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Layout général</title>

    <link th:href="@{/styles/app.css}" rel="stylesheet" />
  </head>
  <body>
    <header th:insert="fragments/header">
    </header>

    <main id="content">
      <p>Contenu de la page</p>
    </main>

    <footer th:insert="fragments/footer">
    </footer>
  </body>
</html>
```

Remplacement des balises

```html
<header th:fragment="header">
    <h1 id='page-title'>Google</h1>
</header>
```

```html
<footer th:fragment="footer">
    Made with ☕ by Afpa
</footer>
```

- # Thymeleaf Layout Dialect :
- Documentation : **https://ultraq.github.io/thymeleaf-layout-dialect/processors/decorate/**
- Permet d'indiquer des balises d'une page **dynamiquement modifiées** en fonction d'un autre template

layout.html

```
<header th:insert="fragments/header">
</header>

<main id="content" layout:fragment="content">
  <p>Le contenu de la page se positionnera ici</p>
</main>

<footer th:insert="fragments/footer">
</footer>
```

Sous-template : **home.html**

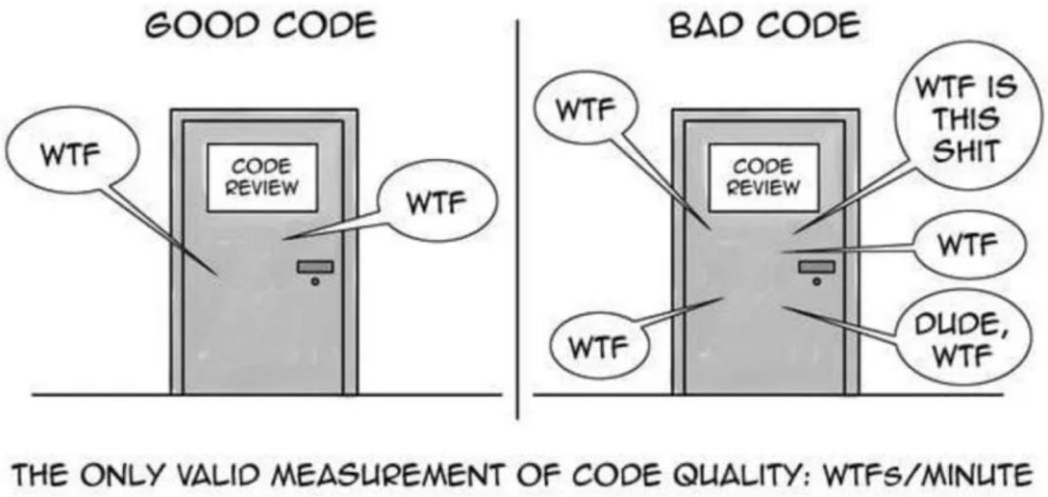
```
<!DOCTYPE html>
<html layout:decorate="~{layout}">

<head>
    <title>Page d'accueil</title>
</head>

<body>
    <main layout:fragment="content">
        <p>Ceci est une page d'accueil</p>
    </main>
</body>

</html>
```

- # Pourquoi limiter les redondances ?
- Favoriser la **maintenabilité** du code
- Augmenter la **qualité** de l'application
- Simplifier le travail **collaboratif**
- Simplifier le développement de **tests unitaires**
- Limiter la consommation de café et de doliprane



Agence nationale pour la formation professionnelle des adultes
Établissement public à caractère industriel et commercial
Tour Cityscope
3 rue Franklin, 93100 Montreuil
824 228 142 RCS BOBIGNY