

Thymeleaf

Démonstration 5 du module 4

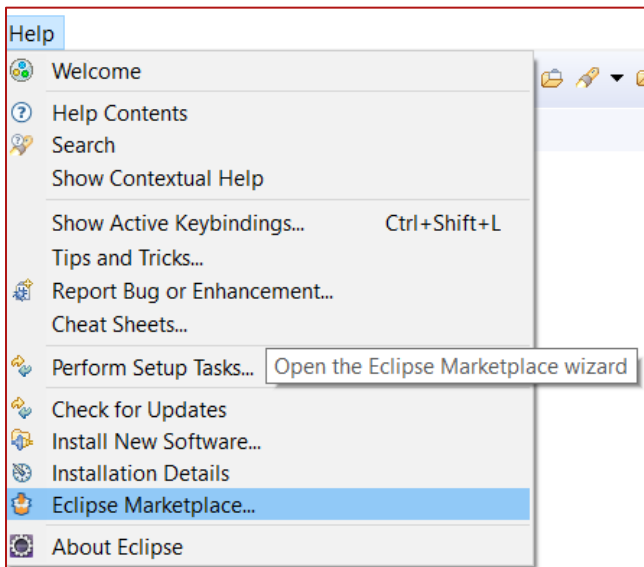
Les objectifs de cette démonstration sont

- La mise en place du plugin Thymeleaf dans Eclipse
- Le mise en pratique des bases de Thymeleaf pour construire les vues d'une application web classique

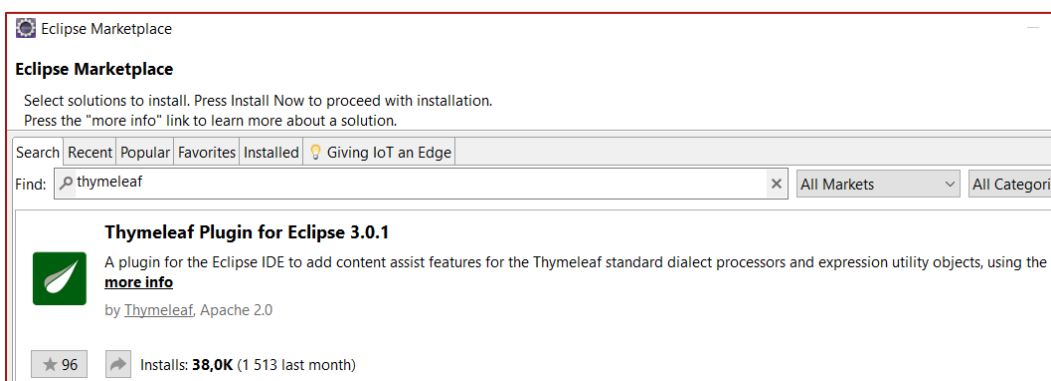
Déroulement

Installation dans Eclipse du plugin Thymeleaf

- Sélectionner le menu *Help* → *Eclipse Marketplace...*



- Rechercher *Thymeleaf* et installer "Thymeleaf Plugin for Eclipse 3.0.1"



- Redémarrer Eclipse pour valider l'installation...

Contexte

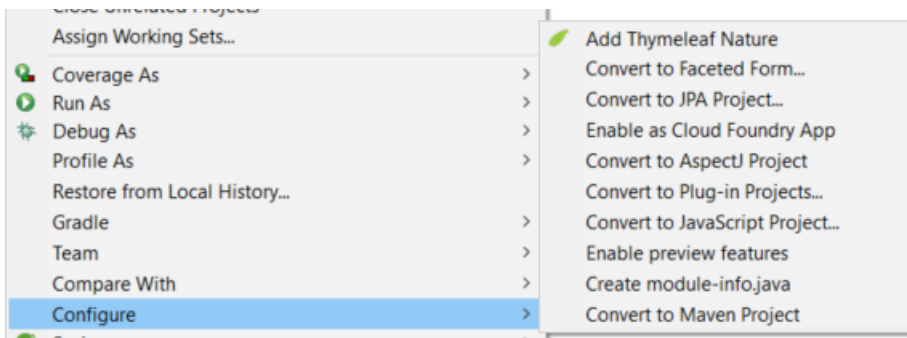
Compléter le projet précédent

- 2 nouveaux fichiers de CSS sont disponibles dans ressources ; ils complètent le précédent :
 - general.css (à utiliser sur toutes les pages)
 - demo-table.css (à utiliser pour les pages avec un « table »)
- L'icône de l'ENI est aussi à disposition dans les ressources
- Amélioration de notre application avec la mise en place de Thymeleaf
- Pour manipuler les données et afficher les données du modèle dans les templates.

Déroulement

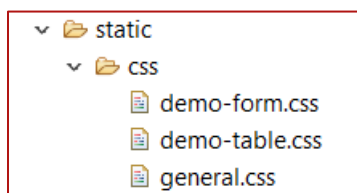
Configuration de Thymeleaf

- Sous Eclipse ; Pour utiliser l'auto-complétions dans vos templates pour Thymeleaf ; il faut ajouter ses propriétés aux projets
- Clic droit sur le projet → configure → Add Thymeleaf Nature

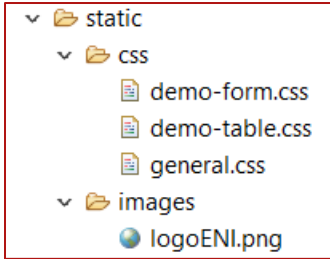


Modification d'index.html

- Intégration de la CSS
 - Ajouter les 2 fichiers de CSS dans static/css/



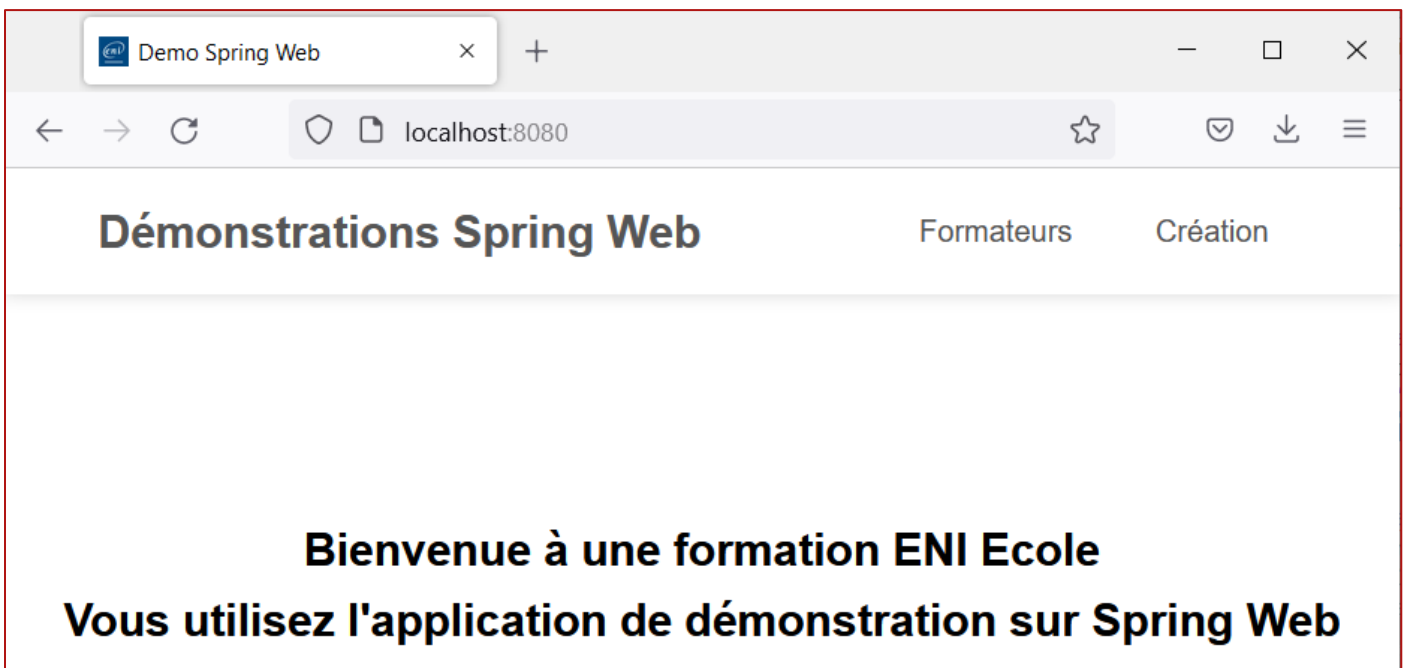
- Ajouter la CSS general.css à la page
- Intégration de l'icône
 - Ajouter l'icône dans un répertoire images sous static



- Ajouter le lien dans la vue pour l'icône
- Intégration d'une barre de navigation

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
<link rel="stylesheet" href="/css/general.css">
<link rel="icon" href="/images/LogoENI.png">
</head>
<body>
  <!-- Navbar -->
  <header>
    <h1 id="nav-title"><a href="/">Démonstrations Spring Web</a></h1>
    <nav>
      <ul>
        <li><a href="/trainers">Formateurs</a></li>
        <li><a href="/trainers/Create">Création</a></li>
      </ul>
    </nav>
  </header>
  <main class="center-main">
    <section class="center-section">
      <h1 style="text-align: center;">Bienvenue à une formation ENI Ecole</h1>
      <h2 style="text-align: center;">Vous utilisez l'application de démonstration sur Spring Web</h2>
    </section>
  </main>
</body>
</html>
```

- Intégration des Nouvelles CSS, de l'icône et de la barre de navigation dans tous les templates existants



Affichage de la liste des formateurs

1. Ajout au modèle de la liste des formateurs

- Modifier la méthode allTrainers de TrainerController
- Passer en paramètre Model
- Injecter la liste des formateurs grâce à la couche BLL

```
@GetMapping
public String allTrainers(Model model) {
    List<Trainer> lstTrainers = trainerService.findAll();
    model.addAttribute("trainers", lstTrainers);
    return "view-trainers";
}
```

2. Utilisation de Thymeleaf pour afficher la liste des formateurs dans une table

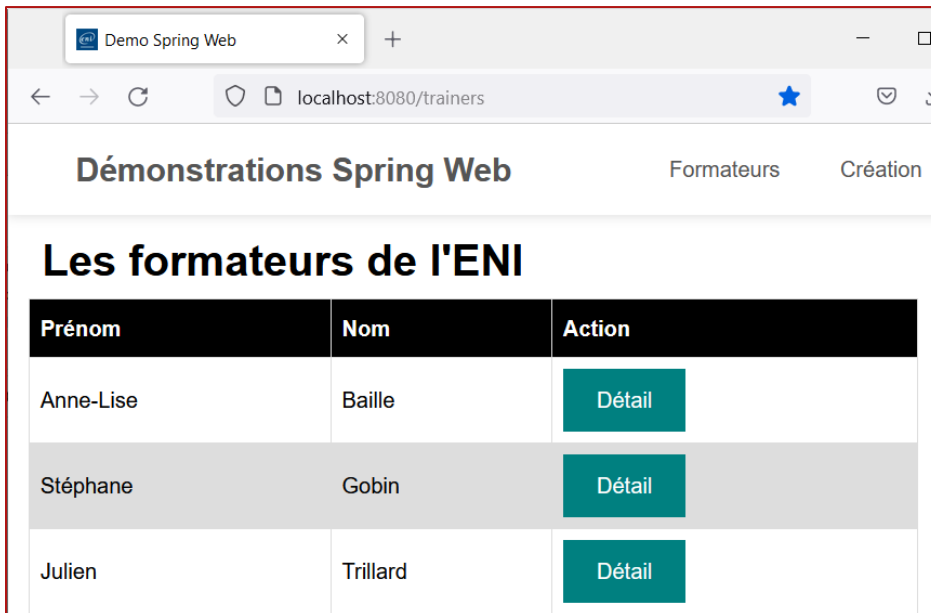
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"><head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
<link rel="stylesheet" href="/css/general.css">
<link rel="stylesheet" href="/css/demo-table.css">
<link rel="icon" href="/images/LogoENI.png">
</head>
<body>
    <header>
        <h1 id="nav-title"><a href="/">Démonstrations Spring Web</a></h1>
        <nav>
            <ul>
                <li><a href="/trainers">Formateurs</a></li>
                <li><a href="/trainers/create">Création</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Les formateurs de l'ENI</h1>
        <table>
            <thead>
                <tr>
                    <th>Prénom</th>
                    <th>Nom</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <!--build html table based on trainers -->
                <tr data-th-each="current : ${trainers}">
                    <td data-th-text="${current.firstName}"></td>
                    <td data-th-text="${current.lastName}"></td>
                    <td><a data-th-href="@{/trainers/detail(email=${current.email})}">Détail</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </main>
</body>
</html>
```

- Mise à jour de la vue ; ajout
 - de la navigation,
 - des CSS générale et spécifique aux tableaux
 - et de l'icône

- Intégration de la balise Thymeleaf :

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

- Utilisation de la balise `data-th-each` pour manipuler une liste provenant de modèle
 - Récupération de l'attribut du modèle avec `#{trainers}`
 - Déclaration d'une variable `current` pour accéder à chaque élément (similaire au `foreach` de Java)



The screenshot shows a web browser window titled 'Demo Spring Web' at the URL 'localhost:8080/trainers'. The page has a header with 'Démonstrations Spring Web' and navigation links 'Formateurs' and 'Création'. The main content is titled 'Les formateurs de l'ENI' and contains a table with three columns: 'Prénom', 'Nom', and 'Action'.

Prénom	Nom	Action
Anne-Lise	Baille	Détail
Stéphane	Gobin	Détail
Julien	Trillard	Détail

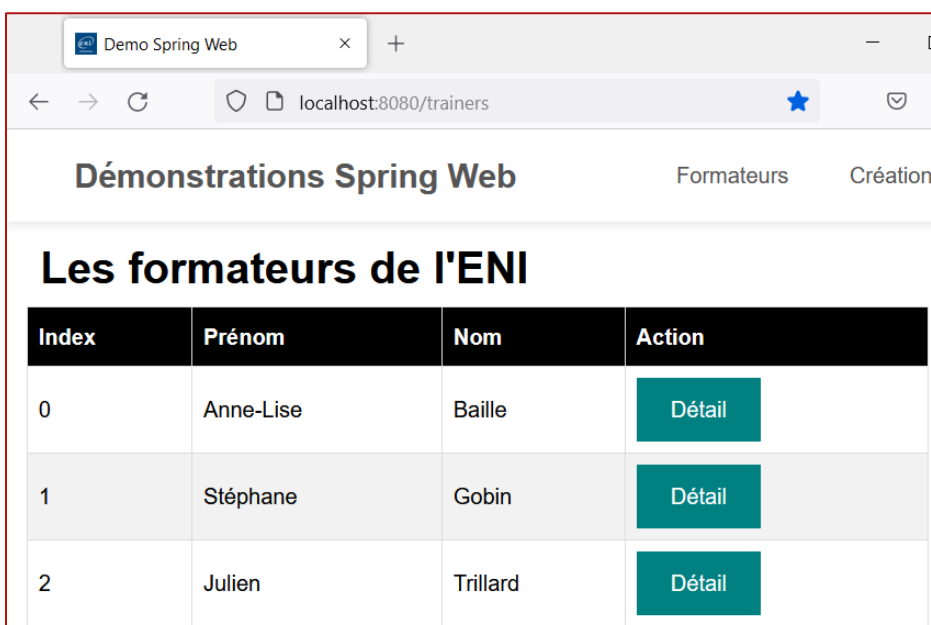
- Thymeleaf permet aussi de récupérer la position dans la liste :
 - Ajouter un entête pour placer la colonne index en premier dans le tableau

```
<th>Index</th>
```

- Modifier le `data-th-each` pour accéder à l'index

```
<tr data-th-each="current, iter : #{trainers}">
  <td data-th-text="#{iter.index}"></td>
```

- `iter`, permet d'accéder à index qui est la position actuelle de l'élément.



The screenshot shows the same web browser window as before, but the table now includes an 'Index' column as the first column. The data rows are numbered 0, 1, and 2.

Index	Prénom	Nom	Action
0	Anne-Lise	Baille	Détail
1	Stéphane	Gobin	Détail
2	Julien	Trillard	Détail

L'utilisation de frameworks pour le développement avec Java EE

- `current` et `iter` sont des variables Thymeleaf locales, visibles uniquement dans la balise dans laquelle elle sont déclarées (balise `tr` dans l'exemple ci-dessus).
- Il est possible de déclarer une variable locale avec le mot clé `with`:
 - Faire de la concaténation de chaînes de caractères

```
<p data-th-with=firstTrainer =${trainers[0]}>  
  Premier formateur de la liste <span data-th-text="${firstTrainer.firstName + ' ' +  
  firstTrainer.lastName }"></span>
```

Premier formateur de la liste Anne-Lise Baille

Commenter ce dernier test une fois réalisé.

Dans un futur, il pourrait produire des erreurs si la liste est vide

- Création d'un lien avec les données dynamiques

```
<a data-th-href="@{/trainers/detail(email=${current.email})}">Détail</a>
```

- Création d'une URL `@{}`
- Utilisation de `(email=${current.email})` pour créer un paramètre à l'URL avec l'email du formateur courant
- Ainsi, nous allons pouvoir afficher le détail de chaque formateur maintenant.

3. Détail de Thymeleaf dans le formulaire du formateur

- Mise à jour de la vue :
 - Ajout de la navigation, la CSS générale et de l'icône

```
<!DOCTYPE html>  
<!-- Ajout du moteur de template Thymeleaf -->  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
<meta charset="UTF-8">  
<title>Demo Spring Web</title>  
<link rel="stylesheet" href="/css/general.css">  
<link rel="stylesheet" href="/css/demo-form.css">  
<link rel="icon" href="/images/LogoENI.png">  
</head>  
<body>  
  <header>  
    <h1 id="nav-title"><a href="/">Démonstrations Spring Web</a></h1>  
    <nav>  
      <ul>  
        <li><a href="/trainers">Formateurs</a></li>  
        <li><a href="/trainers/create">Création</a></li>  
      </ul>  
    </nav>  
  </header>  
  <main>  
    <form action="/trainers" method="post" >  
    <h1>Détail du formateur </h1>  
    <ul class="flex-outer">  
      <li>
```

```

<label for="inputFirstN">Prénom : </label>
<input type="text" name="firstName" id="inputFirstN" required
  data-th-value="${trainer.firstName}"/>
</li>
<li>
  <label for="inputLastN">Nom : </label>
  <input type="text" name="lastName" id="inputLastN" required
    data-th-value="${trainer.lastName}"/>
</li>
<li>
  <label for="inputEmail">Email : </label>
  <input type="text" name="email" id="inputEmail" required
    data-th-value="${trainer.email}"/>
</li>
<li>
  <button type="submit">Enregistrer</button>
</li>
</ul>
</form>
</main>
</body>
</html>

```

- Utilisation des attributs de Thymeleaf pour manipuler l'objet Trainer mis dans le modèle de Spring
- `data-th-value="${trainer.firstName}"` ;
 - `trainer` permet de récupérer l'attribut du modèle, le nom correspond à la clef dans l'attribut
 - Une fois l'objet récupéré, il est possible d'accéder à ses attributs : `trainer.firstName`
 - Et `data-th-value` permet de préciser sur quel attribut HTML du composant, il faut injecter la donnée. Dans le cas présent sur l'attribut value du champ input.
- Tester l'affichage des différents formateurs, en cliquant sur le bouton détail de la vue « view-trainers.html »

The screenshot shows a web browser window with the title 'Demo Spring Web'. The address bar shows 'localhost:8080/trainers/detail?email=abaille@campus-eni'. The page content includes a header 'Démonstrations Spring Web' with links 'Formateurs' and 'Création'. The main section is titled 'Détail du formateur' and contains a form with the following data:

Prénom :	Anne-Lise
Nom :	Baille
Email :	abaille@campus-eni.fr

At the bottom right of the form is a button labeled 'ENREGISTRER'.

Démonstrations Spring Web

Formateurs Création

Détail du formateur

Prénom :

Nom :

Email :

ENREGISTRER

Démonstrations Spring Web

Formateurs Création

Détail du formateur

Prénom :

Nom :

Email :

ENREGISTRER

Pour toute informations supplémentaire, se référer à la documentation de Thymeleaf (thymeleaf.org) :

<https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>