

# Utiliser un Converter

## 1. Démonstration 9 du module 4

Les objectifs de cette démonstration sont

- Création d'un Converter
- Son utilisation sur un formulaire avec une association 1-n

## Contexte

- Compléter le projet précédent
- Création d'un formulaire pour créer un nouveau formateur associé à ses cours qu'il dispense.

## Déroulement

### 1. Création du Converter

- Ajouter un nouveau package : fr.eni.demo.mmi.converter
- Créer la classe StringToCourseConverter qui implémente Converter<String, Course>
  - Annotée là avec @Component, pour qu'elle devienne un bean de Spring
  - Il faut lui injecter le service CourseService
  - Et sa méthode convert permettra à partir de l'identifiant du cours de retrouver et retourner le cours associé

```
package fr.eni.demo.mmi.converter;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.converter.Converter;
import org.springframework.stereotype.Component;

import fr.eni.demo.bll.CourseService;
import fr.eni.demo.bo.Course;

@Component //Définir le converter en tant que bean Spring
public class StringToCourseConverter implements Converter<String, Course>{ //Implémente Converter
    private CourseService service;

    @Autowired
    public void setCourseService(CourseService service) {
        System.out.println("StringToCourseConverter - setCourseService");
        this.service = service;
    }

    @Override
    public Course convert(String id) {
        System.out.println("Convert - "+ id );
        Integer theId = Integer.parseInt(id);
        return service.findById(theId);
    }
}
```

## 2. Liste de cours

- Notre application, a déjà une liste de cours en session, grâce :
  - La méthode `getCourses` de `TrainerController`. Annotée `@ModelAttribute`(« `allCourses` »)
  - Et l'annotation `@SessionAttributes`
  - Il n'est donc pas nécessaire de créer un bean spécifique dans une classe de configuration

## 3. Méthodes du contrôleur pour le formulaire

- Création de 2 méthodes dans `TrainerController` :
  - Celle pour afficher le formulaire, mappée sur le Get et l'url « `create` »
    - Elle injecte l'instance de l'objet du formulaire « `trainer` »
  - Celle pour traiter le post sur le formulaire, l'ajout du nouveau formateur et la redirection vers l'affichage de la liste des formateurs
    - Elle se fait injecter l'objet du formulaire `@ModelAttribute`(« `trainer` »)

```
// Création d'un nouveau formateur
@GetMapping("/create")
public String createTrainer(Model model) {
    Trainer trainer = new Trainer();
    // Ajout de l'instance dans le modèle
    model.addAttribute("trainer", trainer);
    return "view-newtrainer-form";
}

// Récupération de l'objet trainer du formulaire
//Traçage de la liste des cours associés via Converter
//sauvegarde
@PostMapping("/create")
public String createTrainer(@ModelAttribute("trainer") Trainer trainer) {
    System.out.println(trainer.getLstCourses());
    trainerService.create(trainer);
    return "redirect:/trainers";
}
```

## 4. Le formulaire : `view-newtrainer-form.html`

```
<!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>
        <th:block data-th-if="${SpringSession != null}">
            <span data-th-text="${'[' + SpringSession + ']'}"></span>
        </th:block>
    </h1>
```

```

<nav>
  <ul>
    <li><a href="/trainers">Formateurs</a></li>
    <li><a href="/trainers/create">Création</a></li>
    <li><a href="/session">Connexion</a></li>
    <li><a href="/session/invalidate">Déconnexion</a></li>
  </ul>
</nav>
</header>
<main>
  <form data-th-action="@{/trainers/create}" data-th-object="${trainer}" method="post">
    <h1>Détail du formateur</h1>
    <ul class="flex-outer">
      <li><label for="inputFirstN">Prénom : </label>
        <input type="text" data-th-field="{firstName}" id="inputFirstN" required/>
      </li>
      <li><label for="inputLastN">Nom : </label>
        <input type="text" data-th-field="{lastName}" id="inputLastN" required/>
      </li>
      <li><label for="inputEmail">Email : </label>
        <input type="text" data-th-field="{email}" id="inputEmail" required/>
      </li>
      <li><label for="LstCourses">Cours : </label>
        <select id="LstCourses" data-th-field="{LstCourses}" required
          multiple="multiple">
          <option data-th-each="course : ${allCourses}"
            data-th-text="{course.id} + ' ' + {course.title}"
            data-th-value="{course.id}"></option>
        </select>
      </li>
      <li>
        <button type="submit">Enregistrer</button>
      </li>
    </ul>
  </form>
</main>
</body>
</html>

```

**Exécution :**

- Dans la console ; nous pouvons remarquer la création du bean du Converter, dès le démarrage

```
StringToCourseConverter - setCourseService
```

- Lors du clic sur le lien « Création »
  - Il y a l'affichage du nouveau formulaire
  - Avec la liste des cours chargés
  - Dans la console, il est possible de constater l'appel automatique du Converter sur chaque élément de cette liste :

```

Convert - 10
Convert - 20
Convert - 30
Convert - 50
Convert - 60
Convert - 130
Convert - 140
Convert - 220

```

**Démonstrations Spring Web**    Formateurs    Création    Connexion    Déconnexion

## Détail du formateur

Prénom :

Nom :

Email :

Cours : 

10 Algorithmique  
20 Initiation à la programmation  
30 POO  
50 SQL

**ENREGISTRER**

- Création d'un nouveau formateur, en sélectionnant plusieurs cours qu'il dispensent :

**Démonstrations Spring Web**    Formateurs    Création    Connexion    Déconnexion

## Détail du formateur

Prénom :

Nom :

Email :

Cours : 

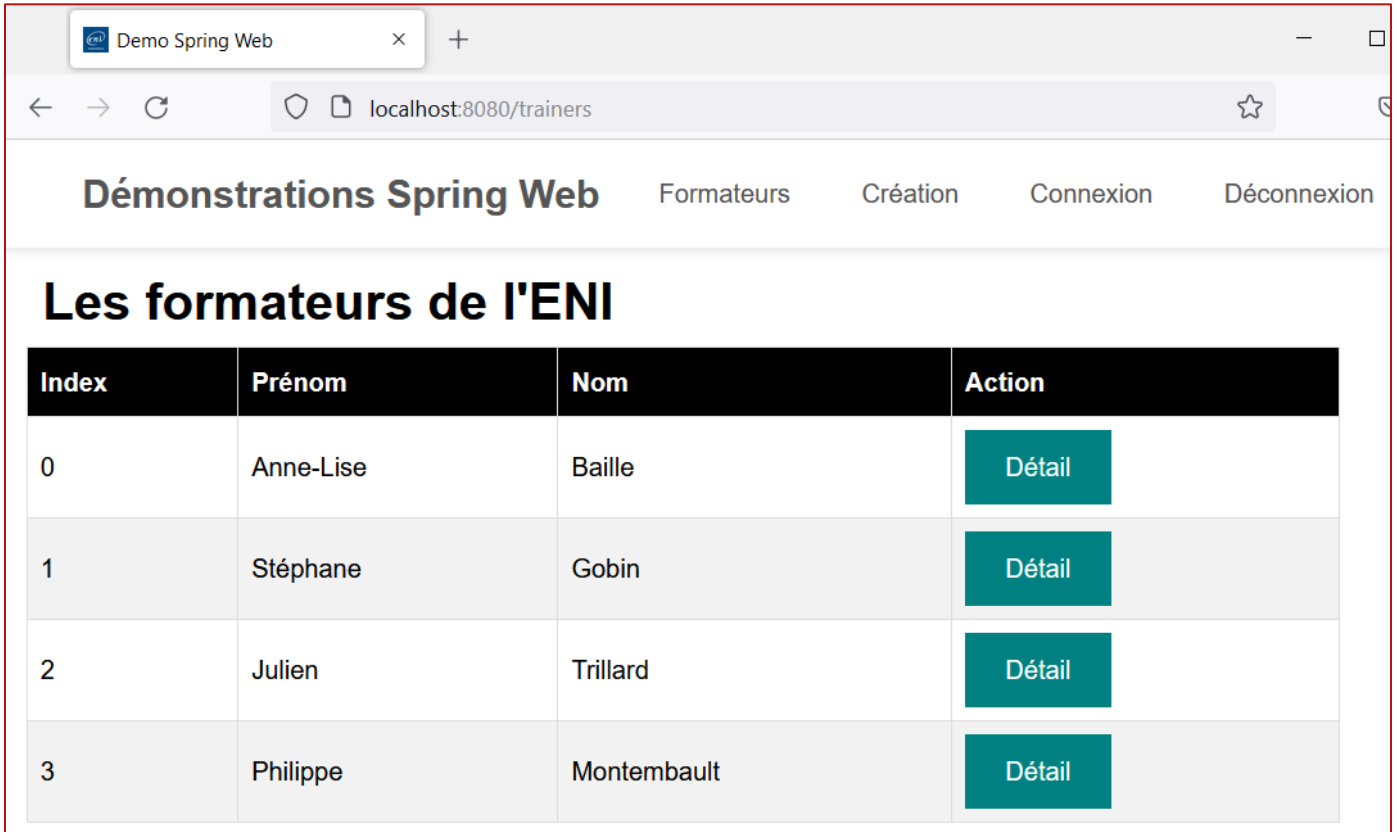
60 PL-SQL  
130 Web Client  
140 Symfony  
220 frameworks Java EE

**ENREGISTRER**

- Lors du clic sur le bouton enregistrer ; constater dans la console
  - L'appel du Convertir sur les 2 cours sélectionnés
  - et la trace des 2 cours affichés depuis la méthode du contrôleur :

```
Convert - 130
Convert - 140
[course [id=130, title=Web Client, duration=5], course [id=140, title=Symfony, duration=10]]
```

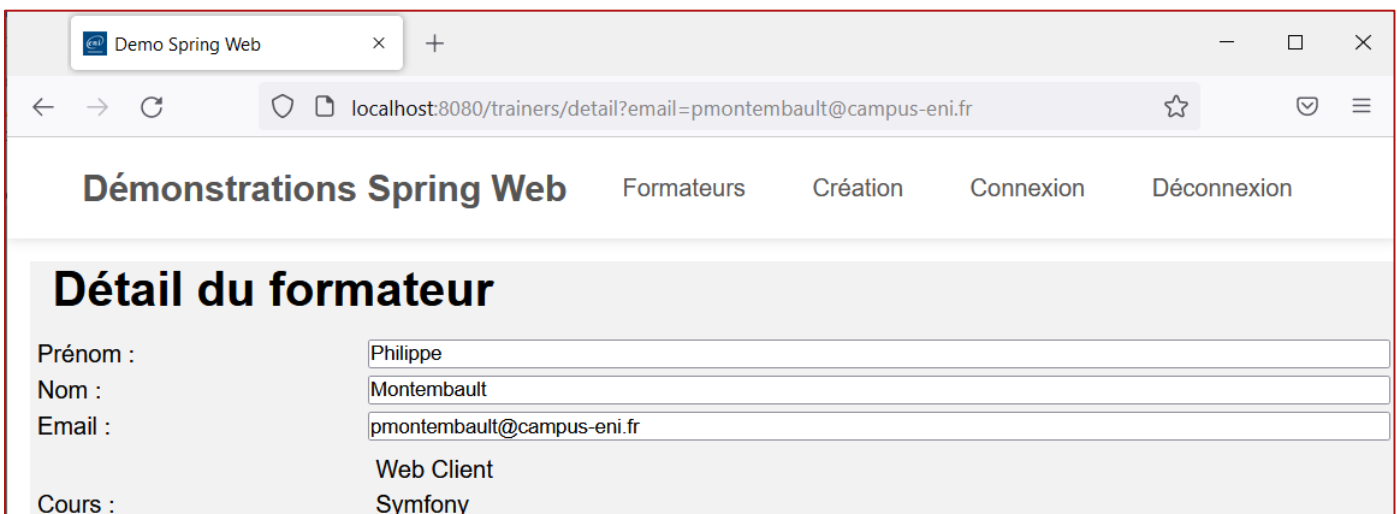
- Le formateur apparaît bien dans la liste des formateurs :



The screenshot shows a web browser window with the title 'Demo Spring Web'. The address bar shows 'localhost:8080/trainers'. The page has a navigation bar with links: 'Formateurs', 'Création', 'Connexion', and 'Déconnexion'. The main heading is 'Les formateurs de l'ENI'. Below it is a table with four columns: 'Index', 'Prénom', 'Nom', and 'Action'.

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

- En cliquant sur le détail du formateur, sa liste des cours est affichée :



The screenshot shows the same web browser window, but the address bar now shows 'localhost:8080/trainers/detail?email=pmontembault@campus-eni.fr'. The navigation bar is the same. The main heading is 'Détail du formateur'. Below it, the details of the trainer are displayed in a form:

Prénom : Philippe  
 Nom : Montembault  
 Email : pmontembault@campus-eni.fr  
 Cours : Web Client, Symfony

## 5. Allez plus loin

- Prouver que sans le Converter, il ne serait pas possible de manipuler cette association dans l'objet trainer du formulaire
  - Mettre en commentaire l'annotation @Component sur la StringToCourseConverter
- Cliquer sur « Création »
  - A cette étape, l'application son comportement similairement ; le formulaire c'est bien affiché.

L'utilisation de frameworks pour le développement avec Java EE

- Mais il n'y a rien dans la console.
- Renseigner les champs

Démonstrations Spring Web    Formateurs    Création    Connexion    Déconnexion

## Détail du formateur

Prénom :

Nom :

Email :

Cours : 

10 Algorithmique

20 Initiation à la programmation

30 POO

50 SQL

**ENREGISTRER**

- Cliquer sur enregistrer :

## Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Mar 22 17:33:10 CET 2022  
There was an unexpected error (type=Bad Request, status=400).  
Validation failed for object="trainer". Error count: 1  
org.springframework.validation.BindException: org.springframework.validation.BeanPropertyBindingResult: 1 errors  
Field error in object 'trainer' on field 'lstCourses': rejected value [10,20]; codes [typeMismatch.trainer.lstCourses,typeMismatch.lstCourses,typeMismatch.java.util.List,typeMismatch]; arguments [org.springframework.context.support.DefaultMessageSourceResolvable: codes [trainer.lstCourses,lstCourses]; arguments []; default message [lstCourses]]; default message [Failed to convert property value of type 'java.lang.String[]' to required type 'java.util.List' for property 'lstCourses'; nested exception is java.lang.IllegalStateException: Cannot convert value of type 'java.lang.String' to required type 'fr.eni.demo.bo.Course' for property 'lstCourses[0]': no matching editors or conversion strategy found]

- Il n'arrive pas à convertir : java.lang.String en fr.eni.demo.bo.Course.
- Ce qui confirme que dès qu'il y a une association à gérer au travers d'un seul formulaire, il faudra gérer un Converter
- Réactiver l'annotation du Converter.