

# Les paramètres et variables http

## Démonstration 2 du module 4

Les objectifs de cette démonstration sont :

- de montrer comment lire les paramètres de la requête http envoyées dans la requête en tant que paramètres http
- de montrer comment traiter les variables de la requête http

## Contexte

- Un fichier de CSS est fourni dans les ressources.
  - Il permettra de formater l'affichage du formulaire

## Compléter le projet précédent

- Créer un répertoire sous « static » nommé « css »
- Copier le fichier de CSS fourni

## Déroulement

### Passage et lecture de paramètres

#### 1. Page view-trainers.html

- Ajout des 3 liens qui nous permettront d'afficher le détail d'un formateur dans un nouveau template.
  - Pour le moment les URL de ces liens sont statiques. [Nous les ferons évoluer dans une démonstration future]

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
</head>
<body>
  <h1>Les formateurs de l'ENI</h1>
  <a href="trainers/detail?email=abaille@campus-eni.fr">Accéder au détail du formateur</a><br>
  <a href="trainers/detail?email">Paramètre n'a pas de valeur associée</a><br>
  <a href="trainers/detail">Accéder à la vue sans le paramètre</a><br>
</body>
</html>
```

- La page définit
  - un **paramètre http** email envoyé en Get via le premier lien hypertexte avec la valeur : abaille@campus-eni.fr
  - un **paramètre http** email envoyé en Get via le deuxième lien hypertexte **sans valeur** pour tester le comportement du membre « default » de l'annotation
  - un lien sans le paramètre pour valider le membre required à false de l'annotation

## 2. Récupérer le paramètre http « email » dans le contrôleur

### Utilisation de @RequestMapping sur la classe

- Gestion globale sur la classe TrainerController de l'url par défaut `"/trainers"`
- Pour cela ; ajout de l'annotation @RequestMapping sur la classe
  - Ainsi, la méthode allTrainers n'a besoin d'aucune url.
- Et les autres méthodes de la classe sont au moins sur cette url par défaut

```
...
import org.springframework.web.bind.annotation.RequestMapping;
...
@Controller
//Url par défaut pour toutes les méthodes du contrôleur
@RequestMapping("/trainers")
public class TrainerController {
    @GetMapping
    public String allTrainers() {
        System.out.println("Nous chargerons la liste des formateurs dans une autre démonstration");
        return "view-trainers";
    }
}
```

### Ajout d'une méthode pour gérer le mapping avec le paramètre http

- Création de la méthode detailTrainer
  - qui va récupérer le paramètre email
  - le rendre non obligatoire
  - et mettre par défaut « coach@campus-eni.fr » si aucune valeur

```
...
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

//@Controller --> permet de définir la classe comme un bean Spring de type Controller
@Controller
//Url par défaut pour toutes les méthodes du contrôleur
@RequestMapping("/trainers")
public class TrainerController {
    @GetMapping
    public String allTrainers() {
        System.out.println("Nous chargerons la liste des formateurs dans une autre démonstration");
        return "view-trainers";
    }

    @GetMapping("/detail")
    public String detailTrainer(
        @RequestParam(name = "email",
            required = false,
```

```

        defaultValue = "coach@campus-eni.fr") String emailTrainer) {
    System.out.println("Le paramètre - " + emailTrainer);
    return "view-trainer-form";
}

```

- Pour le moment, il est impossible de mettre les informations dans le formulaire sauf en dur. [Nous le ferons évoluer dans une démonstration future]
- A cette étape, le formulaire sera vide dans les 3 cas.
- Seuls les traces de la console montreront les 3 versions différentes.

### Création de la vue view-trainer-form.html

- création d'un nouveau template.
- Il représente un formulaire qui pourra servir à créer un nouveau formateur ou modifier un formateur existant.
  - Pour le moment, il est impossible de mettre les informations dans le formulaire sauf en dur. [Nous le ferons évoluer dans une démonstration future]
  - A cette étape, le formulaire sera vide dans les 3 cas.

```

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
<link rel="stylesheet" href="/css/demo-form.css">
</head>
<body>
    <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 />
            </li>
            <li>
                <label for="inputLastN">Nom : </label>
                <input type="text" name="lastName" id="inputLastN" required />
            </li>
            <li>
                <label for="inputEmail">Email : </label>
                <input type="text" name="email" id="inputEmail" required />
            </li>
            <li>
                <button type="submit">Enregistrer</button>
            </li>
        </ul>
    </form>
</body>
</html>

```

- Exécuter l'application et tester le comportement, les traces de la console sont :

```
Le paramètre - abaille@campus-eni.fr
Le paramètre - coach@campus-eni.fr
Le paramètre - coach@campus-eni.fr
```

- Dans le premier cas, le paramètre est renseignés à « abaille@campus-eni.fr », le contrôleur le réceptionne parfaitement.
- Dans les deux autres cas ; (où le paramètre est vide ou pas de paramètre) ; Spring utilise la valeur de « default » = « coach@campus-eni.fr »

#### Aller plus loin

- Supprimer le membre « default » de l'annotation @RequestParam
- Vérifier le comportement des 3 liens
- L'application fonctionne toujours les traces de la console sont :

```
Le paramètre - abaille@campus-eni.fr
Le paramètre - 
Le paramètre - null
```

- Dans le cas, où le paramètre est dans l'URL sans valeur, Spring considère que la valeur est la chaîne de caractère vide.
- Dans le cas, où le paramètre n'est pas dans l'URL, il considère que la valeur est nulle.
- Une fois le test réalisé, remettre le membre « default »

### 3. Récupérer les paramètres http du formulaire dans le contrôleur

- Ces paramètres sont obligatoires dans la vue
- Il faut donc les rendre obligatoire dans le contrôleur, en plaçant le membre « required » à vrai
- Ajouter la méthode createOrUpdateTrainer, elle est annotée @PostMapping
  - [Dans des démonstrations à venir, nous y associerons les comportements back nécessaire]
  - Pour le moment, elle va tracer les 3 paramètres http et retourner sur la vue view-trainers.html

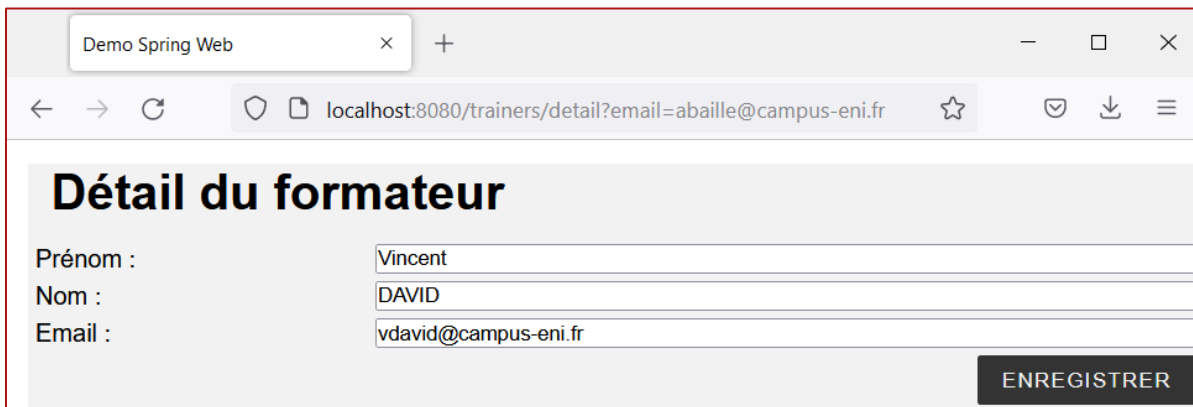
```
package fr.eni.demospringbootwebmvc.mmi.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

//@Controller --> permet de définir la classe comme un bean Spring de type Controller
@Controller
//Url par défaut pour toutes les méthodes du contrôleur
@RequestMapping("/trainers")
public class TrainerController {
    @GetMapping
    public String allTrainers() {
        System.out.println("Nous chargerons la liste des formateurs dans une autre démonstration");
        return "view-trainers";
    }
}
```

```
@GetMapping("/detail")
public String detailTrainer(
    @RequestParam(name = "email",
        required = false,
        defaultValue = "coach@campus-eni.fr") String emailTrainer) {
    System.out.println("Le paramètre - " + emailTrainer);
    return "view-trainer-form";
}

@PostMapping
public String createOrUpdateTrainer(
    @RequestParam(required = true) String email,
    @RequestParam(required = true) String firstName,
    @RequestParam(required = true) String lastName) {
    System.out.println("Les paramètres");
    System.out.println("Email - " + email);
    System.out.println("FirstName - " + firstName);
    System.out.println("LastName - " + lastName);
    return "view-trainers";
}
}
```



The screenshot shows a web browser window titled 'Demo Spring Web' with the address bar displaying 'localhost:8080/trainers/detail?email=abaille@campus-eni.fr'. The main content area has a heading 'Détail du formateur' and a form with three input fields: 'Prénom' (containing 'Vincent'), 'Nom' (containing 'DAVID'), and 'Email' (containing 'vdavid@campus-eni.fr'). A dark button labeled 'ENREGISTRER' is located at the bottom right of the form.

- Les traces de la console, avec les champs renseignées ; sont :

```
Les paramètres
Email - vdavid@campus-eni.fr
FirstName - Vincent
LastName - DAVID
```

## Passage et lecture d'une variable

### 1. Page view-trainers.html

- Ajout des 2 liens pour tester le passage d'une variable et obtenir un résultat similaire au passage d'un paramètre
  - Comme précédemment, les url sont statiques

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demo Spring Web</title>
</head>
<body>
  <h1>Les formateurs de l'ENI</h1>
  <a href="trainers/detail?email=abaille@campus-eni.fr">Accéder au détail du formateur</a><br>
  <a href="trainers/detail?email">Paramètre n'a pas de valeur associée</a><br>
  <a href="trainers/detail">Accéder à la vue sans le paramètre</a><br>

  <h2>Variable de la requête : </h2>
  <a href="trainers/detail/variable/sgobin@campus-eni.fr">Accéder au détail du formateur</a><br>
  <a href="trainers/detail/variable/">Accéder à la vue sans la variable</a>
</body>
</html>
```

- La page a été complétée avec :
  - Un lien hypertexte contenant une **variable http** `sgobin@campus-eni.fr` envoyée en Get
  - Un deuxième lien hypertexte avec la même URI mais sans variable pour valider le membre `required` à `false` de l'annotation

### 2. Récupérer le paramètre http « email » dans le contrôleur

#### Ajout d'une méthode pour gérer le mapping avec la variable http

- Création de la méthode `detailTrainer2`
  - Elle aura 2 URLs possibles : avec la variable `{email}` et sans
  - Elle va récupérer la variable `email`
  - Elle va la rendre non obligatoire
  - et mettre par défaut « `coach@campus-eni.fr` » si aucune valeur

```
...
import org.springframework.web.bind.annotation.PathVariable;
...
@Controller
@RequestMapping("/trainers")
public class TrainerController {
...
  //Pour utiliser le membre required = false
  //Il faut mettre les 2 url supportées +
  @GetMapping({"detail/variable/", "detail/variable/{email}"})
  public String detailTrainer2(
    @PathVariable(name = "email", required = false) String emailTrainer) {
    //Il n'est pas possible de passer une valeur par défaut si la variable n'existe pas dans URL
    //Il est possible de le faire par codage en testant null
    if(emailTrainer == null) {
      emailTrainer = "coach@campus-eni.fr";
    }
  }
}
```

```
System.out.println("La variable - " + emailTrainer);
return "view-trainer-form";
}
```

- Exécuter l'application et tester le comportement, les traces de la console sont :

```
La variable - sgobin@campus-eni.fr
La variable - coach@campus-eni.fr
```

- Dans le premier cas, la variable email est renseignés à « sgobin@campus-eni.fr », le contrôleur la réceptionne parfaitement.
- Dans le deuxième cas ; (il n'y a pas de valeur) ; Spring retourne nulle dans la variable et le code ajouter permet de retourner une valeur par défaut = « coach@campus-eni.fr »

### Aller plus loin

- Supprimer le membre « required » de l'annotation @PathVariable
- Vérifier le comportement des 2 liens
  - Le premier avec la variable bien passée, même comportement que précédemment.
  - Dans le cas, où il n'y a pas de valeur, il y a une erreur :

localhost:8080/trainers/detail/variable/

## Whitelabel Error Page

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

Thu Mar 17 10:35:44 CET 2022

There was an unexpected error (type=Internal Server Error, status=500).

Required URI template variable 'email' for method parameter type String is not present

org.springframework.web.bind.MissingPathVariableException: Required URI template variable 'email' for method parameter type String is not present

- Spring considère que la variable 'email' est obligatoire. Et le précise.
  - Attention, à bien mettre le membre required = false dans le cas d'une variable optionnelle.
- Une fois le test réalisé, remettre le membre « required = false »