Spring Boot : services web REST

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Pojo : Plain Old Java Object

- Une classe Java
- avec un constructeur sans paramètres
- des attributs privés
- des getters et setters publiques
- qui n'hérite d'aucune classe ni interface système (comme `Serializable`)

Problématique

- Dans le monde réel, on utilise très souvent les Pojos dans nos applications
- Une application évolue et un Pojo peut subir de changements (plusieurs modifications dans le code source)
- Ces changements sont souvent statiques : ajout d'un attribut $\Rightarrow$ génération de getters / setters + [méthodes déléguées] $\Rightarrow$ nouveau constructeur $\Rightarrow$ génération d'un nouveau `toString()` ...

Spring Boot & Lombok

Création de projet Spring Boot

- Aller dans `File > New > Other`
- Chercher `Spring`, dans `Spring Boot` sélectionner `Spring Starter Project` et cliquer sur `Next >`
- Saisir
  - `SpringBootLombok` dans `Name`,
  - `com.example` dans `Group`,
  - `SpringBootLombok` dans `Artifact`,
  - `com.example.demo` dans `Package`
- Cliquer sur `Next >`
- Chercher et cocher les cases correspondantes aux `Spring Web`, `Spring Data JPA`, `MySQL Driver`, `DevTools` et `Lombok` puis cliquer sur `Next >`
- Valider en cliquant sur `Finish`

Et si on part d'un projet existant, il faudra ajouter les dépendances suivantes

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <scope>runtime</scope>
</dependency>
```

Spring Boot & Lombok

Première utilisation de Lombok

- Aller dans `C:/Users/User/.m2/repository/org/projectlombok/lombok/1.18.4`
- Faire double clic sur `lombok-1.18.4` et démarrer l'installation (ça peut prendre plusieurs minutes)
- Redémarrer Eclipse

Spring Boot & Lombok

Créons le modèle `Personne` **dans** `com.example.demo.model`

```
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Spring Boot & Lombok

Créons le modèle `Personne` **dans** `com.example.demo.model`

```
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Créons un contrôleur `LombokController` **dans** `com.example.demo.controller`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
  }
}
```

Spring Boot & Lombok

Créons le modèle `Personne` **dans** `com.example.demo.model`

```
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Créons un contrôleur `LombokController` **dans** `com.example.demo.controller`

```
@Controller
public class LombokController {
   @GetMapping(value="/lombok")
   public void sayHello() {
      Personne personne = new Personne();
   }
}
```

Vérifier que les attributs de `Personne` sont directement accessibles (`personne.num = 100;`)

Spring Boot & Lombok

Pour rendre les attributs privés, ajoutons l'annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Spring Boot & Lombok

Pour rendre les attributs privés, ajoutons l'annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Le contrôleur `LombokController`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
  }
}
```

Spring Boot & Lombok

Pour rendre les attributs privés, ajoutons l'annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Le contrôleur `LombokController`

```
@Controller
public class LombokController {
   @GetMapping(value="/lombok")
   public void sayHello() {
      Personne personne = new Personne();
   }
}
```

Vérifier que les attributs de `Personne` ne sont plus accessibles (`personne.num = 100;`)

Spring Boot & Lombok

Pour avoir les getters et setters, ajoutons les annotations suivantes

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Spring Boot & Lombok

Pour avoir les getters et setters, ajoutons les annotations suivantes

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Le contrôleur `LombokController`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
    personne.setNom("wick");
    personne.setPrenom("john");
    System.out.println(personne.getPrenom()+ "␣" + personne.getNom());
  }
}
```

Spring Boot & Lombok

Pour définir un `toString` pour tous les attributs, ajoutons l' annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Spring Boot & Lombok

Pour définir un `toString` pour tous les attributs, ajoutons l' annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Le contrôleur `LombokController`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
    personne.setNom("wick");
    personne.setPrenom("john");
    System.out.println(personne);
  }
}
```

Spring Boot & Lombok

Pour définir un `toString` pour une sélection d'attributs, ajoutons l'annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString(of= {"nom","prenom"})
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Spring Boot & Lombok

Pour définir un `toString` pour une sélection d'attributs, ajoutons l'annotation suivante

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString(of= {"nom","prenom"})
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Le contrôleur `LombokController`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
    personne.setNom("wick");
    personne.setPrenom("john");
    System.out.println(personne);
  }
}
```

Spring Boot & Lombok

Les quatre annotations suivantes

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Spring Boot & Lombok

Les quatre annotations suivantes

```
@FieldDefaults(level=AccessLevel.PRIVATE)
@Setter
@Getter
@ToString
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

peuvent être remplacées par l'annotation `@Data`

```
@Data
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Spring Boot & Lombok

Pour définir un constructeur avec tous les attributs, ajoutons l'annotation suivante

```
@Data
@AllArgsConstructor
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Spring Boot & Lombok

Pour définir un constructeur avec tous les attributs, ajoutons l'annotation suivante

```
@Data
@AllArgsConstructor
public class Personne {
  Integer num;
  String nom;
  String prenom;
}
```

Dans `LombokController`, le constructeur sans paramètre est souligné en rouge car il n'existe plus

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
    personne.setNom("wick");
    personne.setPrenom("john");
    System.out.println(personne);
    Personne personne2 = new Personne(200,"mike","bob");
    System.out.println(personne2);
  }
}
```

Pour redéfinir le constructeur sans paramètre, ajoutons l'annotation suivante

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Personne {
   Integer num;
   String nom;
   String prenom;
}
```

Spring Boot & Lombok

Pour définir un constructeur avec seulement les attributs non nuls, ajoutons les annotations suivantes

```
@Data
@AllArgsConstructor
@NoArgsConstructor
@RequiredArgsConstructor
public class Personne {
   Integer num;
   @NonNull
   String nom;
   @NonNull
   String prenom;
}
```

Ne pas confondre `@NonNull` et `@NotNull` qu'on utilise pour la validation de formulaire.

Utilisation de trois constructeurs dans `LombokController`

```
@Controller
public class LombokController {
  @GetMapping(value="/lombok")
  public void sayHello() {
    Personne personne = new Personne();
    personne.setNom("wick");
    personne.setPrenom("john");
    System.out.println(personne);
    Personne personne2 = new Personne(200,"mike","bob");
    System.out.println(personne2);
    Personne personne3 = new Personne("green","ben");
    System.out.println(personne3);
  }
}
```