Swagger

Achref El Mouelhi

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

elmouelhi.achref@gmail.com

Spring Boot Rest & Swagger

Swagger UI

- Générateur de documentation au format **JSON** pour les services **REST**
- Permettant également de visualiser la documentation des ressources exposées
- Documentation officielle : `https://swagger.io/tools/swagger-ui/`

Spring Boot Rest & Swagger

Création de projet Spring Boot

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

Spring Boot Rest & Swagger

Explication

- Le package contenant le point d'entrée de notre application (la classe contenant le `puclic static void main`) est `com.example.demo`
- Tous les autres packages `dao`, `model`... doivent être dans le package `demo`.

Spring Boot Rest & Swagger

Explication

- Le package contenant le point d'entrée de notre application (la classe contenant le `puclic static void main`) est `com.example.demo`
- Tous les autres packages `dao`, `model`... doivent être dans le package `demo`.

Pour la suite, nous considérons

- une entité `Personne` à définir dans `com.example.demo.model`
- une interface DAO `PersonneRepository` à définir dans `com.example.demo.dao`
- un contrôleur REST `PersonneController` à définir dans `com.example.demo.dao`

Créons une entité `Personne` dans `com.example.demo.model`

```
package com.example.demo.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@AllArgsConstructor
@Data
@Entity
public class Personne {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long num;
  private String nom;
  private String prenom;
}
```

Préparons notre interface DAO `PersonneRepository`

```
package com.example.demo.dao;

import org.springframework.data.jpa.repository.
   JpaRepository;

import com.example.demo.model.Personne;

public interface PersonneRepository extends
   JpaRepository<Personne, Long> {

}
```

Spring Boot Rest & Swagger

Créons le contrôleur REST suivant

```
@RestController
public class PersonneRestController {

  @Autowired
  private PersonneRepository personneRepository;

  @GetMapping("/personnes")
  public List<Personne> getPersonnes() {
    return personneRepository.findAll();
  }

  @GetMapping("/personnes/{id}")
  public Personne getPersonne(@PathVariable("id") long id) {
    return personneRepository.findById(id).orElse(null);
  }

  @PostMapping("/personnes")
  public Personne addPersonne(@RequestBody Personne personne) {
    return personneRepository.save(personne);
  }
}
```

Spring Boot Rest & Swagger

Dans `application.properties`, ajoutons les données permettant la connexion à la base de données et la configuration de `Hibernate`

```
spring.datasource.url = jdbc:mysql://localhost:3306/boot?serverTimezone
  =UTC
spring.datasource.username = root
spring.datasource.password = root
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql = true
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.
  MySQL8Dialect
spring.jpa.hibernate.naming.physical-strategy = org.hibernate.boot.
  model.naming.PhysicalNamingStrategyStandardImpl
```

L'ajout de la propriété `spring.jpa.hibernate.naming.physical-strategy` permet de forcer **Hibernate** à utiliser les mêmes noms pour les tables et les colonnes que les entités et les attributs.

Spring Boot Rest & Swagger

Pour tester

Il faut aller sur `http://localhost:8080/personnes` ou sur `http://localhost:8080/personnes/1`.

Spring Boot Rest & Swagger

Pour tester

Il faut aller sur `http://localhost:8080/personnes` ou sur `http://localhost:8080/personnes/1`.

Constat

Toutes nos ressources sont exposées : pour récupérer les données, il suffit de connaître l'URL.

Spring Boot Rest & Swagger

Pour utiliser **Swagger**, il faut ajouter deux dépendances

1. une première pour activer **Swagger** dans le projet **Spring Boot**
2. une deuxième pour visualiser les ressources exposées ainsi que les verbes **HTTP** autorisés

Spring Boot Rest & Swagger

Ajoutons la première dépendance dans `pom.xml`

```
<!-- https://mvnrepository.com/artifact/io.springfox
   /springfox-swagger2 -->
<dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>2.9.2</version>
</dependency>
```

Spring Boot Rest & Swagger

Dans `com.example.demo`**, ajoutons l'annotation** `@EnableSwagger2` **à la classe** `SpringBootSwaggerApplication`

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import springfox.documentation.swagger2.annotations.EnableSwagger2;


@EnableSwagger2
@SpringBootApplication
public class SpringBootSwaggerApplication {

        public static void main(String[] args) {
                SpringApplication.run(SpringBootSwaggerApplication.
                  class, args);
        }

}
```

Spring Boot Rest & Swagger

Explication

Pour consulter la documentation au format **JSON** généré par **Swagger**, allez à `http://localhost:8080/v2/api-docs`.

Spring Boot Rest & Swagger

Explication

Pour consulter la documentation au format **JSON** généré par **Swagger**, allez à `http://localhost:8080/v2/api-docs`.

Comment faire mieux ?

Ajouter la deuxième dépendance

Spring Boot Rest & Swagger

Ajoutons la deuxième dépendance dans `pom.xml`

```
<!-- https://mvnrepository.com/artifact/io.springfox
   /springfox-swagger-ui -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

Spring Boot Rest & Swagger

Explication

Pour visualiser les API exposées, allez à `http://localhost:8080/swagger-ui.html`.

Spring Boot Rest & Swagger

Explication

Pour visualiser les API exposées, allez à `http://localhost:8080/swagger-ui.html`.

Question

Comment faire pour ne pas visualiser `base-error-controller`

Spring Boot Rest & Swagger

Dans `com.example.demo.config`, on définit la classe `SwaggerConfig` en précisant le package contenant les contrôleurs à exposer

```
package com.example.demo.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.example.demo.controller"))
                .paths(PathSelectors.any())
                .build();
    }
}
```

Spring Boot Rest & Swagger

Pour filtrer certaines méthodes (celles qui acceptent un paramètre par exemple)

```
package com.example.demo.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {
    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.example.demo.controller"))
                .paths(PathSelectors.regex("/personnes$"))
                .build();
    }
}
```

On peut aussi annuler l'affichage d'une méthode dans la documentation en utilisant l'annotation `@ApiIgnore`

```
@RestController
public class PersonneRestController {

  @Autowired
  private PersonneRepository personneRepository;

  @GetMapping("/personnes")
  public List<Personne> getPersonnes() {
    return personneRepository.findAll();
  }

  @GetMapping("/personnes/{id}")
  public Personne getPersonne(@PathVariable("id") long id) {
    return personneRepository.findById(id).orElse(null);
  }

  @ApiIgnore
  @PostMapping("/personnes")
  public Personne addPersonne(@RequestBody Personne personne) {
    return personneRepository.save(personne);
  }
}
```

Pour commenter une méthode, on utilise l'annotation `@ApiOperation`

```
@RestController
public class PersonneRestController {

  @Autowired
  private PersonneRepository personneRepository;

  @ApiOperation(value = "retourne la des personnes stockées dans la BD"
     )
  @GetMapping("/personnes")
  public List<Personne> getPersonnes() {
    return personneRepository.findAll();
  }

  @GetMapping("/personnes/{id}")
  public Personne getPersonne(@PathVariable("id") long id) {
    return personneRepository.findById(id).orElse(null);
  }

  @PostMapping("/personnes")
  public Personne addPersonne(@RequestBody Personne personne) {
    return personneRepository.save(personne);
  }
}
```

Pour commenter un paramètre d'une méthode, on utilise l'annotation `@ApiParam`

```
@RestController
public class PersonneRestController {

  @Autowired
  private PersonneRepository personneRepository;

  @GetMapping("/personnes")
  public List<Personne> getPersonnes() {
    return personneRepository.findAll();
  }

  @GetMapping("/personnes/{id}")
  public Personne getPersonne(@PathVariable("id") long id) {
    return personneRepository.findById(id).orElse(null);
  }

  @PostMapping("/personnes")
  @ApiParam(value="${PersonneRestController.addPersonne.personne}",
     required = true)
  public Personne addPersonne(@RequestBody Personne personne) {
    return personneRepository.save(personne);
  }
}
```

Spring Boot Rest & Swagger

Remarque

Pour consulter la liste d'annotations disponibles, allez à `https://springfox.github.io/springfox/docs/current/#docket-spring-java-configuration`.