

Aida Magou DIOP

M1 GLSI

Développement d'applications Java

Lab 2 : Premières pas avec Spring-boot

Exercice 1 : API REST pour gérer des cafés (sb-coffee-1)

1-Créer le projet Spring Boot

spring init --dependencies=web --type=gradle-project --groupId=esp.dgi --artif

--artifactId=sb-coffee-1 --name=CoffeeApplication

```
LENOVO@DESKTOP-8N3U8ER MINGW64 ~/Desktop/IPDL_Toure
$ spring init --dependencies=web --type=gradle-project --groupId=esp.dgi --artif
actId=sb-coffee-1 --name=CoffeeApplication
Using service at https://start.spring.io
Content saved to 'sb-coffee-1.zip'
```

```
LENOVO@DESKTOP-8N3U8ER MINGW64 ~/Desktop/IPDL_Toure
$ unzip sb-coffee-1.zip -d sb-coffee-1
Archive: sb-coffee-1.zip
  inflating: sb-coffee-1/build.gradle
    creating: sb-coffee-1/src/
    creating: sb-coffee-1/src/test/
    creating: sb-coffee-1/src/test/java/
    creating: sb-coffee-1/src/test/java/esp/
    creating: sb-coffee-1/src/test/java/esp/dgi/
    creating: sb-coffee-1/src/test/java/esp/dgi/sb_coffee_1/
  inflating: sb-coffee-1/src/test/java/esp/dgi/sb_coffee_1/CoffeeApplicationTest
s.java
    creating: sb-coffee-1/src/main/
    creating: sb-coffee-1/src/main/resources/
  inflating: sb-coffee-1/src/main/resources/application.properties
    creating: sb-coffee-1/src/main/resources/templates/
    creating: sb-coffee-1/src/main/resources/static/
    creating: sb-coffee-1/src/main/java/
    creating: sb-coffee-1/src/main/java/esp/
    creating: sb-coffee-1/src/main/java/esp/dgi/
    creating: sb-coffee-1/src/main/java/esp/dgi/sb_coffee_1/
  inflating: sb-coffee-1/src/main/java/esp/dgi/sb_coffee_1/CoffeeApplication.jav
a
```

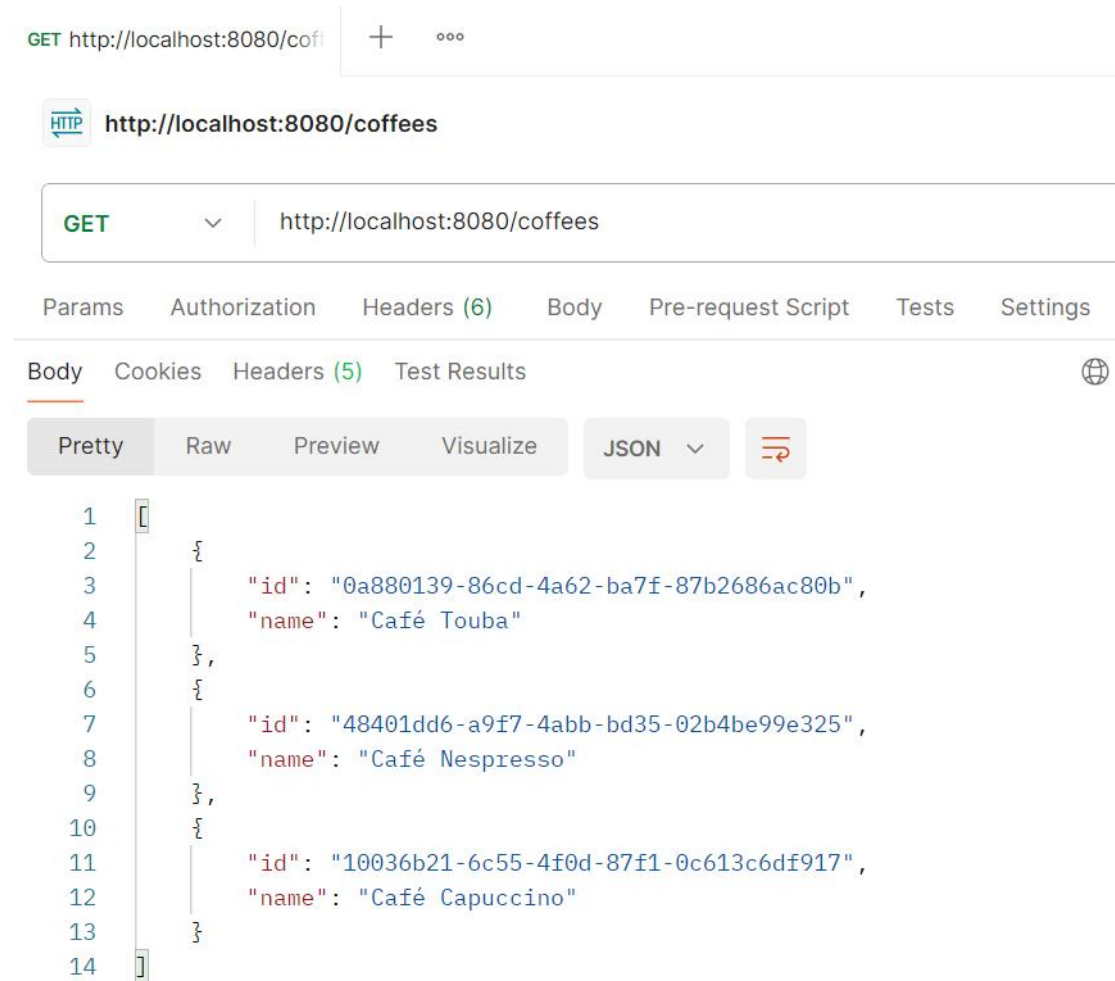
2- Créer la classe Coffee

```
CoffeeApplication.java X Coffee.java X RestCoffeeController.java
src\main\java\org.springframework.samples.petclinic> Coffee.java > Language Support for
1 package org.springframework.samples.petclinic;
2
3 import java.util.UUID;
4
5 public class Coffee {
6     private String id;
7     private String name;
8
9     public Coffee() {
10    }
11
12    public Coffee(String id, String name) {
13        this.id = id;
14        this.name = name;
15    }
16
17    public Coffee(String name) {
18        this.id = UUID.randomUUID().toString();
19        this.name = name;
20    }
21
22    // Getters et Setters
23    public String getId() {
```

3- Créer le contrôleur REST RestCoffeeController

5- Tester l'API avec Postman

*Lister tous





*Lister un cafe

HTTP <http://localhost:8080/caffe/48401dd6-a9f7-4abb-bd35-02b4be99e325>

GET <http://localhost:8080/caffe/48401dd6-a9f7-4abb-bd35-02b4be99e325>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (5) Test Results  200


Pretty Raw Preview Visualize JSON 


```
1 {
2   "id": "48401dd6-a9f7-4abb-bd35-02b4be99e325",
3   "name": "Café Nespresso"
4 }
```

*Modifions un cafe :



HTTP <http://localhost:8080/caffe/1cdb27e6-78a3-4ca6-bb7e-4669fd094bd0>


PUT <http://localhost:8080/caffe/1cdb27e6-78a3-4ca6-bb7e-4669fd094bd0>

Params Authorization Headers (8) Body  Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON 

```
1 {
2   "id": "1cdb27e6-78a3-4ca6-bb7e-4669fd094bd0",
3   "name": "Mon Café Touba..."
4 }
```

Body Cookies Headers (5) Test Results  200 OK 6 ms 237 B 

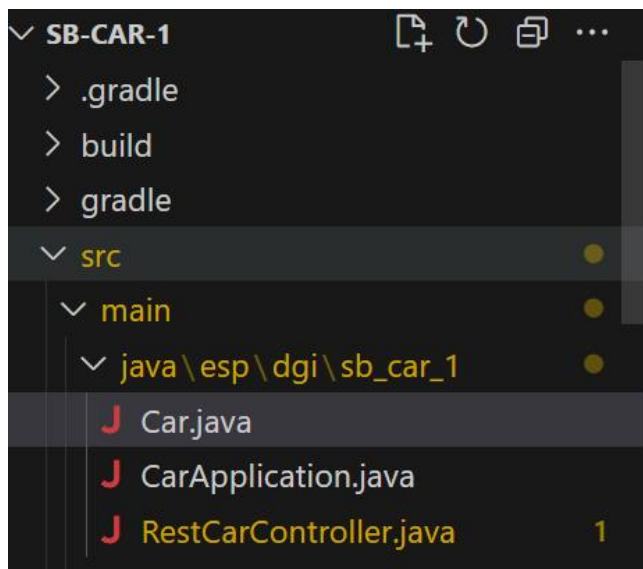
Pretty Raw Preview Visualize JSON 

```
1 {
2   "id": "1cdb27e6-78a3-4ca6-bb7e-4669fd094bd0",
3   "name": "Mon Café Touba..."
4 }
```

*Supprimons le cafe : cafe Capuccino


```
[
  {
    "id": "d51776a8-97ba-49be-ad7e-4ab27725f790",
    "name": "Café Touba"
  },
  {
    "id": "1cdb27e6-78a3-4ca6-bb7e-4669fd094bd0",
    "name": "Mon Café Touba..."
  }
]
```

Exercice 2 : API REST pour gérer des voitures (sb-car-1)



```
J Car.java    J RestCarController.java 1 X    application.properties    J CarA

src > main > java > esp > dgi > sb_car_1 > J RestCarController.java > ...
1  package esp.dgi.sb_car_1;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Optional;
6
7  import org.springframework.web.bind.annotation.DeleteMapping;
8  import org.springframework.web.bind.annotation.GetMapping;
9  import org.springframework.web.bind.annotation.PathVariable;
10 import org.springframework.web.bind.annotation.PostMapping;
11 import org.springframework.web.bind.annotation.PutMapping;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.RequestMapping;
14 import org.springframework.web.bind.annotation.RestController;
15
16 @RestController
17 @RequestMapping("/cars")
18 public class RestCarController {
19
20     private List<Car> cars = new ArrayList<>();
21
22     public RestCarController() {
23         System.out.println(x: "🚀 RestCarController chargé !");
```

```
J RestCarController.java 1    application.properties    J CarApplication.java X    D v

src > main > java > esp > dgi > sb_car_1 > J CarApplication.java > ...
1  package esp.dgi.sb_car_1;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class CarApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(primarySource:CarApplication.class, args);
11     }
12
13 }
14
```


```
src > main > java > esp > dgi > sb_car_1 > Car.java > Language Support for Java(TM) by Red Hat

1  package esp.dgi.sb_car_1;
2
3  import java.util.UUID;
4
5  public class Car {
6      private String id;
7      private String brand;
8      private String model;
9
10     public Car() {
11     }
12
13     public Car(String id, String brand, String model) {
14         this.id = id;
15         this.brand = brand;
16         this.model = model;
17     }
18
19     public Car(String brand, String model) {
20         this.id = UUID.randomUUID().toString();
21         this.brand = brand;
22         this.model = model;
23     }
}
```

```
src > main > resources > application.properties

1  spring.application.name=CarApplication
2  server.port=8081
3
4
```


Test avec Postman :

 http://localhost:8081/cars

GET

▼


http://localhost:8081/cars

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualize

JSON ▼



```
1  [
2    {
3      "id": "b3a65fed-5c65-4611-804e-8dd1b7cf75ba",
4      "brand": "Toyota",
5      "model": "Corolla"
6    },
7    {
8      "id": "59b802bd-7764-49f0-b778-772e6562847f",
9      "brand": "Peugeot",
10     "model": "208"
11   },
12   {
13     "id": "9d766f8d-ee5e-4d99-8d55-88976db7ede2",
14     "brand": "Renault",
15     "model": "Clio"
16   }
17 ]
```

GET

▼


http://localhost:8081/cars/9d766f8d-ee5e-4d99-8d55-88976db7ede2

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualize

JSON ▼



```
1  {
2    "id": "9d766f8d-ee5e-4d99-8d55-88976db7ede2",
3    "brand": "Renault",
4    "model": "Clio"
5  }
```


df

