BULZAN STEFAN

SPRING WEB

Building a web application with Spring Boot

WEB APPLICATION

- In order to implement REST protocol we need to link the resources to url (http://host:port/application/persons)
- For this we will declare a Controller
- A Controller is a special type of Spring Bean that will link an url to a class and methods
- @RestController : In order to create a REST controller controller you need to annotate the class with it
- @RequestMapping(“<my resource path>”) : you specify the url this class is linked to
- @GetMapping(“<path>”) : links the method to the url specified + GET
- @PostMapping(“<path>”) : links the method to the url specified + GET
- @PatchMapping(“<path>”) : links the method to the url specified + GET
- @PutMapping(“<path>”) : links the method to the url specified + GET
- @DeleteMapping(“<path>”) : links the method to the url specified + GET

REQUEST MAPPING

- @PathVariable : in order to have dynamic urls (like /persons/23) you should have a parameter annotated with @PathVariable and a placeholder in the url: /persons/{id}
  - ! The name of the annotated parameter should be the same as the name of the placeholder (id)
- @RequestParam : if you annotate a parameter with it, it will represent the url parameter with the same name
  - Be aware, by default it is required, so the request fails if the parameter is not present
  - Usually the params are optional, so just put @RequestParam(required=false)

REST VERBS

- GET : use it to get one or more resources
  - Can be called on resources (/persons) or resources with id (/persons/2)
- POST : use it to create a new resource.
  - Call it usually when you don’t have that resource (not for update) (/persons)
- PUT : use it to FULLY update a resource
  - Call it when you know the id of the resource you want to update (/persons/2)
- PATCH : use it to partially update a resource
  - You send only the fields you want to change in the body, not the whole resource
- DELETE: use it to delete a resource
  - eg. DELETE /persons/2
  - Never delete all resources... only if you really need to

REST BODY

- By default Spring converts the objects into and from json, so using spring you shouldn’t be too concerned by this
- But if you need to access the object that makes the conversions, use ObjectMapper from Jackson

DEMO

BUILDING A SIMPLE REST SERVICE

REST CLIENTS

- So for WWW, the client is the browser
- But when the application outputs JSON, how do we consume that?
  - The browser can show us json data, so GET requests can be tested in the browser
    - Install a browser extension (JSON Formatter) to show it better
- For more complex operations there are REST clients:
  - Postman (https://www.postman.com/)
  - Using this you can manually build a REST request

SPRING REST CLIENT

- RestTemplate is the Spring Web Client
- In order to have it you must include

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>5.2.5.RELEASE</version>
</dependency>
```

- In order to make a request you need to
  - instantiate it: new RestTemplate()
  - Call .getForObject(url, class). To get the return body object directly
  - getForEntity(url, class) will return the full return (response code, headers, body)
  - Every verb has a specific method: postForObject, putForObject, delete
  - There is a generic method that can be customised: exchange(url, method, requestEntity, responseType, uriVars)

DEMO

USING SPRING REST CLIENT

SPRING REST CLIENT

- When we fetch a collection of elements, things get more complicated
- We need to specify to RestTemplate what type the response should be transformed to, but the type is a List<Person> for example
- Generics are erased at runtime, so we need a workaround
- The workaround is ParameterizedTypeReference
  - This is a wrapper that will retain the type at runtime
- so, to get all persons in our application:

```
List<Person> body = restTemplate.exchange(
"http://localhost:8080/hello/list",
HttpMethod.GET,
new HttpEntity<>(null),
new ParameterizedTypeReference<List<Person>>() {})
.getBody();
```

DEMO

FETCHING COLLECTIONS WITH RESTTEMPLATE