Workshop - Spring rest

Durante questo workshop useremo Spring WEB per creare dei servizi REST con trasposto dati di tipo Json e RestTemplate per simulare il client nei test.

Building a RESTful Web Service;
https://spring.io/guides/gs/rest-service/
Consuming a RESTful Web Service:
https://spring.io/guides/gs/consuming-rest/
Jackson 2.1.3 Wiki
https://github.com/FasterXML/jackson

Parte 1 – Spring MVC - REST

Esercizio 1:

Importa in Eclipse il progetto Spring-REST-Training.
Estendiamo la classa `UserController` aggiungendo un nuovo metodo `getUserByd(Integer id)`

Il nuovo servizio deve rispondere all seguente URL:

```
curl -X GET http://localhost:8080/users/1/
```

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

`Hint: per intercettare il parametro id` utilizza `@PathVariable`

Esercizio 2

Estendiamo la classa `UserController` aggiungendo un nuovo metodo `addUser(String firstName, String lastName)`
Il servizio deve restituire l'utente appena inserito.

Fai un test del nuovo servizio usando:

```
curl -X POST http://localhost:8080/users/nome/cognome
```

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

Esercizio 3

Estendiamo la classa `UserController` aggiungendo un nuovo metodo `saveUser(Integer userID, String firstName, String lastName)` dove sono non nulle il nome e/o il cognome da modificare
Il servizio deve restituire l'utente appena modificato.

Fai un test del nuovo servizio usando:

```
curl -X PUT http://localhost:8080/users/id/nome/cognome
```

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

Esercizio 4

Estendiamo la classa `UserController` aggiungendo un nuovo metodo `deleteUser(Integer id)`

Fai un test del nuovo servizio usando:

```
curl -X DELETE http://localhost:8080/users/9/
```

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

Esercizio 5 (Bonus)

Estendiamo la classa `UserController` aggiungendo un nuovo metodo `addPhoneNumber(Integer userID, String phoneNumber, String type)`

Il servizio deve restituire l'utente appena modificato.

Fai un test del nuovo servizio usando:

```
curl -X POST http://localhost:8080/users/addPhoneNumber/1/55555/mobile
```

e

```
curl -X GET http://localhost:8080/users/1
```

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

Esercizio 6 (Bonus)

Estendiamo la classa `UserController` aggiungendo un nuovo metodo `saveUser(Integer userID, User user)` dove user è la versione aggiornata dell'utente che deve essere salvata nel sistema. Il metodo deve intercettare il metodo HTTP PUT

esempio:

```
curl -X PUT http://localhost:8080/users/1
```

Lo user da aggiornare deve essere nel body della request.

Crea almeno un test per il il nuovo metodo nella classe di test `UsersRestTest` utilizzando `RestTemplate`.

Hint: per intercettare il parametro user utilizza `@RequestBody User user`