Εργαστηριακή άσκηση 4: HTTP services

ΑΣΚΗΣΗ:

Spring Boot REST Example

The REST application follows the REST architectural approach. We use the REST application for developing and designing networked applications. It generates the HTTP request that performs CRUD operations on the data. Usually, it returns data in JSON or XML format.

Spring Boot REST API Example

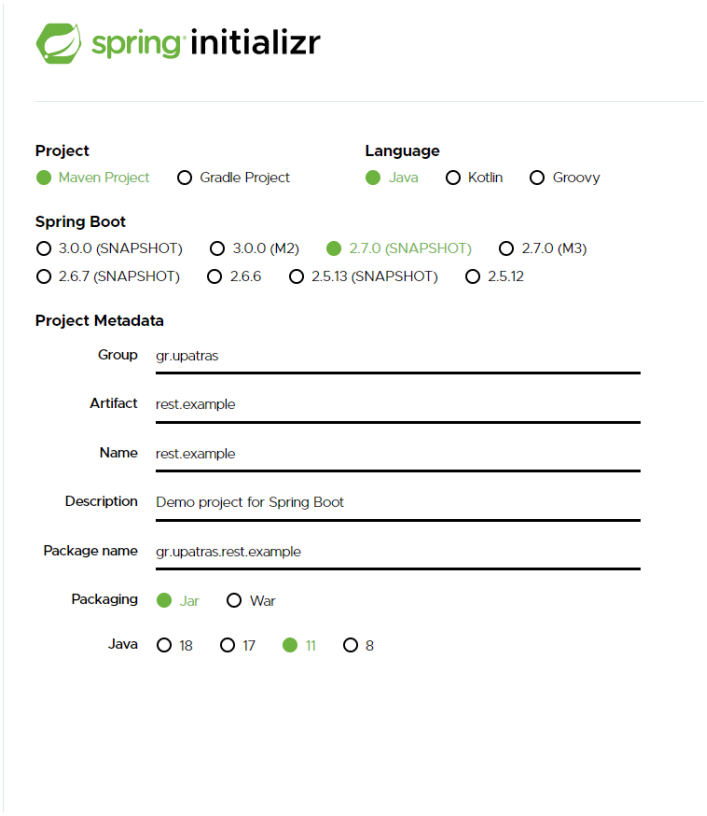
In the following example, we are going to create a REST application. In this application, we have created a list of products and return the same list. It returns the data in JSON format.

Step 1: Open the Spring Initializr https://start.spring.io/

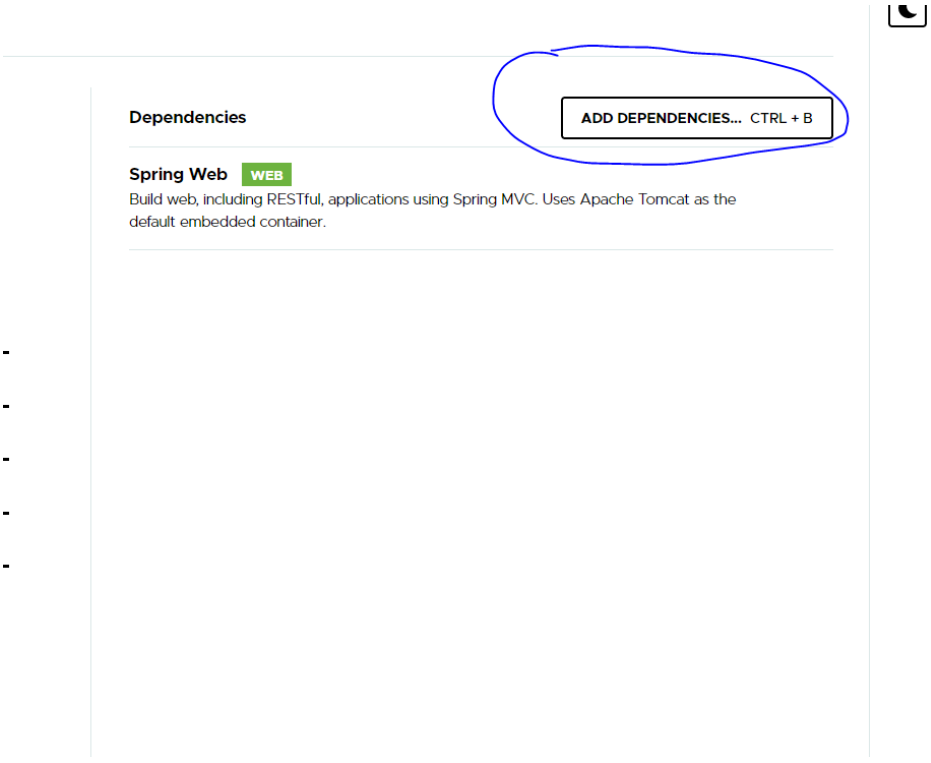
Step 2: Select the Spring Boot version **2.7.0 (SNAPSHOT)**

Step 3: Provide the **Group** name. We have provided the Group name **gr.upatras**

Step 4: Provide the **Artifact**. We have provided the **Artifact rest.example**.



Step 5: Add the **Spring Web** dependency.



-
-
-
-
-

Step 6: Click on the Generate button. When we click on the **Generate** button, it wraps all the specifications related to application into a **zip** file and downloads it to the local system.

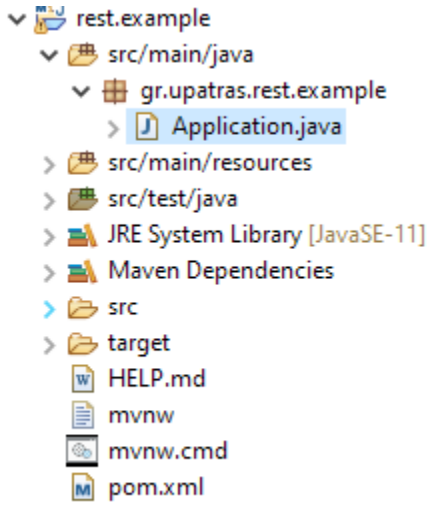
Step 7: Extract the zip file.

Step 8: copy the folder somewhere in your workspace

Step 9: Import the project.

File -> Import -> Maven->Existing Maven Projects -> Next -> Browse -> Select the folder rest-example -> Select Folder -> Finish

When the project imports successfully, we can see it in the **Package Explorer** section of the IDE.



Step 10: Create a **model** class in the package **gr.upatras.rest.example**. We have created a model class with the name **Product**. In this class, do the following:

- Create five private fields variable: id, pname, batchno, price, and noofproduct.

```
private int id;
private String pname;
private String batchno;
private double price;
private int noofproduct;
```

- Create a default constructor.

- Generate **Constructor using Fields**.
  Right-click on the file -> Source -> Generate Constructor using Fields -> Select All -> Generate
- Generate **Getters** and Setters.

Right-click on the file -> Source -> Generate Getters and Setters -> Select All -> Generate

Now, we need to create a controller.

Step 11: In the **gr.upatras.rest.example** package, create a class (This will be the Controller). We have created a class with the name **ProductController**.

- Annotate the class with the annotation **@RestController**.
- We have autowired the **IProductService** interface. We will create it in the next step.
- We have created a mapping **/product** by using the annotation **@GetMapping**.
- We have mapped a method **getProduct()** to the **/product**. The method returns a list of products.

ProductController.java

```
package gr.upatras.rest.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ProductController {
	@Autowired
	private IProductService productService;

//mapping the getProduct() method to /product
	@GetMapping(value = "/product")
	public List<Product> getProduct() {
//finds all the products
		List<Product> products = productService.findAll();
//returns the product list
		return products;
	}
}
```

Step 12: Create an interface in the package gr.upatras.rest.example with the name IProductService and define the findAll() method that returns a List of products.

IProductService.java

```
package gr.upatras.rest.example;

import java.util.List;
/**
 * @author ctranoris
 *
 */
public interface IProductService
{
List<Product> findAll();
}
```

Step 13: Create a **Service** class. We have created a service class in the package **gr.upatras.rest.example** with the name **ProductService**.

- Annotate the class with the annotation **@Service** and implements the **IProductService** interface.
- In this class, override the **findAll()** method by using the annotation **@Override**. The findAll() method of the ProductService class overrides the findAll() method of the **IProductService** interface.
- Create an object of **ArrayList**.
- **Add** the products in the ArrayList.
- Return the **list** of the products.

ProductService.java

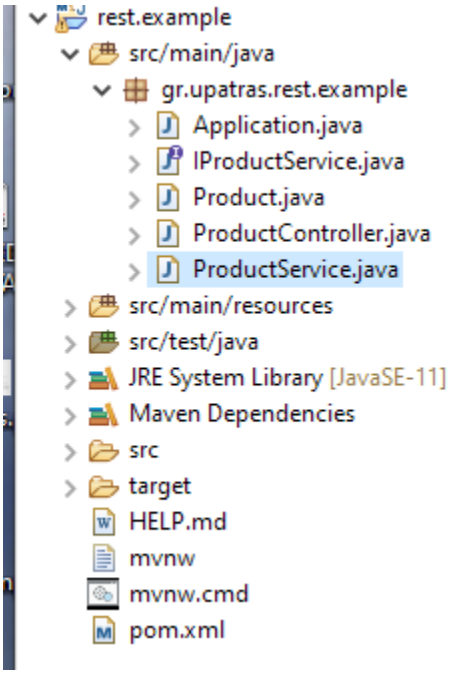
```
  package gr.upatras.rest.example;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;

@Service
public class ProductService implements IProductService {
	@Override
	public List<Product> findAll() {
//creating an object of ArrayList
```

```
            ArrayList<Product> products = new ArrayList<Product>();
//adding products to the List
            products.add(new Product(100, "Mobile", "CLK98123", 9000.00, 6));
            products.add(new Product(101, "Smart TV", "LGST09167", 60000.00, 3));
            products.add(new Product(102, "Washing Machine", "38753BK9", 9000.00,
7));
            products.add(new Product(103, "Laptop", "LHP29OCP", 24000.00, 1));
            products.add(new Product(104, "Air Conditioner", "ACLG66721", 30000.00,
5));
            products.add(new Product(105, "Refrigerator ", "12WP9087", 10000.00,
4));
//returns a list of product
            return products;
      }
}
```

Now we have created all the files and folders. After creating all the files, the project directory looks like the following:



Step 15: Open the **Application.java** file and run it as Java

Application. By default, it runs on port **8080**.

Application.java

```
package gr.upatras.rest.example;
```

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

@SpringBootApplication
public class Application {

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

}
```

Step 16: Open the browser and invoke the URL

Click on the link **Get all Products**. It returns a list of products in **JSON** format and the URL change to http://localhost:8080/product.

- Δειτε και με το Browser τα Network interactions

Για ένα πλήρες REST tutorial πχ: https://www.twilio.com/blog/create-rest-apis-java-spring-boot