

LUNAFACTORY :

Create a web application with Spring Boot

**Introduction**

Lunafactory project is a small Web Application, developed with IntelliJ and Spring Boot using Maven. The goal of this Application is to call a Web API, sort the datas in a specific way and display them on a web page.

• Requirements :

* Show a list of furnitures already assembled, sorted in alphabetical order and remove the duplicates
* Show a list of the 15th most expensive products

**Web Application content**

The Web Application has 3 pages :

* Home page (<url:localhost:9090/Lunafactory>), this page has two buttons. They allow you to access to the two different sorted lists of products
* Price Sorted Page (url:localhost:9090/TopProducts), this page show you a table with the 15th most expensive products from the whole list.
* Assembled sorted page (<url:localhost:9090/Products>), this page display a table with all assembled products of the list. All duplicates have been removed

**Code Structure**

**•** As a Spring Boot application, the project is launched using the class with @SpringBootApplication annotation. In this project this class is LunafactoryApplication.class located in com.exercise.lunafactory package.

• To create and manage the web pages we use a controller class ProductController.class with @RestController annotation to be able to use RESTFul web services easier.  
  
• In ProductController.class :

* To display the views on the website we have three different methods, each corresponds to a specific webpage. At the beginning of a method we specify the URL that will call this method using @RequestMapping annotation.  
  Those functions create a view from an html page converted into a ModelAndView Object.
* In the method productList() we call the Rest API using a restTemplate object and we execute the function *exchange* on it. In this function we put in parameters the url of the API request, we also add a header to specify our private token.  
  The response we get is a String with a JSON format containing all the products.  
  Then we parse this String into a Java Array of Product objects, making the access to the products very easy in the program.
* To sort those products following the requirements, we use few functions with small loops algorithms that check if items has the required attribute values and we store them into another list that we show on the webpages.

• To store the datas we get from the API in the code, Product and Dimension classes have been created. They have exact same attributes as the products we receive in the server response. With those classes it makes easier to store the datas in Java Object and manipulate them.