

Seminar 5

Spring framework

Useful links:

<http://spring.io/guides/gs/rest-service/>

<https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html>

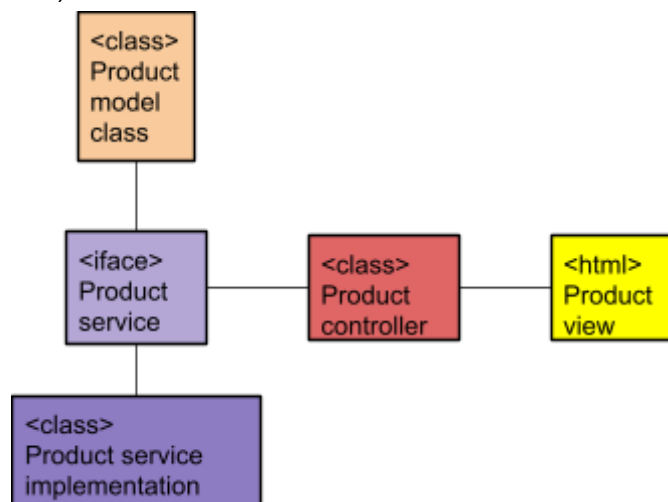
<https://stackify.com/spring-mvc/>

Assignment 0 (Run the test project for testing Your environment readiness for Spring Framework)

- Open Git and clone the demo-2 project from (or download it)
 - ssh - [git@github.com:KarinaSkirmante/FirstSpring_example.git](ssh:git@github.com:KarinaSkirmante/FirstSpring_example.git);
 - https - https://github.com/KarinaSkirmante/FirstSpring_example.git
- Import demo-2 in Your IDE (Sprint Tool Suite, Eclipse, Visual Studio Code, etc)
- Run demo-2 as Spring Boot Application and get the text “**Hello World in Spring - JAVA Course**” on <http://localhost:8080/testmsg/> url.

Seminar assignments

Create a Spring Application where it is possible to see all products in store, see only one product by id, add new product, delete product by id, update product by id. Possible classes, interface and view see below. Create project packages related to modules (models, services, servicesimpl, controllers)



Assignment 1 (Create model class related with Product)

- Create Product class with id, title, price, description, quantity, surname, age and id attributes. id should be unique;
- Create no-argument constructor (can be empty);
- Create argument constructor;
- Create functions:
 - Set;
 - Get;
 - toString;

Assignment 2 (Create interface related with Product Services)

- Create interface related to all actions by product, for example
 - selectAllProducts,
 - selectOneProductById,
 - insertNewProductByObject,
 - insertNewProductByParameters,
 - updateNewProductByObject,
 - updateNewProductByParameters,
 - deleteProductById
 - Others
- Don't forget - only function declaration is needed in interface (without function body) and all functions in interface are public and abstract.

Assignment 3 (Create interface implementation class related with Product Services)

- Create class which will implements interface from Assignment 2;
- Before class definition add @Service annotation (which annotates classes at the service layer). @Service documentation here: <https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/Service.html>
- Create ArrayList of Products where all products will be store;
- Override all functions from Assignment 2, for example, function selectOneProductById can be implemented

```
@Override
public Product selectOneProductById(int id) throws Exception {
    if (id >= 0 && id < productList.size()) {
        for (int i = 0; i < productList.size(); i++) {
            if (productList.get(i).getP_id() == id)
                return productList.get(i);
        }
    }
}
```

```
throw new Exception("Product are not in list");}
```

Assignment 4 (Create controller class related to Product)

- Create ProductController class with @Controller annotation;
- It is possible to create RequestMapping for all functions of the controller class. For example,

```
@RequestMapping("/product")
```

- Create where several methods related with Product are implemented

```
@GetMapping("/select")
```

```
public String selectAll(Model model)
```

```
@GetMapping("/select/{id}")
```

```
public String selectById(@PathVariable(name="id") int  
id, Model model)
```

```
@GetMapping("/insert")
```

```
public String insertNewProductGet(Product product)
```

```
@PostMapping("/insert")
```

```
public String insertNewProductPost(Product product)
```

```
@GetMapping("/update/{id}")
```

```
public String updateProductGet(@PathVariable(name="id")  
int id, Product product, Model model)
```

```
@PostMapping("/update/{id}")
```

```
public String updateProductPost(Product product)
```

```
@GetMapping("/delete/{id}")
```

```
public String deleteById(@PathVariable(name="id") int  
id, Model model)
```

- Create a ProductServiceImpl object and call service functions when it is needed.
- To pass arguments from controller to view Model object can be used. The addAttribute() function defines all attributes which will be used in view.

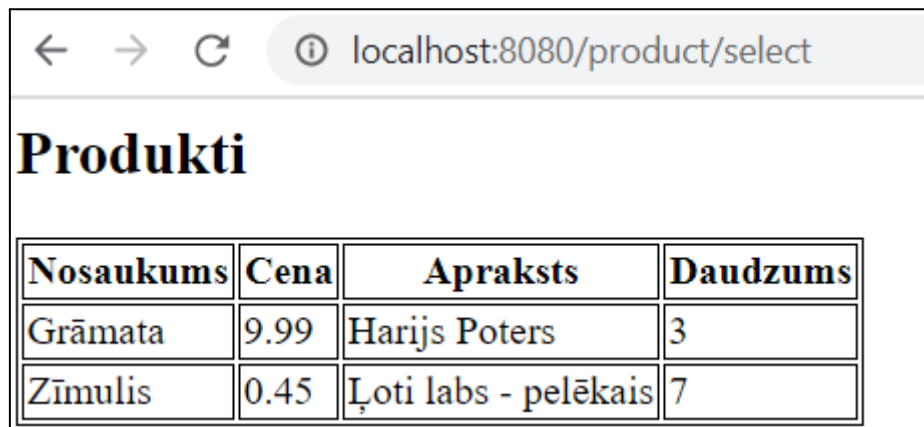
- For simple use, all functions will return a String variable such as html file name. For example,

```
@GetMapping("/select")
public String selectAll(Model model){
    model.addAttribute("productList",
        productServiceImpl.selectAllProducts());
    return "select-all-product";} //html file
```

select-all-products.html exists in src/main/resources/templates/

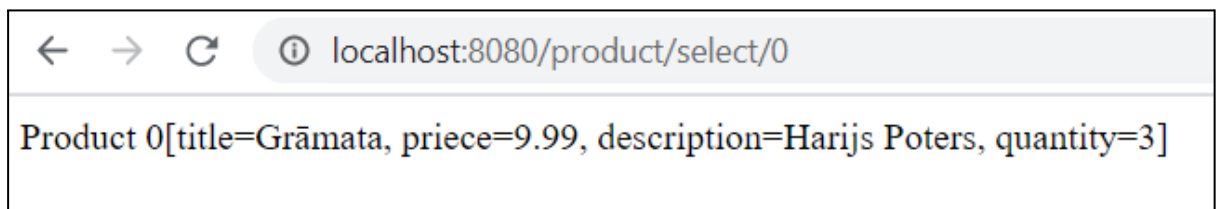
Assignment 5 (Create views related to Product)

- Create views related with Product/s
 - select-all-products.html where all products will be shown;



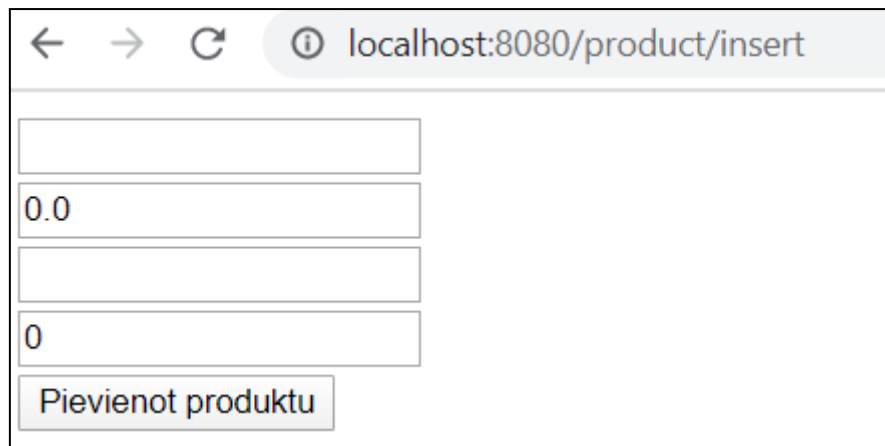
Nosaukums	Cena	Apraksts	Daudzums
Grāmata	9.99	Harijs Poters	3
Zīmulis	0.45	Ļoti labs - pelēkais	7

- select-one-product.html where only one passed product will be shown;

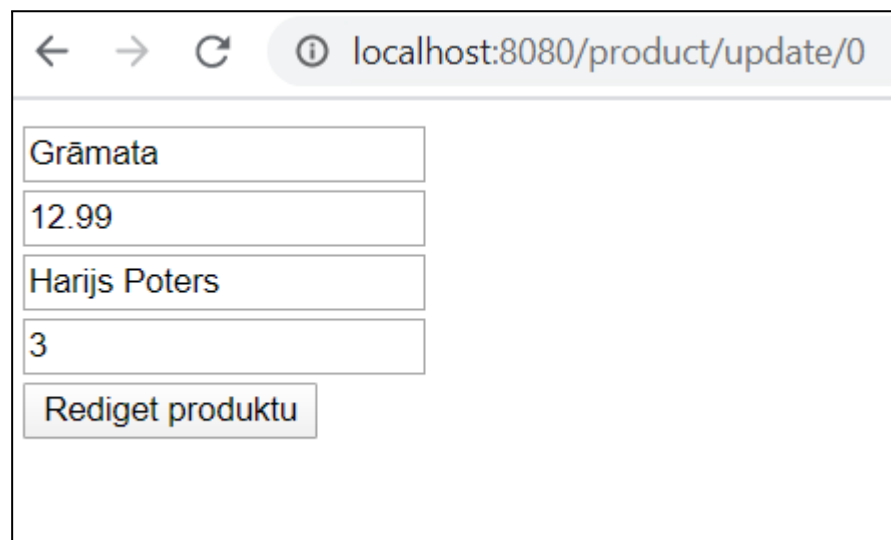


Product 0 [title=Grāmata, price=9.99, description=Harijs Poters, quantity=3]
--

- insert-product.html where submit button and 4 empty input fields are created (one input field per each Product variable).



- update-product.html which is similar like insert-product.html but with specific product variables inserted in input fields.

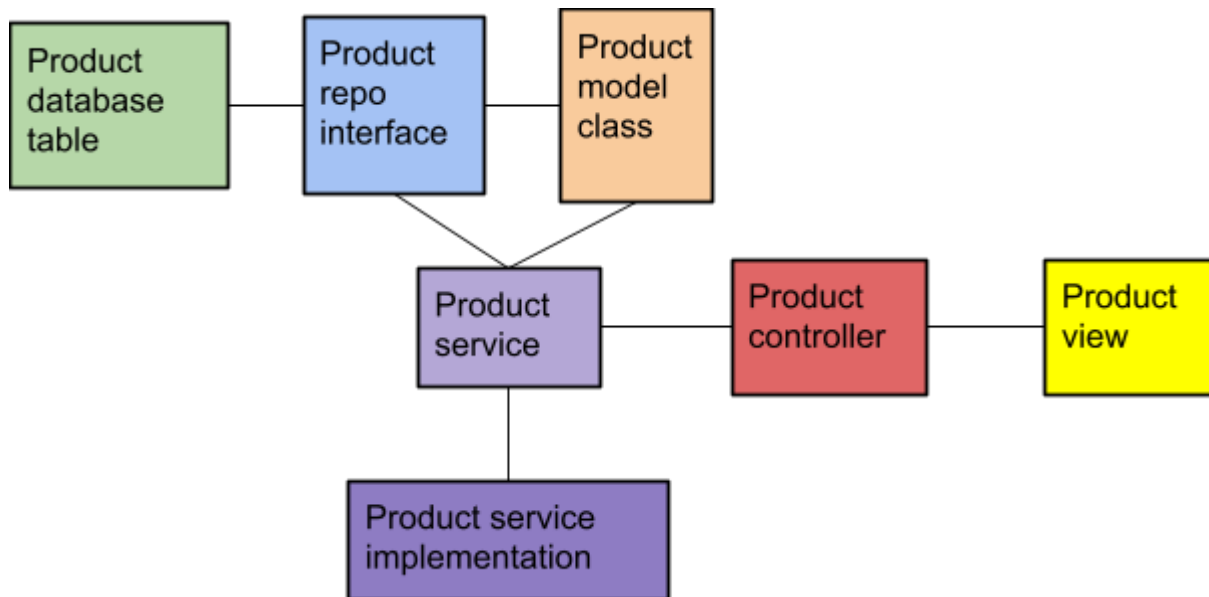


*Assignment 6 (validation)

Create validation annotations to Product model class and add validation in the controller and views. Take a look to here: <https://spring.io/guides/gs/validating-form-input/>

*Assignment 7 (Spring JPA and databases)

Implement functionality that allows to save all products in the database (the type of the database You can choose by Your own).



*Assignment 8 (add logger in project)

- Add a logger in project using **log4j** library (it could be add using maven dependencies or properties)
- Create a Logger object and add Error, Debug, Info type of messages in each controller method
- Create a **log4j.properties** file, and put it in the **resources** folder

The example of log4j.properties file:

Root logger option

log4j.rootLogger=DEBUG, stdout, file

Redirect log messages to console

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n

Redirect log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

#outputs to Tomcat home

log4j.appender.file.File=\${catalina.home}/logs/myapp.log

log4j.appender.file.MaxFileSize=5MB

log4j.appender.file.MaxBackupIndex=10

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n

-
- Create a simple controller to return a welcome page.

*Assignment 9 (upload the file)

- Create the controller which is responsible of the file uploading in view.
- Create the InputStream object for file upload in programm
- Copy file in Spring project using IOUtils.copy() function
- Flush the response buffer