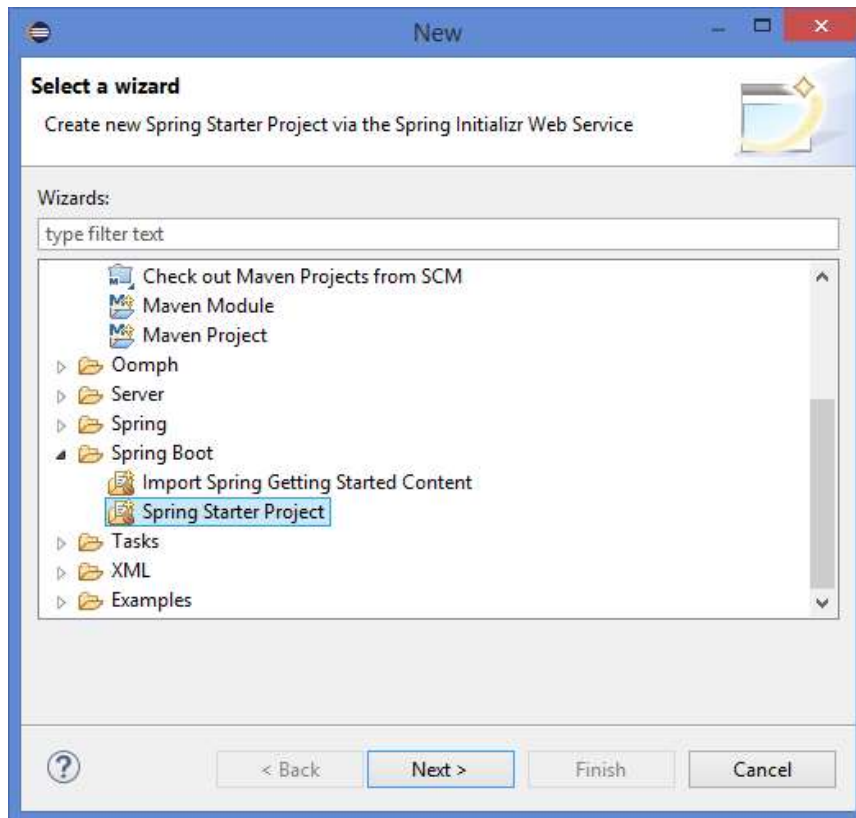


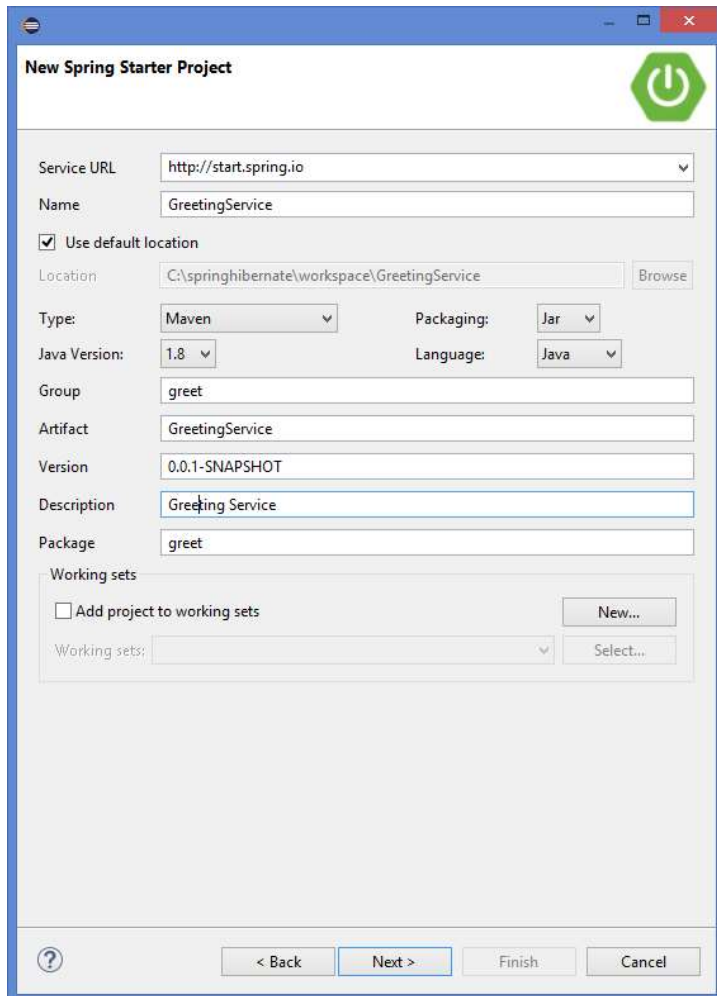
## Lab assignment 2: Application architecture

### Exercise 1: Spring Boot REST

In Eclipse select the menu items **File->New->Other** and select **Spring Boot->Spring Starter Project**.



Click **Next**.



**New Spring Starter Project**

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

Description:

Package:


Working sets

☐ Add project to working sets

Working sets:

Fill in the details as given in the picture above and click **Next**.

New Spring Starter Project Dependencies



Spring Boot Version: 2.0.3

Frequently Used:

☐ Config Client

☐ HSQLDB

☐ Reactive Web

☐ Eureka Discovery

☐ JPA

☒ Web

☐ Eureka Server

☐ Kafka Streams

☐ Web Services

Available:

Type to search dependencies

▶ Cloud Core

▶ Cloud Discovery

▶ Cloud Messaging

▶ Cloud Routing

▶ Cloud Tracing

▶ Core

▶ I/O

▶ Integration

▼ NoSQL

☐ Redis

☐ Reactive Redis

☒ MongoDB

☐ Reactive MongoDB

☐ Embedded MongoDB

☐ Elasticsearch


Selected:

X MongoDB

X Web

Make Default

Clear Selection



< Back

Next >

Finish

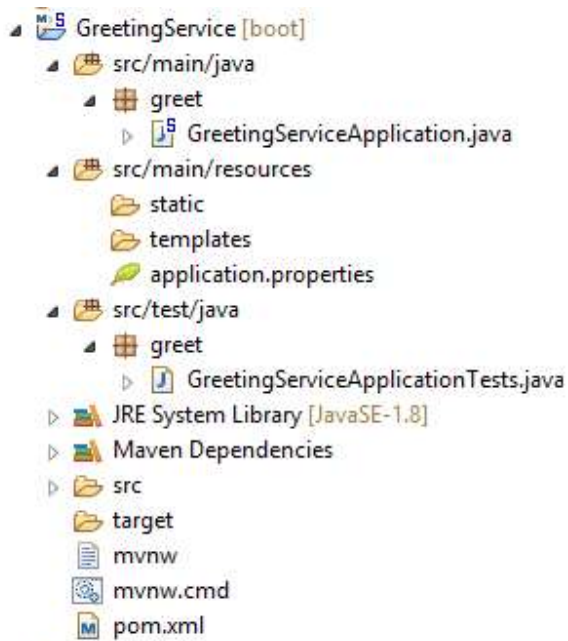
Cancel

Select the checkbox for **Web** and the checkbox for **Mongo**.  
Also select Spring Boot version **2.0.3**.

Then click **Finish**.

Wait till the **GreetingService** project is created.

The following GreetingService project is created:



In the created GreetingService project, write the following classes:

```
@RestController
public class GreetingController {

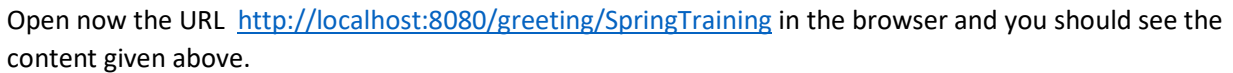
    @RequestMapping("/greeting/{message}")
    public ResponseEntity<?> getGreeting(@PathVariable("message") String message) {
        Greeting greeting = new Greeting("");
        greeting.setContent("Hello World "+message);
        return new ResponseEntity<Greeting>(greeting, HttpStatus.OK);
    }
}
```

```
public class Greeting {  
    private String content;  
  
    public Greeting(String content) {  
        this.content = content;  
    }  
  
    public String getContent() {  
        return content;  
    }  
  
    public void setContent(String content) {  
        this.content = content;  
    }  
}
```

```
@SpringBootApplication  
public class HelloServiceApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(HelloServiceApplication.class, args);  
    }  
}
```

Right-click the **HelloServiceApplication.java** file and select **Run as->Spring Boot App**.

...

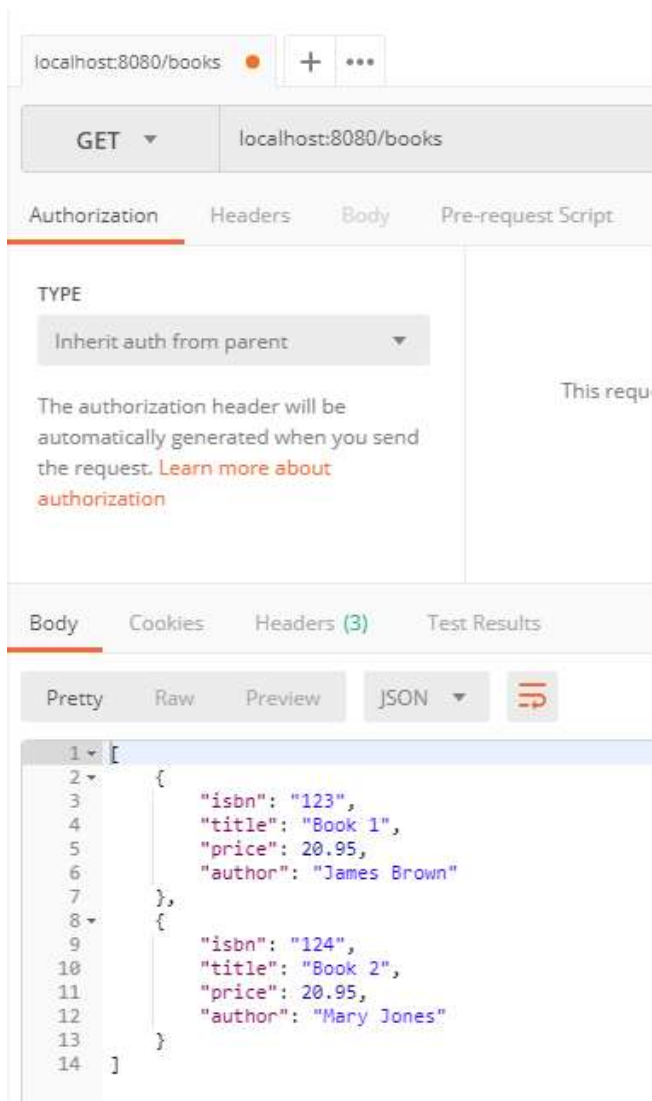


```
addBook(Book book);
deleteBook(String isbn);
getBook(String isbn);
getAllBooks();
```

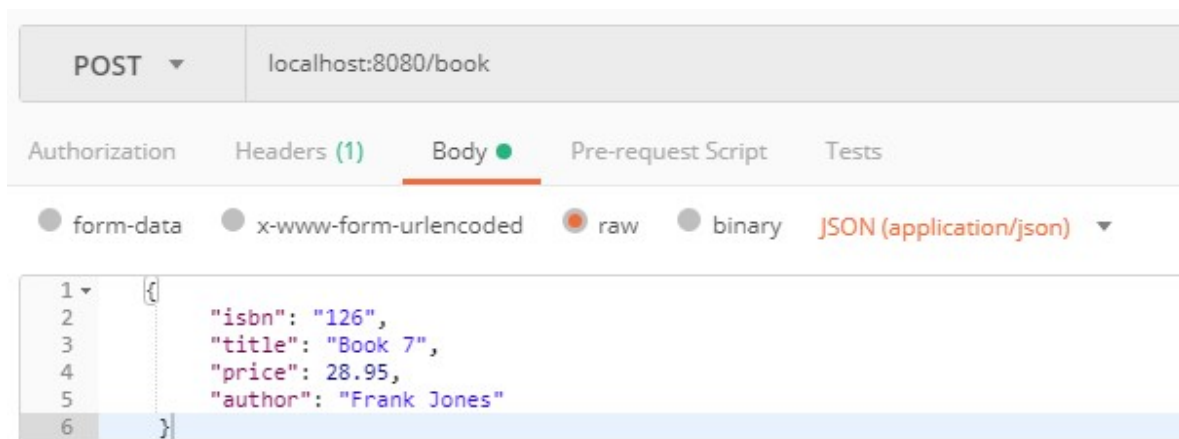
To test the BookService, you first have to install **Postman**. Download Postman from <https://www.getpostman.com/apps> and install Postman.

### Test the BookService with Postman:

Get all the books



Posting a new book:



Deleting an existing book:

DELETE ▼	localhost:8080/book/126			
Authorization	Headers	Body	Pre-request Script	Tests



## Exercise 2: Spring Mongo

Now we want to store the products of exercise 2 to the Mongo database.

First we have to start the mongo database by running  
**C:\architecturetraining\mongodb\bin\startmongo.bat.**

Modify the application of Exercise 1 as such that the Books are stored in the Mongo database.

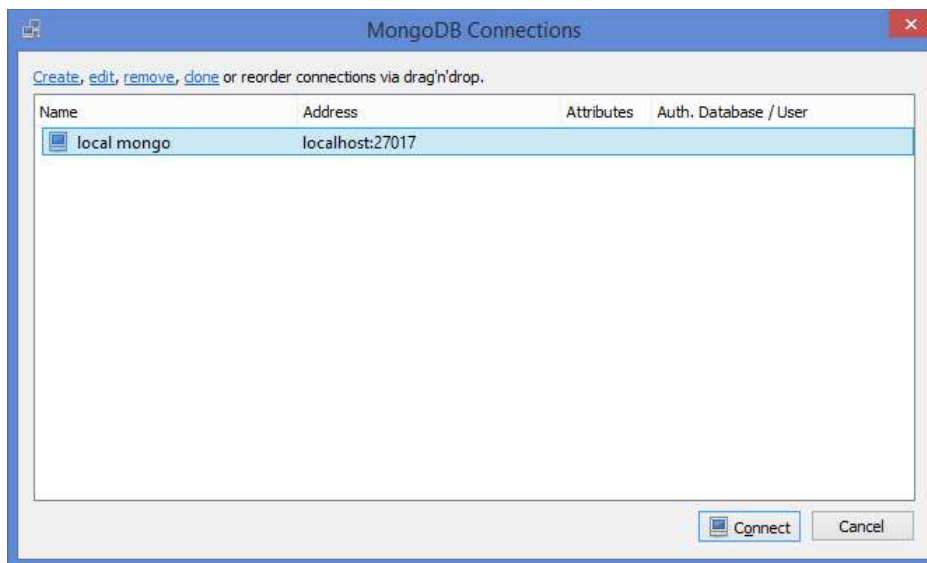
Create a BookRepository interface

Add the following properties to application.properties:

**spring.data.mongodb.host=localhost**  
**spring.data.mongodb.port=27017**  
**spring.data.mongodb.database=testdb**

Modify the BookService class so that the books are stored in the database.

Now run **C:\architecturetraining\robo3t\robo3t.exe**. This application is a Client application that allows you to look into the mongo database.



Click **Connect**.

In the **testdb** you should see now all the books.

Add some new books and check if they are stored in the database.

### **Exercise 3: Webshop implementation**

Implement the ProductCatalogService from the webshop design of Lab 2 with Spring Boot. This ProductCatalogService has the following (subset) of methods:

```
public void addProduct(String productnumber, String description, double price)  
public Product getProduct(String productnumber)  
public void setStock(String productnumber, int quantity, String locationcode)
```

The products are stored in the mongo database.

Use REST to call the methods on the ProductCatalogService

In the same application, implement the ShoppingService from the webshop design of Lab 2 with Spring Boot with the following (subset) of methods:

```
public void addToCart(String cartId, String productnumber, int quantity)  
public ShoppingCart getCart(String cartId)
```

The shoppingcart is stored in the mongo database.

Use REST to call the methods on the ShoppingService