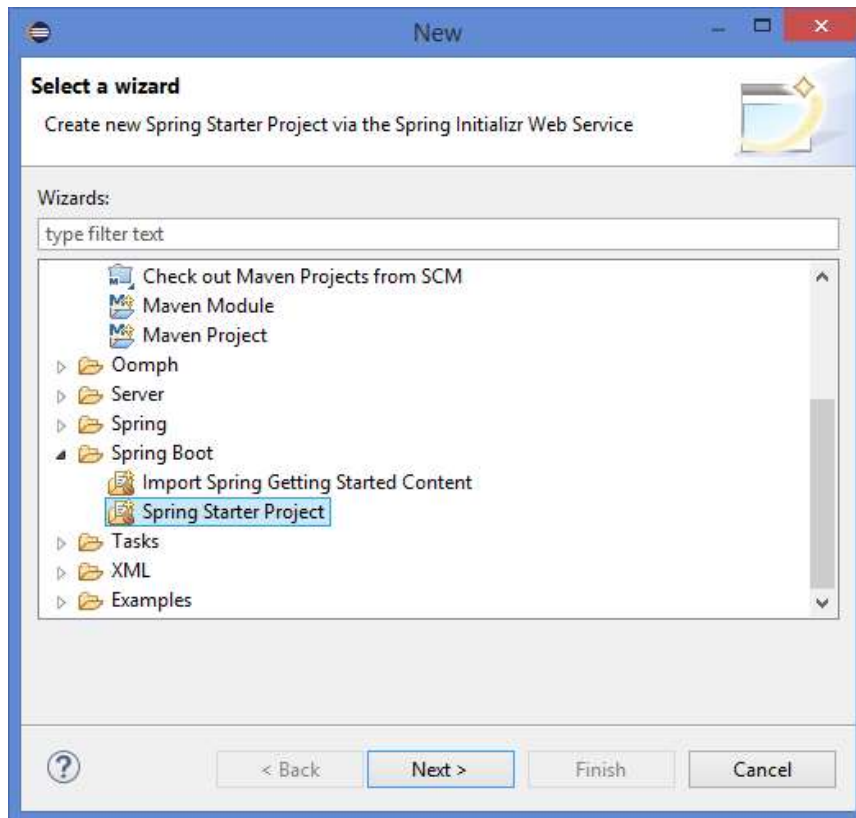


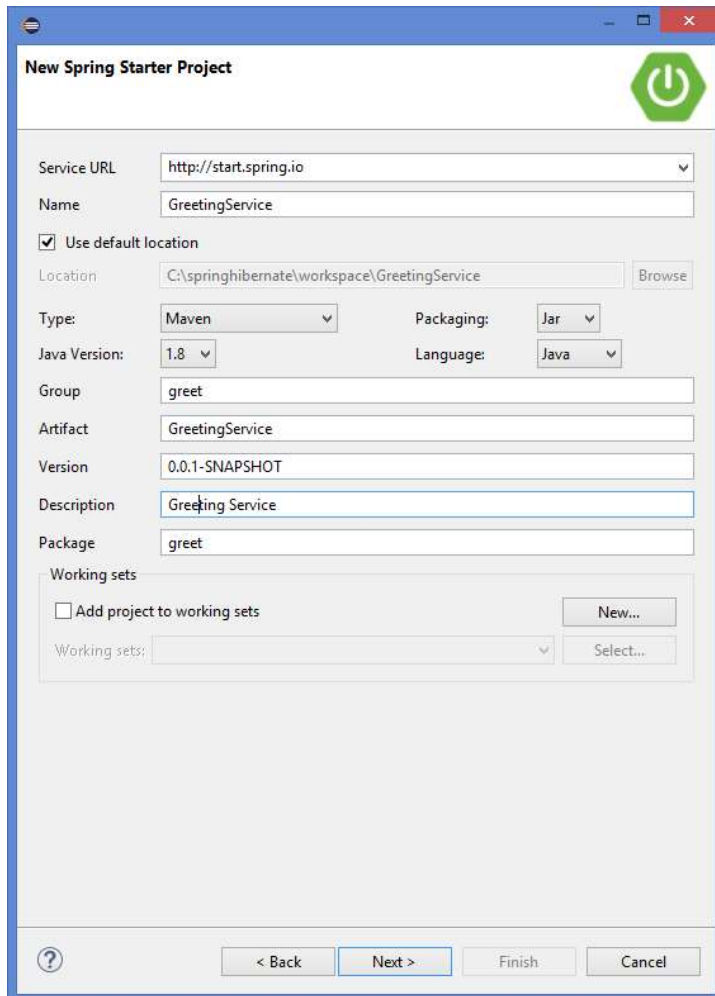
## Lab assignment 2: Application architecture

### Exercise 1: Spring Boot REST

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



Click **Next**.




The image shows a 'New Spring Starter Project' dialog box with the following fields and options:

- Service URL:** `http://start.spring.io`
- Name:** `GreetingService`
- ☒ **Use default location**
- Location:** `C:\springhibernate\workspace\GreetingService` (with a 'Browse' button)
- Type:** `Maven`
- Packaging:** `Jar`
- Java Version:** `1.8`
- Language:** `Java`
- Group:** `greet`
- Artifact:** `GreetingService`
- Version:** `0.0.1-SNAPSHOT`
- Description:** `Greeting Service`
- Package:** `greet`
- Working sets:**
  - ☐ **Add project to working sets** (with a 'New...' button)
  - Working sets:** (empty dropdown menu with a 'Select...' button)

At the bottom, there is a help icon (?) and four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted.


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

☐ Eureka Discovery

☐ Eureka Server

☐ HSQLDB

☐ JPA

☐ Kafka Streams

☐ Reactive Web

☒ Web

☐ 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

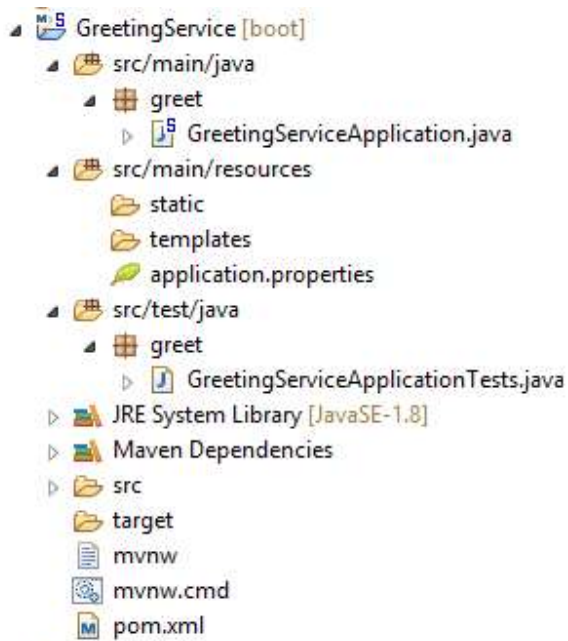
Cancel

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 {

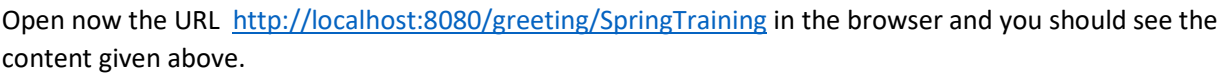
    @RequestMapping("/greeting/{message}")
    public ResponseEntity<?> getGreeting(@PathVariable("message") String message) {
        Greeting greeting = new Greeting("");
        greeting.setContent("Hello World "+message);
        return new ResponseEntity<Greeting>(greeting, HttpStatus.OK);
    }
}
```

```
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**.

• • •



The Book class has the following properties: isbn, author, title, price

## Test the BookService with Postman:

Get all the books

localhost:8080/books

GET localhost:8080/books

Authorization Headers Body Pre-request Script

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This requ

Body Cookies Headers (3) Test Results

Pretty Raw Preview JSON

```
1 [
2   {
3     "isbn": "123",
4     "title": "Book 1",
5     "price": 20.95,
6     "author": "James Brown"
7   },
8   {
9     "isbn": "124",
10    "title": "Book 2",
11    "price": 20.95,
12    "author": "Mary Jones"
13  }
14 ]
```

Posting a new book:

POST localhost:8080/book

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "isbn": "126",
3   "title": "Book 7",
4   "price": 28.95,
5   "author": "Frank Jones"
6 }
```

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.

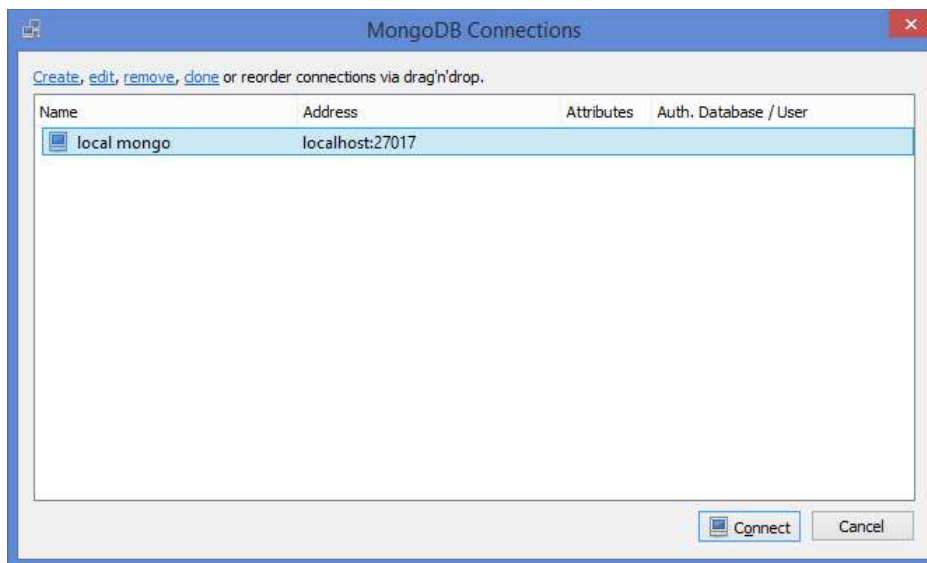
Create a BookRepository interface

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

**IMPORTANT: You only learn from this lab if you do this lab yourself. If you copy a solution from someone else, then this is in violation with the academic honesty policy of the university and the penalty will be a NC for the course.**

### **What to hand in?**

You can submit a zip file with your solutions in sakai. All labs that you submit in sakai should contain a **readme.txt** file with the following statement:

*I hereby declare that this submission is my own original work and to the best of my knowledge it contains no materials previously published or written by another person.*

*[your name as signature]*

**If you do not have this readme.txt file, you don't get credit for your lab submission.**