<http://localhost:8080> – така си достъпвам моята машина ако има сървър

http://localhost е същото като <http://127.0.0.1>

CSS – да използваме bootstrap.css – доста неща прави – една торба с информация наготово за това как трябва да изглежда страницата ни.

<https://www.google.com/search?q=spring> – преди въпросчето е URL параметър, след въпросчето е query параметър

# (MVC) Model-View Controller

## **-Views**

**-** presentation / Render (UI)UserInterface (produce HTML)

May use templates to dynamically generate HTML

## **- Controllers (logic) –**

Prepare UI (presentation logic) and Update database (business logic)

Process the requests

A set of classes that handles

* Communication from the user
* Overall application flow
* Application-specific logic (business logic)

Every controller has one or more "actions"

## **- Models (data)**

Data **access classes** or ORM (**Object-relational mapping**)

------------- ----------------- --------------

# Spring MVC == open source Web MVC framework for Java

## **Spring Boot with Maven – Maven VIEW**

<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>Phonebook</**groupId**>  
 <**artifactId**>phonebook</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 <**packaging**>jar</**packaging**>  
  
 <**name**>phonebook</**name**>  
 <**description**>Demo project for Technology Fundamentals Course @SoftUni</**description**>  
  
 <**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.0.4.RELEASE</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  
  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**project.reporting.outputEncoding**>UTF-8</**project.reporting.outputEncoding**>  
 <**java.version**>11</**java.version**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-devtools</**artifactId**>  
 <**scope**>runtime</**scope**>  
 </**dependency**>  
 </**dependencies**>  
  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
  
</**project**>

## **SpringBoot в public static void main** - src/main/java/app/MvcAppExample.java:

**package** phonebook;  
  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
**public class** PhonebookApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(PhonebookApplication.**class**, args);  
 }  
}

## **Класът Contact**

**package** phonebook.entity;  
  
**public class** Contact {  
 **private** String **name**;  
 **private** String **number**;  
  
 **public** Contact(String name, String number) {  
 **this**.**name** = name;  
 **this**.**number** = number;  
 }  
  
 **public** Contact() {}  
  
 **public** String getName() {  
 **return this**.**name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getNumber() {  
 **return this**.**number**;  
 }  
  
 **public void** setNumber(String number) {  
 **this**.**number** = number;  
 }  
}

## **Spring Controllers**

MVC controllers hold actions, mapped to URL by annotations

@Controller  
**public class** ContactController {  
 **private** List<Contact> **contacts**;

@GetMapping(**"/"**) *//вземи данните за всички***public** ModelAndView index(ModelAndView modelAndView) {  
………………….  
}

}

## **Controller Actions**

@GetMapping(**"/"**) *//вземи данните за всички спрямо URL мапинга***public** ModelAndView index(ModelAndView modelAndView) {  
 modelAndView.setViewName(**"index"**);  
 modelAndView.addObject(**"contacts"**, **this**.**contacts**);  
  
 **return** modelAndView;  
}

@PostMapping(**"/"**) *//покажи всички***public** String storeContact(Contact contact) {  
 **this**.**contacts**.add(contact);  
  
 **return "redirect:/"**; *//върни ме на началната страница*}

@DeleteMapping(**"/contacts/{name}"**) *//това взема от URL-а***public** String deleteContact(@PathVariable String name) {  
 **this**.**contacts** = **this**.**contacts** .stream()  
 .filter(c -> !c.getName().equals(name)) //листът става с 1 по-малко  
 .collect(Collectors.*toList*());  
  
 **return "redirect:/"**; *//върни ме на началната страница*}

## **Thymeleaf – Template View Engine**

Thymeleaf allows us to:

* Use **variables** / **collections** in our views
* Execute **operations** on our variables
* **Iterate** over **collections**

All Thymeleaf tags and attributes begin with **th:**

Example - <p **th**:text="Example">…</p>

**th**:block is an attribute container that disappears in the HTML - <**th scope="col"**>Name</**th**>

### **Thymeleaf Variable Expressions** в index.html

Variable Expressions are executed on the context variables

<**td th:text="${contact.name}"**></**td**>

### **Thymeleaf Link Expressions** в index.html

Link Expressions are used to build URLs

#### Example:

<a th:href="@{/register}">Register</a>

#### Passing query string parameters:

<a th:href="@{/details(id=${game.id})}">Details</a>

#### Create dynamic URLs:

<a th:href="@{/games/{id}/edit(id=${game.id})}">Edit</a>

### In Thymeleaf you can create HTML forms:

<form th:action="@{/user}" th:method="post">

<input type="number" name="id"/>

<input type="text" name="name"/>

<input type="submit">

</form>

<**form class="form-horizontal" th:method="POST"**> ….. **</form**>

<**form th:action="@{/contacts/{name}(name=${contact.name})}" th:method="DELETE"**>  
 <**input type="submit" value="Delete Contact"**/>  
 </**form**>

### You can parse the input as an object

@PostMapping("/user")

public ModelAndView register(@ModelAttribute User user) { … }

### You can use if statements in thymeleaf using **th:if**

<div th:if="${…}">

<p>The statement is true"</p>

</div>

### You can create inverted if statements using **th:unless**

<div th:unless="${…}">

<p>The statement is false"</p>

</div>

### For loop

<div th:each="element :

${#numbers.sequence(start, end, step)}">

<p th:text="${element}"></p>

</div>

<div th:each="element : ${#numbers.sequence(1, 5, 1)}">

<p th:text="${element}"></p>

</div>

*//1 2 3 4 5*

### For-each loop

<div th:each="item : ${collection}">

<p th:text="${item.property}"></p>

</div>

<div th:each="book : ${books}">

<p th:text="${book.name}"></p>

<p th:text="${book.author}"></p>

<p th:text="${book.price}"></p>

</div>

## Задачата, частта index.html

<**div class="container"**>  
 <**legend**>All Contacts</**legend**>  
 <**table class="table"**>  
 <**thead**>  
 <**tr**>  
 <**th scope="col"**>Name</**th**>  
 <**th scope="col"**>Number</**th**>  
 <**th scope="col"**>Actions</**th**>  
 </**tr**>  
 </**thead**>  
 <**tbody**>  
 <**tr th:each="contact : ${contacts}"**>  
 <**td th:text="${contact.name}"**></**td**>  
 <**td th:text="${contact.number}"**></**td**>  
 <**td**>  
 <**form th:action="@{/contacts/{name}(name=${contact.name})}" th:method="DELETE"**>  
 <**input type="submit" value="Delete Contact"**/>  
 </**form**>  
 </**td**>  
 </**tr**>  
 </**tbody**>  
 </**table**>  
  
 <**form class="form-horizontal" th:method="POST"**>  
 <**fieldset**>  
 <**legend**>New Contact</**legend**>  
 <**div class="form-group"**>  
 <**label for="name" class="col-lg-2 control-label"**>Name</**label**>  
 <**div class="col-lg-10"**>  
 <**input type="text" autofocus="autofocus" name="name" title="Name" class="form-control"  
 id="name"**/>  
 </**div**>  
 </**div**>  
 <**div class="form-group"**>  
 <**label for="number" class="col-lg-2 control-label"**>Number</**label**>  
 <**div class="col-lg-10"**>  
 <**input type="text" autofocus="autofocus" name="number" title="Number" class="form-control"  
 id="number"**/>  
 </**div**>  
 </**div**>  
 <**div class="form-group"**>  
 <**div class="col-lg-10 col-lg-offset-2"**>  
 <**button type="submit" class="btn btn-primary"**>Add</**button**>  
 </**div**>  
 </**div**>  
 </**fieldset**>  
 </**form**>  
  
 <**hr**/>  
  
</**div**>

## Summary - **Passing Attributes to View**

### Passing a collection to the view

<**tbody**>  
 <**tr th:each="contact : ${contacts}"**>  
 <**td th:text="${contact.name}"**></**td**>  
 <**td th:text="${contact.number}"**></**td**>  
 </**tr**>  
</**tbody**>

@GetMapping(**"/"**) *//вземи данните за всички спрямо URL мапинга***public** ModelAndView index(ModelAndView modelAndView) {  
 modelAndView.setViewName(**"index"**);  
 modelAndView.**addObject**(**"contacts"**, **this**.**contacts**); //добавя целият списък до момента  
  
 **return** modelAndView;  
}

### Passing a String to the view

<body>

<p>Hello, <span th:text="${name}"></span></p>

</body>

@GetMapping("/hello")

public ModelAndView hello(ModelAndView modelAndView) {

modelAndView.setViewName("hello");

modelAndView.**addObject**("name", "Peter");

return modelAndView;

}