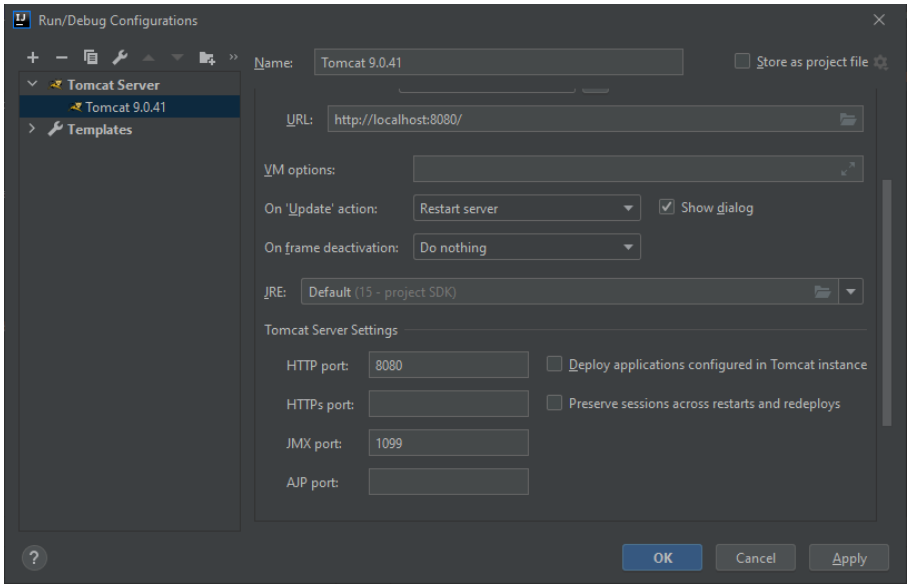
20. Spring Framework (MVC)

- Создание “model-view-controller” приложений
- Использование шаблонизатора страниц Thymeleaf

Pom.xml (добавление зависимостей):

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <spring.version>5.2.1.RELEASE</spring.version>
</properties>

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.thymeleaf</groupId>
        <artifactId>thymeleaf-spring5</artifactId>
        <version>3.0.11.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>4.0.1</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Добавление tomcat-сервера:



Сценарий: пример на основе калькулятора. Отправка запросов с параметрами в виде чисел и операций над ними, и отображение результата в html-странице с помощью Thymeleaf.

Структура проекта

config.DispatchcerServletInitializer (класс для конфигурации DispatcherServlet, указание config класса и основного маппинга):

```
public class MySpringMvcDispatcherSerlvetIntitializer
        extends AbstractAnnotationConfigDispatcherServletInitializer {
    @Override
    protected Class<?>[] getRootConfigClasses() {
        return null;
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[]{SpringConfig.class};
    }

    @Override
    protected String[] getServletMappings() {
        return new String[]{"/"};
    }
```

config.SpringConfig (класс конфигурации приложения, создание контекста, параметров отображения веб-страниц, аннотация @EnableWebMvc, классы Spring Resource Template Resolver, Spring Template Engine):

```java
@Configuration
@ComponentScan("org.example.springCourse")
@EnableWebMvc
public class SpringConfig implements WebMvcConfigurer {

    private final ApplicationContext applicationContext;

    @Autowired
    public SpringConfig(ApplicationContext applicationContext) {
        this.applicationContext = applicationContext;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver templateResolver = new
SpringResourceTemplateResolver();
        templateResolver.setApplicationContext(applicationContext);
        templateResolver.setPrefix("/WEB-INF/views/");
        templateResolver.setSuffix(".html");
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine() {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver());
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }

    @Override
    public void configureViewResolvers(ViewResolverRegistry registry) {
        ThymeleafViewResolver resolver = new ThymeleafViewResolver();
        resolver.setTemplateEngine(templateEngine());
        registry.viewResolver(resolver);
    }
}
```

controllers.Calculator: (основной класс Калькулятора, задание маппингов для каждой операции, возврат шаблона веб-страницы, аннотация @Controller, @RequestMapping, @GetMapping, @RequestParam):

```java
@Controller
@RequestMapping("/first")
public class Calculator {
    @GetMapping("/calculator")
    public String calculation(@RequestParam(value = "a") int a,
                              @RequestParam(value = "b") int b,
                              @RequestParam("action") String action,
                              Model model) {
        double result;

        switch (action) {
            case ("addition"):
                result = a + b;
                break;
            case ("subtraction"):
                result = a - b;
```

```
                break;
            case ("miltiplication"):
                result = a * b;
                break;
            case ("division"):
                result = a / (double) b;
                break;
            default:
                result = 0;
        }
        model.addAttribute("result", result);
        return "first/calculator";

    }
```

views.calculator.html (отображение результата по адресу first/calculator, вывод на экран переменной result):

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">
<head>
    <meta charset="ISO-8859-1">
    <title>My calculator</title>
</head>
<body>
Result:
<p th:text="${result}"/>
</body>
</html>
```

Результат

Отправка запроса с со следующими параметрами:

?a=100&b=100&action=addition