Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
Кафедра «Информатика»

Практический пример №7
архитектура REST в Spring. REST-клиент.

Составил: Старший преподаватель кафедры «Информатика»
Черниговский Алексей Сергеевич

Красноярск 2020

Содержание

Предисловие....3
1 Создание web-приложения поддерживающего архитектуру REST....3
1.1 Создание Maven проекта....3
1.2 Зависимости....3
1.3 Структура проекта....4
1.4 Контроллер StudentsController....5
1.5 applicationContext.xml конфигурация....5
1.6 SpringConfig.java конфигурация....6
1.7 web.xml конфигурация....7
1.8 show.ftl страница....8
1.9 StudentService.java файл....8
1.10 Описание....9
2 Создание элементарного REST-клиента....9
2.1 Создание Maven проекта....9
2.2 Зависимости....9
2.3 Структура проекта....9
2.4 App.java....10
2.5 Описание....10

Предисловие

В процессе разработки на Spring могут возникнуть неоднозначности. В случае использования других версий библиотек, отсутствия необходимых библиотек или неправильной настройки могут возникать непредвиденные ошибки. Не стесняйтесь спрашивать у преподавателя как на занятии, так и в оффлайн режиме по любым доступным каналам связи.

1 Создание web-приложения поддерживающего архитектуру REST

1.1 Создание Maven проекта

Создаем новый Maven проект в IDE Eclipse из архетипа webapp
Зададим ему Artifact Id: rest.

1.2 Зависимости

Добавим в pom.xml следующие зависимости:

```
 <properties>
       <spring.version>5.2.8.RELEASE</spring.version>
</properties>

<dependencies>
 <dependency>
       <groupId>org.springframework</groupId>
       <artifactId>spring-jdbc</artifactId>
       <version>${spring.version}</version>
</dependency>
<dependency>
       <groupId>org.springframework.data</groupId>
       <artifactId>spring-data-jpa</artifactId>
       <version>2.3.3.RELEASE</version>
</dependency>
<dependency>
       <groupId>org.postgresql</groupId>
       <artifactId>postgresql</artifactId>
       <version>42.2.15</version>
</dependency>

<dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-core</artifactId>
       <version>5.4.20.Final</version>
</dependency>

<dependency>
<groupId>javax.persistence</groupId>
<artifactId>javax.persistence-api</artifactId>
<version>2.2</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.springframework</groupId>
        <artifactId>spring-context-support</artifactId>
        <version>${spring.version}</version>
</dependency>

<dependency>
        <groupId>org.freemarker</groupId>
        <artifactId>freemarker</artifactId>
        <version>2.3.30</version>
</dependency>

<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
         <version>2.10.2</version>
</dependency>

<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.10.2</version>
</dependency>
</dependencies>
```

1.3 Структура проекта

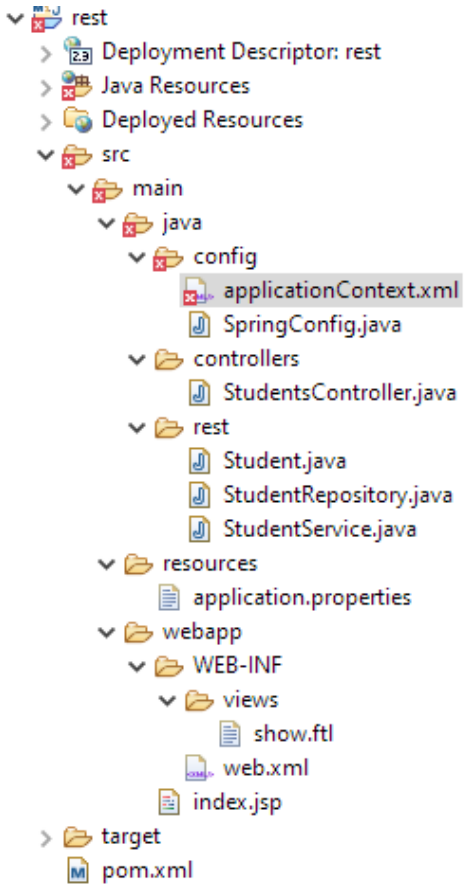

Рисунок 1 — структура проекта rest

1.4 Контроллер StudentsController

```
@Controller
RequestMapping("/students")
public class StudentsController {

@Autowired
private StudentService studentsService;

@RequestMapping(value="/{id}", method=RequestMethod.GET, headers =
{"Accept=application/json"})
public @ResponseBody Student getStudent(@PathVariable("id") int id) {
	return studentsService.getStudentById(id);
}

@RequestMapping(value="/{id}", method=RequestMethod.GET, headers = {"Accept=text/
html"})
public String getStudent(@PathVariable("id") int id, Model model) {
	model.addAttribute(studentsService.getStudentById(id));
	return "show";
}
}
```

1.5 applicationContext.xml конфигурация

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
```

```
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd">

       <mvc:annotation-driven/>

       <context:component-scan base-package="config, controllers"/>

       <bean id="contentNegotiationManager"
class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
         <property name="favorPathExtension" value="true" />
         <property name="ignoreAcceptHeader" value="true"/>
         <property name="useJaf" value="false"/>
         <property name="defaultContentType" value="text/html" />

         <property name="mediaTypes">
             <map>
                <entry key="html" value="text/html"/>
                <entry key="json" value="application/json"/>
            </map>
         </property>
    </bean>

       <bean
class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">
       <property name="contentNegotiationManager" ref="contentNegotiationManager"/>
             <property name="viewResolvers">
          <list>
            <bean
class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver"/>
          </list>
       </property>
       </bean>

       <bean id="freeMarkerConfig"
class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
             <property name="templateLoaderPath" value="/WEB-INF/views"/>
             <property name="defaultEncoding" value="UTF-8"/>
             <property name="freemarkerSettings">
                   <props>
                         <prop key="default_encoding">UTF-8</prop>
                   </props>
             </property>
       </bean>
       <bean id="templateResolver"
class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver">
             <property name="suffix" value=".ftl"/>
             <property name="contentType" value="text/html; charset=utf-8"/>
             <property name="cache" value="false"/>
       </bean>

      </beans>
```

1.6 SpringConfig.java конфигурация

```
@Configuration
@EnableJpaRepositories("rest")
@EnableTransactionManagement
@ComponentScan("rest")
@PropertySource("classpath:application.properties")
public class SpringConfig {

@Autowired
private Environment env;
@Bean
DataSource dataSource() {
	DriverManagerDataSource dataSource = new DriverManagerDataSource();

dataSource.setDriverClassName(env.getProperty("dataSource.driverClassName"));
	dataSource.setUrl(env.getProperty("dataSource.url"));
	dataSource.setUsername(env.getProperty("dataSource.username"));
	dataSource.setPassword(env.getProperty("dataSource.password"));
	return dataSource;
}

@Bean
public EntityManagerFactory entityManagerFactory() {
	HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
	vendorAdapter.setGenerateDdl(true);
	LocalContainerEntityManagerFactoryBean factory = new
	LocalContainerEntityManagerFactoryBean();
	factory.setJpaVendorAdapter(vendorAdapter);
	factory.setPackagesToScan("rest");
	factory.setDataSource(dataSource());
	factory.afterPropertiesSet();
	return factory.getObject();
}
@Bean
public PlatformTransactionManager transactionManager() {
	JpaTransactionManager txManager = new JpaTransactionManager();
	txManager.setEntityManagerFactory(entityManagerFactory());
	return txManager;
}
}
```

1.7 web.xml конфигурация

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >
<	web-app>
<display-name>Archetype Created Web Application</display-name>

 <servlet>
	<servlet-name>dispatcher</servlet-name>
	<servlet-class>org.springframework.web.servlet.
		DispatcherServlet</servlet-class>
	<init-param>
		<param-name>contextConfigLocation</param-name>
		<param-value>
			config/applicationContext.xml
		</param-value>
	</init-param>
```

```
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

1.8 show.ftl страница

```
<#import "/spring.ftl" as spring/>
<meta charset="UTF-8">
<html>
<head>
<title>Отображение студента</title>
</head>
<body>
<div>
   <fieldset>
      <legend>Отображение студента</legend>
      <form name="student" action="">
       <table cellspacing="0">
              <tr>
                    <th><label>Фамилия:</label><th>
                    <td><@spring.formInput "student.lastName" "" "text"/>
                    </td>
              </tr>
              <tr>
                    <th><label>Имя:</label><th>
                    <td><@spring.formInput "student.firstName" "" "text"/>
                    </td>
              </tr>
              <tr>
                    <th><label>Отчество:</label><th>
                    <td><@spring.formInput "student.patronymic" "" "text"/>
                    </td>
              </tr>
              <tr>
                    <th><label>Средний балл:</label><th>
                    <td><@spring.formInput "student.avgMark" "" "text"/>
                    </td>
              </tr>
       </table>
      </form>
   </fieldset>
  </div>
</body>
   </html>
```

1.9 StudentService.java файл

```
@Component
public class StudentService {

@Autowired
StudentRepository rep;

public List<Student> getAll(){
```

```
        return rep.findAll();
    }

    public Student getStudentById (Integer id){
        Optional<Student> result = rep.findById(id);
        if(result.isPresent()){
            return result.get();
        }else{
            return null;
        }
    }
    }
```

1.10 Описание

Приложение представляет собой простейшее REST приложение предоставляющее ресурс Student (код класса Student не представлен в документе так как представляет собой сущность, которая не влияет на приложение, альтернативным может быть любой другой пользовательский класс-сущность).

Класс StudentService представляет собой прослойку между ComputerRepository и приложением.

По адресу /rest/students/{id}.html (обратите внимание на `headers = {"Accept=text/html"}` в аннотации @RequestMapping) будет возвращаться представление show, которое обрабатывается TemplateResolver`ом Freemarker.

По адресу /rest/students/{id} может обращаться любое другое приложение для получения ресурса Student, рассмотрим такое приложение.

2 Создание элементарного REST-клиента

2.1 Создание Maven проекта

Создаем новый Maven проект в IDE Eclipse из архетипа maven-archetype-quickstart

Зададим ему Artifact Id: restclient.

2.2 Зависимости

Добавим в pom.xml следующие зависимости:

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>5.2.8.RELEASE</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.2.8.RELEASE</version>
</dependency>
```

2.3 Структура проекта

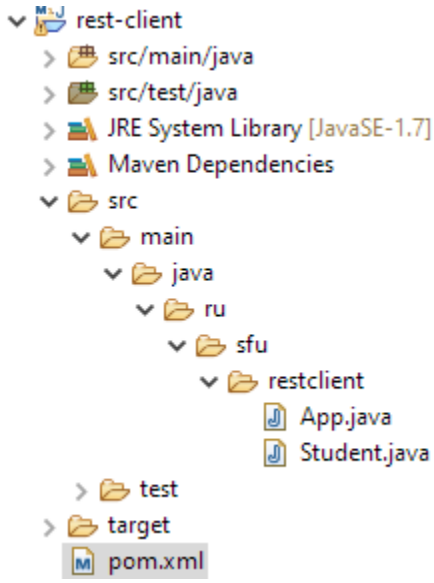

Рисунок 2 — Структура проекта restclient

2.4 App.java

```
public class App
{
	public static void main( String[] args ){
		String ret = retrieveStudent(1);
		System.out.println(ret.toString());
	}

	public static String retrieveStudent(int id) {
		return new RestTemplate().getForObject(
		"http://localhost:8081/rest/students/{id}",
		String.class, id);
	}
}
```

2.5 Описание

Приложение представляет собой элементарный REST-клиент, который отправляет GET-запрос на ресурс `http://localhost:8081/rest/students/{id}` посредством RestTemplate, получает объект и выводит его на консоль.