### JAVA第二阶段—DAY20-JAVA案例

1. 使用JS原生方式实现Ajax

步骤：

创建核心对象

建立连接

发送请求

接受并处理来自服务器的响应结果

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

创建项目day05\_ajax

步骤二：创建html页面

在WebContent目录下创建包index.html

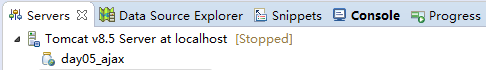
|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>原生js实现ajax</title>  <script type=*"text/javascript"*>  //点击请求按钮执行  **function** fun() {  //1.创建XMLHttpRequest  **var** xmlhttp;  **if** (window.XMLHttpRequest) {// code for IE7+, Firefox, Chrome, Opera, Safari  xmlhttp = **new** XMLHttpRequest();  } **else** {// code for IE6, IE5  xmlhttp = **new** ActiveXObject("Microsoft.XMLHTTP");  }  /\*2使用对象的open(method,url,async)方法建立连接  method：请求的类型；GET 或 POST  url：文件在服务器上的位置  async：true（异步）或 false（同步  \*/  xmlhttp.open("GET", "ajaxServlet?username=wangcai", **true**);  //3.发送请求  xmlhttp.send();  //接收并处理来自服务器的响应结果  xmlhttp.onreadystatechange = **function**() {  **if** (xmlhttp.readyState == 4 && xmlhttp.status == 200) {    //获取服务器的响应数据  **var** data = xmlhttp.responseText;    alert(data);    }  }  }  </script>  </head>  <body>  <input type=*"button"* value=*"发送Ajax请求"* onclick="fun();">  <input type=*"text"* name=*"username"*>  </body>  </html> |

步骤三：创建servlet

在src目录下创建包com.bailiban.web.servlet ,定义过滤器AjaxServlet

|  |
| --- |
| **package** com.bailiban.web.servlet;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.annotation.WebServlet;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  @WebServlet("/ajaxServlet")  **public** **class** AjaxServlet **extends** HttpServlet {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;      **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {    //获取请求参数  String username = request.getParameter("username");  //让服务器睡5秒  **try** {  Thread.*sleep*(5000);  } **catch** (InterruptedException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  System.***out***.println(username);  //响应数据  response.getWriter().write("hi!"+username);    }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  // **TODO** Auto-generated method stub  doGet(request, response);  }  } |

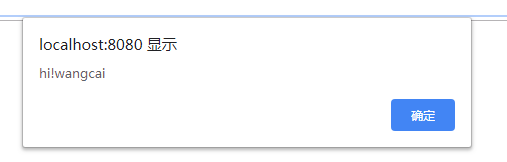
步骤四：将项目部署到Tomcat,启动服务器



步骤五：浏览器访问

在浏览器地址输入：http://localhost:8080/day05\_ajax/index.html

查看效果



1. 使用Jquery方式实现Ajax

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

基于项目day05\_ajax

步骤二：创建html页面

在WebContent目录下创建包index\_ajax.html、index\_ajax\_get.html、index\_ajax\_post.html

index\_ajax.html页面

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Jquery实现Ajax</title>  <script type=*"text/javascript"* src=*"js/jquery1.12.4.min.js"*></script>  <script type=*"text/javascript"*>  //点击请求按钮执行  **function** fun() {    //使用$.ajax()发送异步请求  $.ajax({  url:"ajaxServlet" , // 请求路径  type:"POST" , //请求方式  //data: "username=jack&age=23",//请求参数  data:{"username":"wangcai\_ajax","age":18},  success:**function** (data) {  alert(data);  },    dataType:"text"//设置接受到的响应数据的格式  });  }  </script>  </head>  <body>  <input type=*"button"* value=*"发送Ajax请求"* onclick="fun();">  <input type=*"text"* name=*"username"*>  </body>  </html> |

index\_ajax\_get.html页面

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Jquery实现ajax\_get</title>  <script type=*"text/javascript"* src=*"js/jquery1.12.4.min.js"*></script>  <script type=*"text/javascript"*>  //点击请求按钮执行  **function** fun() {      //使用$.get()发送ajax请求  $.get("ajaxServlet",{"username":"wangcai\_get","age":18},**function** (data) {  alert(data);  },"text");  }  </script>  </head>  <body>  <input type=*"button"* value=*"发送Ajax请求"* onclick="fun();">  <input type=*"text"* name=*"username"*>  </body>  </html> |

index\_ajax\_post.html页面

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>Jquery实现Ajax\_post</title>  <script type=*"text/javascript"* src=*"js/jquery1.12.4.min.js"*></script>  <script type=*"text/javascript"*>  //点击请求按钮执行  **function** fun() {  //使用$.post()发送异步请求  $.post("ajaxServlet",{"username":"wangcai\_post"},**function** (data) {  alert(data);  },"text");  }  </script>  </head>  <body>  <input type=*"button"* value=*"发送Ajax请求"* onclick="fun();">  <input type=*"text"* name=*"username"*>  </body>  </html> |

步骤三：浏览器访问

<http://localhost:8080/day05_ajax/index_ajax.html>

[http://localhost:8080/day05\_ajax/index\_ajax\_get.html](http://localhost:8080/day05_ajax/index_ajax.html)

[http://localhost:8080/day05\_ajax/index\_ajax\_post.html](http://localhost:8080/day05_ajax/index_ajax.html)

1. 演示JSON的语法：定义格式，数据获取

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

基于项目day05\_ajax

步骤二：创建html页面

在WebContent目录下创建包json数据语法.html、json数据语法\_遍历.html文件

json数据语法.html

|  |
| --- |
| <!DOCTYPE html>  <html lang=*"en"*>  <head>  <meta charset=*"UTF-8"*>  <title>Title</title>  <script>  //1.基本的JSON对象  **var** person = {"name": "如花", age: 18, 'sex': **true**};  //获取name的值  //var name = person.name;  **var** name = person["name"];  // alert(name);    // 2.嵌套 JSON 对象  **var** addr = {"address": {"province":"湖北","city":"武汉"}};  **var** city = addr.address.city;  // alert(city);    //3.JSON值为数组 {}———> []  **var** persons = {  "persons": [  {"name": "如花", "age": 18, "sex": **true**},  {"name": "旺财", "age": 19, "sex": **true**},  {"name": "小强", "age": 20, "sex": **false**}  ]  };  // alert(persons);  //获取小强  **var** name1 = persons.persons[2].name;  // alert(name1);  //4.JSON对象数组 []———> {}  **var** ps = [{"name": "如花", "age": 18, "sex": **true**},  {"name": "旺财", "age": 19, "sex": **true**},  {"name": "小强", "age": 20, "sex": **false**}];  //获取旺财  //alert(ps);  alert(ps[1].name);  </script>  </head>  <body>  </body>  </html> |

json数据语法\_遍历.html

|  |
| --- |
| <!DOCTYPE html>  <html lang=*"en"*>  <head>  <meta charset=*"UTF-8"*>  <title>Title</title>  <script>  //遍历JSON  **var** p ={"name": "如花", "age": 18, "sex": **true**};  **for**(**var** key **in** p){    alert("键:"+key+"-"+"值："+p[key]);  }    //遍历 json对象数组  **var** ps = [{"name": "如花", "age": 18, "sex": **true**},  {"name": "旺财", "age": 19, "sex": **true**},  {"name": "小强", "age": 20, "sex": **false**}];  //获取person对象中所有的键和值  //for in 循环  /\*for(var i in person){  //i就是数组中元素的索引  alert(i+":"+person[i]);  }\*/  //获取ps中的所有值  **for** (**var** i = 0; i < ps.length; i++) {  **var** p = ps[i];  **for**(**var** key **in** p){  alert(key+":"+p[key]);  }  }  </script>  </head>  <body>  </body>  </html> |

步骤三：访问页面测试效果

1. 演示 Java对象转换JSON相互转换

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

基于项目day05\_ajax

步骤二：导依赖Jar包

导入依赖jar包



步骤三：编写Junit测试类

在src目录下创建包com.bailiban.test ,编写JacksonTest测试类

|  |
| --- |
| **package** com.bailiban.test;  **import** java.io.File;  **import** java.io.FileWriter;  **import** java.util.ArrayList;  **import** java.util.Date;  **import** java.util.HashMap;  **import** java.util.List;  **import** java.util.Map;  **import** org.junit.Test;  **import** com.bailiban.domain.Person;  **import** com.fasterxml.jackson.databind.ObjectMapper;  **public** **class** JacksonTest {  // Java对象转为JSON字符串  @Test  **public** **void** test1() **throws** Exception {  // 创建Person对象  Person p = **new** Person();  p.setName("如花");  p.setAge(18);  p.setSex("女");  // 创建Jackson的核心对象 ObjectMapper  ObjectMapper mapper = **new** ObjectMapper();  // 转换  // writeValueAsString(obj):将对象转为json字符串  String json = mapper.writeValueAsString(p);  System.***out***.println(json);// {"name":"如花","age":18,"sex":"女"}  // writeValue，将数据写到a.txt文件中  mapper.writeValue(**new** File("a.txt"), p);  // writeValue.将数据关联到Writer中  mapper.writeValue(**new** FileWriter("b.txt"), p);  }  //测试注解  @Test  **public** **void** test2() **throws** Exception {  // 创建Person对象  Person p = **new** Person();  p.setName("如花");  p.setAge(18);  p.setSex("女");  p.setBirthday(**new** Date());  // 转换  ObjectMapper mapper = **new** ObjectMapper();  String json = mapper.writeValueAsString(p);  System.***out***.println(json);// {"name":"张三","age":23,"gender":"男","birthday":1530958029263}  // {"name":"如花","age":18,"sex":"女","birthday":"2020-07-12"}  }  //测试转换list集合  @Test  **public** **void** test3() **throws** Exception {  //创建Person对象  Person p1 = **new** Person();  p1.setName("如花");  p1.setAge(18);  p1.setSex("女");  p1.setBirthday(**new** Date());  Person p2 = **new** Person();  p2.setName("小强");  p2.setAge(18);  p2.setSex("男");  p2.setBirthday(**new** Date());  Person p3 = **new** Person();  p3.setName("旺财");  p3.setAge(28);  p3.setSex("男");  p3.setBirthday(**new** Date());  //创建List集合  List<Person> ps = **new** ArrayList<Person>();  ps.add(p1);  ps.add(p2);  ps.add(p3);  //2.转换  ObjectMapper mapper = **new** ObjectMapper();  String json = mapper.writeValueAsString(ps);  // [{},{},{}]  //[{"name":"如花","age":18,"sex":"女","birthday":"2020-07-12"},{"name":"小强","age":18,"sex":"男","birthday":"2020-07-12"},{"name":"旺财","age":28,"sex":"男","birthday":"2020-07-12"}]  System.***out***.println(json);  }  //测试转换map集合  @Test  **public** **void** test4() **throws** Exception {  //1.创建map对象  Map<String,Object> map = **new** HashMap<String,Object>();  map.put("name","张三");  map.put("age",23);  map.put("gender","男");  //2.转换  ObjectMapper mapper = **new** ObjectMapper();  String json = mapper.writeValueAsString(map);  //{"name":"张三","age":23,"gender":"男"}  System.***out***.println(json);//{"gender":"男","name":"张三","age":23}  }  //演示 JSON字符串转为Java对象  @Test  **public** **void** test5() **throws** Exception {  //初始化JSON字符串  String json = "{\"sex\":\"男\",\"name\":\"如花\",\"age\":23}";  //创建ObjectMapper对象  ObjectMapper mapper = **new** ObjectMapper();  //转换为Java对象 Person对象  Person person = mapper.readValue(json, Person.**class**);  System.***out***.println(person);  }  } |

步骤四：运行测试方法

1. 添加用户校验用户名是否存在

需求：

1.添加时输入用户名，当失去焦点 Ajax异步验证 用户名是否已经存在

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

基于day05\_ajax项目

步骤二：创建实体类 User

创建包com.bailiban.domain,在该包下创建User类

|  |
| --- |
| **package** com.bailiban.domain;  **public** **class** User {  **private** **int** id; //用户编号  **private** String username;//用户名  **private** String password; //密码      **public** User() {  **super**();  }    **public** User(String username, String password) {  **super**();  **this**.username = username;  **this**.password = password;  }  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getUsername() {  **return** username;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** String getPassword() {  **return** password;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }      } |

在包com.bailiban.servlet中创建LoginServlet,设置访问路径为/login

步骤三：编写工具类JDBCUtils

创建包com.bailiban.utils,在该包下创建JDBCUtils类

|  |
| --- |
| **package** com.bailiban.utils;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.sql.Connection;  **import** java.sql.SQLException;  **import** java.util.Properties;  **import** javax.sql.DataSource;  **import** com.alibaba.druid.pool.DruidDataSourceFactory;  /\*\*  \* JDBC工具类 使用Durid连接池  \*/  **public** **class** JDBCUtils {  **private** **static** DataSource *ds*;  // 静态代码块  **static** {  **try** {  // 1.加载配置文件  Properties pro = **new** Properties();  // 使用ClassLoader加载配置文件，获取字节输入流  InputStream is = JDBCUtils.**class**.getClassLoader().getResourceAsStream("db.properties");  pro.load(is);  // 2.初始化连接池对象  *ds* = DruidDataSourceFactory.*createDataSource*(pro);  } **catch** (IOException e) {  e.printStackTrace();  } **catch** (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 获取连接池对象  \*/  **public** **static** DataSource getDataSource() {  **return** *ds*;  }  } |

步骤四：编写dao层

创建包com.bailiban.dao,在该包下创建UserDao接口

|  |
| --- |
| **package** com.bailiban.dao;  **import** com.bailiban.domain.User;  **public** **interface** UserDao {  /\*\*  根据用户名查询用户  \*/  User findUserByUserName(String username);  } |

创建包com.bailiban.dao.impl,在该包下创建UserDaoImpl实现类

|  |
| --- |
| **package** com.bailiban.dao.impl;  **import** org.springframework.dao.DataAccessException;  **import** org.springframework.jdbc.core.BeanPropertyRowMapper;  **import** org.springframework.jdbc.core.JdbcTemplate;  **import** com.bailiban.dao.UserDao;  **import** com.bailiban.domain.User;  **import** com.bailiban.utils.JDBCUtils;  /\*\*  \* 操作数据库中User表的类  \*/  **public** **class** UserDaoImpl **implements** UserDao {  //声明JDBCTemplate对象共用  **private** JdbcTemplate template = **new** JdbcTemplate(JDBCUtils.*getDataSource*());  @Override  **public** User findUserByUserName(String username) {  //定义Sql  String sql ="select \* from t\_user where username = ? ";  //执行查询  User user;  **try** {  user = template.queryForObject(sql, **new** BeanPropertyRowMapper<User>(User.**class**),username);  **return** user;  } **catch** (DataAccessException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  **return** **null**;  }    }    } |

步骤五：编写service层

创建包com.bailiban.service,在该包下创建UserService接口

|  |
| --- |
| **package** com.bailiban.service;  **import** com.bailiban.domain.User;  **public** **interface** UserService {  /\*\*  根据用户名查询用户  \*/  User findUserByUserName(String username);  } |

创建包com.bailiban.service.impl,在该包下创建UserServiceImpl实现类

|  |
| --- |
| **package** com.bailiban.service.impl;  **import** com.bailiban.dao.UserDao;  **import** com.bailiban.dao.impl.UserDaoImpl;  **import** com.bailiban.domain.User;  **import** com.bailiban.service.UserService;  **public** **class** UserServiceImpl **implements** UserService {    //依赖dao  **private** UserDao userDao = **new** UserDaoImpl();  /\*\*  根据用户名查询用户  \*/  @Override  **public** User findUserByUserName(String username) {  **return** userDao.findUserByUserName(username);  }  } |

步骤六：创建CheckUserServlet编写CheckUserServlet中的代码

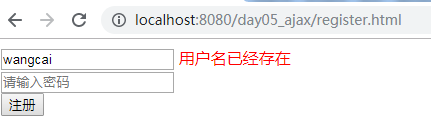
在包com.bailiban.web.servlet中创建CheckUserServlet

|  |
| --- |
| **package** com.bailiban.web.servlet;  **import** com.bailiban.domain.User;  **import** com.bailiban.service.UserService;  **import** com.bailiban.service.impl.UserServiceImpl;  **import** com.fasterxml.jackson.databind.ObjectMapper;  **import** javax.servlet.ServletException;  **import** javax.servlet.annotation.WebServlet;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **import** java.io.IOException;  **import** java.util.HashMap;  **import** java.util.Map;  @WebServlet("/checkUser")  **public** **class** CheckUserServlet **extends** HttpServlet {    **private** **static** **final** **long** ***serialVersionUID*** = 1L;  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  //1.获取用户名  String username = request.getParameter("username");  //2.调用service层判断用户名是否存在  UserService userService = **new** UserServiceImpl();  User user = userService.findUserByUserName(username);    //设置响应的数据格式为json  response.setContentType("application/json;charset=utf-8");  Map<String,Object> map = **new** HashMap<String,Object>();  **if**(user!=**null**){  //存在  map.put("isExsit",**true**);  map.put("msg","用户名已经存在");  }**else**{  //不存在  map.put("isExsit",**false**);  map.put("msg","用户名可用");  }  //将map转为json，并且传递给客户端  //将map转为json  ObjectMapper mapper = **new** ObjectMapper();  //并且传递给客户端  mapper.writeValue(response.getWriter(),map);  }  **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  **this**.doPost(request, response);  }  } |

步骤七：创建register.html页面发送Ajax异步请求

|  |
| --- |
| <!DOCTYPE html>  <html lang=*"en"*>  <head>  <meta charset=*"UTF-8"*>  <title>注册页面</title>  <script src=*"js/jquery1.12.4.min.js"*></script>  <script>  //在页面加载完成后  $(**function** () {  //给username绑定blur事件  $("#username").blur(**function** () {  //获取username文本输入框的值  **var** username = $(**this**).val();  //发送ajax请求  $.get("checkUser",{username:username},**function** (data) {  **var** span = $("#s\_username");  **if**(data.isExsit){  //用户名存在  span.css("color","red");  span.html(data.msg);  }**else**{  //用户名不存在  span.css("color","green");  span.html(data.msg);  }  });  });  });  </script>  </head>  <body>  <form>  <input type=*"text"* id=*"username"* name=*"username"* placeholder=*"请输入用户名"*>  <span id=*"s\_username"*></span>  <br>  <input type=*"password"* name=*"password"* placeholder=*"请输入密码"*><br>  <input type=*"submit"* value=*"注册"*><br>  </form>  </body>  </html> |

步骤八：测试





1. 完善图书管理系统的图书添加，添加图书时异步验证图书名不能重复

* 参考答案

实现此案例需要按照如下步骤进行。

步骤一：创建JavaEE项目

复制day04\_book项目 取名 day05\_book

导入JSON转换相关jar包



步骤二：编写dao层

|  |
| --- |
| **package** com.bailiban.dao;  **import** java.util.List;  **import** com.bailiban.domain.Book;  **public** **interface** BookDao {      //根据书名查询书籍  Book findBookByName(String name);  } |

|  |
| --- |
| **package** com.bailiban.dao.impl;  **import** java.util.List;  **import** org.springframework.dao.DataAccessException;  **import** org.springframework.jdbc.core.BeanPropertyRowMapper;  **import** org.springframework.jdbc.core.JdbcTemplate;  **import** com.bailiban.dao.BookDao;  **import** com.bailiban.domain.Book;  **import** com.bailiban.utils.JDBCUtils;  **public** **class** BookDaoImpl **implements** BookDao {  //声明JDBCTemplate对象共用  **private** JdbcTemplate template = **new** JdbcTemplate(JDBCUtils.*getDataSource*());  //根据书名查询书籍  @Override  **public** Book findBookByName(String name) {  //定义Sql  String sql ="select \* from t\_book where name = ? ";  //执行查询  Book book = **null**;  **try** {  book = template.queryForObject(sql, **new** BeanPropertyRowMapper<Book>(Book.**class**),name);  } **catch** (DataAccessException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();    }  **return** book;  }    } |

步骤三：编写service层

|  |
| --- |
| **package** com.bailiban.service;  **import** java.util.List;  **import** com.bailiban.domain.Book;  **public** **interface** BookService {  // 根据书名查询书籍  Book findBookByName(String name);  } |

|  |
| --- |
| **package** com.bailiban.service.impl;  **import** java.util.List;  **import** com.bailiban.dao.BookDao;  **import** com.bailiban.dao.impl.BookDaoImpl;  **import** com.bailiban.domain.Book;  **import** com.bailiban.service.BookService;  **public** **class** BookServiceImpl **implements** BookService {  // 创建dao  **private** BookDao bookDao = **new** BookDaoImpl();    // 根据书名查询书籍  @Override  **public** Book findBookByName(String name) {    **return** bookDao.findBookByName(name);  }  } |

步骤四：创建CheckBookServlet编写CheckBookServlet中的代码

在包com.bailiban.servlet中创建CheckBookServlet

|  |
| --- |
| **package** com.bailiban.servlet;  **import** java.io.IOException;  **import** java.util.HashMap;  **import** java.util.Map;  **import** javax.servlet.ServletException;  **import** javax.servlet.annotation.WebServlet;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **import** com.bailiban.domain.Book;  **import** com.bailiban.service.BookService;  **import** com.bailiban.service.impl.BookServiceImpl;  **import** com.fasterxml.jackson.databind.ObjectMapper;  /\*\*  \* Servlet implementation class CheckBookServlet  \*/  @WebServlet("/checkBook")  **public** **class** CheckBookServlet **extends** HttpServlet {  **private** **static** **final** **long** ***serialVersionUID*** = 1L;      **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  //1.获取书名  String name = request.getParameter("name");  //2.调用service层判断书名是否存在  BookService BookService = **new** BookServiceImpl();  Book book = BookService.findBookByName(name);    //设置响应的数据格式为json  response.setContentType("application/json;charset=utf-8");  Map<String,Object> map = **new** HashMap<String,Object>();  **if**(book!=**null**){  //存在  map.put("isExsit",**true**);  map.put("msg","\*书名已经存在");  }**else**{  //不存在  map.put("isExsit",**false**);  map.put("msg","\*书名可用");  }  //将map转为json，并且传递给客户端  //将map转为json  ObjectMapper mapper = **new** ObjectMapper();  //并且传递给客户端  mapper.writeValue(response.getWriter(),map);  }  **protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  // **TODO** Auto-generated method stub  doGet(request, response);  }  } |

步骤五：修改book-add.jsp页面发送Ajax异步请求

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*  pageEncoding=*"UTF-8"*%>  <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>图书添加</title>  <!--导入框架样式文件-->  <link rel=*"stylesheet"* href=*"*${pageContext.request.contextPath}*/css/bootstrap.min.css"* />  <script type=*"text/javascript"* src=*"*${pageContext.request.contextPath}*/js/jquery1.12.4.min.js"*></script>  <script type=*"text/javascript"* src=*"*${pageContext.request.contextPath}*/js/bootstrap.min.js"*></script>  <script>  //在页面加载完成后  $(**function** () {  //给name绑定blur事件  $("#name").blur(**function** () {  //获取name文本输入框的值  **var** name = $(**this**).val();  //发送ajax请求  $.get("checkBook",{"name":name},**function** (data) {  **var** span = $("#s\_name");  **if**(data.isExsit){  //用图书名存在  span.css("color","red");  span.html(data.msg);  }**else**{  //图书名不存在  span.css("color","green");  span.html(data.msg);  }  });  });  });  </script>  </head>  <body>  <div class=*"container"*>  <div class=*"panel panel-default"*>  <div class=*"panel-heading"*>  <h3 class=*"panel-title"*>图书添加</h3>  </div>  <div class=*"panel-body"*>  <form class=*"form-horizontal"* enctype=*"multipart/form-data"* method=*"post"* action=*"*${pageContext.request.contextPath}*/book/add"*>  <div class=*"form-group"*>  <label for=*"name"* class=*"col-sm-2 control-label"*>图书名称</label>  <div class=*"col-sm-10"*>  <input type=*"text"* class=*"form-control"* id=*"name"* name=*"name"* placeholder=*"请输入图书名称"* required >  <span id=*"s\_name"*></span>  </div>  </div>  <div class=*"form-group"*>  <label for=*"inputEmail3"* class=*"col-sm-2 control-label"*>图书价格</label>  <div class=*"col-sm-10"*>  <input type=*"text"* class=*"form-control"* name=*"price"* placeholder=*"请输入图书价格"* required >  </div>  </div>  <div class=*"form-group"*>  <label for=*"inputEmail3"* class=*"col-sm-2 control-label"*>图书作者</label>  <div class=*"col-sm-10"*>  <input type=*"text"* class=*"form-control"* name=*"author"* placeholder=*"请输入图书作者"* required >  </div>  </div>  <div class=*"form-group"*>  <label for=*"inputEmail3"* class=*"col-sm-2 control-label"*>出版日期</label>  <div class=*"col-sm-10"*>  <input type=*"date"* class=*"form-control"* name=*"publishDate"* placeholder=*"请输入出版日期"* required>  </div>  </div>  <div class=*"form-group"*>  <label for=*"inputEmail3"* class=*"col-sm-2 control-label"*>图书图片</label>  <div class=*"col-sm-10"*>  <input type=*"file"* class=*"form-control"* name=*"image"* placeholder=*"请输入图书图片"* required>  </div>  </div>    <div class=*"form-group"*>  <div class=*"col-sm-offset-2 col-sm-10"*>  <button type=*"submit"* class=*"btn btn-default"*>添加</button>  </div>  </div>  </form>  </div>  </div>  </div>  </body>  </html> |

步骤六：测试

