## 静态资源文件的引用（mvc:resources）

web.xml中配置的DispatcherServlet请求映射为”/”，则Spring MVC将捕获Web容器所有的请求，也包括静态资源请求，由于找不到对用处理器，所以引用的静态资源无法访问。

### WebRoot目录下创建/statics/js目录放js文件

### 将jquery的js文件放到js目录下

### index.jsp引入jquery的js

|  |
| --- |
| <script type="text/javascript"  src="${pageContext.request.contextPath}/statics/js/jquery-2.1.4.min.js">  </script> |

### 配置<mvc:resources>

mapping: 将静态资源映射到指定路径下

location: 本地静态资源文件所在的目录

springmvc-servlet.xml

|  |
| --- |
| <mvc:resources mapping="/statics/\*\*" location="/statics/"></mvc:resources> |

## 异常处理（局部异常处理）

局部异常处理，仅能指定Controller中的异常。使用@ExceptionHandler注解实现

### HelloController

|  |
| --- |
| package cn.bdqn.controller;  import java.util.Map;  import javax.servlet.http.HttpServletRequest;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.ExceptionHandler;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import cn.bdqn.entity.User;  @Controller  @RequestMapping("welcome")  public class HelloController {    @RequestMapping(value="hello",method=RequestMethod.GET)  public String hello(User user,Map<String,Object> map){    System.out.println("HelloController类的hello方法");  System.out.println("username="+user.getUsername());  System.out.println("password="+user.getPassword());  if(!"admin".equals(user.getUsername())){  throw new RuntimeException("用户名错误");  }  map.put("username", user.getUsername());  return "welcome";  }  @ExceptionHandler(value={RuntimeException.class})  public String errorHandler(RuntimeException e,HttpServletRequest request){  request.setAttribute("e", e);  return "error";  }  } |

@ExceptionHandler可以指定多个异常。

指定当控制器发送RuntimeException异常时，使用error视图进行异常信息显示。

### error.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>Title</title> </head> <body>  ${e.message} </body> </html> |

## 异常处理（全局异常处理）

全局异常处理可使用SimpleMappingExceptionResolver来实现

### HelloController

|  |
| --- |
| package cn.bdqn.controller;  import java.util.Map;  import javax.servlet.http.HttpServletRequest;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.ExceptionHandler;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import cn.bdqn.entity.User;  @Controller  @RequestMapping("welcome")  public class HelloController {    @RequestMapping(value="hello",method=RequestMethod.GET)  public String hello(User user,Map<String,Object> map){    System.out.println("HelloController类的hello方法");  System.out.println("username="+user.getUsername());  System.out.println("password="+user.getPassword());  if(!"admin".equals(user.getUsername())){  throw new RuntimeException("用户名错误");  }  map.put("username", user.getUsername());  return "welcome";  }  } |

去掉了handlerexception方法

### springmvc-servlet.xml

|  |
| --- |
| <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">  <property name="exceptionMappings">  <props>  <prop key="java.lang.RuntimeException">error</prop>  </props>  </property>  </bean> |

### error.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %> <html> <head>  <title>Title</title> </head> <body>  ${exception} </body> </html> |

## Ajax（复习）

使用JavaScript异步发送请求，不用每次请求都重新加载页面，发送请求时可以继续其他操作，在服务器响应完后，在使用JavaScript将响应展示给用户。

## JSON（复习）

### fastjson(复习)

### 5.2.引入处理JSON数据工具的jar文件（使用fastjson-1.2.13.jar）

## 在处理方法上加@ResponseBody注解

一般情况下，@ResponseBody都会在异步获取数据时使用，被其标注的处理方法返回的数据将输出到响应流中，客户端获取并显示数据。

### HelloController

|  |
| --- |
| package cn.bdqn.controller;  import java.util.HashMap;  import java.util.Map;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.ResponseBody;  import com.alibaba.fastjson.JSON;  import cn.bdqn.entity.User;  @Controller  @RequestMapping("welcome")  public class HelloController {  @ResponseBody  @RequestMapping(value="hello",method=RequestMethod.POST,  produces={"application/json;charset=UTF-8"})  public String hello(User user){  Map<String,String> map=new HashMap<String,String>();  System.out.println("HelloController类的hello方法");  if("admin".equals(user.getUsername())){  map.put("message", "登录成");  }else{  map.put("message", "登录失败");  }  return JSON.toJSONString(map);  }  } |

### index.jsp

|  |
| --- |
| <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <base href="<%=basePath%>">    <title>My JSP 'index.jsp' starting page</title>  <meta http-equiv="pragma" content="no-cache">  <meta http-equiv="cache-control" content="no-cache">  <meta http-equiv="expires" content="0">  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  <meta http-equiv="description" content="This is my page">  <!--  <link rel="stylesheet" type="text/css" href="styles.css">  -->  <script  type="text/javascript"  src="${pageContext.request.contextPath }/statics/js/jquery-2.1.4.min.js"></script></head>  <script type="text/javascript">  $(function(){  $("#button").click(function(){  var username=$("#username").val();  $.post("${pageContext.request.contextPath}/welcome/hello",{'username':username},function(data){    alert(data.message);  })  })  })  </script>  </head>  <body>  <form action="" method="get">  用户名：<input type="text" name="username" id="username"/>  密码：<input type="password" name="password">  <input type="button" value="提交" id="button">  </form>  </body>  </html> |