# 实验十 SpringMVC数据绑定及拦截器应用

## 实验目的

1. 掌握SpringMVC的数据绑定
2. 掌握拦截器的使用

## 实验内容

通过客户端请求input.jsp页面提交学生信息，内容包括学生学号、姓名、性别、出生日期、成绩（包括计算机导论、C语言程序设计、Java程序设计三门成绩）数据提交后在show.jsp页面可以查看，如果数据没有提交，show.jsp不会显示数据，会跳转到input.jsp页面提示输入数据。

## 实验代码：

Web.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  
 version="4.0">  
 <servlet>  
 <servlet-name>DispatcherServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-mvc.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>DispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>  
   
 <filter>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
 <init-param>  
 <param-name>encoding</param-name>  
 <param-value>UTF-8</param-value>  
 </init-param>  
 <init-param>  
 <param-name>forceEncoding</param-name>  
 <param-value>true</param-value>  
 </init-param>  
 </filter>  
 <filter-mapping>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
</web-app>

spring-mvc.xml

*<?*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:jdbc="http://www.springframework.org/schema/jdbc"  
 xmlns:jee="http://www.springframework.org/schema/jee"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xmlns:util="http://www.springframework.org/schema/util"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.2.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd  
 http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-3.2.xsd  
 http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.2.xsd  
 http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd  
 http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.2.xsd  
 http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd  
 http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-3.2.xsd">  
 <context:component-scan base-package="com.shf.controller"/>  
 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/jsp/"/>  
 <property name="suffix" value=".jsp"/>  
 </bean>  
  
 <bean id="converterService" class="org.springframework.context.support.ConversionServiceFactoryBean">  
 <property name="converters">  
 <array>  
 <bean class="com.shf.Converter.DateConverter"/>  
 </array>  
 </property>  
 </bean>  
 <mvc:annotation-driven conversion-service="converterService"/>  
  
 *<!-- 配置拦截器,其中可以配置多个拦截器 -->* <mvc:interceptors>  
  
 <mvc:interceptor>  
  
 *<!-- 配置拦截器的拦截路径，拦截/user下的全部处理器方法映射  
 比如：http://localhost:8080/Springmvc/user/login.do这个请求就会被拦截  
 -->* <mvc:mapping path="/\*\*"/>  
  
 *<!-- 配置不被该拦截器拦截器的controller方法,这个是可选配置  
 比如：http://localhost:8080/Springmvc/user/index.do将不会被拦截器  
 -->* <mvc:exclude-mapping path="/input"/>  
  
 *<!-- 配置拦截器的bean，指定的是全类名 -->* <bean class="com.shf.Interceptor.SubmitInterceptor"></bean>  
 </mvc:interceptor>  
  
 </mvc:interceptors>  
</beans>

Input.jsp

*<%--  
 Created by IntelliJ IDEA.  
 User: shuho  
 Date: 2022/5/16  
 Time: 11:00  
 To change this template use File | Settings | File Templates.  
--%>*<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
<h1>  
  
 <%=request.getParameter("errmsg")==null?"":request.getParameter("errmsg")%></h1>  
 <form action="/submit" method="post">  
 学号：<input type="text" name="sno"> <br>  
 姓名：<input type="text" name="sname"> <br>  
 性别：<input type="radio" value="1" name="sex"> 男  
 <input type="radio" value="0" name="sex"> 女 <br>  
 出生日期： <input type="date" name="birthday"> <br>  
 成绩（计算机导论）： <input type="text" name="score[0]"> <br>  
 成绩（C语言程序设计）： <input type="text" name="score[1]"> <br>  
 成绩（Java程序设计）： <input type="text" name="score[2]"> <br>  
 <input type="submit" value="提交">  
 </form>  
</body>  
</html>

Show.jsp

*<%--  
 Created by IntelliJ IDEA.  
 User: shuho  
 Date: 2022/5/16  
 Time: 11:00  
 To change this template use File | Settings | File Templates.  
--%>*<%@ page contentType="text/html;charset=UTF-8" language="java" %>  
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>  
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
 <h1>提交成功</h1>  
 <h2>学号: ${requestScope.sno }</h2>  
 <h2>姓名: ${requestScope.sname }</h2>  
 <h2>性别: ${requestScope.sex==1?"男":"女" }</h2>  
 <h2>出生日期:<fmt:formatDate value="${requestScope.birthday}" pattern="yyyy-MM-dd"/></h2>  
  
 <h2>计算机导论成绩: ${requestScope.score[0] }</h2>  
 <h2>C语言程序设计成绩: ${requestScope.score[1] }</h2>  
 <h2>Java程序设计论成绩: ${requestScope.score[2] }</h2>  
</body>  
</html>

Student.java

package com.shf.pojo;  
  
import lombok.Data;  
  
import java.util.Date;  
import java.util.List;  
  
@Data  
public class Student {  
 private String sno;  
 private String sname;  
 private Integer sex;  
 private Date birthday;  
 private List<Double> score;  
}

CheckParametersUtil.java

package com.shf.utils;  
  
import java.util.Collection;  
import java.util.HashMap;  
import java.util.Map;  
  
public class CheckParametersUtil {  
  
 Map<String, Object> map = new HashMap<>();  
  
 */\*\*  
 \* 添加需要校验的参数  
 \* @param object 实参  
 \* @param parameterName 参数名称  
 \* @return CheckParametersUtil  
 \* @author: XuWei  
 \*/* public CheckParametersUtil put(Object object, String parameterName) {  
 map.put(parameterName, object);  
 return this;  
 }  
 */\*\*  
 \* 获取CheckParametersUtil实例  
 \* @return CheckParametersUtil  
 \* @author: XuWei  
 \*/* public static CheckParametersUtil getInstance(){  
 return new CheckParametersUtil();  
 }  
  
 */\*\*  
 \* 校验  
 \* @return DataMessage  
 \* @author: XuWei  
 \* @throws Exception  
 \*/* public void checkParameter() throws Exception {  
 for(Map.Entry<String, Object> entry : map.entrySet()) {  
 if(isEmptyTrim(entry.getValue())){  
 throw new Exception("param [" + entry.getKey() + "] can't null" );  
 }  
 }  
 }  
  
 public String toString(Object object) {  
 return object == null ? "" : object.toString();  
 }  
  
 public boolean isEmpty(Collection collection) {  
 return collection == null || collection.isEmpty();  
 }  
  
 public boolean isEmpty(Map map) {  
 return map == null || map.isEmpty();  
 }  
  
 public boolean isEmpty(String string) {  
 return toString(string).isEmpty();  
 }  
  
 public boolean isEmptyTrim(String string) {  
 return toString(string).trim().isEmpty();  
 }  
  
 public boolean isEmpty(Object object) {  
 return toString(object).isEmpty();  
 }  
  
 public boolean isEmptyTrim(Object object) {  
 return toString(object).trim().isEmpty();  
 }  
  
 public <T> boolean isEmpty(T[] array) {  
 return array == null || array.length == 0;  
 }  
}

DateConverter.java

package com.shf.Converter;  
  
  
import org.springframework.core.convert.converter.Converter;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
public class DateConverter implements Converter<String, Date> {  
 private String datePattern = "yyyy-MM-dd";  
  
 @Override  
 public Date convert(String score) {  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat(datePattern);  
 try {  
 return simpleDateFormat.parse(score);  
 } catch (ParseException e) {  
 return null;  
 }  
 }  
}

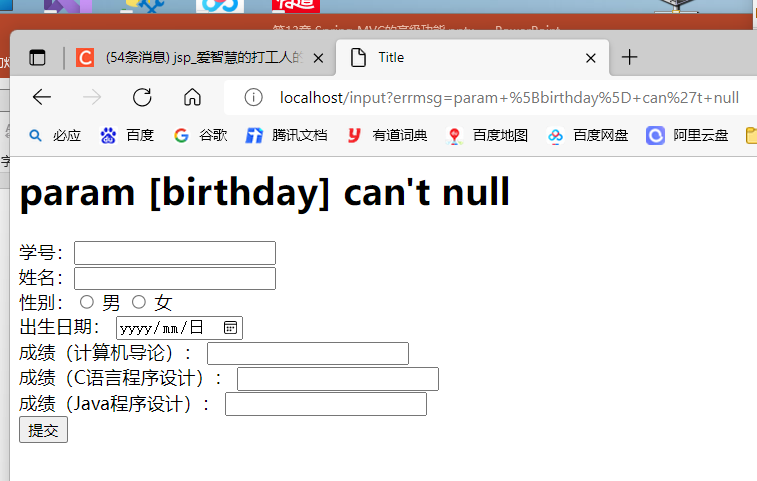
SubmitInterceptor.java

package com.shf.Interceptor;  
  
import org.springframework.web.servlet.HandlerInterceptor;  
import org.springframework.web.servlet.ModelAndView;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import javax.servlet.http.HttpSession;  
import java.util.Enumeration;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class SubmitInterceptor implements HandlerInterceptor {  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
 request.setCharacterEncoding("UTF-8");  
 response.setCharacterEncoding("UTF-8");  
 response.setContentType("text/html;charset=UTF-8");  
  
 if(request.getRequestURI()=="/submit"&&request.getMethod()=="GET"){  
 if (request.getSession().getAttribute("isSubmit")=="ok"){  
 return true;  
 } else {  
 response.sendRedirect("/input");  
 return false;  
 }  
 }  
 return true;  
 }  
  
 @Override  
 public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {  
 HandlerInterceptor.super.postHandle(request, response, handler, modelAndView);  
 }  
  
 @Override  
 public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {  
 HandlerInterceptor.super.afterCompletion(request, response, handler, ex);  
 }  
}

SubmitController.java

package com.shf.controller;  
  
import com.shf.pojo.Student;  
import com.shf.utils.CheckParametersUtil;  
import org.springframework.cglib.beans.BeanMap;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
import javax.servlet.http.HttpSession;  
import java.util.Arrays;  
  
@Controller  
public class SubmitController {  
 @RequestMapping("/")  
 public String toInput(){  
 return "redirect:/input";  
 }  
  
 @RequestMapping("/input")  
 public String input(HttpSession session){  
 session.removeAttribute("isSubmit");  
 return "input";  
 }  
  
 @RequestMapping("/submit")  
 public String submit(Model model, Student student, HttpSession session){  
 System.*out*.println(student);  
 try {  
 CheckParametersUtil.*getInstance*()  
 .put(student.getSno(), "sno")  
 .put(student.getSname(), "sname")  
 .put(student.getSex(), "sex")  
 .put(student.getBirthday(), "birthday")  
 .put(student.getScore(), "score")  
 .checkParameter();  
 } catch (Exception e) {  
 model.addAttribute("errmsg",e.getMessage());  
 return "redirect:/input";  
 }  
  
 model.addAllAttributes(BeanMap.*create*(student));  
 session.setAttribute("isSubmit", "ok");  
 return "show";  
 }  
}

## 运行截图





# 实验小结

通过本次实验我学会了在springMVC中复杂数据的绑定绑定方法，包括数组绑定、集合绑定、复制POJO绑定及JSON数据绑定；并且掌握页面跳转的方式，包括返回值为void类型的页面跳转、返回值为String类型的页面跳转及返回值为ModelAndView类型的页面跳转。