springmvc

1:j简单的入门案例

springmvc.xml

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



<context:component-scan base-package="com.baidu">

</context:component-scan>

<bean id="internalResourceViewResolver" class="
org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix" value="WEB-INF/pages/"></property>
<property name="suffix" value=".jsp"></property>
</bean>

</beans>
```

pom,xml

```

  <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>5.0.2.RELEASE</spring.version>
  </properties>
  <dependencies>
    <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>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>

    <dependency>
      <groupId>javax.servlet.jsp</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.0</version>
      <scope>provided</scope>
    </dependency>

  </dependencies>
```

web,.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.5"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

<servlet >
  <servlet-name> dispartservlet</servlet-name>
  <servlet-class> org.springframework.web.servlet.DispatcherServlet</servlet-
class>
```

```
  <init-param>

  <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
  <servlet-name>dispartservlet</servlet-name>
  <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

controller

```
package com.baidu;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.DispatcherServlet;

@Controller
public class AccountController {

  @RequestMapping("/hello")
  public String a(){

    return "success";
  }
}
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>

hello
<a href="/testspringmvc2/hello">dsd</a>
```

```
</body>
</html>
```

success.jsp沈略

流程:

- 启动Tomcat容器会加载dispartservlet同时加了一个initparam,将springmvc.xml加载了
- springmvc配置了注解扫描
- 扫描controller,将其配置成一个bean
-

@Requestmapping注解介绍

- 参数:
-   String[] params() : 必须传一个属性,否则不能请求到
- RequestMethod[] method()枚举类型: 值为GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE

解决乱码过滤器

```
<filter-mapping>
  <filter-name>characterEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
  </filter-mapping>
```

参数绑定集合

```
 <form action="/testspringmvc2/hello" method="post">
      绑定list集合
```

```
        密码：<input type="text" name="list[0].password" /><br/>
<!--         金额：<input type="text" name="money" /><br/> -->
        用户姓名：<input type="text" name="list[0].username" /><br/>

    绑定map集合 </br>
  密码：<input type="text" name="map['one'].username" /><br/>
<!--         金额：<input type="text" name="money" /><br/> -->
        用户姓名：<input type="text" name="map['one'].password" /><br/>

        <input type="submit" value="提交" />
    </form>
```

Account有list和map

```java
package com.baidu.entity;

import java.util.List;
import java.util.Map;

public class Account {
  String username;

  String password;

  List<User>list;
  Map<String, User>map;
  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;
  }
  public List<User> getList() {
    return list;
  }
  public void setList(List<User> list) {
    this.list = list;
  }
  public Map<String, User> getMap() {
    return map;
  }
  public void setMap(Map<String, User> map) {
    this.map = map;
  }
  @Override
  public String toString() {
    return "Account [username=" + username + ", password=" + password + ",
list=" + list + ", map=" + map + "]";
  }



}
```

自定义转换器

spring提供了ConversionServiceFactoryBean进行类型转换,

只需要在xml中配置即可,

```
<!--配置转换器-->

<bean id="converservice"
class="org.springframework.context.support.ConversionServiceFactoryBean">
  <!--给这个转换器注入一个set类型的属性-->
<property name="converters">
<set>
  <!--自定义的转换器-->
<bean class="com.baidu.entity.Myconvert"></bean>
</set>

</property>

</bean>

<!-- 开启SpringMVC框架注解的支持 -->
    <mvc:annotation-driven conversion-service="converservice"/>
```

```
package com.baidu.entity;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import org.springframework.core.convert.converter.Converter;

public class Myconvert implements Converter<String, Date>{
  public Date convert(String source) {
    // TODO Auto-generated method stub
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
    try {
      Date date = sdf.parse(source);
      return date;
    } catch (ParseException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return null;
  }

}
```

常用注解：

@requerstparam：用于controller和jsp中参数名称不一致的情况，requerstparam里面，加上参数的名称

@requerstbody：用于获取post请求方式的请求体，get没有请求体

@PathVariable 一般用于Restful风格，取参数

@modelattribute

浏览器一般只支持get和post请求，也可以装插件发put，delete请求，

如果没有装插件，怎么让浏览器发送put，delete请求呢，springmvc使用

HiddentHttpMethodFilter这个过滤器可以实现（了解就行）

响应结果重定向转发

```
@RequestMapping("/testforwardorredict")
  public String jj(HttpServletResponse res){
    //使用重定向或转发将不会经过视图解析器
    //使用重定向

    return "redirect:/index.jsp";

    //使用转发
    //return "forward:/WEB-INF/pages/success.jsp";
  }
```

复习下ajax

```
window.onload=function(){

  $.ajax(

  {
    url:"/testspringmvc2/hello",
    type:post,

    data:{username:"zhangsan",userage:"18" },
    success:function(data){
    }
  }
  );
}
```

文件上传

- 前提
  - 请求方式必须是post
  - form表单的enctype 必须是multipart/form-data
  - 选取文件 未选择文件

异常解析器

- 创建自定义异常

```
package com.baidu.utils;

public class MyExeception extends Exception{


private String message;

public String getMessage() {
  return message;
}

public void setMessage(String message) {
  this.message = message;
}

public MyExeception(String message) {
  // TODO Auto-generated constructor stub
  this.message=message;
}

}
```

- 创一个实现HandlerExceptionResolver的类

```
package com.baidu.utils;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerExceptionResolver;
import org.springframework.web.servlet.ModelAndView;

public class HanderExeception implements HandlerExceptionResolver{

  public ModelAndView resolveException(HttpServletRequest request,
HttpServletResponse response, Object handler,
      Exception ex) {
    // TODO Auto-generated method stub
    Exception e=null;

    if(ex instanceof MyExeception){
      e=new MyExeception("系统正在维护");

    }
    ModelAndView mv=new ModelAndView();

    mv.setViewName("error");
    mv.addObject("errormsg",e.getMessage());
    return mv;
  }

}
```

- springmvc.xml中添加

```
<bean class="com.baidu.utils.HanderExeception" id="handerExeception">
</bean>
```

- 在controller制作一个异常

```
@Controller
public class Acountcontroller {
  @RequestMapping("/hello")
  public String d() throws MyExeception{
    try {
      int i=1/0;

    } catch (Exception e) {
      // TODO: handle exception

      throw new MyExeception("系统正在维护");
    }

    return "success";
  }

```

-

原理:默认不处理，异常会一层层抛，最后抛到浏览器显示，如果在异常抛到dispartservlet时，用一个异常处理器，处理抛出来的异常，一旦出现了异常，前端控制器会指定异常处理器，如果我们定义了一个异常处理器实现了HandlerExceptionResolver并注册成了一个bean，异常处理器就会被调用然后跳转到指定页面。

拦截器

拦截器和过滤器的区别：

- 拦截器是springmvc才有的，filter是所有javaee项目都有的
- springmvc只能拦截controller的资源，filter加/*可以过滤所有资源

使用方法

- 创建一个拦截器类，实现HandlerInterceptor.

```
package com.baidu.utils;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerInterceptor;
```

```
public class Myhanderinteceptor implements HandlerInterceptor{

  //访问controller前的操作
  public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler)
      throws Exception {
    // TODO Auto-generated method stub
    System.out.println("拦截器执行了...1");
    return true;
  }


}
```

- 配置springmvc.xml

```
<mvc:interceptors>

<mvc:interceptor>
<mvc:mapping path="/user/*"/>
<bean class="com.baidu.utils.Myhanderinteceptor"></bean>

</mvc:interceptor>
</mvc:interceptors>
```

  -