工作上用的比较多，所以开始学SpringMVC时，到不觉得陌生，不过另一方面SpringMVC本身也不难。比较意外的是，我原本以为SpringMVC是一个单独的框架需要重新找页面下载，结果它就集成在spring framework里。

那么省略下载的过程直接写个hello world项目吧。

于是又喜闻乐见的一丝头绪都没有... ...

Spring是用来整合的框架，获取各种各样乱七八糟的东西，并给他们加统一的玩意。Mybatis & Hibernate是持久层框架，连接数据库，增删查改。那SpringMVC呢？是web框架，用于接受request，返回response，是Servlet的进化版（当然本质还是Servlet）。这么一来问题简单不少，如何写一个SpringMVC的hello world？请问如何写一个Servlet的hello world？

* 在web.xml中配置<servlet>与<servlet-mapping>标签
* 写XXXServlet
* 写XXX.jsp
* 启动Tomcat

SpringMVC也没什么特别的步骤，就是多加了个xml。按顺序

web.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"*

xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee*

*http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"*

id=*"WebApp\_ID"* version=*"3.1"*>

<display-name>framework</display-name>

<welcome-file-list>

<welcome-file>/admin/hello</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:mvc.xml</param-value> 配置文件所在地

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

* 配置文件那个参数是必须的，不写的话SpringMVC会自动去ServletContext那要，没有

就shutdown。

* classpath: & classpath\*:

classpath: 就表示去类路径下找（就我个人而言，包路径贴切一点）

\* 就是通配符，表示会去多个类路径下寻找。好高端诶，我还没遇到过。

* “/”是我觉得最恶心的，只能写“/”，“/\*”是错的。。。

另外，“/”代表所有请求，因此图片，.js文件，.css文件这种没有指定分发的请求都会失败，我不禁想起了我写的第一个Filter，也是把所有请求都过滤了。

<mvc:resources mapping=*"/resources/\*\*"* location=*"/images/"*/>

<mvc:resources mapping=*"/resources2/\*\*"* location=*"/css/"*/>

以上便是解决方案，也就是说，URL中 */resources/\*\** 这个部分会被定向到*/images/*

SpringMVC.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"*

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"*>

<context:component-scan base-package=*"com.jojo"* />

<mvc:annotation-driven/>

<!-- 视图解析器 -->这个东西可以将Controller里return的东西解析成jsp

<bean id=*"viewResolver"*

class=*"org.springframework.web.servlet.view.UrlBasedViewResolver"*>

<property name=*"viewClass"*

value=*"org.springframework.web.servlet.view.JstlView"* />

<property name=*"prefix"* value=*"/WEB-INF/view/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

</beans>

SpringMVC运行至少需要两个部分，**视图解析器**与**注解扫描**，再少就要出事了。

本来是想和applicationContext.xml放在一起的，但出问题了，怎么说吧，就算可以放一起，也不建议这么做，管理起来太麻烦。

Controller

**package** com.jojo.controller;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

@Controller

@RequestMapping(value="/admin")

**public** **class** HelloController {

@RequestMapping(value="/hello", method=RequestMethod.***GET***)

**public** String hello(){

**return** "helloWorld";

}

}

老相识了，@Controller 的作用就是将该类变成 Spring 中的一个 <bean> 标签。至于另一个。。。解释起来好麻烦，总之就相当于<url-pattern>不过他是怎么实现的呢？

jsp不贴了，贴了也没意义，访问的话，靠这个url

localhost:8080/framework/admin/hello

前面那个framework要怎样才可以不写呢？BookCombined项目是怎么做到的。

这个流程简单梳理一下是这样的，根据配置，所有的url都会先访问org.springframework.web.servlet.DispatcherServlet这个Servlet，然后根据注解中写明的url分发给具体的Controller处理（也有可能是xml，但我这里写的是注解），Controller处理完后再转发给某某jsp。注解与转发之所以会起作用全靠springMVC的配置文件，注解扫描器与视图解析器。背下来就好，原理有空自己去研究。

***ModelAndView***

一次都没用过，是真的一次都没用过，我全用Model的。姑且写一下。

**public** ModelAndView list(){

ModelAndView mav = **new** ModelAndView();

mav.addObject("XXX", "XXX");

mav.setViewName("hello");

**return** mav;

}

怎么看怎么别扭，第一印象这东西果然重要啊。我不用它的另一个理由是Model也完全够我用了。

**return** "redirect:/hello";

mav.setViewName("redirect:/hello");

这种写法可以直接重定向，forward同理。

***编码***

<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>

</filter>

<filter-mapping>

<filter-name>characterEncodingFilter</filter-name>

<url-pattern>/\*</url-pattern> 他奶奶的，你现在倒是能用 /\* 了

</filter-mapping>

解决中文环境下post提交的乱码问题，唉，get提交也有乱码呀！

总之得贴上去，不能少。

***<form:form>***

自动封装对象（这个功能不用该标签，springMVC也支持）

自动生成id与name

自动foreach

***RESTful分格URL***

// 坦白讲我也算不上很懂

这是一种很优雅的URL格式，同样是将参数带在URL上，此风格没有“？param=xxx”这种难看的后缀 <https://www.zhihu.com/collection/19928423> 诺，像那个19928423就是参数。

使用起来大概这种感觉：

@RequestMapping(value = "detail/{id}", method = RequestMethod.***GET***)

**public** String details(@PathVariable(name="id") Integer id, Model model){

// **TODO**

model.addAttribute("XXX", "XXX");

**return** "some url";

}

要注意这个注解啊@PathVariable

***文件上传***

要准备Apache Commons fileupload以及io插件（管这两个东西叫插件到底合不合适呢），我看的教程里准备的jar包是这两个

com.springsource.org.apache.commons.fileupload-1.2.0.jar

com.springsource.org.apache.commons.io-1.4.0.jar

看名字像是Spring封装过的，不过我直接用的Commons，也没报错，想必是SpringMVC在升级过程中解决了这个不兼容的问题吧，毕竟谁高兴管理那么多jar包。

mvc.xml里新增一个<bean>

<!-- 解析上传文件 -->

<bean id=*"multipartResolver"*

class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<property name=*"defaultEncoding"* value=*"UTF-8"*/>

<property name=*"maxUploadSize"* value=*"10000000"*/>

</bean>

jsp里就一个form，method还有那个enctype千万别忘了。

<form action=*"*${pageContext.request.contextPath}*/admin/upload/singleFile"*

method=*"post"* enctype=*"multipart/form-data"*>

<input type=*"file"* name=*"file"*/>

<input type=*"submit"*>

</form>

Controller，注意形参的类型，MultipartFile

@Controller

@RequestMapping("/admin/upload")

**public** **class** UploadController {

@RequestMapping(value = "singleFile", method = RequestMethod.***POST***)

**public** String singleFile(@RequestParam(value="file", required=**true**)MultipartFile file

, HttpServletRequest request) **throws** Exception{

String path = request.getServletContext().getRealPath("");

System.***out***.println(path);

// file.transferTo(new File("D:\\Workspace\\2.txt"));

**return** "success";

}

}

这样一来就可以获取被上传的文件。

***注解***

* @RequestParam

用在形参前，括号内属性可以有多个，用逗号隔开。

value = “abc” 写完这个后，使用该注解的参数，前端必须传名为abc的参数。

required = true/false 必须要有,或是可以没有，自动填null，0什么的。

* @ModelAttribute

结合<form:form>使用，可以从表单生成一个对象

前端与后端的数据一定要保持一致，比如说age这个字段是int型，那么就只能填数字。

* @RequestMapping

value = “abc” 这个代表访问的URL

method = RequestMethod.GET 这个代表HTTP请求方法，有好多个

* @ResponseBody

有了他可以自动将对象转成json格式，但小峰说这玩意不灵活，因为当model类中涉及到嵌套时，这玩意有可能给不出想要的格式。而且这玩意也不是拿来就能用的，得做点准备。

贴上jackson的jar包

springMVC.xml里加上 <mvc:annotation-driven/> 。

### [@RequestBody](http://blog.csdn.net/sbangm/article/details/43022377)

### [@RequestHeader](http://blog.csdn.net/sbangm/article/details/43022377)

### @CookieValue

* @SessionAttributes
* @ModelAttribute