2016/6/15

SpringMVC

1. 一.Spring MVC 优点

Spring MVC 和其他的Web框架一样，基于MVC设计理念

采用松散耦合可拔插组件结构，比其他MVC框架更具有哦灵活性和扩展性。

Spring MVC还支持REST风格的URL请求：注解驱动以及REST风格的SpringMVC

是Spring3.0最出彩的功能之一，此外Spring MVC在数据绑定、视图解析、本地化

以及静态资源处理都有许多不俗的表现。它在框架的设计上已经超越了Struts、webwork

等框架。

1. 二.工作原理

a.DispathcerServlet是其核心，相当于Spring MVC中的大管家。在web.xml中配置后，拦截所有的HTTP请求

b.DispatcherServlet接受到请求后将其交给HandlerMapping处理，相当于路由，HandlerMapping将请求映射到相应的Handler可以看成是目标主机

c.DispatcherServlet收到HandlerMapping返回的Handler后，通过HandlerAdapter对Hanlder进行封装，然后返回ModeAndView包含了视图逻辑名和模型数据

d.DispatcherSerlvet收到ModeAndView后将其交给ViewResolver解析，将逻辑视图名得到正真的视图对象View

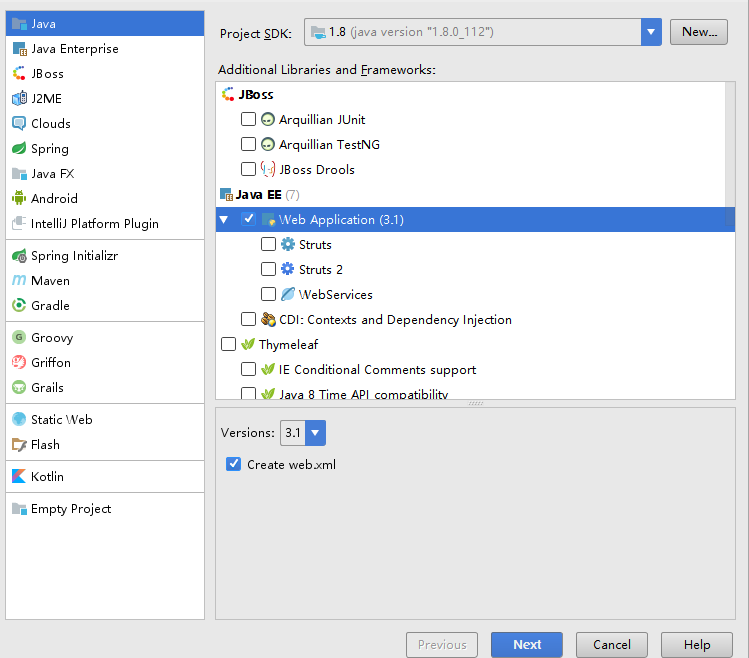
c.DispatcherServelt得到真正视图对象View后，就使用View和ModeAndView中的模型数据进行视图渲染。

d.最终展现给客户端--》图片、pdf文档、html页面、xml、json数据等

三.web工程配置

3.1 新建在IDEA中创建JavaEE Web工程

File->new Project->Java->JavaEE->Web Application-->工程名/工程保存位置

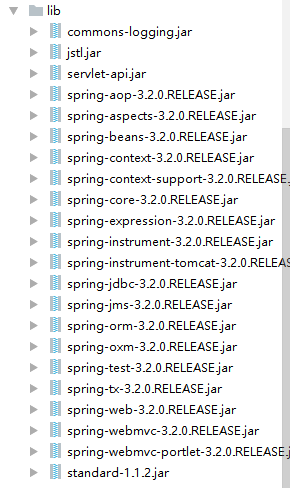


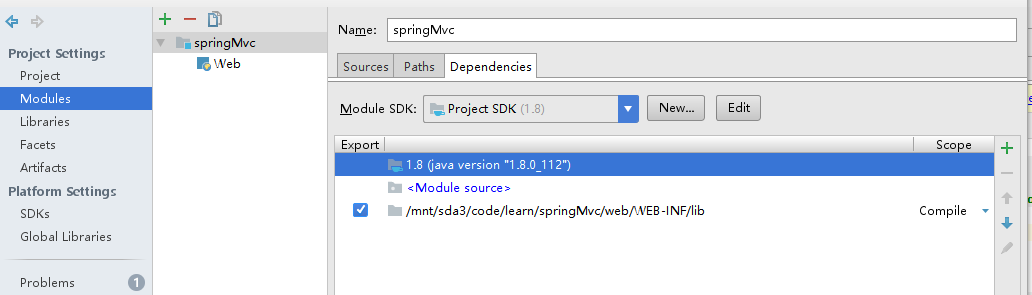
3.2 导入SpringMVC中需要的jar包

Project Structure->Project Settings->Moules->Add(jar包的路径)

这里添加jar的方式跟Eclipse有点不同，但是思想都是一样的。

我这里将jar添加到WEB-INF的lib下边（lib目录不存在需要手动建立）





3.3HelloWorld快速入门

**a.****web.xml springmvc默认读取/WEB-INF/<servlet-name>-servlet.xml**

<servlet>

<servlet-name>springmvc</servlet-name>

<!-- springMVC的入口，分发器，管家 ,分发器默认读取/WEB-INF/<servlet-name>-servlet.xml文件-->

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

<!-- tomcat启动的时候，springmvc也初始化 -->

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern><!-- /表示拦截所有请求，也可以 \*.do \*.html 等 -->

</servlet-mapping>



**b.将配置文件布置到其他路径下 如放置到src/configs下**

<servlet>

<servlet-name>springmvc</servlet-name>

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

<!-- 读取指定目录下的配置文件，名字可以改变 \*表示加载该目录下的所有文件，不加\*也可以-->

<init-param>

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

<param-value>classpath\*:configs/springmvc-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern><!-- /表示拦截所有请求，也可以 \*.do \*.html 等 -->

</servlet-mapping>

**b.springmvc-servlet.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"**

**xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"**>

*<!--配置bean-->*

<**bean name="/test/helloworld" class="com.learn.controller.HelloWorldController"**></**bean**>

*<!--配置视图解析器，springMVC需要在配置文件这中配置试图解析器才正常解析视图-->*

<**bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>

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

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

</**bean**>

</**beans**>

4.spring MultiActionController 实现在一个Controller中写多个方法

步骤1：写一个Controller类extends MultiActionController

步骤2：自定义方法参数为 HttpServletRequest request,HttpServletResponse response

public ModelAndView add(HttpServletRequest request,HttpServletResponse response){

System.out.println("add");

return new ModelAndView("multi","method","add");

}

步骤3：在配置文件中配置参数方法解析器

<!-- 配置参数方法解析器 action为方法参数名

http://localhost:8888/springmvc2/test1/multi?action=update

-->

<bean id="paramMethodResolver" class="org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver">

<property name="paramName" value="action"></property>

</bean>

步骤4：配置自定义Controller类

<!-- 使用参数方法解析器， 引入方法参数解析器，将请求参数映射到具体的方法

http://localhost:8888/springmvc2/test1/multi?action=add

-->

<bean name="/test1/multi" class="com.mcao.web.controller.MultiController">

<property name="methodNameResolver">

<ref bean="paramMethodResolver"/>

</property>

</bean>

步骤5：启动web服务器，访问http://localhost:8888/springmvc2/test1/multi?action=add

action的值就是请求方法

5.spring mvc对静态资源的访问

由于我们在web.xml中配置了如下，会拦截所有的请求，对于静态资源的话，浏览器是会重新发送一次请求，也会被拦截

我们有没有写Controller处理这些静态资源的请求，所以我们需要在springmvc-serlvet.xml中添加

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern><!-- /表示拦截所有请求，也可以 \*.do \*.html 等 这样的话就只会拦截到.do 或\*.html请求 -->

</servlet-mapping>

在springmvc-serlvet.xml中添加，我们就可以顺利访问到我们的静态资源

<!-- 对静态资源的访问 声明在/imgs/\*\* 的所有资源都不要拦截-->

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

html部分：

<body>

<h2>访问图片</h2>

<div>

< img alt="图片未找到" src="../imgs/05.jpg">

</div>

</body>

6.spring注解配置

springmvc-annotation-servlet.xml

3.0之前

<!-- 注解扫描包 ,扫描该包下的所有Controller-->

<context:component-scan base-package="com.mcao.web.controller.annotation" />

<!-- 负责根据url找方法 -->

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"></bean>

根据扫描包和url找类

<bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"></bean>

<!-- 配置视图解析器 -->

<bean id="viewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/jsps/"></property><!-- 前缀/代码项目跟目录 -->

<property name="suffix" value=".jsp"></property><!--后缀 -->

</bean>

3.0之后

<!--开启注解 3.0之后-->

<mvc:annotation-driven/>

扫描包下边的类的注解配置

步骤1：定义一个普通的类 配置注解@Controller //使用注解标示为一个Controller

步骤2：配置请求映射@RequestMapping("/user2") //配置该类的url的路经

步骤3：方法的返回值为逻辑视图名或者ModeAndView

@Controller //使用注解标示为一个Controller

@RequestMapping("/user2") //配置该类的url的路经

public class UserController2 {

@RequestMapping("/addUser") //value的值为url地址

public ModelAndView addUser(){

String result = "this is addUser----优化版";

return new ModelAndView("annotation","result",result);

}

//一般不是涉及一些安全性的东西不需要配置请求方式，就是GET,POST都可以

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

@RequestMapping("/delUser")

public ModelAndView deleteUser(){

String result = "this is deleteUser----优化版";

return new ModelAndView("annotation","result",result);

}

//直接返回路经

@RequestMapping("/toUser")

public String toUer(HttpServletRequest request){

String result = "this is toUser----优化版";

request.setAttribute("result", result);

return "annotation";

}

}

7.springmvc参数传递

1.通过前台页面的属性名与后台方法中的方法参数保持一样就可以将参数值注入进来

如：前台页面

<form action="/springmvc6/user/data/addUser">

姓名：<input type="text" name="userName">

年龄：<input type="text" name="age">

<input type="button" value="提交" onclick="addUser()">

</form>

方法

@RequestMapping("/addUser") //value的值为url地址

public String addUser(String userName,String age,HttpServletRequest request){ //参数名与前台页面一直，springmvc会将值注入

request.setAttribute("userName", userName);

request.setAttribute("age", age);

return "userManager";

}

2.通过实体类，注入，实体类中提供gett.. sett