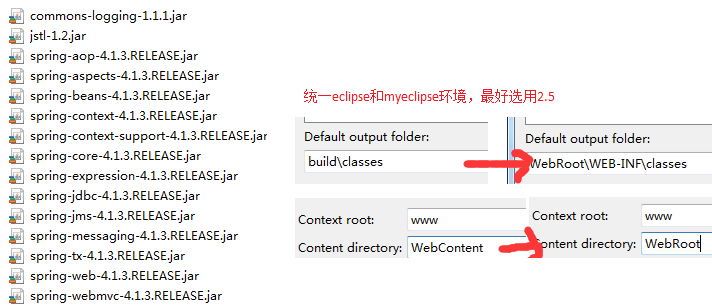
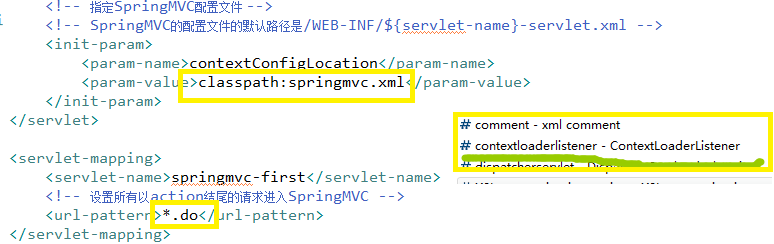
# 入门程序之helloworld—全注解

## 加入 jar 包



## web.xml--前端控制器DispatcherServlet



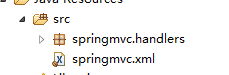
<!--

可以配置/ ，此工程 所有请求全部由springmvc解析，此种方式可以实现 RESTful方式，需要特殊处理对静态文件的解析不能由springmvc解析

可以配置\*.do或\*.action，所有请求的url扩展名为.do或.action由springmvc解析，此种方法常用

-->

### DispatcherServlet



<!--配置DispatcherServlet -->

<!-- The front controller of this Spring Web application, responsible for handling all application requests -->

<servlet>

<servlet-name>springDispatcherServlet</servlet-name>

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

<!-- 配置DispatcherServlet 一个初始化参数：配置SpringMVC配置文件的名称和位置 -->

<init-param>

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

<param-value>classpath:springmvc.xml</param-value>

</init-param>

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

</servlet>

<!-- Map all requests to the DispatcherServlet for handling -->

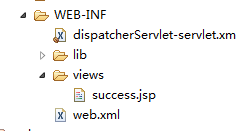
<servlet-mapping>

<servlet-name>springDispatcherServlet</servlet-name>

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

</servlet-mapping>

### 使用默认路径



<!--配置DispatcherServlet -->

<!-- The front controller of this Spring Web application, responsible for handling all application requests -->

<servlet>

<servlet-name>dispatcherServlet</servlet-name>

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

<!--

实际上也可以不通过 contextConfigLocation 来配置 SpringMVC 的配置文件, 而使用默认的.

默认的配置文件为: /WEB-INF/<servlet-name>-servlet.xml

-->

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

</servlet>

<!-- Map all requests to the DispatcherServlet for handling -->

<servlet-mapping>

<servlet-name>dispatcherServlet</servlet-name>

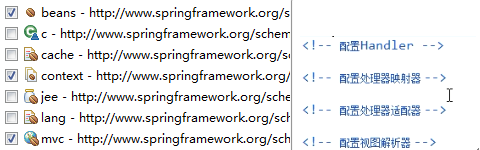
<url-pattern>/</url-pattern>

</servlet-mapping>

## springmvc.xml--加入 Spring MVC 的配置文件

### 引入约束

在springmvc.xml中配置springmvc架构三大组件（处理器映射器、适配器、视图解析器）



### 配置自动扫描的包

<!-- 配置自定义扫描包 -->

<context:component-scan base-package=*"springmvc"*></context:component-scan>

### 配置视图解析器

视图名称解析器：将视图逻辑名解析为: /WEB-INF/pages/<viewName>.jsp

<!-- 配置视图解析器：如何把handler 方法返回值解析为实际的物理视图 -->

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

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

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

</bean>

@Controller

**public** **class** HelloWorld {

/\*\*

\* 1. 使用 **@RequestMapping** 注解来映射请求的 URL

\* 2. 返回值会通过视图解析器解析为实际的物理视图, 对于 InternalResourceViewResolver 视图解析器, 会做如下的解析:

\* 通过 prefix + returnVal + 后缀 这样的方式得到实际的物理视图, 然会做转发操作

\* /WEB-INF/views/success.jsp

\*/

@RequestMapping("/helloworld")

**public** String hello(){

System.*out*.println("hello world");

**return** "success";

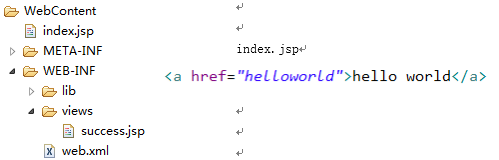
}

}

@RequestMapping("/helloworld"):对应jsp页面action的地址

"success"根据返回值返回success.jsp

## 编写视图



# Web.xml

## 解决post乱码问题

提交发现，保存成功，但是保存的是乱码

在web.xml中加入：

<!-- 解决post乱码问题 -->

<filter>

<filter-name>encoding</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<!-- 设置编码参是UTF8 -->

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encoding</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

以上可以解决post请求乱码问题。

对于get请求中文参数出现乱码解决方法有两个：

修改tomcat配置文件添加编码与工程编码一致，如下：

<Connector URIEncoding="utf-8" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

另外一种方法对参数进行重新编码：

String userName new

String(request.getParamter("userName").getBytes("ISO8859-1"),"utf-8")

ISO8859-1是tomcat默认编码，需要将tomcat编码后的内容按utf-8编码

# Springmvc.xml.

## 组件扫描器

使用组件扫描器省去在spring容器配置每个Controller类的繁琐。

使用<context:component-scan>自动扫描标记@Controller的控制器类，

在springmvc.xml配置文件中配置如下：

<!-- 配置controller扫描包，多个包之间用,分隔 -->

<context:component-scan base-package=*"cn.itcast.springmvc.controller"* />

## 注解驱动

### 注解处理器映射器

注解式处理器映射器，对类中标记了@ResquestMapping的方法进行映射。根据@ResquestMapping定义的url匹配@ResquestMapping标记的方法，匹配成功返回HandlerMethod对象给前端控制器。

HandlerMethod对象中封装url对应的方法Method。

从spring3.1版本开始，废除了DefaultAnnotationHandlerMapping的使用，推荐使用RequestMappingHandlerMapping完成注解式处理器映射。

在springmvc.xml配置文件中配置如下：

<!-- 配置处理器映射器 -->

<bean

class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"* />

注解描述：

@RequestMapping：定义请求url到处理器功能方法的映射

### 注解处理器适配器

注解式处理器适配器，对标记@ResquestMapping的方法进行适配。

从spring3.1版本开始，废除了AnnotationMethodHandlerAdapter的使用，推荐使用RequestMappingHandlerAdapter完成注解式处理器适配。

在springmvc.xml配置文件中配置如下：

<!-- 配置处理器适配器 -->

<bean

class=*"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"* />

### 注解驱动

直接配置处理器映射器和处理器适配器比较麻烦，可以使用注解驱动来加载。

SpringMVC使用<mvc:annotation-driven>自动加载RequestMappingHandlerMapping和RequestMappingHandlerAdapter

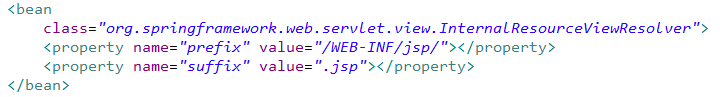
可以在springmvc.xml配置文件中使用<mvc:annotation-driven>替代注解处理器和适配器的配置。

<!-- 注解驱动 -->

<mvc:annotation-driven />

## 视图解析器

### jsp



3种返回方式



返回void，原生



### ftl

<**bean id="viewResolver" class="org.springframework.web.servlet.view.freemarker.FreeMarkerViewResolver"**>  
 <**property name="cache" value="true"** />  
 <**property name="prefix" value=""** />  
 <**property name="suffix" value=".ftl"** />  
 <**property name="viewClass" value="com.kun.framework.web.web.FreeMarkerView"** />  
</**bean**>

# 注意

## Springmvc.xml不配置视图解析器

Springmvc不配置视图解析器，由原生controller控制

