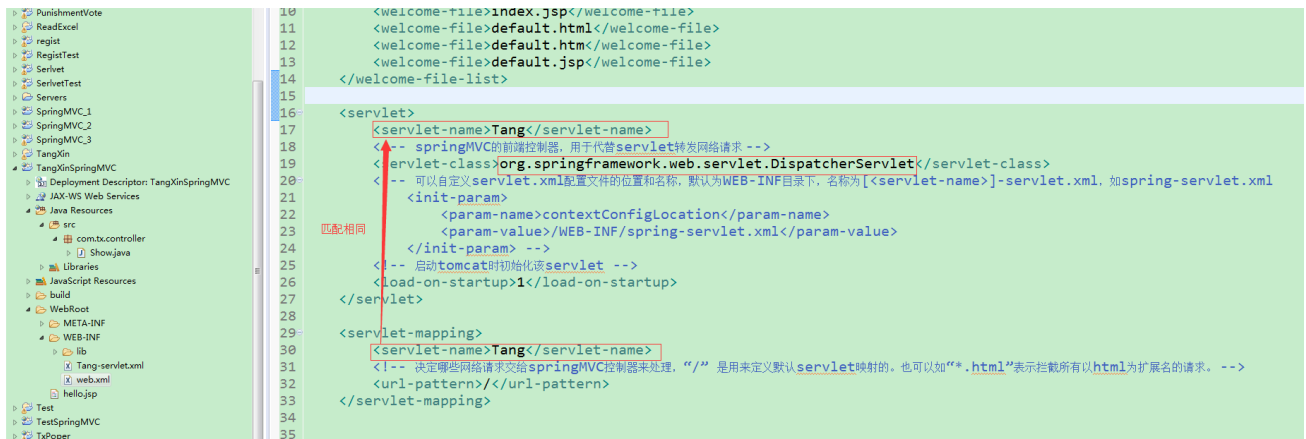
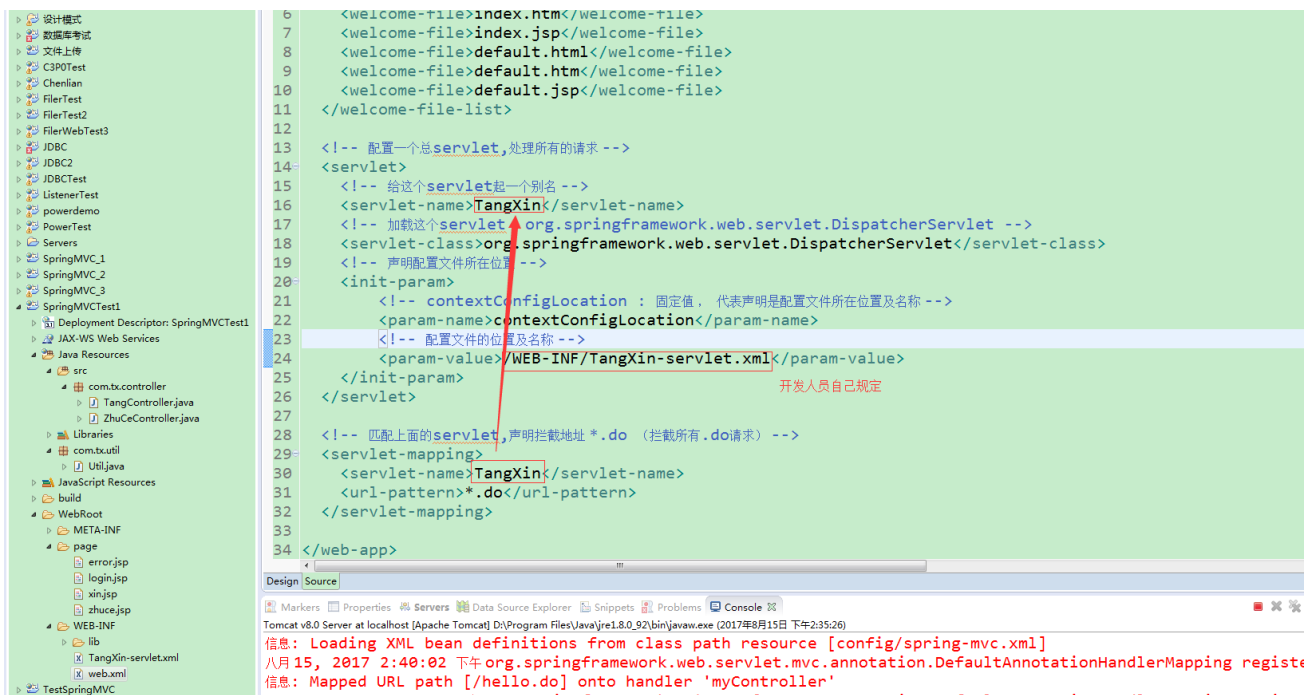


SpringMVC(1)

SpringMVC的配置及实现



```
10 <welcome-file>index.jsp</welcome-file>
11 <welcome-file>default.html</welcome-file>
12 <welcome-file>default.htm</welcome-file>
13 <welcome-file>default.jsp</welcome-file>
14 </welcome-file-list>
15
16 <servlet>
17   <servlet-name>Tang</servlet-name>
18   <!-- springMVC的前端控制器, 用于代替servlet转发网络请求 -->
19   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
20   <!-- 可以自定义servlet.xml配置文件的位置和名称, 默认为WEB-INF目录下, 名称为[<servlet-name>]-servlet.xml, 如spring-servlet.xml -->
21   <init-param>
22     <param-name>contextConfigLocation</param-name>
23     <param-value>/WEB-INF/spring-servlet.xml</param-value>
24   </init-param> -->
25   <!-- 启动tomcat时初始化该servlet -->
26   <load-on-startup>1</load-on-startup>
27 </servlet>
28
29 <servlet-mapping>
30   <servlet-name>Tang</servlet-name>
31   <!-- 决定哪些网络请求交给springMVC控制器来处理, "/" 是用来定义默认servlet映射的。也可以如 "*.html"表示拦截所有以html为扩展名的请求。 -->
32   <url-pattern>/</url-pattern>
33 </servlet-mapping>
34
35
```



```
6 <welcome-file>index.htm</welcome-file>
7 <welcome-file>index.jsp</welcome-file>
8 <welcome-file>default.html</welcome-file>
9 <welcome-file>default.htm</welcome-file>
10 <welcome-file>default.jsp</welcome-file>
11 </welcome-file-list>
12
13 <!-- 配置一个总servlet, 处理所有的请求 -->
14 <servlet>
15   <!-- 给这个servlet起一个别名 -->
16   <servlet-name>TangXin</servlet-name>
17   <!-- 加载这个servlet -->
18   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
19   <!-- 声明配置文件所在位置 -->
20   <init-param>
21     <!-- contextConfigLocation : 固定值, 代表声明是配置文件所在位置及名称 -->
22     <param-name>contextConfigLocation</param-name>
23     <!-- 配置文件的位置及名称 -->
24     <param-value>/WEB-INF/TangXin-servlet.xml</param-value>
25   </init-param>
26 </servlet>
27
28 <!-- 匹配上面的servlet, 声明拦截地址 *.do (拦截所有.do请求) -->
29 <servlet-mapping>
30   <servlet-name>TangXin</servlet-name>
31   <url-pattern>*.do</url-pattern>
32 </servlet-mapping>
33
34 </web-app>
35
```

Tomcat v8.0 Server at localhost [Apache Tomcat/8.0.92] bin\java.exe (2017年8月15日 下午2:35:26)

信息: Loading XML bean definitions from class path resource [config/spring-mvc.xml]

八月 15, 2017 2:40:02 下午 org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping register: 信息: Mapped URL path [/hello.do] onto handler 'myController'

```
<servlet>
  <servlet-name>Tang</servlet-name>
  <!-- springMVC的前端控制器, 用于代替servlet转发网络请求 -->
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <!-- 可以自定义servlet.xml配置文件的位置和名称, 默认为WEB-INF目录下, 名称为[<servlet-name>]-servlet.xml, 如spring-servlet.xml -->
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-servlet.xml</param-value>
  </init-param> -->
  <!-- 启动tomcat时初始化该servlet -->
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>Tang</servlet-name>
  <!-- 决定哪些网络请求交给springMVC控制器来处理, "/" 是用来定义默认servlet映射的。也可以如 "*.html"表示拦截所有以html为扩展名的请求。 -->
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
4       xmlns:mvc="http://www.springframework.org/schema/mvc"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
6       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd
7       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">
8
9     <!-- 表示将请求的URL和Bean名字映射, 如URL为"/上下文/hello", 则Spring配置文件必须有一个名字为"/hello"的Bean, 上下文默认忽略。-->
10    <bean
11        class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping" />
12
13    <!-- 表示所有实现了org.springframework.web.servlet.mvc.Controller接口的Bean可以作为Spring
14        Web MVC中的处理器。-->
15    <bean
16        class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter" />
17
18
19    <!-- 配置控制器的Bean节点(在spring的配置文件中, 注册自己定义的控制器的)。name:控制器的路径, class:控制器的包名与类名 -->
20    <bean name="/hello.do" class="com.tx.controller.Show" />
21
22
23    <!-- 这里的class属性是固定写法 -->
24    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
25        <!-- JstlView表示JSP模板页面需要使用JSTL标签库, classpath中必须包含jstl的相关jar包 -->
26        <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />
27
28        <!-- 查找视图页面的前缀和后缀(前缀【逻辑视图名】后缀), 比如传进来的逻辑视图名为hello, 则该jsp视图页面应该存放在"/WEB-INF/jsp/hello.jsp" -->
29        <property name="prefix" value="/" />
30        <property name="suffix" value=".jsp" />
31    </bean>
32
33 </beans>
34

```

域名空间

固定写法
url与java文件的映射

声明控制器

设置访问地址并且给访问地址匹配控制器

```

8
9
10 <!-- 允许创建url地址来分发处理请求 -->
11 <!-- <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"></bean> -->
12 <!-- 允许给url地址匹配对应的class来处理请求 -->
13 <bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"></bean>
14
15 <!-- 创建一个URL地址, 并且给这个地址匹配对应的处理类 -->
16 <bean name="/login" class="com.tx.controller.TangController"></bean>
17 <bean name="/zhuce" class="com.tx.controller.ZhuCeController"></bean>
18
19 <!-- 允许创建多个url地址来分发处理请求 -->
20 <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
21     <property name="mappings">
22         <props>
23             <!-- 控制器的访问路径, HelloWorld.do:控制器的名称 -->
24             <prop key="/requestpath.do">login</prop>
25             <prop key="/zhuce.do">zhuce</prop>
26         </props>
27     </property>
28 </bean>
29
30 <!-- 给控制层controller匹配对应的视图层地址 -->
31 <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
32     <!-- 声明视图层需要使用jstl -->
33     <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />
34     <!-- 指定视图层的位置 -->
35     <property name="prefix" value="/page/" />
36     <!-- 指定视图层的类型 -->
37     <property name="suffix" value=".jsp" />
38 </bean>
39 </beans>

```

```

<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <!-- 控制器的访问路径, HelloWorld.do:控制器的名称 -->
            <prop key="/hello">abc</prop>
            <prop key="/second">second</prop>
        </props>
    </property>
</bean>

<!-- 配置控制器的Bean节点(在spring的配置文件中, 注册自己定义的控制器的)。
    name:控制器的路径, class:控制器的包名与类名 -->
<bean name="abc" class="com.share.controller.MyController"/>
<bean name="second" class="com.share.controller.Second"/>

```

```

<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:applicationContext.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>Dispatcher</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

```

文件地址直接在src路径下

捕获所有的.do请求

```

<bean id="urlMapping"
  class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="/Login.do">loginController</prop>
    </props>
  </property>
</bean>

<bean id="LoginController" class="com.share.controller.LoginController">
  <property name="successView" value="showAccount"></property>
  <property name="failView" value="Login"></property>
</bean>

<bean id="viewResolver"
  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="prefix" value="/"></property>
  <property name="suffix" value=".jsp"></property>
</bean>
</beans>

```

请求地址

映射名称

处理登录请求的控制器

参数名称

参数的值

进入控制器时传入的参数

设置流程控制中转发的页面路径及页面格式

注解方式实现SpringMVC

- 1、web.xml配置不变
- 2、在spring-mvc.xml配置文件中
 - 1) 开启注解
 - 2) 设置视图层位置及地址
 - 3) 加载所有的注解--扫描所有注解所在的包

```

<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-4.1.xsd
  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">

  <!-- base-package="com.share.controller"为设置需要进行注解扫描的包 -->
  <context:component-scan base-package="com.share.controller" />

  <!-- 开启Spring的注解 -->
  <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />
  <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping" />

  <!-- 这里的class属性是固定写法 -->
  <bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <!-- JstlView表示JSP模板页面需要使用JSTL标签库，classpath中必须包含jstl的相关jar包 -->
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />

    <!-- 查找视图页面的前缀和后缀（前缀[逻辑视图名]后缀），比如传进来的逻辑视图名为hello，则该jsp视图页面应该存放在"WEB-INF/jsp/hello.jsp" -->
    <property name="prefix" value="/page/" />
    <property name="suffix" value=".jsp" />
  </bean>

</beans>

```

第三步，设置扫描的包
将所有有注解的类放在一个或多个包下，扫描这些包，让注解生效

第一步开启注解，加载这两个固定的类

第二步，指定视图层位置及类型
prefix:指定视图的位置
suffix:指定视图的类型

- 3、控制器编写

```

8
9 /**
10  * @Controller: 当前类是控制器
11  * */
12  @Controller
13  public class MyController {
14      /**
15       * @RequestMapping: 设置控制器的访问路径
16       * 浏览器通过/hello.do直接访问
17       * */
18      @RequestMapping(value = "/hello.do")
19      public ModelAndView abc(){
20          System.out.println("这是注解的控制器");
21          ModelAndView view = new ModelAndView();
22          //转发到first
23          view.setViewName("first");
24          return view;
25      }
26      /**
27       * 通过浏览器直接访问/second
28       * */
29      @RequestMapping(value = "/second.do")
30      public ModelAndView bcd(){
31          ModelAndView view = new ModelAndView();
32          view.setViewName("sencond");
33          System.out.println("第二个页面");
34          return view;
35      }
36  }
37

```

告诉虚拟机，这个类是控制器

最好以controller结尾，命名满足命名规范即可

给方法绑定对应的URL，浏览器可以通过访问这个URL获取对应的数据及跳转到对应的界面

方法名称满足命名规范即可

返回值：根据返回值类型确定界面的跳转还是数据的输出
返回值类型：String, ModelAndView, Void, javaBean
String: 视图名称
ModelAndView: 设置视图名称，携带数据，根据视图名称跳转界面，并且携带对应的数据，数据放在请求对象中
Void:
1、不需要跳转界面，直接通过响应对象输出数据 2、需要跳转界面，会将请求地址作为视图名称。例如 hello.do, 返回值是void, 那么跳转界面名称为hello

JavaBean: 将请求地址作为视图名称，并且将这个javaBean添加到对应的视图

错误集:

```

4
5 <servlet>
6     <servlet-name>spring</servlet-name>
7     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
8     <init-param>
9         <!-- contextConfigLocation:不能修改 -->
10        <param-name>contextConfigLocation</param-name>
11        <param-value>classpath:config/spring-mvc.xml</param-value>
12    </init-param>
13    <load-on-startup>1</load-on-startup>
14 </servlet>
15 <servlet-mapping>
16     <servlet-name>spring</servlet-name>
17     <url-pattern>*.do</url-pattern>
18 </servlet-mapping>

```

错误集1: 所有的请求一定要经过总的servlet

```

6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7 <title>Insert title here</title>
8 </head>
9 <body>
10     <a href="hello.do">hello world</a>
11     <br>
12     <a href="second.do">second</a>
13 </body>
14 </html>

```

Http://127.0.0.1:8080/SpringMVC_3/page/hello.do 一般都是错误的，出现404

/SpringMVC_3/请求地址

如果视图在自定义目录下，默认请求地址一定会在此目录下寻找

```

2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/
3     <display-name>SpringMVC_3</display-name>
4
5     <servlet>
6         <servlet-name>spring</servlet-name>
7         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
8         <init-param>
9             <!-- contextConfigLocation:不能修改 -->
10            <param-name>contextConfigLocation</param-name>
11            <param-value>classpath:config/spring-mvc.xml</param-value>
12        </init-param>
13        <load-on-startup>1</load-on-startup>
14    </servlet>
15    <servlet-mapping>
16        <servlet-name>spring</servlet-name>
17        <url-pattern>*.do</url-pattern>
18    </servlet-mapping>
19
20    <welcome-file-list>
21        <welcome-file>index.html</welcome-file>
22        <welcome-file>index.htm</welcome-file>
23        <welcome-file>index.jsp</welcome-file>

```

```

3 <beans xmlns="http://www.springframework.org/schema/beans"
4       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:mvc="http://www.springframework.org/schema/mvc"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
7       http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.1.xsd
8       http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">
9
10    <!-- base-package="com.share.controller"为设置需要进行注解扫描的包-->
11    <context:component-scan base-package="com.share.controller" />
12
13    <!-- 开启Spring的注解 -->
14    <bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />
15    <bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping" />
16
17    <!-- 这里的class属性是固定写法 -->
18    <bean
19        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
20        <!-- JstlView表示JSP模板页面需要使用JSTL标签库，classpath中必须包含jstl的相关jar包 -->
21        <property name="viewClass"
22            value="org.springframework.web.servlet.view.JstlView" />
23
24        <!-- 查找视图页面的前缀和后缀（前缀[逻辑视图名]后缀），比如传进来的逻辑视图名为hello，则该jsp视图页面应该存放在"WEB-INF/jsp/hello.jsp" -->
25        <property name="prefix" value="/page/" />
26        <property name="suffix" value=".jsp" />
27    </bean>
28
29
30 </beans>

```

启动时Tomcat一定会报错，在控制台一定会有报错输出，从最开始报错的位置开始找，找第一个异常所在，一般报错根源会接在末尾

使用SpringMVC流程：

- 1、导入对应的架包（20个架包+3个依赖包）
- 2、在web.xml里面声明一个总的servlet---org.springframework.web.servlet.DispatcherServlet
- 3、设置servlet拦截的请求

注：springmvc的原理：用的一个servlet 处理所有请求，写分发地址，将对应的请求交给对应的地址来处理，从而达到控制层，dao层，service层分离的效果

4、写springMVC的配置文件，

- 1) 编写对应的域名空间
- 2) 开启springMvc的注解
 - (1)*HandlerMapping: 允许创建分发的地址

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

- (2) *HandlerAdapter: 允许将地址绑定给对应的处理事件

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

3) 设置扫描的包路径（注解所在的地址）

```
<context:component-scan base-package="com.share.controller" />
```

4) 设置视图层的配置

- 1) 设置视图的地址

```
<property name="prefix" value="/page/" />
```

- 2) 设置视图的类型

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

- 3) 视图界面可能会用JSTL标签

```
<property name="viewClass"
    value="org.springframework.web.servlet.view.JstlView" />
```

5、编写控制器

- 1、创建一个自己的类
- 2、给这个类加上控制层注解（想要注解生效，控制器所在的包一定要被扫描）
- 3、给对应的方法上面添加请求地址

```
@RequestMapping(value="/test-mapping")
```

4、方法的名称随意，方法的参数可以为（Javabean,request,response,session,String等），系统会根据请求自动将对应的数据传入对应的参数中

5、根据返回值来跳转界面