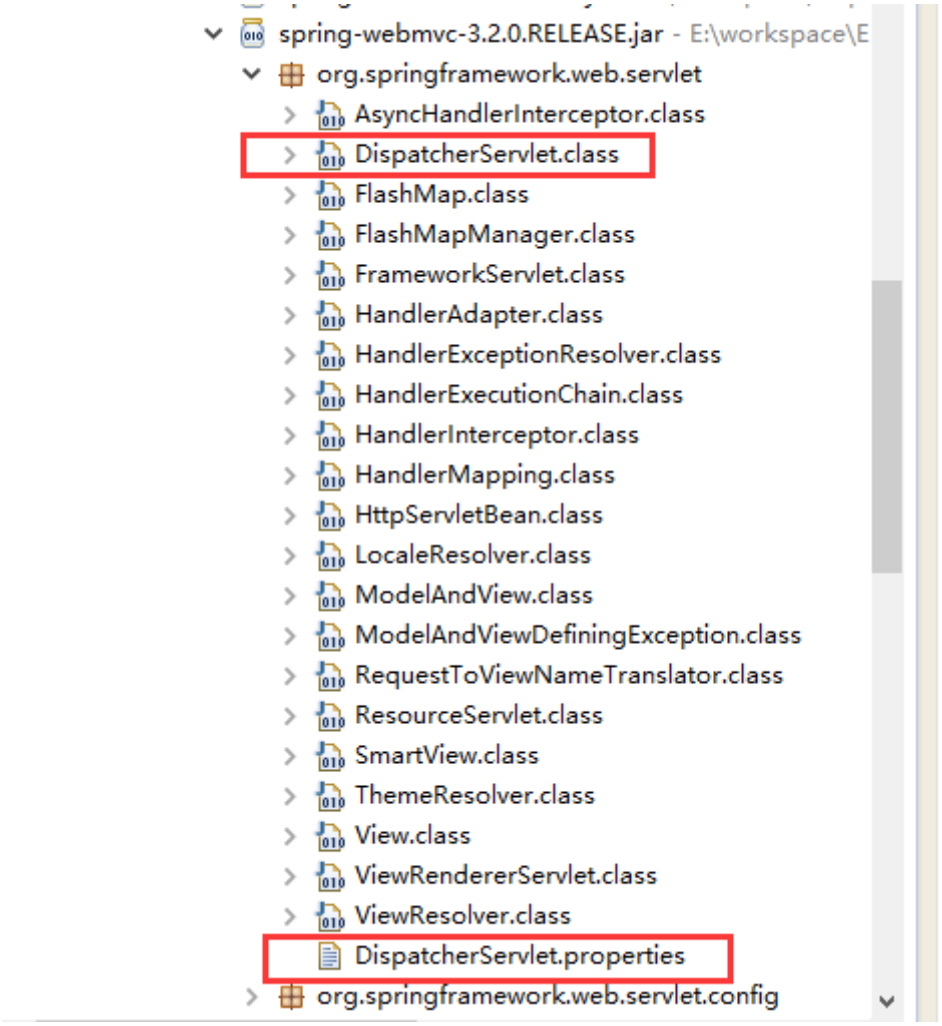
SpringMVC从入门到精通之第三章_慕课手记

好吧对于好多朋友说上一张不好懂，我感到抱歉，上次写的时候那个点还没下班，已经深夜了脑子有点短路。但是只要能跟着复制进自己的程序里面是可以运行的，每次都是能保证复制即可运行的。不然我没有必要贴代码了，直接截图多方便。
上一章节中首先讲解前端控制器DispatcherServlet的配置包括加载springmvc文件、拦截什么样的请求等
还有两个组件：分别是适配器和映射器（另外再补充一组）

```
非注解的处理器映射器
处理器映射器：
org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping
另一个映射器：
org.springframework.web.servlet.handler.SimpleUrlHandlerMapping
补充：多个映射器可以并存，前端控制器判断url能让哪些映射器映射，就让正确的映射器处理。
```

```
非注解的处理器适配器：
org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter
要求编写的Handler实现 Controller接口。
org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter
要求编写的Handler实现 HttpRequestHandler接口。
```

可能有人没有配置上面的那些发现程序能够运行。那么知识点来了：
本章的第一个知识点在一个配置文件中是springmvc框架里面的一个配置文件，它的作用就是当没有配置上面的适配器和处理器框架会根据这个文件默认使用一组适配器和映射器。



我们来简单的看下这个文件里面的内容：

要先看下面这个翻译：

```
# Default implementation classes for DispatcherServlet's strategy interfaces.
# Used as fallback when no matching beans are found in the DispatcherServlet context.
# Not meant to be customized by application developers.

org.springframework.web.servlet.LocaleResolver=org.springframework.web.servlet.i18n.AcceptHeaderLocaleResolver

org.springframework.web.servlet.ThemeResolver=org.springframework.web.servlet.theme.FixedThemeResolver
```

这个就是默认加载的处理器映射器有两个：第一个是根据Bean的名字URL的映射器，第二个是默认的注解处理器映射器（等下会讲注解的）

```
org.springframework.web.servlet.HandlerMapping=org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping,\
```

```
org.springframework.web.servlet.mvc.annotation.DefaultAnnotationH
andlerMapping
这个是默认加载的处理器适配器有三个：http请求相关的、简单的处理器适配器、默认
的注解处理器适配器（等下会说到注解的）
org.springframework.web.servlet.HandlerAdapter=org.springframewor
k.web.servlet.mvc.HttpRequestHandlerAdapter,\

org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapte
r,\

org.springframework.web.servlet.mvc.annotation.AnnotationMethodHa
ndlerAdapter

org.springframework.web.servlet.HandlerExceptionResolver=org.spri
ngframework.web.servlet.mvc.annotation.AnnotationMethodHandlerExc
eptionResolver,\

org.springframework.web.servlet.mvc.annotation.ResponseStatusExce
ptionResolver,\

org.springframework.web.servlet.mvc.support.DefaultHandlerExcepti
onResolver

org.springframework.web.servlet.RequestToViewNameTranslator=org.s
pringframework.web.servlet.view.DefaultRequestToViewNameTranslato
r

org.springframework.web.servlet.ViewResolver=org.springframework.
web.servlet.view.InternalResourceViewResolver

org.springframework.web.servlet.FlashMapManager=org.springframewo
rk.web.servlet.support.SessionFlashMapManager
```

上面的文件里面涉及到两个注解的组件分别是：

```
注解的处理器映射器
org.springframework.web.servlet.mvc.annotation.DefaultAnnotationH
```

```
andlerMapping
注解的处理器适配器
org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter
这个两个是在spring3.1版本之前用的，在3.1之后使用的都是和http请求相关的两个组件
分别是：
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping注解映射器。
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter注解适配器。
```

程序配置：注解的这个两个必须**成对**出现，要么全部出现、要么都不出现

```
    <!-- 知识点二：=========注解映射器====================== -->
    <!-- 说明 ：对类中标记@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。   -->
    <!-- 注意：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。  -->
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

    <!-- 知识点三：=========注解适配器====================== -->
    <!-- 说明：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。  -->
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>
```

实例程序：

```
/**
```

```
 * @Title: ItemList3.java
 * @Package com.springapp.web.controller
 * @Description: TODO
 * @author Hanson
 * @date 2015-10-25
 */
package com.springapp.web.controller;

import java.util.ArrayList;
import java.util.List;

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

import com.springapp.pojo.Items;

/**
 *
 * @ClassName: ItemList3
 * @Description: TODO(注解的处理器映射器和注解的适配器)
 * @author: Hanson.Q
 * @date: 2016年1月24日 下午4:22:44
 *
 */
@Controller
public class ItemList3 {
    /**
     *
     * @Title: queryItem
     * @Description: TODO (定义请求url到处理器功能方法的映射)
     * @Author: Hanson.Q
     * @Create Date: 2016年1月24日 下午4:23:11
     * @History: 2016年1月24日 下午4:23:11 Hanson.Q Created.
     *
     * @return
     *
     */
```

```
    @RequestMapping("/queryItem.action")
    public ModelAndView queryItem() {
        // 商品列表
        List<Items> itemsList = new ArrayList<Items>();
        Items items_2 = new Items();
        items_2.setName("苹果手机");
        items_2.setPrice(5000f);
        items_2.setDetail("iphone6s苹果手机！");
        itemsList.add(items_2);
        // 创建modelAndView准备填充数据、设置视图
        ModelAndView modelAndView = new ModelAndView();
        // 填充数据
        modelAndView.addObject("itemsList", itemsList);
        // 视图
        modelAndView.setViewName("order/itemsList");
        return modelAndView;
    }
}
```

当然光配置上面两个注解相关的组件还是不够的为什么呢？因为spring容器并不知道哪些类是处理器，这时候需要一个配置来把处理器交给容器管理。
知识点四：

```
    <!-- 组件扫描器：可以扫描@Controller、@Service、@Repository等等 -
->
    <context:component-scan base-
package="com.springapp.web.controller"/>
```

那么可能还有点懒说配置那两个注解相关的组件还是不方便，还是得写那么长。框架又帮我们想好了。
知识点五：真实开发的时候使用下面的配置可以代替上面配置的两个注解的组件

```
    <!-- springmvc使用<mvc:annotation-driven> -->
    <!-- 自动加载RequestMappingHandlerMapping和
RequestMappingHandlerAdapter， -->
    <!-- 可用在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。 -->
```

```
    <mvc:annotation-driven/>
```

这样以后这个配置文件中只要配置如下内容就可以了：

没有使用注解时候的配置文件：

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-
3.2.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-
3.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context-3.2.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-
3.2.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-
3.2.xsd ">
    <!-- 配置简单的控制器处理适配器 -->
    <!-- SimpleControllerHandlerAdapter：即简单控制器处理适配器， -->
    <!-- 所有实现了org.springframework.web.servlet.mvc.Controller
接口的Bean作为Springmvc的后端控制器。 -->
    <bean
class="org.springframework.web.servlet.mvc.SimpleControllerHandle
rAdapter"/>
    <bean
class="org.springframework.web.servlet.mvc.HttpRequestHandlerAdap
ter" />
```

```
    <!-- 配置BeanNameUrl处理器映射器 -->
    <!-- BeanNameUrlHandlerMapping：表示将定义的Bean名字作为请求的
url，需要将编写的controller在spring容器中进行配置， -->
    <!-- 且指定bean的name为请求的url，且必须以.action结尾。 -->
    <bean
class="org.springframework.web.servlet.handler.BeanNameUrlHandler
Mapping" />
    <bean
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMa
pping"/>

    <!-- controller配置 -->
    <!-- name="/items1.action"：前边配置的处理器映射器为
BeanNameUrlHandlerMapping， -->
    <!-- 如果请求的URL 为“上下文/items1.action”将会成功映射到ItemList1
控制器。 -->
    <bean name="/items1.action" id="itemList1"
class="com.springapp.web.controller.ItemList1"/>
    <bean name="/items2.action" id="itemList2"
class="com.springapp.web.controller.ItemList2"/>

    <!-- 配置视图解析器 -->
    <!-- InternalResourceViewResolver：支持JSP视图解析 -->
    <!-- viewClass：JstlView表示JSP模板页面需要使用JSTL标签库，所以
classpath中必须包含jstl的相关jar包； -->
    <!-- prefix 和suffix：查找视图页面的前缀和后缀，最终视图的址为： -->
    <!-- 前缀+逻辑视图名+后缀，逻辑视图名需要在controller中返回
ModelAndView指定，比如逻辑视图名为hello，-->
    <!-- 则最终返回的jsp视图地址 "WEB-INF/jsp/hello.jsp" -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewR
esolver">
        <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
```

```
</beans>
```

使用注解之后的配置文件：

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-
3.2.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-
3.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-
context-3.2.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-
3.2.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-
3.2.xsd ">

    <!-- ======================组件扫描器====================== --
>
    <context:component-scan base-
package="com.springapp.web.controller" />

    <!-- springmvc使用<mvc:annotation-driven> -->
    <!-- 自动加载RequestMappingHandlerMapping和
RequestMappingHandlerAdapter， -->
    <!-- 使用<mvc:annotation-driven>替代注解处理器和适配器的配置。 -->
    <mvc:annotation-driven />
```

```
    <!-- 配置视图解析器 -->
    <bean

class="org.springframework.web.servlet.view.InternalResourceViewR
esolver">
        <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView"
/>
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

把这个配置和之前的对比一下看看是不是简单的多了。
之前写那么多我原本的意思是想让大家明白的深入一点，不能只知道怎么使用但是本身原理还是不知道，那长期以往后果就是你是一个没有想法的程序员。而不是开发工程师。
在这边还要感谢慕课，让我有机会能够与大家分享自己学习技术上的心得。

相关标签：