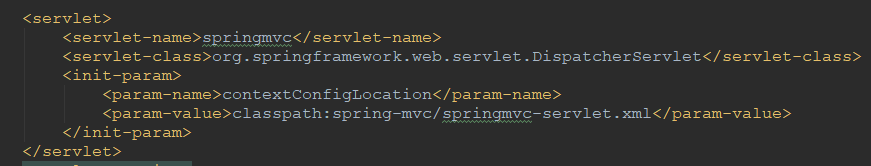
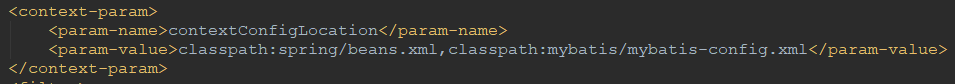
## web.xml当中spring配置文件说明

在web.xml里面，我们可以在两个位置上面指定对spring配置文件的加载，

一个是：



另一个是：



这两者有什么联系与区别呢？

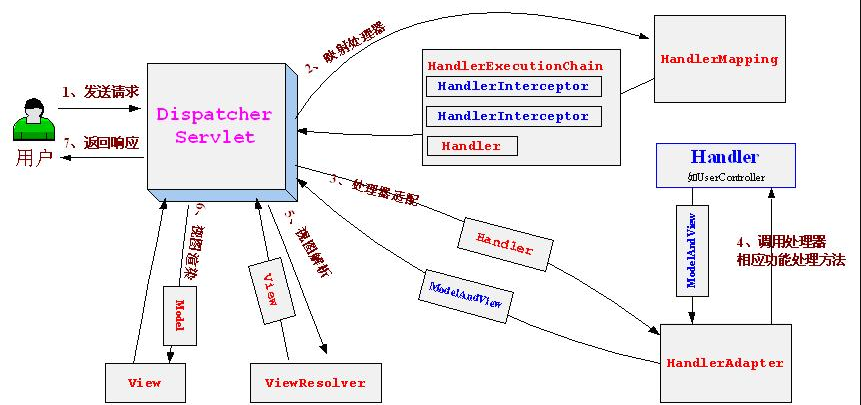
ContextLoaderListener中创建的ApplicationContext主要用于整个web应用程序需要共享的一些组件，比如DAO，数据库的ConnectionFactory等。而由DispatcherServlet创建的ApplicationContext主要用于和该servlet相关的一些组件，比如Controller，ViewResovler，HandlerMapping等。

对于作用范围来讲，在DispathcerServlet可以引用ContextLoaderListener所创建的ApplicationContext，而反过来不行。

从web.xml配置中可以看出来，ContextLoaderListener是会先于DispatcherServlet创建ApplicationContext，DispatcherServlet创建ApplicationContext时会先找到由Listener所创建的ApplicationContext，再将后者的ApplicationContext作为参数传给DispatcherServlet的ApplicationContext的setParent()方法，作为它的父上下文。当需要一个bean对象的时候，先在子ApplicationContext中查找，再在父ApplicationContext中查找。

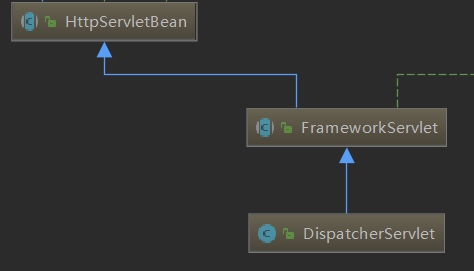
## DispatcherServlet

DispatcherServlet是SpringMVC的一个核心组件，它主要用于职责调度工作，本身主要用于控制流程，其中最重要的一个方法是doDispatch，该方法将请求分发给不同的handler，并将结果进行解析渲染，最终返回给客户端。



处理步骤如下：

1. Tomcat启动后，对DispatcherServlelt进行实例化，DispatcherServlet的类继承图如下所示：

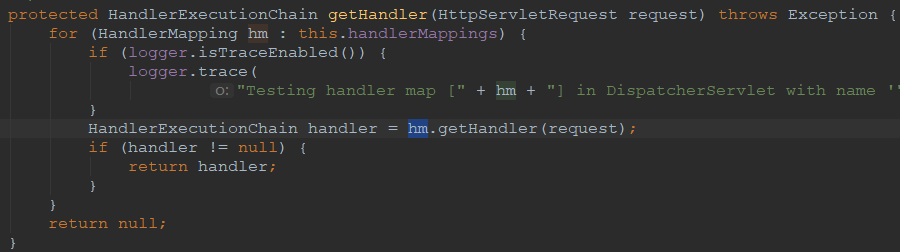


HttpServletBean的init方法所做的主要事情是将web.xml中设置的DispatcherServlet的初始化参数加载到该servlet中，并且在init方法中留给子类FrameWorkServlet一个initServletBean方法去重写。

FrameWorkServlet重写initServletBean的目的主要是用来创建ApplicationContext，该WebApplicationContext加载的xml文件主要配置了只对SpringMVC有效的Bean，例如Controller，HanderMapping，HanderAdapter等。FrameWorkServlet中有一个onRefresh方法，由子类DispatcherServlet去实现。

DispatcherServlet中会调用onRefresh方法来完成初始化，onRefresh方法调用的是initStrategies方法，该方法初始化SpringMVC中的一些策略，例如，initHanderMappings方法就是初始化HandlerMappings，如果ApplicationContext中没有配置HandlerMapping的bean，那么机会默认使用BeanNameUrlHandlerMapping。

1. 客户端发出请求，如果该请求满足web.xml中DispatcherServlet的URL，那么服务器就会将请求交给springMVC处理。
2. 当获取到请求后，会根据当前的request来获取具体的HandlerExecutionChain，HandlerExecutionChain是一个类，里面封装了一个handler对象以及0个或者多个handler拦截器。具体的获取过程如下：



DispatcherServlet在初始化的时候会把所有注册过的HandlerMapping都加载进来，在这个方法里面遍历所有的HandlerMapping，并根据request从中获取到HandlerExecutionChain。这边需要说下HandlerMaping是什么，HandlerMapping定义了请求与HandlerExecutionChain之间的映射关系。这样以来就找到了request所需要的HandlerExecutionChain。之后需要获取到一个支持当前handler的handlerAdapter，该对象是真正调用handler处理的对象，获取的方法还是遍历所有的已经注册的handlerAdapter。

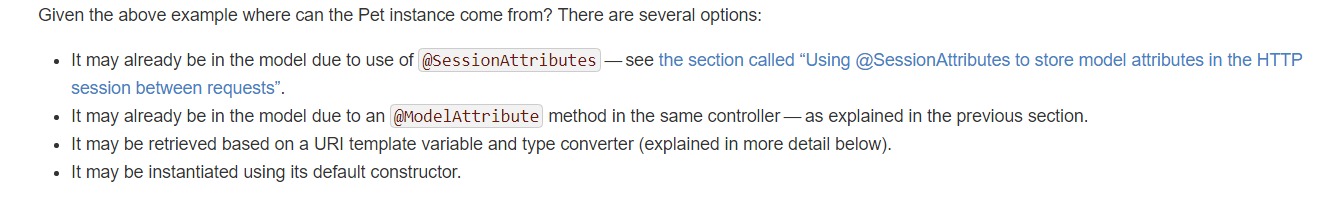
1. 之后，DispatcherServlet处理interceptors的preHandle方法。
2. 接着调用handlerAdapter的handle方法，里面需要传入具体被调用的handler，以及request，resopnse，使用handlerAdapter的原因是在DispatcherServlet当中可以使用统一的接口，而不需要根据不同的handler来调用不同的方法。Handle方法返回一个modelAndview对象。
3. 处理interceptor的postHandle方法。
4. 最后调用processDispatchResutl方法，当中会处理异常或者渲染modelandview。

## @ModelAttribute

ModelAttribute可以放在method上，也可以放在method参数上面。

当放在method上的时候，意味着将该方法的返回值添加到model中，添加有ModelAttribute注解的那些方法会在添加有RequestMapping注解的方法之前执行。

当放在method参数上面的时候，意味着可以从model中抽取匹配的参数并注入到参数对象当中，操作较为方便，当然前提需要request中的name与对象的属性名字匹配。这个参数对象会怎么来呢？官网给出的解释如下：



* 该对象已经存在于SessionAttributes中**了**，可以直接获取到
* 该对象已经存在于ModelAttribute中，可以直接获取
* 它可以基于URI模板变量和类型转化器检索
* 如果上面都没有，那么就创建一个新的对象，并放入model中。

## @SessionAttribute

如果我们需要在多次请求之间保持数据，那么就需要我们将数据存入到HttpSession中，例如多步骤提交表单的时候。SpringMVC提供了@SessionAttributes进行请求之间的透明存储数据。

@SessionAttribute(value={“user”})的含义是是将model中的名字为”user”的对象存储会话中。@SessionAttribute注解一般添加在controller类上。

包含@SessionAttribute注解的controller执行流程如下：

1. 首先根据@SessionAttribute注解信息查找会话内的对象放入到模型数据中；
2. 执行@ModelAttribute注解的方法：如果模型数据中包含同名的数据，@ModelAttribute方法会执行，但是其执行的结果不会添加到model中。否则，会将执行的结果添加到model中。
3. 执行@RequestMapping方法，绑定@ModelAttribute注解的参数：查找模型数据是否有@ModelAttribute注解的同名对象，如果有就直接使用，如果没有就通过反射创建一个，并根据url给对象赋值，最终将该对象存入model中。
4. 如果会话可以销毁了，例如多步骤提交表单的最后一步，此时可以调用SessionStatus对象的setComplete()标识当前会话中指定的数据可以进行清理了，此时当@RequestMapping功能处理方法执行完毕后会进行清理会话数据。
5. 那什么时候将@SessionAttribute注解中指定的值从model中添加到Httpsession中呢？当上述流程走完之后，最后会调用一个requestComplete方法，该方法中会调用updateAccessdSessionAttributes()方法，该方法会将session中被影响到的属性进行更新，如果一开始没有，就会将期望的对象添加进去。

## RequestMappingHandlerAdapter

之前说过，handlerAdapter的作用是用来调用与url相对的那个controller方法，而这个controller是通过handlerMapping找出来的，当然前提是controller得注册到ioc容器当中。之前的Adapter是基于非注解式的，一切都是使用基于xml的方法。但当我们使用注解方式的时候，我们就需要重新设置handlerMapiing跟handlerAdapter。HandlerMapping使用RequestMappingHandlerMapping，handlerAdapter使用RequestMappingHandlerAdapter。

RequestMappingHandlerAdapter类中主要的方法是invokeHandlerMethod方法，该方法主要的动作为：

* 创建一个ServletInvocableHandlerMethod对象，该对象用于最终调用RequestMaping方法。
* 给该对象添加参数解析器、返回值处理器等
* 创建一个ModelFactory对象，并调用该对象的initModel方法，该方法的作用如上上方@SessionAttribute执行流程所示。
* 其他还有一些设置，这边就先忽略
* 最终会调用invocableMethod方法的invokeAndHandle方法，该方法里面会获取@RequestMapping注解方法的参数，通过反射创建该方法所在的对象，在其中调用method.invoke方法，最终达到能够调用url对应handler方法的目的。