


DispatcherServlet

 <http://jinnianshilongnian.iteye.co...>

1、Controller：处理器/页面控制器，做的是MVC中的C的事情，但控制逻辑转移到前端控制器了，用于对请求进行处理；

2、HandlerMapping：请求到处理器的映射，如果映射成功返回一个HandlerExecutionChain对象（包含一个Handler处理器（页面控制器）对象、多个HandlerInterceptor拦截器）对象；如BeanNameUrlHandlerMapping将URL与Bean名字映射，映射成功的Bean就是此处的处理器；

3、HandlerAdapter：HandlerAdapter将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器；如SimpleControllerHandlerAdapter将对实现了Controller接口的Bean进行适配，并且掉处理器的handleRequest方法进行功能处理；

4、ViewResolver：ViewResolver将把逻辑视图名解析为具体的View，通过这种策略模式，很容易更换其他视图技术；如InternalResourceViewResolver将逻辑视图名映射为jsp视图；

5、LocalResover：本地化解析，因为Spring支持国际化，因此LocalResover解析客户端的Locale信息从而方便进行国际化；

6、ThemeResovler：主题解析，通过它来实现一个页面多套风格，即常见的类似于软件皮肤效果；

7、MultipartResolver：文件上传解析，用于支持文件上传；

8、HandlerExceptionResolver：处理器异常解析，可以将异常映射到相应的统一错误界面，从而显示用户友好的界面（而不是给用户看到具体的错误信息）；

9、RequestToViewNameTranslator：当处理器没有返回逻辑视图名等相关信息时，自动将请求URL映射为逻辑视图名；

10、FlashMapManager：用于管理FlashMap的策略接口，FlashMap用于存储一个请求的输出，当进入另一个请求时作为该请求的输入，通常用于重定向场景，后边会细述。

web.xml加载过程

公众号：java思维导图

1 启动WEB项目的时候，容器首先会去它的配置文件web.xml读取两个节点：`<listener>` `</listener>`和`<context-param>` `</context-param>`。

2 紧接着，容器创建一个`ServletContext (application)`，这个WEB项目所有部分都将共享这个上下文。

3 容器以`<context-param>` `</context-param>`的name作为键，value作为值，将其转化为键值对，存入`ServletContext`。

4 容器创建`<listener>` `</listener>`中的类实例，根据配置的class类路径`<listener-class>`来创建监听，在监听中会有`contextInitialized(ServletContextEvent args)`初始化方法，启动Web应用时，系统调用Listener的该方法，在这个方法中获得：
`ServletContext application = ServletContextEvent.getServletContext();`
`context-param的值 = application.getInitParameter("context-param的键");`
得到这个`context-param`的值之后，你就可以做一些操作了。

举例：你可能想在项目启动之前就打开数据库，那么这里就可以在`<context-param>`中设置数据库的连接方式（驱动、url、user、password），在监听类中初始化数据库的连接。这个监听是自己写的一个类，除了初始化方法，它还有销毁方法，用于关闭应用前释放资源。比如：说数据库连接的关闭，此时，调用`contextDestroyed(ServletContextEvent args)`，关闭Web应用时，系统调用Listener的该方法。

5 接着，容器会读取`<filter>` `</filter>`，根据指定的类路径来实例化过滤器。

☺ web.xml配置

☰ 公众号：java思维导图

contextConfigLocation

- 指定Spring IoC容器需要读取的XML文件路径
- 默认会去/WEB-INF/下加载applicationContext.xml

ContextLoaderListener

- Spring监听器
- Spring MVC在Web容器中的启动类，读取applicationContext.xml，负责Spring IoC容器在Web上下文中的初始化

DispatcherServlet

- 前端处理器，接受的HTTP请求和转发请求的类

CharacterEncodingFilter

- 字符集过滤器

IntrospectorCleanupListener

- 防止Spring内存溢出监听器

😊 配置文件标签

📖 公众号：java思维导图

context:annotation-config

他的作用是式地向 Spring 容器注册AutowiredAnnotationBeanPostProcessor、CommonAnnotationBeanPostProcessor、PersistenceAnnotationBeanPostProcessor 以及 RequiredAnnotationBeanPostProcessor 这 4 个BeanPostProcessor。

注册这4个 BeanPostProcessor的作用，就是为了你的系统能够识别相应的注解。

context:component-scan

两个子标签

<context:include-filter> 指定的扫描

<context:exclude-filter> 指定的不扫描

例子：只扫描controller

```
<context:component-scan base-package="tv.huan.weisp.web .controller">
< context:include-filter type=" annotation"
expression="org.springframework.stereotype.Controller"/>
</context:component-scan>
```

注意

如果配置了<context:component-scan>那么<context:annotation-config/>标签就可以不用再xml中配置了，因为前者包含了后者。

context:property-placeholder

加载单独配置的property文件的参数

```
<context:property-placeholder location="classpath:config.properties" />
```

import

引入其他的spring配置文件

```
<import resource="classpath:spring/spring-mybatis.xml" />
```


特殊bean

公众号：java思维导图

HandlerMapping

处理器映射。它会根据某些规则将进入容器的请求映射到具体的处理器以及一系列前处理器和后处理器（即处理器拦截器）上。具体的规则视HandlerMapping类的实现不同而有所不同。其最常用的一个实现支持你在控制器上添加注解，配置请求路径。当然，也存在其他的实现。

HandlerAdapter

处理器适配器。拿到请求所对应的处理器后，适配器将负责去调用该处理器，这使得DispatcherServlet无需关心具体的调用细节。比方说，要调用的是一个基于注解配置的控制器，那么调用前还需要从许多注解中解析出一些相应的信息。因此，HandlerAdapter的主要任务就是对DispatcherServlet屏蔽这些具体的细节。

HandlerExceptionResolver

处理器异常解析器。它负责将捕获的异常映射到不同的视图上去，此外还支持更复杂的异常处理代码。

ViewResolver

视图解析器。它负责将一个代表逻辑视图名的字符串（String）映射到实际的视图类型View上。

LocaleResolver & LocaleContextResolver

地区解析器 和 地区上下文解析器。它们负责解析客户端所在的地区信息甚至时区信息，为国际化的视图定制提供了支持。

ThemeResolver

主题解析器。它负责解析你web应用中可用的主题，比如，提供一些个性化定制的布局等。

MultipartResolver

解析multi-part的传输请求，比如支持通过HTML表单进行的文件上传等。

FlashMapManager

FlashMap管理器。它能够存储并取回两次请求之间的FlashMap对象。后者可用于在请求之间传递数据，通常是在请求重定向的情境下使用。

异常处理方式

HandlerExceptionResolver

可以实现全局异常控制

HandlerExceptionResolver接口中定义了一个resolveException方法，用于处理Controller中的异常。Exception ex参数即Controller抛出的异常。返回值类型是ModelAndView，可以通过这个返回值来设置异常时显示的页面。

SimpleMappingExceptionResolver

Spring提供的一个默认异常实现类SimpleMappingExceptionResolver

@ExceptionHandler

可以实现局局异常控制

如果@ExceptionHandler方法是在控制器内部定义的，那么它会接收并处理由控制器（或其任何子类）中的@RequestMapping方法抛出的异常。

如果你将@ExceptionHandler方法定义在@ControllerAdvice类中，那么它会处理相关控制器中抛出的异常。

web.xml的错误-page标签

简单处理异常和跳转，灵活程度不及HandlerExceptionResolver的异常处理



公众号：java思维导图