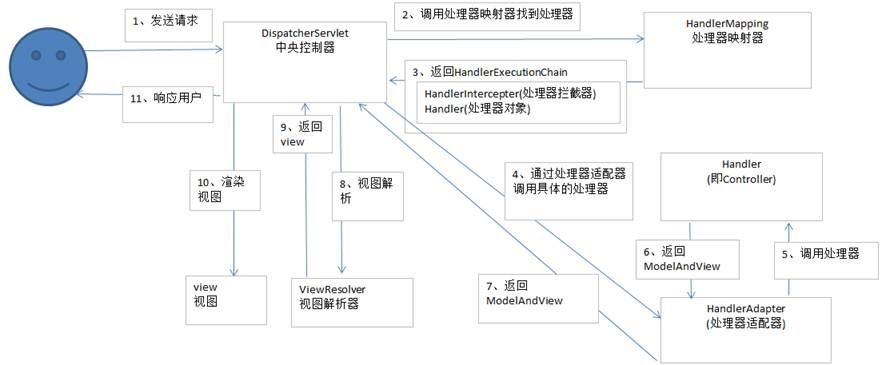
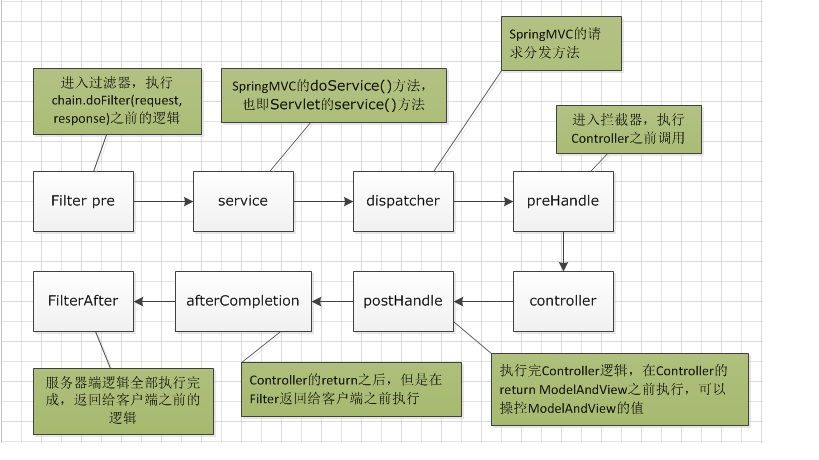
[Spring MVC请求处理流程图](http://676744379-qq-com.iteye.com/blog/1530911)

**博客分类：** [spring mvc](http://676744379-qq-com.iteye.com/category/222868)

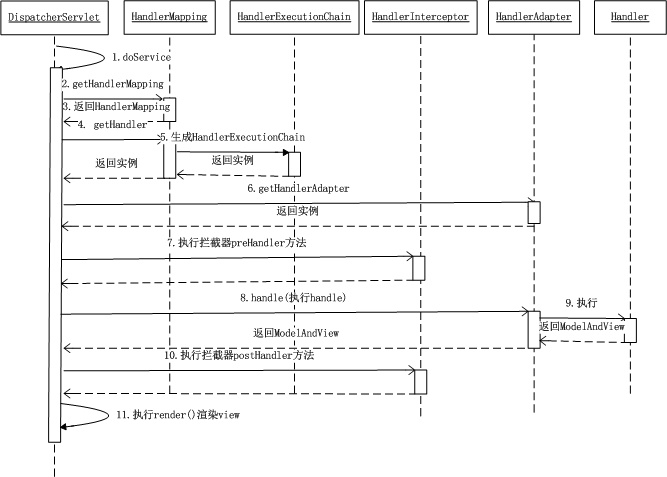
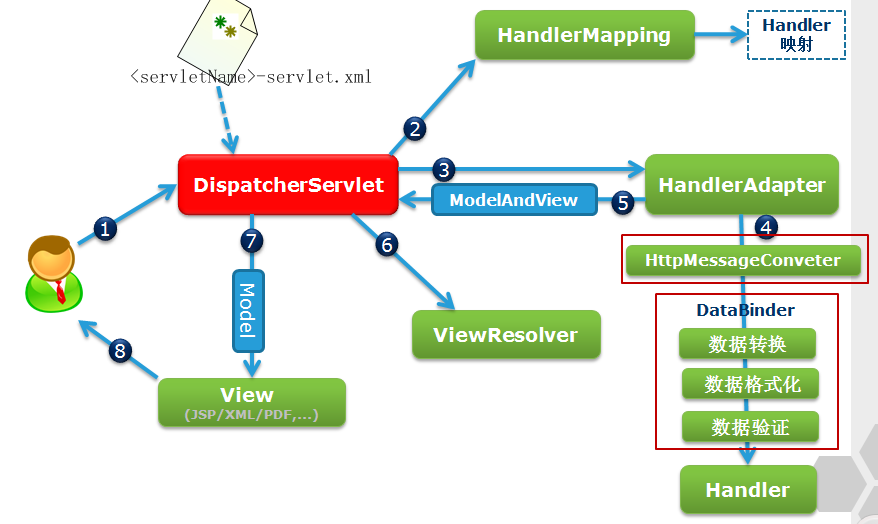


流程   
1、用户发送请求至前端控制器DispatcherServlet   
2、DispatcherServlet收到请求调用HandlerMapping处理器映射器。   
3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。   
4、DispatcherServlet调用HandlerAdapter处理器适配器   
5、HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。   
6、Controller执行完成返回ModelAndView   
7、HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet   
8、DispatcherServlet将ModelAndView传给ViewReslover视图解析器   
9、ViewReslover解析后返回具体View   
10、DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。   
11、DispatcherServlet响应用户

包含过滤器的过程：



当web程序启动的时候,ContextLoaderServlet会把对应的配置文件信息读取出来,通过注射去初始化控制器DispatchServlet. 而当接受到一个HTTP请求的时候, DispatchServlet会让HandlerMapping去处理这个请求.HandlerMapping根据请求URL(不一定非要是URL,完全可以自定义,非常灵活)来选择一个Controller. 然后DispatchServlet会在调用选定的Controller的handlerRequest方法,并且在这个方法前后调用这个Controller的interceptor(假如有配置的话),然后返回一个视图和模型的集合ModelAndView.框架通过ViewResolver来解析视图并且返回一个View对象,最后调用View的render方法返回到客户端



**Spring工作流程描述**

      1. 用户向服务器发送请求，请求被Spring 前端控制Servelt DispatcherServlet捕获；

      2. DispatcherServlet对请求URL进行解析，得到请求资源标识符（URI）。然后根据该URI，调用HandlerMapping获得该Handler配置的所有相关的对象（包括Handler对象以及Handler对象对应的拦截器），最后以HandlerExecutionChain对象的形式返回；

      3. DispatcherServlet 根据获得的Handler，选择一个合适的HandlerAdapter。（**附注**：如果成功获得HandlerAdapter后，此时将开始执行拦截器的preHandler(...)方法）

       4.  提取Request中的模型数据，填充Handler入参，开始执行Handler（Controller)。 在填充Handler的入参过程中，根据你的配置，Spring将帮你做一些额外的工作：

      HttpMessageConveter： 将请求消息（如Json、xml等数据）转换成一个对象，将对象转换为指定的响应信息

      数据转换：对请求消息进行数据转换。如String转换成Integer、Double等

      数据根式化：对请求消息进行数据格式化。 如将字符串转换成格式化数字或格式化日期等

      数据验证： 验证数据的有效性（长度、格式等），验证结果存储到BindingResult或Error中

      5.  Handler执行完成后，向DispatcherServlet 返回一个ModelAndView对象；

      6.  根据返回的ModelAndView，选择一个适合的ViewResolver（必须是已经注册到Spring容器中的ViewResolver)返回给DispatcherServlet ；

      7. ViewResolver 结合Model和View，来渲染视图

      8. 将渲染结果返回给客户端。

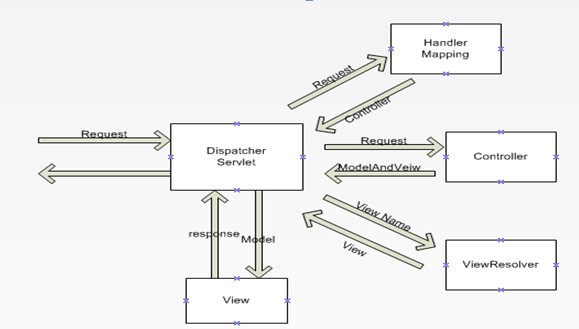
Spring工作流程描述

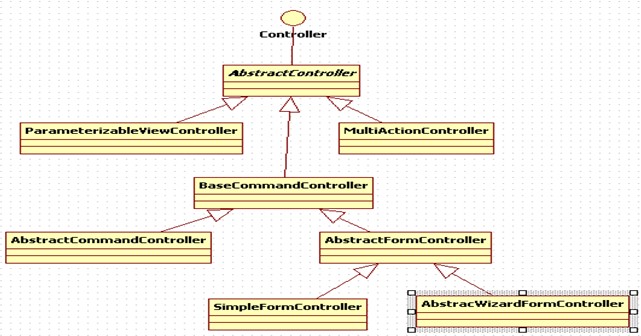
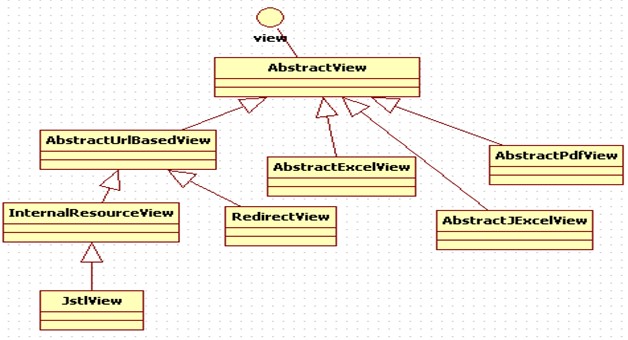
    为什么Spring只使用一个Servlet(DispatcherServlet)来处理所有请求？

     详细见J2EE设计模式-前端控制模式

    Spring为什么要结合使用HandlerMapping以及HandlerAdapter来处理Handler?

    符合面向对象中的单一职责原则，代码架构清晰，便于维护，最重要的是代码可复用性高。如HandlerAdapter可能会被用于处理多种Handler。

  1.springmvc请所有的请求都提交给DispatcherServlet,它会委托应用系统的其他模块负责对请求进行真正的处理工作。   
　　2.DispatcherServlet查询一个或多个HandlerMapping,找到处理请求的Controller.   
　　3.DispatcherServlet将请求提交到目标Controller   
　　4.Controller进行业务逻辑处理后，会返回一个ModelAndView   
　　5.Dispathcher查询一个或多个ViewResolver视图解析器,找到ModelAndView对象指定的视图对象   
　　6.视图对象负责渲染返回给客户端。Dispatcher通过model将ModelAndView()中的参数进行解析最终展现出完整的view并返回给客户端。

1. 当DispatcherServlet接到请求时，他先会查找适当的处理程序来处理请求。DispatcherServlet通过一个或者多个处理程序映射，将每个请求映射到处理程序中。处理程序映射配置在web应用程序的上下文中，是实现了HandlerMapping接口的Bean。它负责为请求返回一个适当的处理程序（也就是Controller）。处理程序映射通常根据请求的URL将请求映射到处理程序（Controller）。   
2. 一旦DispatcherServlet选择了适当的控制器，它就会调用这个控制器来处理请求。   
3. 控制器处理完请求后，会将模型和视图名（有时候是视图对象）返回给DispatcherServlet。模型包含了控制器要传递给视图进行显示的属性。如果返回的是视图名称，它会被解析成视图对象再进行呈现。绑定模型和视图的基本类是ModelAndView   
4. 当DispatcherServlet接收到模型和视图名称时，它会将逻辑视图名称解析成视图对象再进行呈现。DispatcherServlet从一个或者多个视图解析器中解析视图。视图解析器配置在Web应用程序上下文中，是实现了ViewResolver接口的Bean。它的任务是根据逻辑视图名称返回视图对象。   
5. 一旦DispatcherServlet将视图名称解析称为视图对象，它就会呈现视图对象，并传递控制器返回的模型。视图的任务是将模型属性展示给用户。   
  
DispatcherServlet接到请求后如何映射到Controller呢？   
在Spring MVC中，Web请求被Web应用程序上下文中声明的一个或者多个处理程序映射Bean（是吸纳了HandlerMapping接口的Bean）映射到Controller。Spring MVC提供了几种HandlerMapping实现：   
1. BeanNameUrlHandlerMapping(默认情况)，他根据Controller Bean名称中指定的URL模式将请求映射到处理程序上。   
eg. <bean name="/welcome.htm" class="com.kevin.controller.WelcomeController">...</bean>   
当你访问http://\*\*\*\*\*\*/welcome.htm这个URL时，DispatcherServlet通过BeanNameUrlHandlerMapping映射就找到了WelcomeController。   
2. ControllerClassNameHandlerMapping，它是按控制器类名称映射请求。   
3. SimpleUrlHandlerMapping，用定制的映射定义来映射请求。   
  
  
下面我们就来看一下Controller。   
   
  
Controller接口是Spring MVC中所有控制器类的基本接口。通过实现这个接口，你可以创建自己的控制器。在handleRequest()方法中，可以随意地处理Web请求。   
  
AbstractController:如果你想让控制器具备一些基本的特性，如过滤受支持的HTTP方法（GET,POST和HEAD）,以及在HTTP响应中生成cache-control头部等，你可以让它扩展AbstractController类。   
  
ParameterizableViewController:用来创建带有参数化视图的控制器。   
  
SimpleFormController:它支持命令对象的概念（commandName），并且可以将表单域的值绑定到命令对象的同名属性上。   
  
AbstractWizardFormController:为向导表单的处理定义了基本的任务。向导表单有多个单页，因此必须为向导表单控制器定义多个页面视图。然后这个控制器可以跨越所有这些表单页面来管理表单状态。向导表单会有多个动作，不像SimpleFormController只有单个提交动作。AbstractWizardFormController会根据特殊的请求参数决定用户的动作，通常是用提交按钮的名称来指定动作的。   
\_finished: 完成向导表单。   
\_cancel: 取消向导表单。   
\_targetx: 进入目标页面，这里的x是从0开始的页面索引。   
  
MultiActionController：允许你将多个相关的动作分组到一个控制器中。   
  
常见的视图类型：   
   
  
几种常见的ViewResolver的解析：   
1. InternalResourceViewResolver: 根据URL解析视图。通过添加前缀和后缀的方法，将每个视图名称都映射到一个URL上。   
2. XmlViewResolver: 从XML配置文件中解析视图。将视图声明成Spring的Bean，并按他们的Bean Name进行解析。   
3. ResourceBundleViewResolver: 从ResourceBundle中解析视图。   
4. 用多个视图解析器解析视图, 需要注意的是，您需要为你配置的视图解析器配置解析的优先级。<property name="order" value="0"/> value越小，优先级越高。

为什么要使用Spring

　　AOP 让开发人员可以创建非行为性的关注点，称为横切关注点，并将它们插入到应用程序代码中。使用 AOP后，公共服务（比如日志、持久性、事务等）就可以分解成方面并应用到域对象上，同时不会增加域对象的对象模型的复杂性。   
  
　　IOC 允许创建一个可以构造对象的应用环境，然后向这些对象传递它们的协作对象。正如单词 倒置 所表明的，IOC 就像反过来的JNDI。没有使用一堆抽象工厂、服务定位器、单元素（singleton）和直接构造（straightconstruction），每一个对象都是用其协作对象构造的。因此是由容器管理协作对象（collaborator）。   
  
　　Spring即使一个AOP框架，也是一IOC容器。 Spring 最好的地方是它有助于您替换对象。有了Spring，只要用JavaBean属性和配置文件加入依赖性（协作对象）。然后可以很容易地在需要时替换具有类似接口的协作对象。   
  
　　Spring 框架是一个分层架构，由 7 个定义良好的模块组成。Spring模块构建在核心容器之上，核心容器定义了创建、配置和管理bean 的方式，如图 1 所示。   
  
　　组成 Spring 框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。每个模块的功能如下：   
  
　　核心容器：核心容器提供 Spring框架的基本功能。核心容器的主要组件是BeanFactory，它是工厂模式的实现。BeanFactory使用控制反转（IOC）模式将应用程序的配置和依赖性规范与实际的应用程序代码分开。   
  
　　Spring 上下文：Spring 上下文是一个配置文件，向 Spring框架提供上下文信息。Spring上下文包括企业服务，例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。   
  
　　Spring AOP：通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了Spring框架中。所以，可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于Spring的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖EJB组件，就可以将声明性事务管理集成到应用程序中。   
  
　　Spring  DAO：JDBCDAO抽象层提供了有意义的异常层次结构，可用该结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理，并且极大地降低了需要编写的异常代码数量（例如打开和关闭连接）。SpringDAO的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。   
  
　　Spring ORM：Spring 框架插入了若干个 ORM 框架，从而提供了 ORM的对象关系工具，其中包括JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和DAO异常层次结构。   
  
　　Spring Web 模块：Web 上下文模块建立在应用程序上下文模块之上，为基于Web的应用程序提供了上下文。所以，Spring 框架支持与 Jakarta Struts的集成。Web模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。   
  
　　Spring MVC 框架：MVC 框架是一个全功能的构建 Web 应用程序的 MVC实现。通过策略接口，MVC框架变成为高度可配置的，MVC 容纳了大量视图技术，其中包括JSP、Velocity、Tiles、iText 和 POI。 

　　Spring 框架的功能可以用在任何 J2EE服务器中，大多数功能也适用于不受管理的环境。Spring的核心要点是：支持不绑定到特定 J2EE服务的可重用业务和数据访问对象。毫无疑问，这样的对象可以在不同 J2EE 环境 （Web或EJB）、独立应用程序、测试环境之间重用。