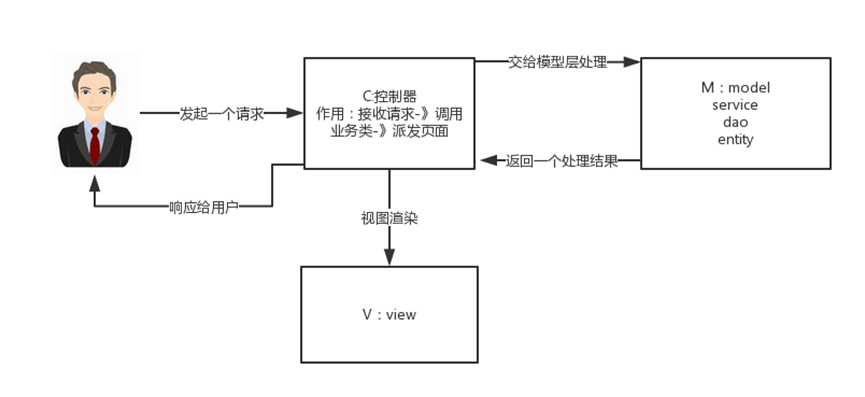
SpringMVC工作原理

1. SpringMVC流程
2. 框架内各部分组件说明
3. 各个组件的工作机制
4. 核心架构的具体流程
5. MVC模式 – MVC模式是一种设计模式

SpringMVC的深度探险 http://www.iteye.com/blogs/subjects/springmvc-explore

# MVC模式



## MVC模式分析

1. M-Model 模型（完成业务逻辑：有javaBean构成，service+dao+entity）
2. V-View 视图（做界面的展示： jsp，html……）
3. C-Controller 控制器（接收请求—>调用模型—>根据结果派发页面）

## 特点

1. 框架本身并不通过某种手段来干预或者控制浏览器发送Http请求的行为方式。服务器后端程序和前端通过HTTP请求进行交互。
2. 页面（View层）和请求处理类（Controller）之间的映射关系通过某一种配置形式维系起来。在SpringMVC中，使用了Annotation注解；在Struts2中，默认采取XML配置文件。不过无论是哪一种配置形式，隐藏在其背后的都是对于请求映射关系的定义。
3. Controller层的设计差异是不同MVC框架之间最主要的差异。SpringMVC使用方法参数来对请求的数据进行映射；而Struts2使用Controller类内部的属性来进行数据请求的映射。
4. 交互关系。在MVC模型中，浏览器端和服务器端的交互关系非常明确：无论采取什么样的框架，总是以一个明确的URL作为中心，辅之以参数请求。在后端可以接收到完整的HTTP请求，关键就是处理HTTP请求。

# 二、 SpringMVC框架介绍

## 出现背景

1. SpringMVC是一个MVC的开源框架,SpringMVC就相当于是Struts2加上Spring的整合。
2. SpringMVC是Spring的一个后续产品，就是在原有基础上，又提供了web应用的MVC模块，可以简单的把SpringMVC理解为是Spring的一个模块（类似AOP，IOC这样的模块）,与之相关的包有spring-web和spring-webmvc。
3. SpringMVC和Spring无缝集成的意思是SpringMVC就是Spring的一个子模块，所以根本不需要同Spring进行整合。

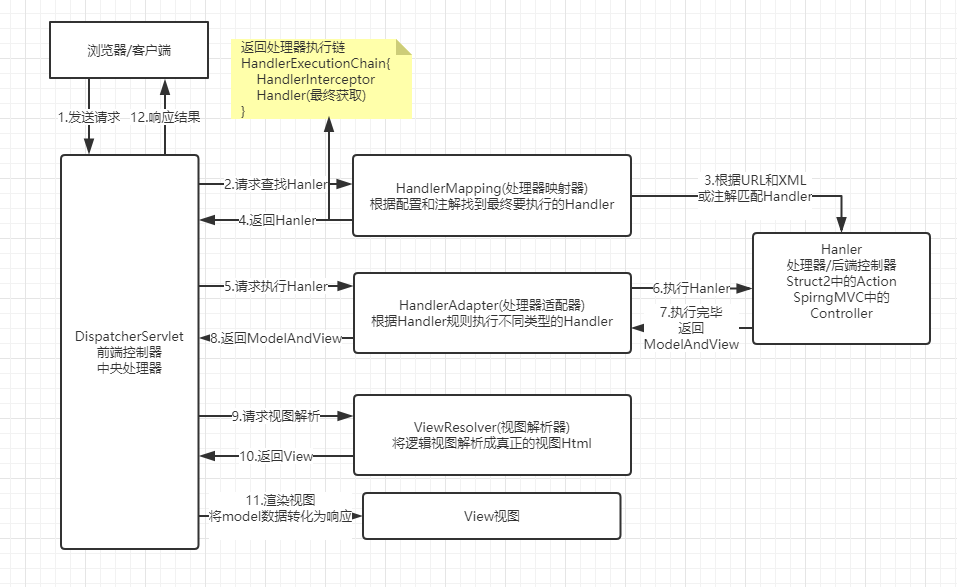
## 框架解决的问题

1. 任何一个框架，都有自己特定的适用领域，框架的设计和实现，必定是为了应付该领域内许多通用的，烦琐的、基础的工作而生。SpringMVC作为一个表现层框架，主要是解决Web开发领域中表现层中的三大问题。

URL到框架的映射、Http请求参数绑定、Http响应的生成和输出

1. 学习一个框架，首要的是要先领会它的设计思想。从抽象、从全局上来审视这个框架。其中最具有参考价值的，就是这个框架所定义的核心接口。**核心接口定义了框架的骨架，也在最抽象的意义上表达了框架的设计思想。**

# 三、SpringMVC基本工作原理



## DispatcherServlet

1. SpringMVC工作原理
2. SpringMVC流程

（DI调用处理器映射器 – 以便找到处理器）

1. 用户发送请求到**前端控制器**DispatcherServlet。

* 从web.xml中获取DispatcherServlet的名称，默认地址在Web-INF下面，默认名称为：[name]-servlet.xml

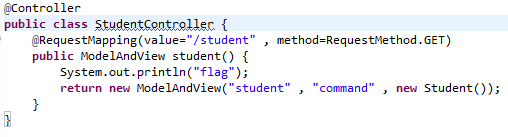
1. DispatcherServlet收到请求调用**处理器映射器**HandlerMapping。
2. 处理器映射器找到具体的**处理器**(可以根据xml配置、注解进行查找)，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

（DI调用处理器适配器 – 以便调用处理器，返回一个ModelAndView）

1. DispatcherServlet调用HandlerAdapter**处理器适配器**。
2. HandlerAdapter经过适配调用具体的**处理器**(**Controller**，也叫后端控制器)。

* 调用具体的java类，使用@Controller表明控制器，使用@RequestMapping表明控制器的访问地址和访问Method，紧跟的方法为访问地址所调用的方法。

1. Controller执行完成返回**ModelAndView**。



1. HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet。

（DI将ModelAndView传给视图解析器，返回一个view）

1. DispatcherServlet将ModelAndView传给ViewReslover**视图解析器**。
2. ViewReslover解析后返回具体View。

（DI渲染视图，响应用户跳到view视图）

1. DispatcherServlet根据Model获取参数数据，进行**渲染视图**（即将模型数据填充至视图中）生成渲染后的view。
2. DispatcherServlet响应用户，返回控制权给DispatcherServlet，最后再把view视图展示给用户。
3. 框架内各部分组件说明

以下组件通常使用框架提供实现：

* 1. DispatcherServlet：作为前端控制器，**整个流程控制的中心**，控制其它组件执行，统一调度，降低组件之间的耦合性，提高每个组件的扩展性。
  2. HandlerMapping：通过扩展处理器映射器实现不同的**映射方式**，例如：配置文件方式，实现接口方式，注解方式等。
  3. HandlAdapter：通过扩展处理器适配器，支持更多类型的处理器。
  4. ViewResolver：通过扩展视图解析器，**支持更多类型的视图解析**，例如：jsp、freemarker、pdf、excel等。

1. 各个组件的工作机制
2. 前端控制器DispatcherServlet（不需要工程师开发）,由框架提供

* 作用：接收请求，响应结果，相当于转发器，中央处理器。有了dispatcherServlet减少了其它组件之间的耦合度。
* 用户请求到达前端控制器，它就相当于mvc模式中的c，dispatcherServlet是整个流程控制的中心，由它调用其它组件处理用户的请求，dispatcherServlet的存在降低了组件之间的耦合性。

1. 处理器映射器HandlerMapping(不需要工程师开发),由框架提供

* 作用：根据请求的url查找Handler
* HandlerMapping负责根据用户请求找到Handler即处理器，springmvc提供了不同的映射器实现不同的映射方式，例如：**配置文件方式，实现接口方式，注解方式**等。

1. 处理器适配器HandlerAdapter

* 作用：按照特定规则（HandlerAdapter要求的规则）去执行Handler
* 通过HandlerAdapter对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理器进行执行。

1. 处理器Handler(需要工程师开发) -- 后端控制器

* 注意：编写Handler时按照HandlerAdapter的要求去做，这样适配器才可以去正确执行Handler
* 作用：Handler 是继DispatcherServlet前端控制器的后端控制器，在DispatcherServlet的控制下Handler对具体的用户请求进行处理。

由于Handler涉及到具体的用户业务请求，所以一般情况需要工程师根据业务需求开发Handler。

1. 视图解析器View resolver(不需要工程师开发)，由框架提供

* 作用：进行视图解析，根据逻辑视图名解析成真正的视图（view）
* View Resolver负责将处理结果生成View视图，View Resolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象，最后对View进行渲染将处理结果通过页面展示给用户。 springmvc框架提供了很多的View视图类型，包括：jstlView、freemarkerView、pdfView等。

一般情况下需要通过页面标签或页面模版技术将模型数据通过页面展示给用户，需要由工程师根据业务需求开发具体的页面。

1. 视图View(需要工程师开发jsp...)

* View是一个接口，实现类支持不同的View类型（jsp、freemarker、pdf...）

1. 需要开发的部分：

Handler：处理器，即后端控制器用controller表示。

View：视图，即展示给用户的界面，视图中通常需要标签语言展示模型数据。

1. 核心架构的具体流程
2. 首先用户发送请求——>DispatcherServlet，前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点，进行全局的流程控制。
3. DispatcherServlet——>HandlerMapping， HandlerMapping 将会把请求映射为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，通过这种策略模式，很容易添加新的映射策略；
4. DispatcherServlet——>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；
5. HandlerAdapter——>处理器功能处理方法的调用，HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理；并返回一个ModelAndView 对象（包含模型数据、逻辑视图名）；
6. ModelAndView的逻辑视图名——> 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；
7. View——>渲染，View会根据传进来的Model模型数据进行渲染，此处的Model实际是一个Map数据结构，因此很容易支持其他视图技术；
8. 返回控制权给DispatcherServlet，由DispatcherServlet返回响应给用户，到此一个流程结束。