spring mvc 原理深度解析

学习一个框架的阶段

1阶段：学会对这个框架的使用
2除段：掌握这个框架的架构思想和它的层次结构
3阶段：掌握底层实现细节,(需要改造的时候在去深入研究)

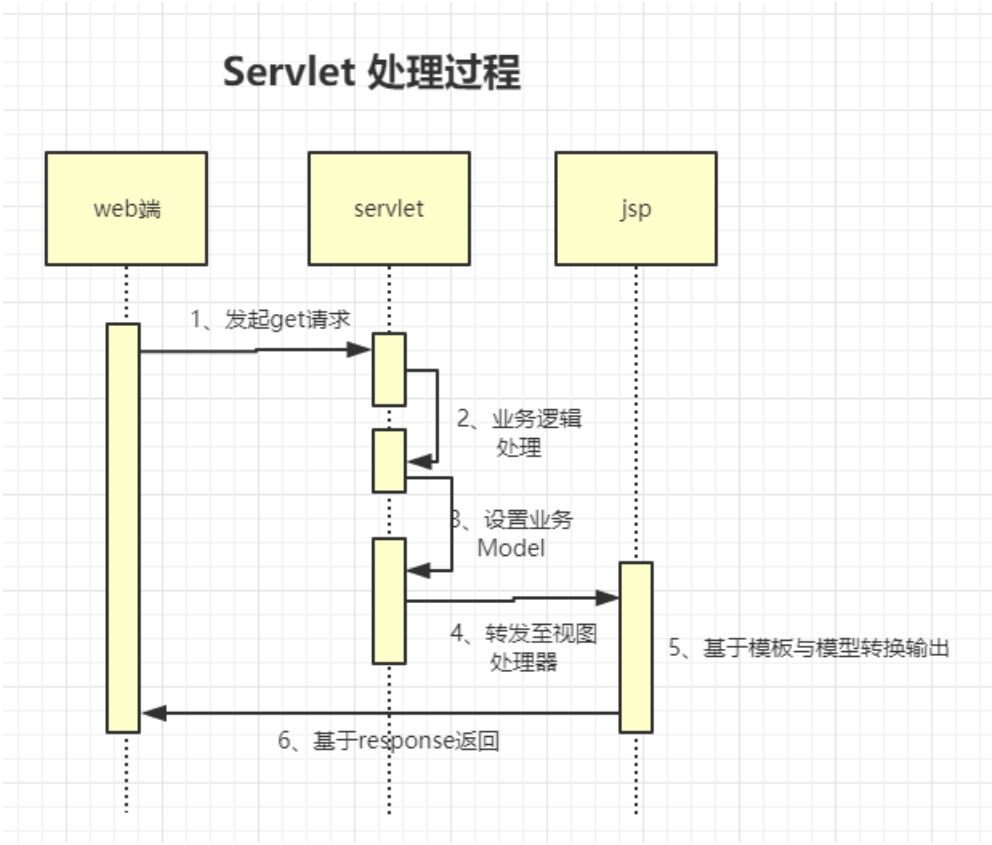
一、理论基础

课程概要

- mvc 理论基础与spring mvc 设计思想
- 从dispatchServlet 出发 讲述mvc 体系结构组成
- mvc 执行流程讲解

mvc 理论基础与spring mvc 设计思想

回顾servlet 与jsp 执行过程（10分钟）

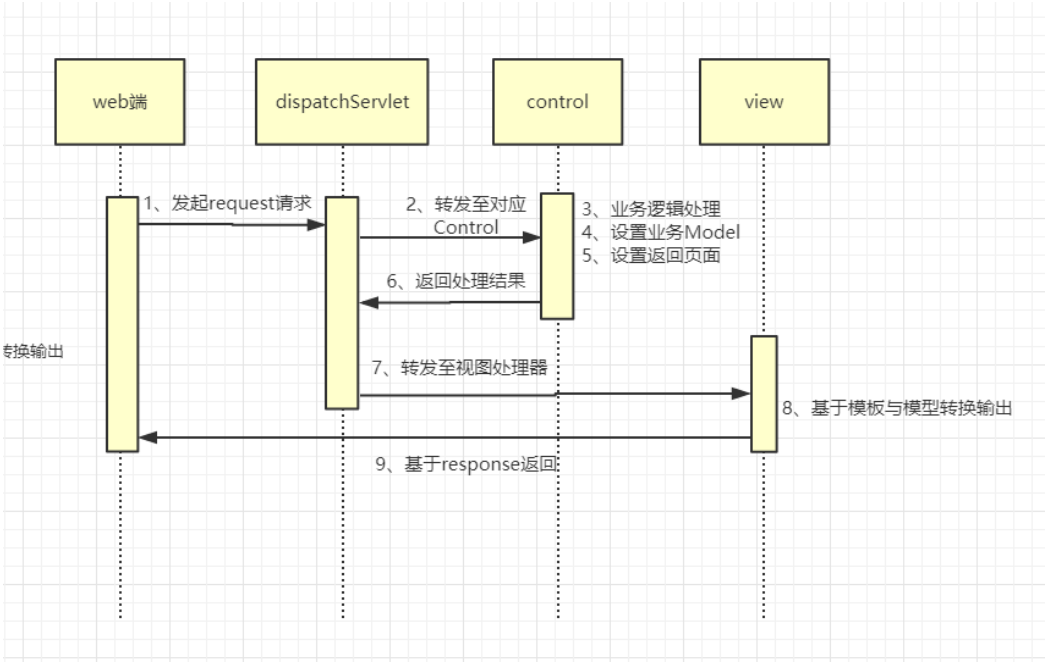


spring mvc本质上是对其进行封装简化了开发流程。易用性、程序逻辑结构更清晰。（5分钟）

a. 基于注解的URL映谢
b. http表单参数转换
c. 全局统一异常处理
d. 拦截器的实现
e. 多视图控制器

从dispatchServlet 出发 讲述mvc 体系结构组成

基于 DispatchServlet来设计一个mvc框架执行流程（画出流程图）（10分钟）



提出问题?

1、dispatchServlet 如何找到对应的Control?

2、如何执行调用Control 当中的业务方法?

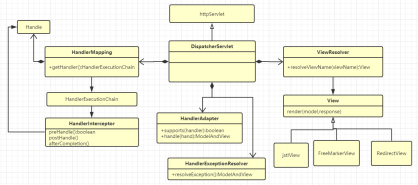
3、如何找到对应的视图进行处理?

4、dispatchServlet本身是如何初始化的?

3、配置一个spring mvc 示例演示 验证上述流程 。(20分钟)

- [x] ~~创建一个Controller 类~~
- [x] ~~配置DispatchServlet~~
- [x] ~~创建spring-mvc.xml 文件~~
- [x] ~~配置SimpleUrlHandlerMapping~~
- [x] ~~配置InternalResourceViewResolver~~

4、基于示例展开dispatchServlet 核心类结构。(30分钟)



1. HandlerMapping: url与控制器的映谢
2. HandlerAdapter: 控制器执行适配器
3. ViewResolver: 视图仓库
4. view: 具体解析视图
5. HandlerExceptionResolver:异常捕捕捉器

- [x] ~~拦截器演示(5分钟)~~
- [x] ~~异常配置演示(5分钟)~~

mvc 执行流程讲解

5、spring mvc 上下文初始过程（10）
webApplicationContext 初始过程
handlerMappings 初始过程
handlerExceptionResolvers 初始过程
viewResolvers 初始过程
handleAdapters 初始过程

6、request请求过程 （10）
1、调用doDispatch()
2、遍历handlerMappings 与request 获取一个执行链 getHandler()
3、遍历handleAdapters 与 handle 获取一个handle 适配器
4、通过执行链 去调用拦截器当中的 preHandle() 方法， 进行预处理。
5、基于handle 适配器 去调用handle 方法,返回 modelAndView
6、通过执行链 去调用拦截器当中的 PostHandle() 方法， 进行拦截处理。
7、processDispatchResult()
7.1 正常：调用render()进行视图解析
7.1.1 基于 遍历 viewResolvers 工与 viewname 获取View
7.1.2 调用view.render() 进行视图解析和返回,设置model 至request
7.2 异常：遍历handlerExceptionResolvers 调用resolveException(),返回mv,最后跳转至异常mv

二、配置与应用

课程概要

- spring mvc 上下文初始化过程
- 日常配置与应用
- requestMaping 实现原理

spring mvc 上下文初始化过程

MVC 上下文即xmlWebApplicationContext ， 依托DispatchServlet 的和contextConfigLocation进行创建和初始化,最后对mvc 进策略初始化。

1、创建DispatchServlet
2、initServletBean
3、createWebApplicationContext
3.1、createBeanFactory
3.2、loadBeanDefinitions (contextConfigLocation)
3.3、registerBeanDefinition
IOC 加载完比
4、initStrategies (初始化mvc 环境)
4.1、initHandlerMappings （初始化 映谢配置)
4.2、initHandlerAdapters(初始化适配器)
4.3、initHandlerExceptionResolvers
4.4、initViewResolvers

日常配置与应用

一、演示最简化的spring MVC 配置示例

- [x] ~~beanName 实现mvc 配置~~

二、演示最常用的spring mvc 配置示例

- [x] ~~注解方式实现mvc~~

三、提出问题？

1、使用了哪些handleMaping 与handleAdapter ?

2、简化的配置如何映谢的，如何调用的?

解决过程：

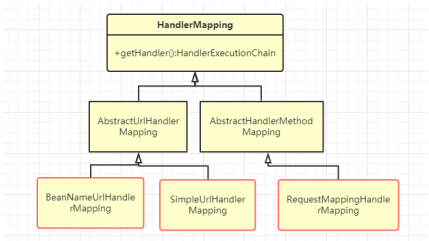
- [x] ~~调试initStrategies 方法，查看已初始化的 Maping 与Adapter~~
- [ ] 认识 <mvc:annotation-driven /> 原理
- [x] ~~认识 NamespaceHandler 接口~~
- [x] ~~查看 MvcNamespaceHandler~~
- [x] ~~查看AnnotationDrivenBeanDefinitionParser~~

结论：

在<mvc:annotation-driven /> 对应的解析器，自动向ioc 里面注册了两个BeanDefinition。分别是：RequestMappingHandlerMapping与BeanNameUrlHandlerMapping

四、MVC可支持的配置

HandlerMapping 实现类结构

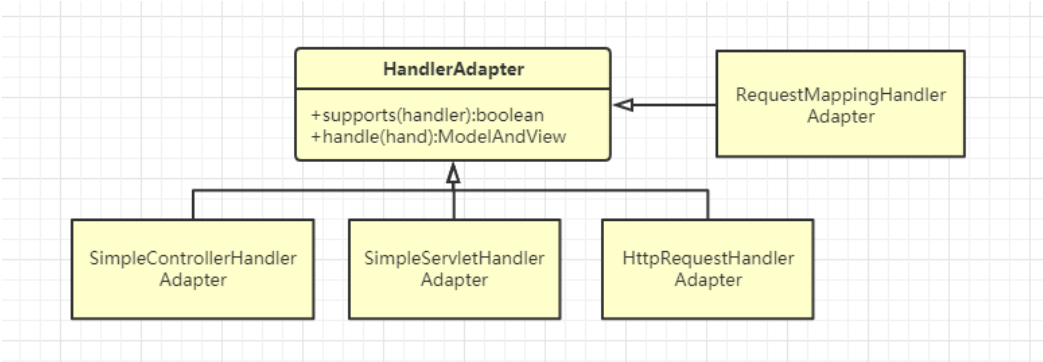


BeanNameUrlHandlerMapping: 基于ioc name 中已/开头的Bean时行 注册至映谢.

SimpleUrlHandlerMapping: 基于手动配置 url 与control 映谢

RequestMappingHandlerMapping: 基于注解方法配置对应映谢

HandlerAdapter 实现类结构图



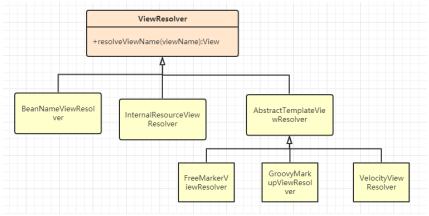
HttpRequestHandlerAdapter:

SimpleServletHandlerAdapter:

SimpleControllerHandlerAdapter:

RequestMappingHandlerAdapter:

ViewResolver 实现类图



BeanNameViewResolver:

InternalResourceViewResolver:

FreeMarkerViewResolver:

演示FreeMarkerViewResolver 配置

- 配置FreeMarkerConfig
- 配置FreeMarkerViewResolver

RequestMapping 实现原理

---

request解析

request执行