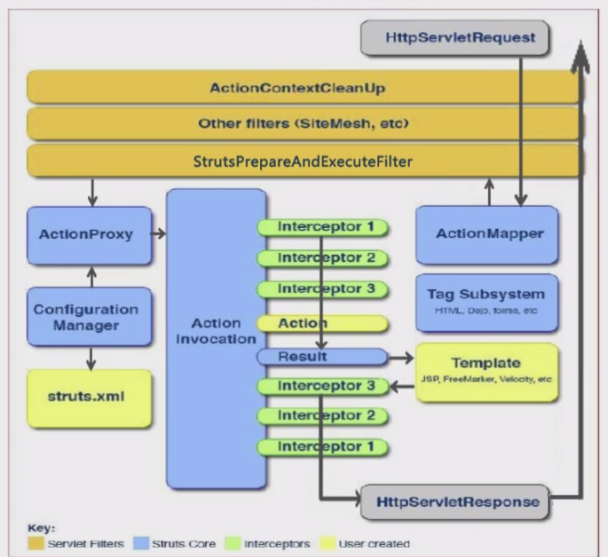
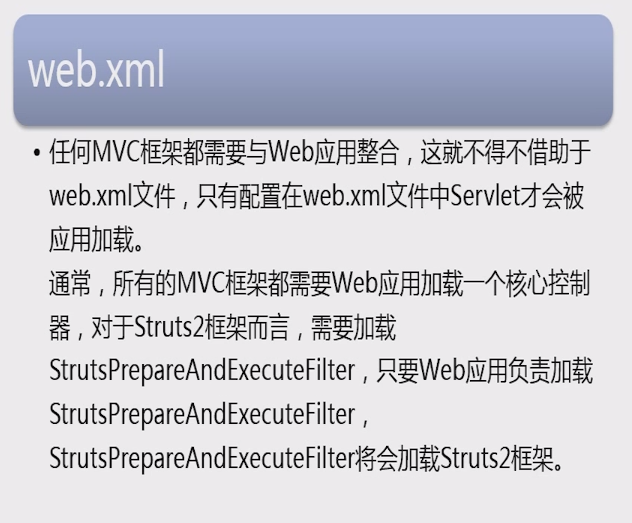
# 搭建环境步骤

1. 下载相应jarbao
2. 创建web项目
3. 创建完善相关配置文件
4. 创建Action并测试启动

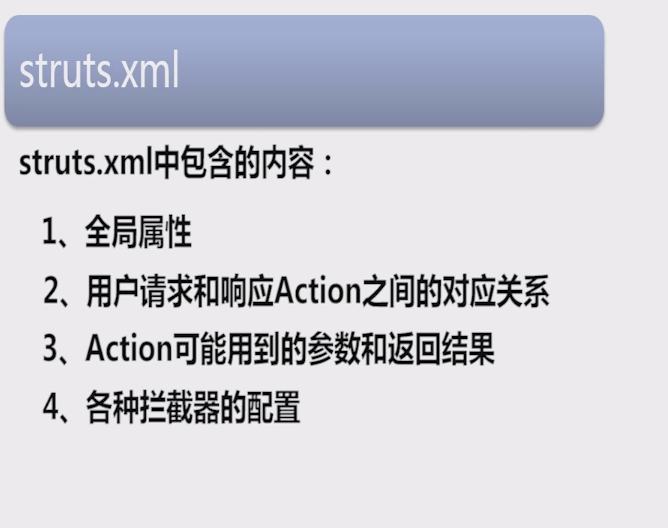
工作原理

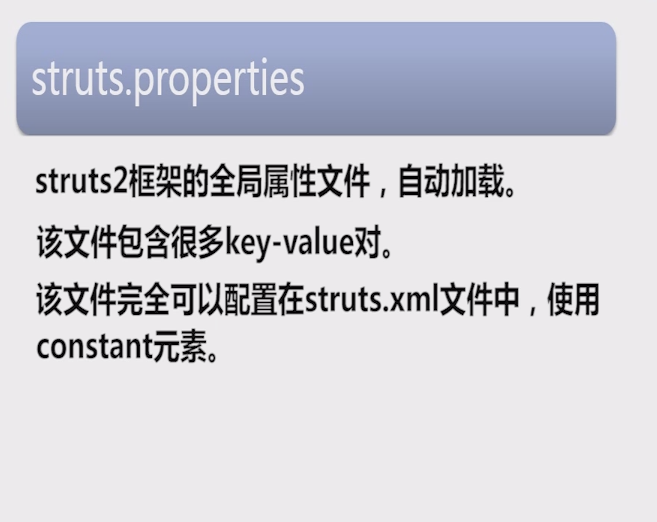


# 核心文件



负责Action文件的映射,该Action包含的Result定义.

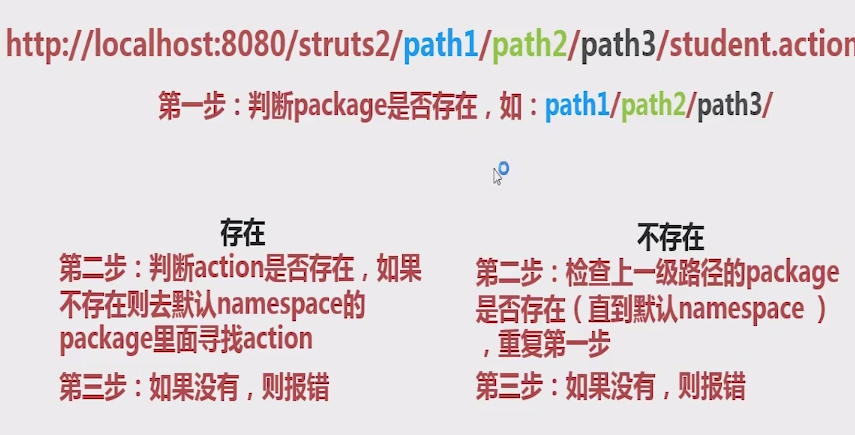




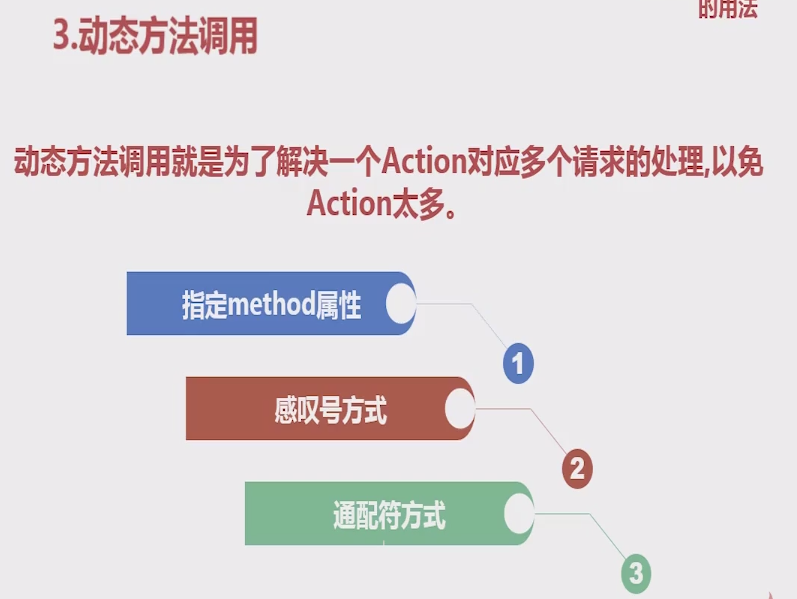
# 访问Servlet的API



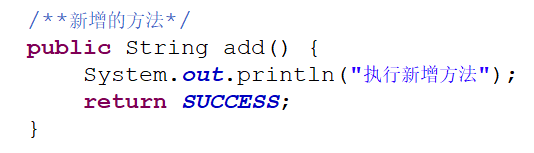
# Action的搜索顺序

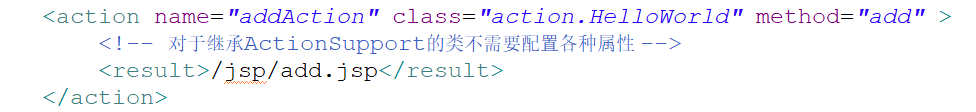


# 动态方法调用

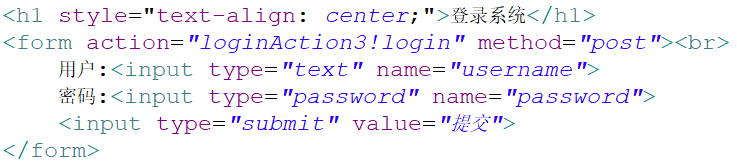


## 通过在Struts配置文件中写方法.

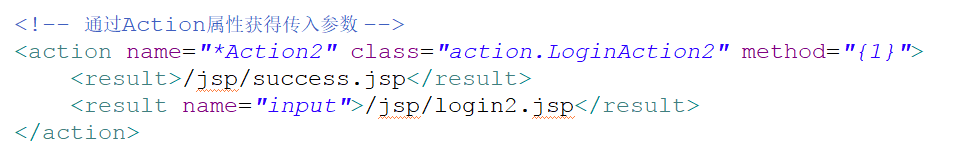




## 通过感叹号的方式直接调取



## 通配符方式调取



# 指定多个配置文件



# 默认Action

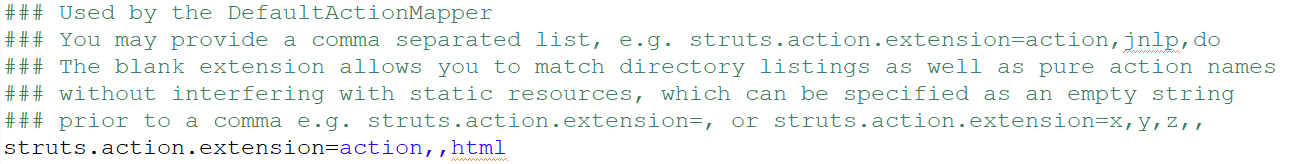


# 后缀修改

1. 可以在xml配置



1. 在struts.properties修改,逗号分隔,为空表示可以缺省后缀



1. 在struts.xml配置,不配置默认为.action或者缺省

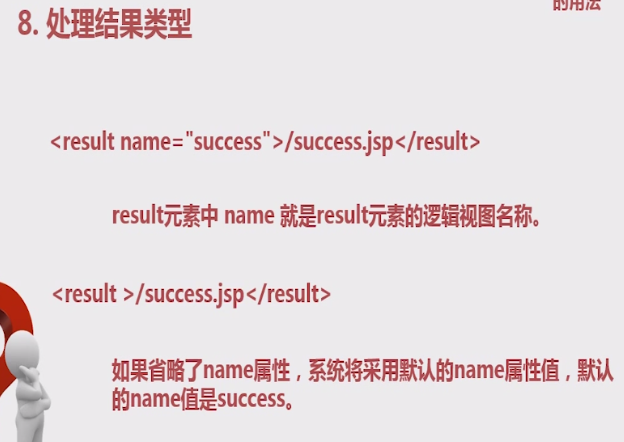


# 接收参数



# 处理结果类型





其他常量



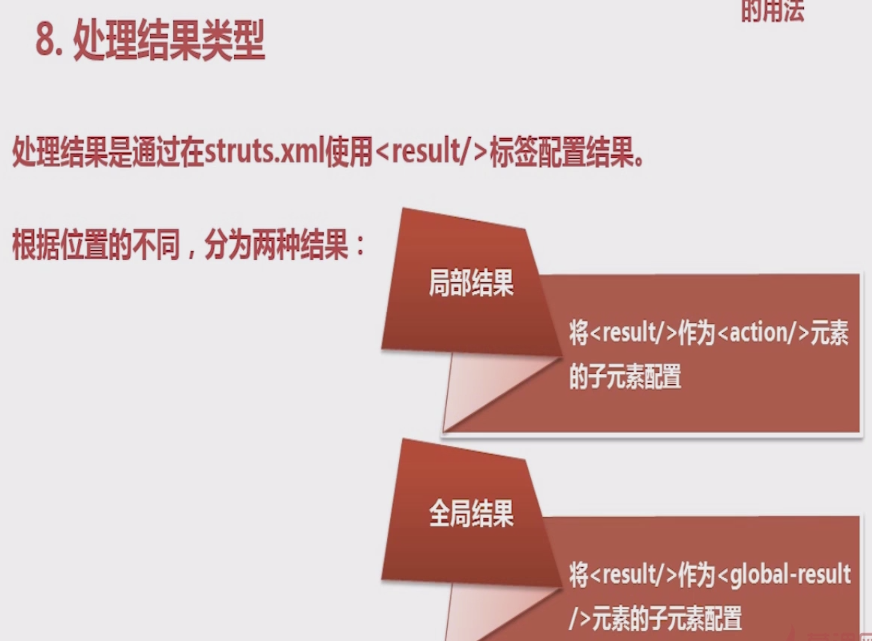
# 验证方式:

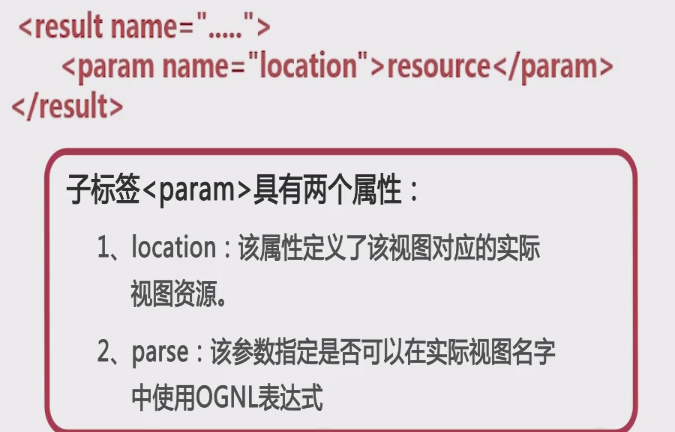
如果表单验证失败,或者对象类型不匹配,重定向到登录界面

验证有两种方式:

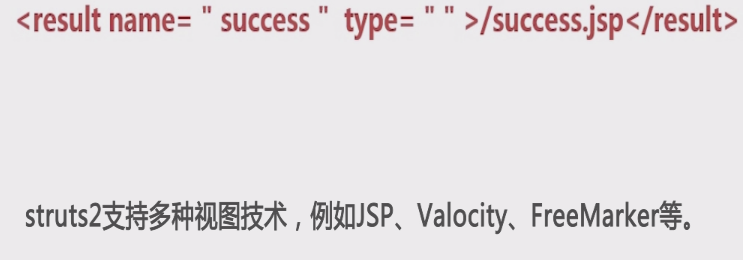
1. 在业务逻辑中验证,返回INPUT或者ACCESS
2. 继承方法,在validate()中验证

处理结果类型:





默认支持OGNL表达式



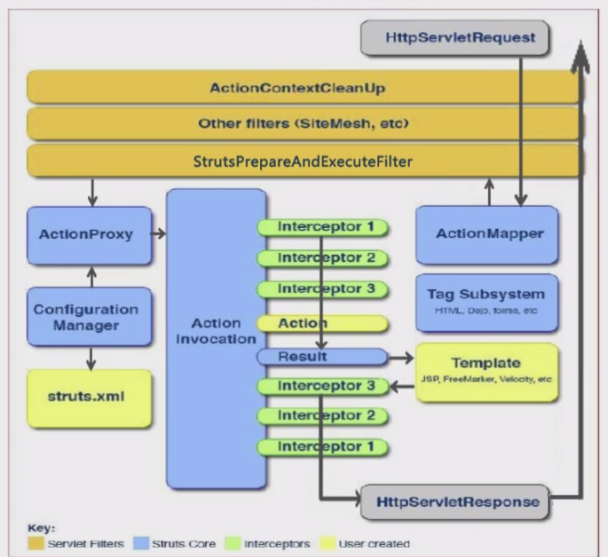
chain:将一个action和另个action链接起来

redirect:重定向,失去传参,并修改url

plainText:显示界面的原始代码

stream:上传下载时,返回一个数据流

# 拦截器工作原理



HttpServletRequest通过Struts2的各种过滤器,到达ActionMapper,判断是否要调用某一个action,如果请求调用的是html,css等静态资源,则不需要调用action;否则,将控制权转交给ActionProxy,代理通过ConfigurationManager(配置管理)读取Struts.xml,找到需要调用的Action,并创建ActionInvocation实例,实例中包含了Action和Result以及前后的Interceptor拦截器,依次执行拦截器后进入业务处理,并返回一个结果,经由Template来调用视图或者另外一个Action,随后反向执行拦截器,最终通过HttpServletResponse响应请求.

拦截器的定义

Struts2大多数功能是通过拦截器实现的,每个拦截器完成某项功能.

拦截器方法在Action执行之前或者之后执行.

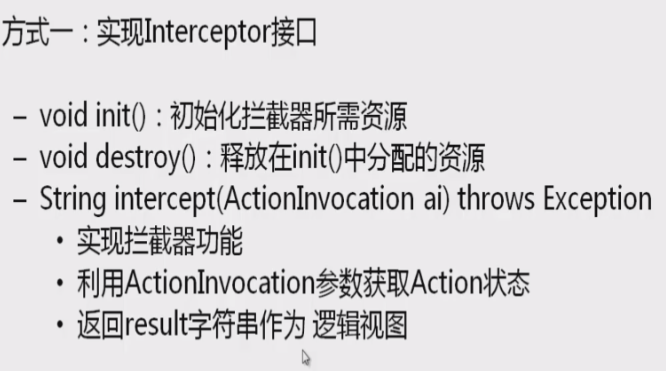
拦截器栈的概念

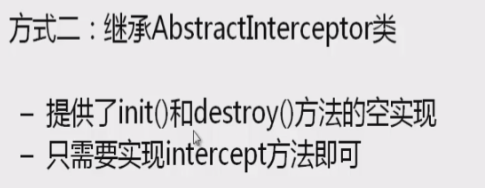
结构:拦截器栈相当于多个拦截器的组合

功能:拦截器栈也是拦截器-

工作过程:相当于递归的过程 拦截器1->拦截器2->execute->result->拦截器2->拦截器1

# 实现方式

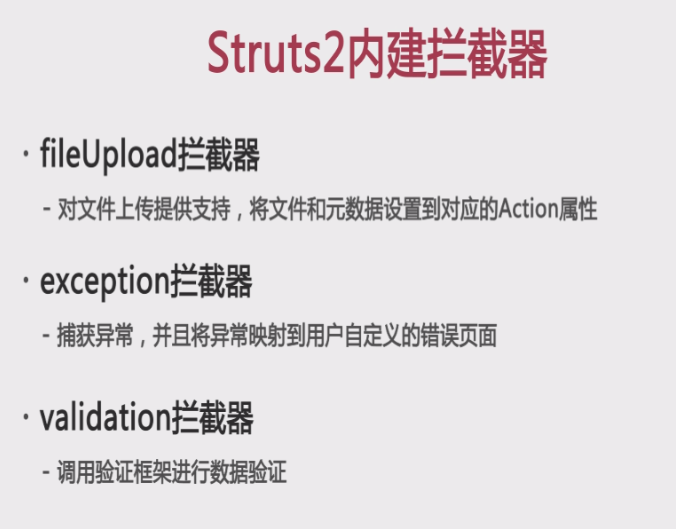
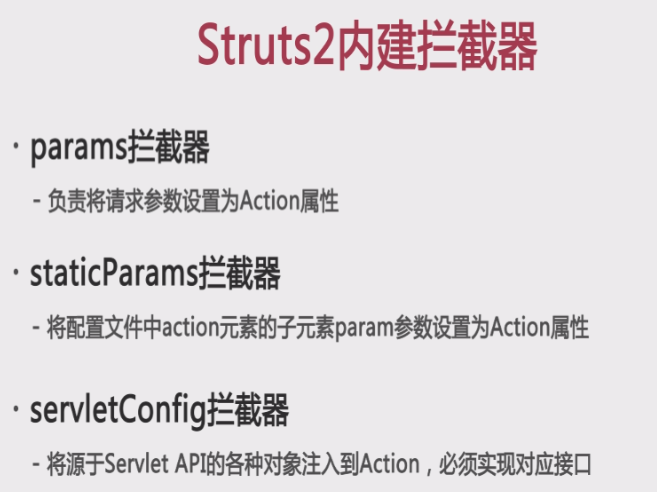




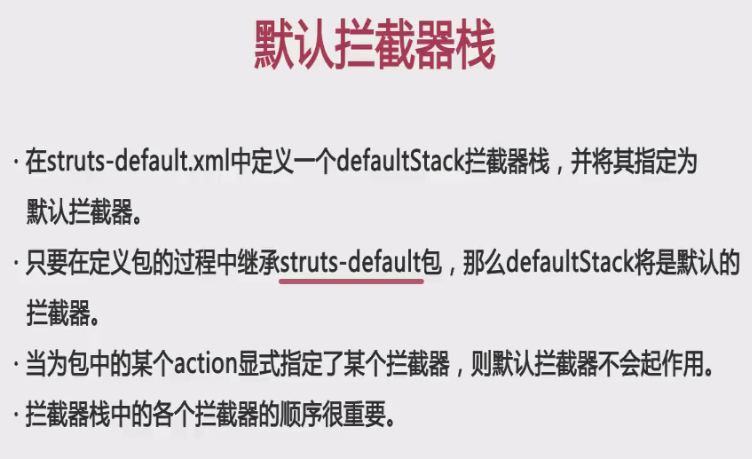
步骤:

1. 创建一个继承自AbstractInterceptor的类
2. 实现类的Intercept方法
3. Struts.xml中注册拦截器
4. 在相应的Action中进行引用

# 内置拦截器



# 默认拦截器的使用要求



# 自定义拦截器栈

