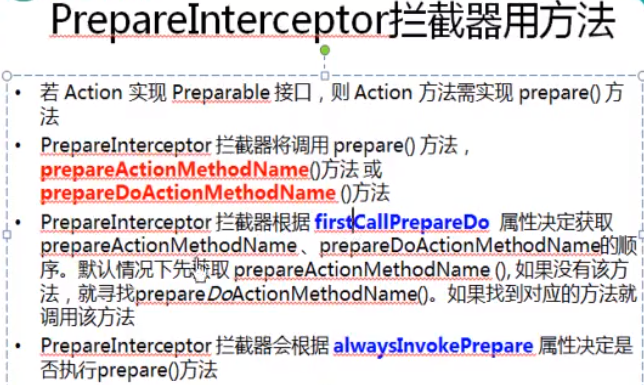
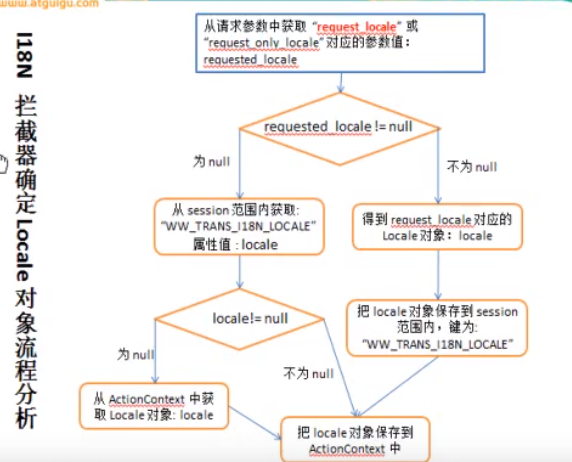
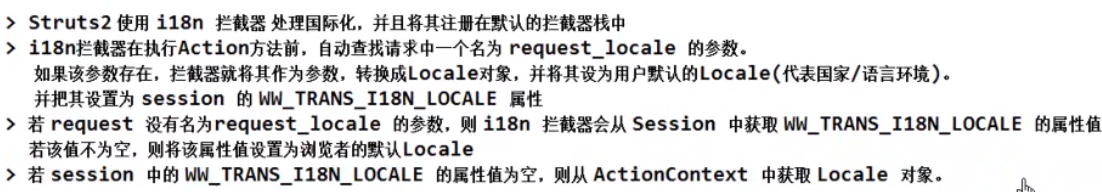
## Struts2

1. MVC(Model View Controller) 模式
   1. 模型：封装应用程序的数据和业务逻辑 pojo(Plain Old Java Object)
   2. 实现应用程序的信息显示功能 jsp
   3. 控制器：接收来自用户的输入，调用模型层，响应对应的视图组件 Servlet Filter
2. Servlet能做的filter都能做
3. Struts2是一个用来开发MVC应用程序的框架，它提供了Web应用程序开发过程中的一些常见问题的解决方案
   1. 对来自用户的输入数据进行合法性验证
   2. 统一的布局
   3. 可扩展性
   4. 国际化和本地化
   5. 支持ajax
   6. 表单的重复提交
   7. 文件的上传和下载
4. Struts2环境步骤：导包，配置web.xml，加上struts.xml文件
5. action 和 Action类
   1. action：代表一个Struts2请求
   2. Action：能够处理Struts2 请求的类
      1. 属性的名字必须遵守与JavaBean属性名相同的命名规则，属性的类型可以是任意类型，从字符串到非字符串（基本数据类型）之间的数据转换可以自动发生
      2. 必须有一个不带参的构造器：通过反射创建实例
      3. 至少有一个供struts在执行这个action时调用的方法
      4. 同一个Action类可以包含多个action方法
      5. Struts2为每一个HTTP请求创建一个新的Action 实例，即Action不是单例的，是线程安全的
6. 什么是WEB资源
   1. HttpServletRequest，HttpSession，ServletContext 等原生的Servlet API
7. 为什么访问WEB资源
   1. B\S应用的Controller 中必然需要访问WEB资源，向域中读写属性，读写cookie，获取realPath...
8. 如何访问
   1. 和servlet API 解耦的方式：只能访问有限的Servlet API对象，且只能访问其有限的方法（读取请求参数，读写域对象的属性，使session失效）
      1. 使用ActionContext
      2. 实现XxxAware接口
      3. 选用的建议：若一个Action类中有多个action方法，且多个方法都需要使用域对象的map 或 parameter，则建议使用aware接口的方式
      4. Session 对应的map实际上是SessionMap类型的，强转后调用其invalidate方法可以使session失效
   2. 和Servlet API 耦合的方式：可以访问更多的Servlet API对象，且可以调用其原生的方法
      1. 使用ServletActionContext
      2. 实现ServletXxxAware 接口
9. ActionSupport是默认的Action类：若某个action节点没有配置class属性，则ActionSupport为待执行的Action类，而excute方法则为默认要执行的方法
   1. 在手工完成字段验证，显示错误消息，国际化等情况下，推荐继承ActionSupport
10. Result
    1. result是action的子节点
    2. Result代表action方法执行后，可能去的一个目的地
    3. 一个action节点可以配置多个result子节点
    4. Result的name属性对应着action方法可能有的一个返回值
11. Result一共有连个属性:name,和type
12. Result的type属性值在struts-default 包的result-types 节点的name属性中定义
    1. Dispatcher（默认的）：转发，同servlet中的转发
    2. Redirect：重定向
    3. RedirectAction：重定向到一个action
    4. Chain：转发到一个action
13. OGNL(Object - Graph - Navigation - Language):对象-图导航语言访问到值栈（ValueStack）里的对象属性
14. 关于值栈
    1. ${productName} 读取productName值时，实际上该属性并不在request等域对象中，而是从值栈中获取的
    2. ValueStack：d
15. 可以从ActionContext中获取值栈对象
16. 值栈分为两个逻辑部分
    1. Map栈：实际上是OgnlContext类型，是个map，也是对ActionContext的一个引用，里边保存着各种map：requestMap，sessionMap，applicationMap，parametersMap，attr
    2. 对象栈：实际上是CompoundRoot 类型，是一个使用ArrayList定义的栈，里面保存着各种和当前Action实例相关的对象，是一个数据结构意义的栈
17. 读取对象栈中的数据：<s:property value=”[1].id”></s:property>
18. 读取map栈中的数据： <s:property value=”#session.product.id”></s:property>
19. AOP:面向切面编程
20. 通用标签
    1. Property ，Url ， set ，push ，iterator ，if-elseif-else ，
    2. Sort，date，a
21. %{ } ：强制进行OGNL解析
22. 表单标签
    1. 使用和html和form标签的感觉差不多
    2. Struts2的form标签会自动生成一个table，以进行自动的排版
    3. 可以对表单提交的值进行回显：从栈顶对象开始匹配属性，并把匹配的属性值赋到对应的标签的value中，若栈顶对象没有对应的属性，则依次向下找相对应的属性
23. Result 标签使用占位符时，最好这样写：”{1}-success”
24. 主题即：风格相近的模板被打包为一个主题
25. Action类必须要有无参构造方法
26. ActionProxy是Action的一个代理类，也就是说Action的调用是通过ActionProxy实现的，其实就是调用了ActionProxy.execute()方法，而该方法又调用了ActionInvocation.invoke()方法
27. ActionInvocation就是Action的调用者。ActionInvocation在Action的执行过程中，负责Interceptor，Action和Result等一系列元素的调度
28. ModelDriven拦截器
    1. 当用户触发add请求时，ModelDriven拦截器将被调用，EmployeeAction对象的getModel方法，并把返回的模型（Employee实例）压入到ValueStack栈
    2. 接下来Parameters拦截器将表单字段映射到ValueStack栈的栈顶对象的各个属性中，因为当时ValueStack栈的栈顶元素是刚被压入模型（Employee）对象，所以该模型将被填充，如果某个对象在模型中没有匹配的属性，Param拦截器将尝试ValueStack栈中的下一个对象
29. ognl.OgnlException: It isn't a simple method which can be called!
    1. 将请求的后缀去掉就不出错了即:emp-add.do 改为：emp-add
30. PrepareInterceptor拦截器方法

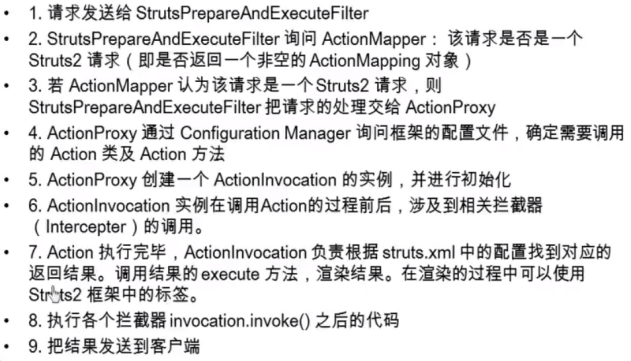
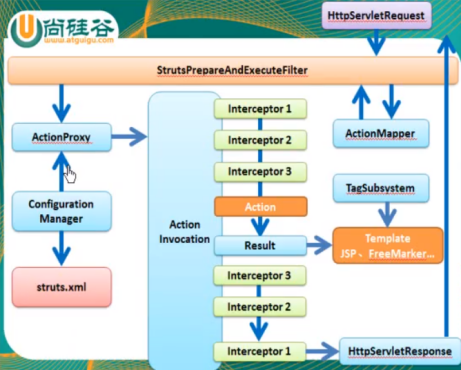


1. 拦截器拦截的是一个Action请求时的操作方法，每次请求都会将所用拦截器中的所有拦截器都调用执行一遍，是否实现了某个拦截器的接口，然后判断，如果实现就会执行拦截器中相对应的方法。
2. 如何自定义类型转换器
   1. 为什么需要自定义类型转换器：因为Struts 不能自动完成字符串到 引用类型的转换
   2. 如何定义类型转换器：
      1. 开发类型转换器的类：扩展StrutsTypeConverter类
      2. 配置类型转换器：
         1. 基于字段的配置
            1. 在所在的Model(可能是一个Action，可能是一个javaBean)的包下，新建一个ModelClassName-conversion.properties
            2. 在文件中输入键值对：fieldName=类型转换器的全类名
            3. 第一次使用类型转换器时创建实例
            4. 类型转换器是单例的
         2. 基于类型的配置
            1. 在src下新建xwork-conversion.properties
            2. 键入：待转换的类型=类型转换器的全类名
            3. 在当前Struts2应用被加载时创建实例
3. 如果i18n出现乱码将文件右下角的CRLF改为CR
4. I18n拦截器示意图

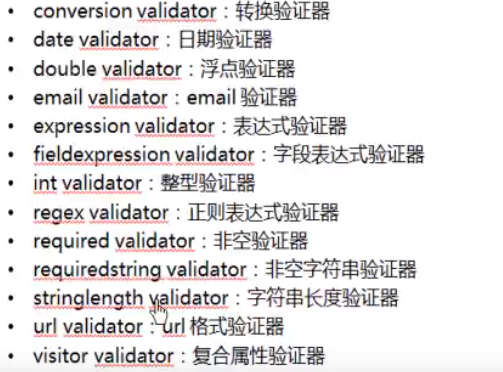




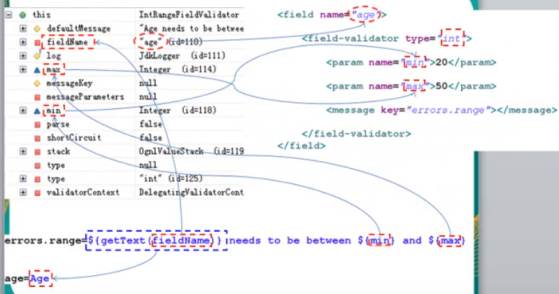
1. Struts2运行流程



1. Struts2内建的验证规则



1. 若一个Action类可以应答多个action请求，多个action请求需要不同的验证规则，怎么办
   1. 为每一个不同的action请求定义其对应的验证文件：ActionClassName-AliasName-validation.xml
   2. 不带别名的配置文件：ActionClassName-validation.xml 中的验证规则依然会发生作用，可以把各个action共有的验证规则配置在其中，但需要注意的是，只适用于某一个action的请求的验证规则就不要在这里配置了
2. 结果反复不对重启服务器
3. 多个验证使用同一个错误消息i18n



1. 自定义验证器
   1. 定义一个验证器的类
      1. 自定义的验证器需要实现Validator接口
      2. 可以选择继承ValidatorSupport 或 FieldValidatorSupport类
      3. 若希望实现一个一般的验证器，则可以继承ValidatorSupport
      4. 若希望实现一个字段验证器，则可以继承FieldValidatorSupport
   2. 在配置文件中配置验证器
   3. 使用
2. 编程式验证需要实现Validateable接口，而ActionSupport类已经实现了
3. 基本的文件上传：直接在Action中定义如下三个属性，并提供getter和setter
   1. 文件对应的File对象：private File [fileFieldName];
   2. 文件类型：private String[fileFieldName]ContentType;
   3. 文件名：private String[fileFieldName]FileName;
4. 限制文件的上传通过配置FileUploadInterceptor拦截器的方式进行限制
   1. MaximumSize : 默认的最大值为2m，上传的单个文件的最大值
   2. AllowedTypes: 允许的上传文件的类型，多个使用 , 分割
   3. AllowedExtensions: 允许的文件扩展名 ，多个使用 , 分割
5. 定制错误消息，可以在国际化文件中定义如下消息：
   1. Struts.messages.error.uploading: 文件上传出错的消息
   2. Struts.messages.error.file.too.large: 文件超过最大值消息
   3. Struts.messages.error.content.type.not.allowed: 文件内容类型不合法的消息
   4. Struts.messages.error.file.extension.not.allowed: 文件文件扩展名不合法的消息
6. 文件下载可以为stream的result指定如下参数
   1. contentType：结果类型
   2. contentLength：下载的文件的长度
   3. contentDisposition:设定content-disposition 响应头该响应头指定接应是一个文件下载类型，一般取值为attachment;filename=”document.pdf”
   4. inputName:指定文件输入流的getter定义的那个属性的名字，默认为inputStream
   5. bufferSize：缓存区的大小，默认为1024
   6. Allowcaching：是否允许使用缓存
   7. contentCharSet：指定下载的字符集
7. 表单重复提交:在不刷新表单页面的情况下
   1. 多次点击提交按钮
   2. 已经提交成功，按回退之后，再点击提交按钮
   3. 在控制器响应页面的形式为转发的情况下，若已经提交成功然后使用f5功能键
8. 不是表单重复提交的情况
   1. 若刷新表单页面，再提交表单不算重复提交
   2. 若使用的是redirect 的响应类型，已经提交成功后，再点击“刷新”，不是表单的重复提交
9. Token 和 tokenSession拦截器
   1. 这两个拦截器均不在默认的拦截器栈中，需要手工配置
   2. 若使用token拦截器，则需要配置一个token.valid 的result
   3. 若使用tokenSession拦截器，则不需要配置result
   4. 使用token拦截器会转到token.valid 这个result
   5. 使用tokenSession拦截器则还会响应那个目标页面，但不会执行目标的action方法，不会执行后续的拦截器
10. 拦截器执行的顺序
    1. Interceptor1->2->3...->Action -> ...3->2->1
11. 定义一个拦截器的类
    1. 可以实现interceptor接口
    2. 继承abstractInterceptor抽象类