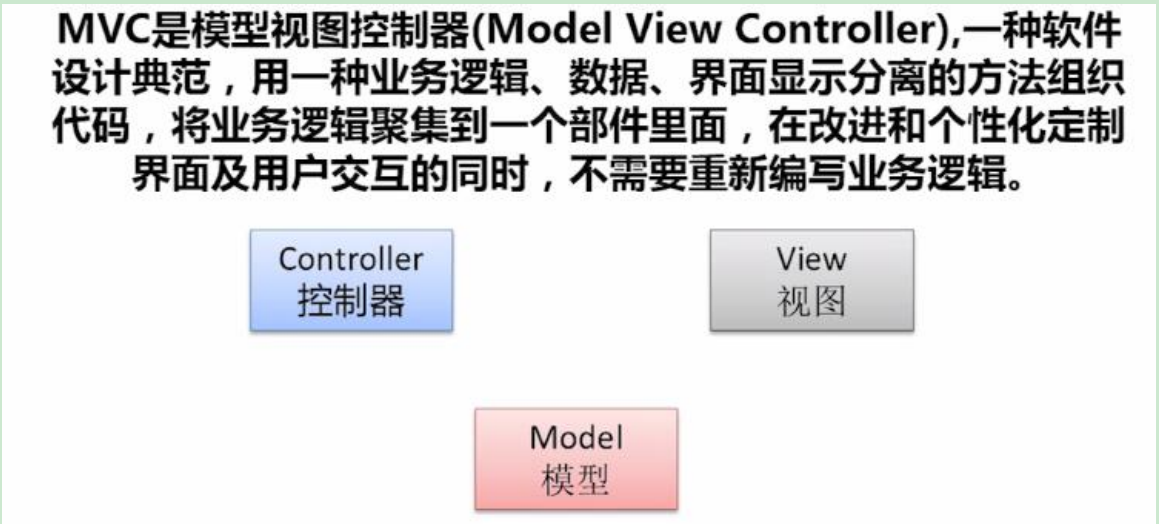
Struts2

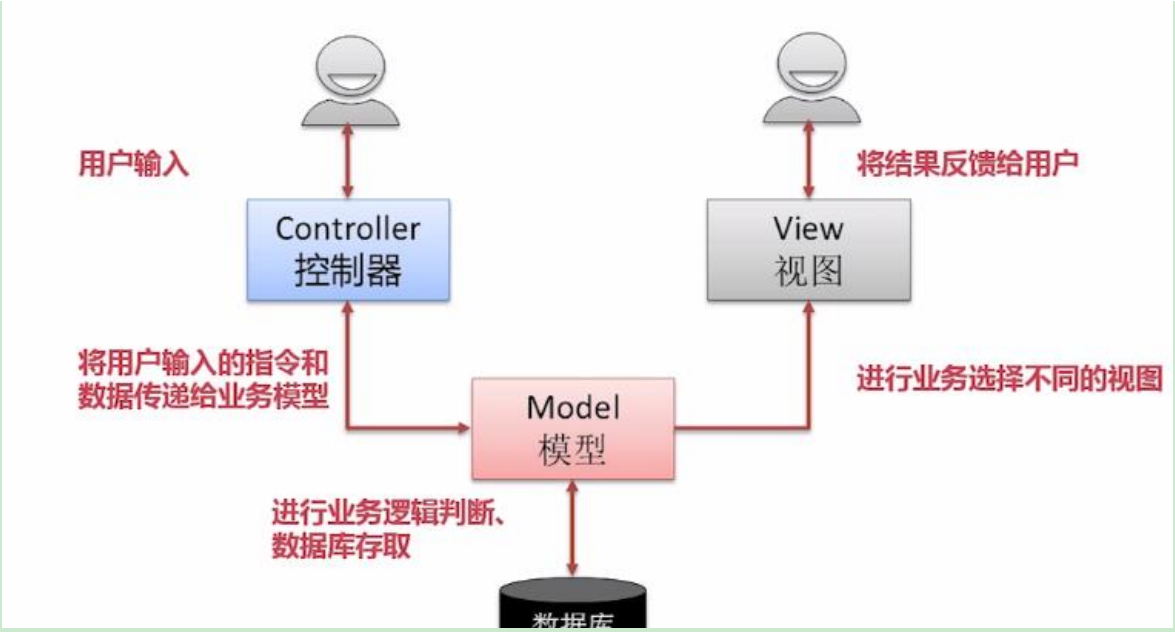
Strusts2是java程序员必学的一项课程
Strusts英文单词是什么意思呢？
翻译： 支柱，支干，来源于建筑和旧式飞机使用的金属支架

Strusts在软件开发中，是一个非常优秀的框架 是关于MVC的框架

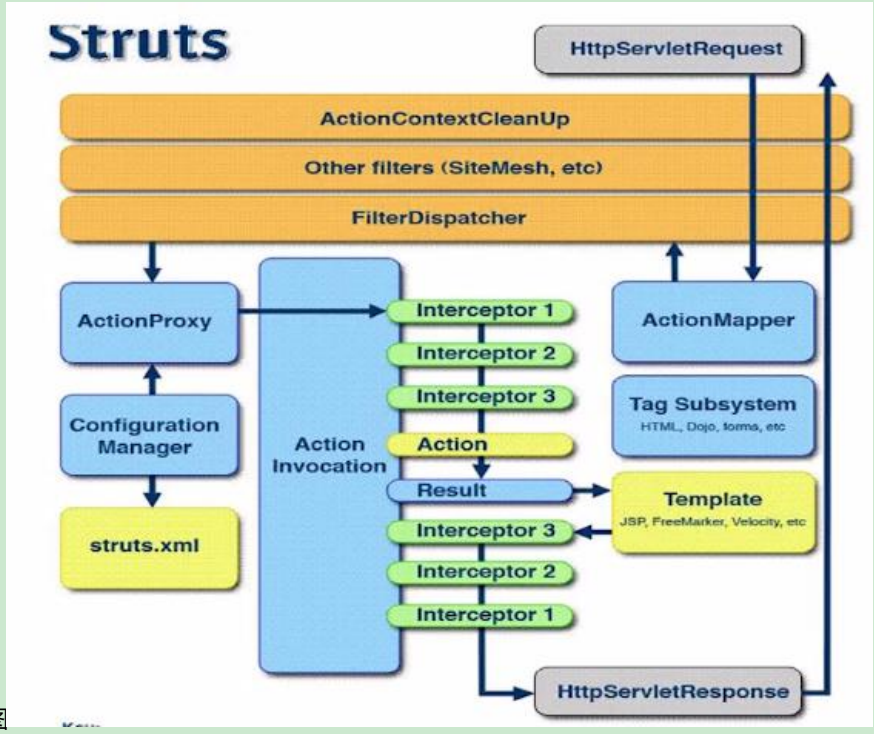
Strusts是流行的与成熟的基于MVC设计模式的Web应用程序设计框架
使用Struts的目的是为了帮助我们减少在运用MVC设计模式来开发web应用的时间

JSP+JAVABEAN =Model1
代码可维护性差
JSP+SERVLET+JAVABEAN= Model2
最典型的MVC





Struts2 不是一个全新的框架，因此稳定性。性能等各个方面都有很好的保证，同时吸收了Struts1余WebWork俩者的优势



Struts2工作原理图

1.客户端初始化一个指向servlet容器的请求。

2.请求经过一系列的过滤器（ActionContextCleanUp、SiteMesh）

3.FilterDispatcher被调用，并询问ActionMapper来决定这个请求是否需要调用某个Action

4.ActionMapper决定要调用那一个Action,FilterDispatcher把请求交给ActionProxy。

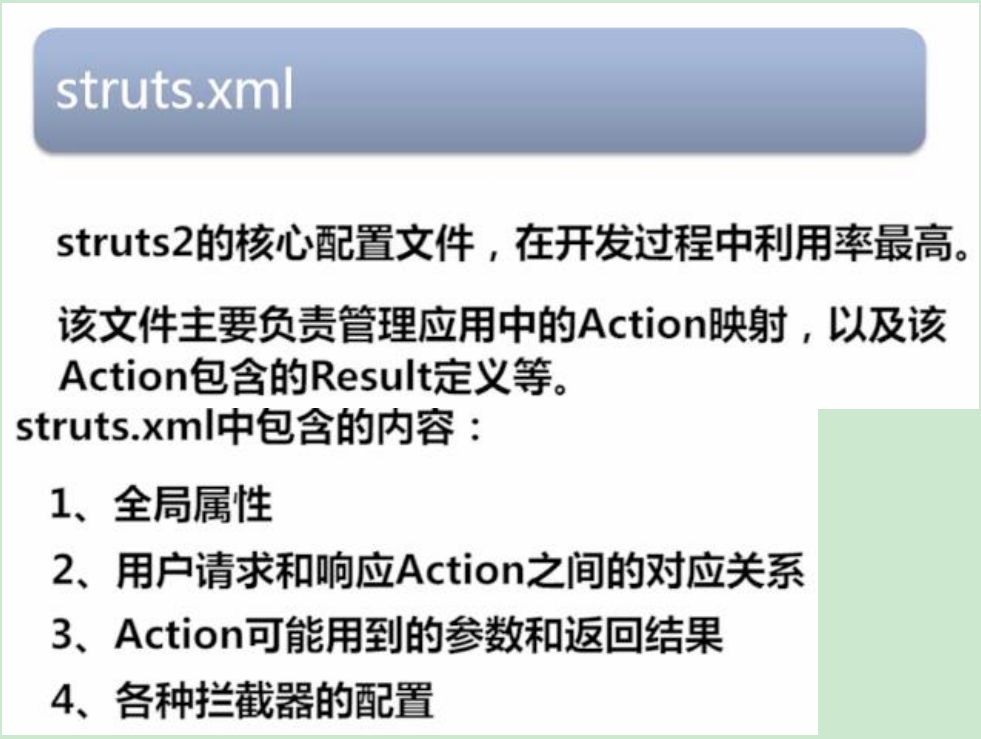
5. ActionProxy通过Configurate Manager询问Struts配置文件，找到要调用的Action类

6. ActionProxy创建一个ActionInvocation实例

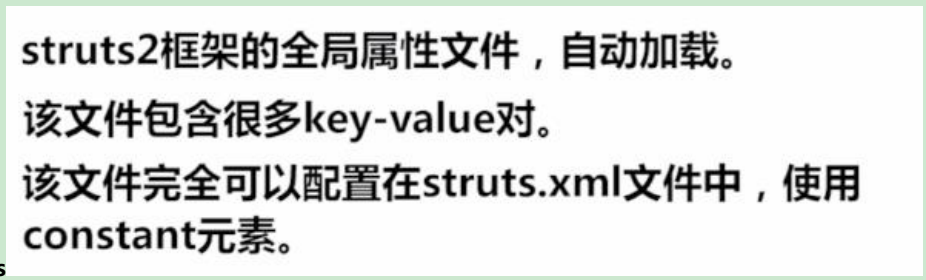
7. ActionInvocation实例使用命令模式来调用，回调Action的exeute方法

8. 一旦Action执行完毕，ActionInvocation负责根据Struts.xml的配置返回结果。

我们需要关键的配置的几个文件
struts2.xml



web.xml



struts.properties

解析**struts.xml**

解析struts.xml——（上）：
1、.dtd就是约束struts.xml中可以有哪些标签不能有哪些标签。
2、<include>
<!-- 可以通过 <include file="">包含其他文件，include节点是struts2中组件化的方式，可以将每个功能模块独立到一个xml配置文件中，然后用Include节点引用 -->
<include file="struts-default.xml"></include>

3、<package>
package提供了将多个Action组织成为一个模块的方式。
package的名字必须是唯一的（配置文件中可以有多个包，但包名要唯一），可以在这个包上加一些拓展的包。
abstract设置package的属性为抽象，抽象的package不能定义action的值，ture或false。
namespace="/test"：http://localhost:8080/项目名/test/xx.action
namespace="/"：http://localhost:8080/项目名/xx.action
<package name="包名" extends="继承的父类的名称" namespace="包的命名空间">
4、拦截器
<interceptors>
<!-- 拦截器定义name（名称）和class（类路径） -->
<interceptor name="" class=""></interceptor>
<!-- 定义拦截器栈 -->
<interceptor-stack name="">
<interceptor-ref name=""></interceptor-ref>
</interceptor-stack>
</interceptors>
<!-- 定义默认的拦截器，每个Action都会自动引用。如果Action中引用了其他的拦截器，默认的拦截器将无效 -->
<default-interceptor-ref name=""></default-interceptor-ref>
5、<global-results>
<!-- 全局results配置 -->
<global-results>
<result name="">/xxx.jsp</result>
</global-results>

解析struts.xml——（下）：
6、<action>
<!--
Action配置：一个Action可以被多次映射（只要action配置中的name不同）
name：action名称，如（假设namespace="/"）：http://localhost:8080/项目名/<action>中属性name值.action
class：对应的类的路径
method：调用Action中的方法名
Ps：每个package包里可以定义多个action
-->\]

<action name="" class="" method="" converter="">
<!-- 引用拦截器：name：拦截器名称或拦截器栈名称 -->
<interceptor-ref name=""></interceptor-ref>

<!-- 节点配置：name：result名称，和Action中返回的值相同（如果不写name，默认SUCCESS或"success"）；
type：result类型，不写则选用superpackage的type struts-default.xml中的默认为dispatcher -->
<result name="" type="">/xxx.jsp</result>

<!-- 参数设置：name：对应Action中的get/set方法 -->
<param name="">值</param>
</action>
7、<constant>
<!-- struts.properties文件中的语句：struts.i18n.reload=true，则对应struts.xml文件中语法如下 -->
<constant name="struts.i18n.reload" value="true"></constant>

这是下面学习的大纲

大纲：

1 访问Servlet API
2 Action搜索顺序
3 动态方法调用
4 指定多个配置文件
5 默认Action
6 Struts2后缀

Servlet API 关于访问Servlet的API

在**servlet**中可以直接调用
HttpServletRequest
HttpServletResponse

在**Struts2**中的**execute**方法中，是不存在

HttpServletRequest
HttpServletResponse

action的搜索顺序

存在的情况：
第二步：如果存在的话，判断action是否存在，如果不存在则去默认namespace的package里面寻找action 如果没有则报错
不存在的情况：
第二步：检查上一级路径的package是否重复（直到默认namespace），重复第一步，如果没有，则报错
建议实际的情况来写，不要多写包名

动态方法调用
动态方法调用是就是为了解决一个action对应多个请求的处理，以免action太多

有三种方式
1 指定method属性：

```
<action name="addAction" method="add" class="com.imooc.action.HelloWorldAction">
    <result >/add.jsp</result>
</action>
```

2 感叹号方式
3 通配符方式

感叹号方式
需要设置动态调用

```
<!-- 支持动态调用 -->
<constant name="struts.enable.DynamicMethodInvocation" value="true"/>
```

```
<result name="update">/update.jsp</result>
```

struts.devMode也就是struts的开发模式，默认值为false，这里修改为true就可以了，以后一旦就该这个文件中的配置就不用去重启tomcat，着实方便许多

```
<constant name="struts.devMode" value="true"/>
```

通配符方式 ：

```
<action name="*_*"  method="{2}" class="hello.{1}Action">
    <result name="success">/index.jsp</result>
    <result name="update">/{2}.jsp</result>

</action>
```

指定多个配置文件：
当action太多的时候，已经无法管理的时候
，我们需要指定多个配置文件

```
<include file="login.xml"></include>
<include file="system.xml"></include>
```

设置编码格式：

```
<constant name="struts.i18n.encoding" value="UTF-8"></constant>
```

默认Action：当别人访问action的时候，找不到action的话，由默认action 指定的action来显示

```
<default-action-ref name="index"></default-action-ref>
 <action name="index">
     <result>/error.jsp</result>
 </action>
```

注意：使用通配符的时候，与默认action冲突，所以输入错误的字符不会自动跳转

修改**Struts2**后缀 后缀就是在浏览器打开的时候的**Struts2**的文件后缀名
这个是可以在**struts2.xml**中修改的

```
<constant name="struts.action.extension" value="html"></constant>
```

还有**struts.propertis**中配置

```
12 struts.action.extension=action,do,struts2,
```

当然也可以在**web.xml**中修改，不常用

```
<filter>
     <filter-name>struts</filter-name>
     <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFi
lter</filter-class>
      <init-param>
          <param-name>struts.action.extension</param-name>
          <param-value>do</param-value>
      </init-param>
</filter>
```

关于从jsp页面form标签接受参数的三种方法：
1 使用Action的属性接受参数
2 使用Domain Model接受参数
3 使用ModelDriven接受参数

1 .直接提交的时候就传过来了，直接在java这边的类接受过来了，只需要getset一下，就能获取到值（这是第一种方式）
2 第二种方式也很简单，是通过放在一个对象中来实现的，

```
User user;

 public User getUser() {
return user;
}

public void setUser(User user) {
this.user = user;
}


public String login(){
     System.out.println(user.getUsername() );
```

```
//....
```

只不过在传递的时候，就要设置好

```
<form action="LoginAction.action" method="post">
用户名：<input type="text" name="user.username">
密码:<input type="password" name="user.password">
    <input type="submit" value="提交">
```

第三种方式需要实现一个接口 ModelDriven

```
public class LoginAction extends ActionSupport  implements ModelDriven<User>{

private User user = new User();


    public String login(){
        System.out.println(user.getUsername());
//....
return SUCCESS;
}

public User getModel() {
return user;
}
}
```

而传递表单的时候不需要再user.password这样了

s

Struts接收参数：

////这里是用的是别人的笔记
用form表单的方式：
1.传到表单的xx.action
2.创建java action类（继承actionsupport，获取execute方法），getset方法封装数据，里面写方法，返回success
private List<String> booklist; //传递数组！
书籍：<input type="text" name="booklist[0]">
//获取值：request.getXXX(“booklist[]”)
Private String username; //传递普通值 然后getset
3.配置Struts文件
关于input：
<result name="input">/login.jsp</result>
功能：当在表单中输入错误值得类型时，跳转到login.jsp页面。

如果提交表单的时候是没有正确匹配的值的时候，列如，

```
年龄:  <input type="text" name="age">
```

这个age在user中对应的是int类型
我们输入的时候变成了其他类型，**转换出错，就会导致，发出的是input**

```
<action name="LoginAction" method="login" class="hello.LoginAction">
    <result>/success.jsp</result>
    <result name="input">/login.jsp</result>
</action>
```

name=input就接受到了 ，跳转到login.jsp页面 ，**说明的就是表单验证不正确**

INPUT信息显示

关于这里的/struts-tags
这个是一个struts标签 这里是引入一个struts标签

```
<%@taglib prefix="s" uri="/struts-tags" %>
```

```
<s:fielderrorname="username"></s:fielderror>
```

还有另外一种方法也是提交input（俩种方法）这俩个方法是为了返回一个INPUT的值
一个是要用到addFieldError这个方法 这个是需要return INPUT
一个是使用继承父类的validate方法的 这个不需要return Input来进行返回信息 因为内部处理好了关于input的返回

```
if(user.getUsername()==null||"".equals(user.getUsername())){
this.addFieldError("username","用户名为空");
    return  INPUT;
}
```

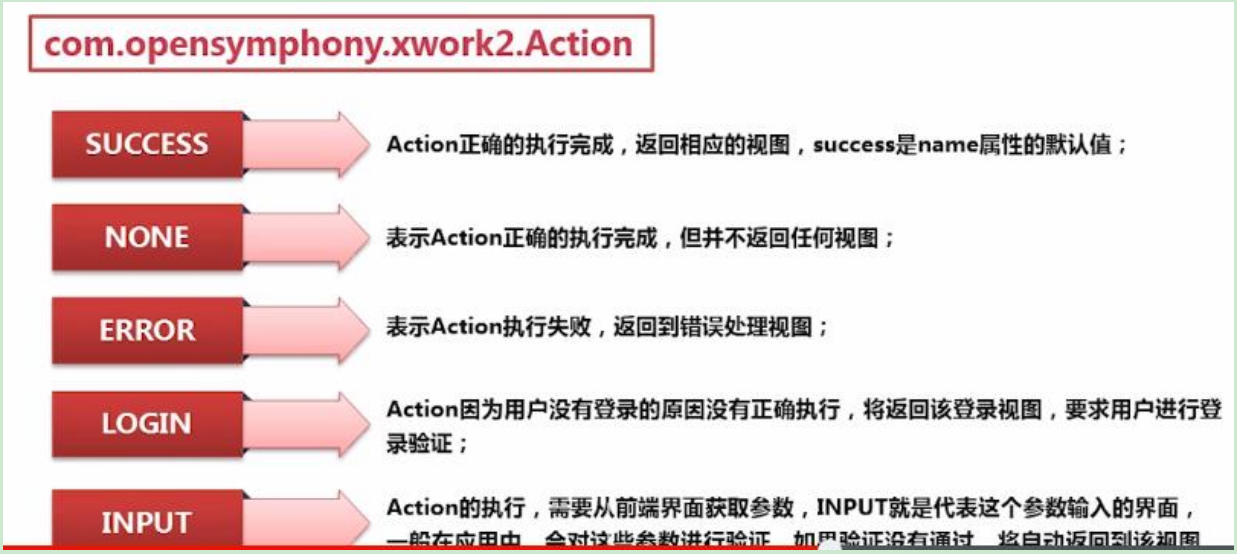
```
@Override
public void validate() {
if(user.getUsername()==null||"".equals(user.getUsername())){
this.addFieldError("username","用户名为空");

}
}
```

关于处理结果类型：
这是一个内部的实现步骤：
用户请求--struts框架--控制器（action）--struts框架---视图资源


```
<action name="addAction" method="add" class="hello.HelloAction">
    <result name="success">/add.jsp</result>
</action>
```

action的具体参数说明
这里的斜杠表示的是绝对路径，上下文路径 如果不指定斜杠的话，就是指定的是当前namespace的指定路径
result元素中的name就是result元素的逻辑视图名称
如果省略了name属性，系统将采用默认的name属性，就是success

在action这个系统对象中



前面的属性是可以随便用（意思就是我们自己也可以定义的），只是struts2定义好了这些常量,我们可以自定义用法。只有最后一个是有特殊性的 就是input。

处理结果类型：
处理结果是通过struts2.xml使用<result/>标签配置结果
根据位置的不同，分为俩种结果
局部结果：将<result/>作为<action/>元素的子元素配置
全局结果：将<result/>作为<global-result/>元素的子配置

<result/>子标签M<param>具有俩个属性
1 location：该属性定义了该视图对应的实际视图资源
2 parse：该参数指定是否可以在实际视图名字中使用OGNL表达式 默认是true，就是支持OGNL表达式

OGNL表达式简单示例
requ.setAttribute("path","add");
${#request.path}.jsp

```
public    String add(){
```

```
request.setAttribute("path","add");

    return  SUCCESS;
}
```

```
<action name="addAction" method="add" class="hello.HelloAction">
    <resultname="success">
        <param name="location">/${#request.path}.jsp</param>
    </result>
</action>
```

关于result元素的属性

result 元素有个type属性 type的默认值为dispatcher，这个类型支持JSP视图

struts2支持多种视图技术，列如JSP，Valocity,FreeMarker等

Result还有一个最最重要的属性type，struts2支持错种视图技术，列如JSP，Valocity，FreeMarker等，当我们没有定义type的时候，默认值为dispatcher，这个类型是支持JSP视图技术

这是支持的type属性：

```
  <result-types>
      <result-type name="chain" class="com.opensymphony.xwork2.ActionChainResul
      <result-type name="dispatcher" class="org.apache.struts2.dispatcher.Servl
      <result-type name="freemarker" class="org.apache.struts2.views.freemarker
      <result-type name="httpheader" class="org.apache.struts2.dispatcher.HttpH
      <result-type name="redirect" class="org.apache.struts2.dispatcher.Servlet
      <result-type name="redirectAction" class="org.apache.struts2.dispatcher.S
      <result-type name="stream" class="org.apache.struts2.dispatcher.StreamRes
      <result-type name="velocity" class="org.apache.struts2.dispatcher.Velocit
      <result-type name="xslt" class="org.apache.struts2.views.xslt.XSLTResult"
      <result-type name="plainText" class="org.apache.struts2.dispatcher.PlainT
      <result-type name="postback" class="org.apache.struts2.dispatcher.Postbac
  </result-types>
```

常用三个：chain，redirect，plaintext。

1、chain：将action和另外一个action链接起来。

2、redirect：重定向（会丢失请求参数）。

3、plaintext：返回网页源代码。

4、stream：返回inputstream用于文件下载。