**Struts:基于mvc模式的应用框架（控制层）**

(就是对servlet封装，提高开发效率；请求的跳转)

使用方法：

1. web项目，引入struts-jar包

8个jar文件。

commons-fileupload-1.2.2.jar 【文件上传相关包】

commons-io-2.0.1.jar

struts2-core-2.3.4.1.jar 【struts2核心功能包】

xwork-core-2.3.4.1.jar 【Xwork核心包】

ognl-3.0.5.jar 【Ognl表达式功能支持表】

commons-lang3-3.1.jar 【struts对java.lang包的扩展】

freemarker-2.3.19.jar 【struts的标签模板库jar文件】

javassist-3.11.0.GA.jar 【struts对字节码的处理相关jar】

1. web.xml中，引入struts核心功能

配置过滤器

1. 开发action
2. 配置action

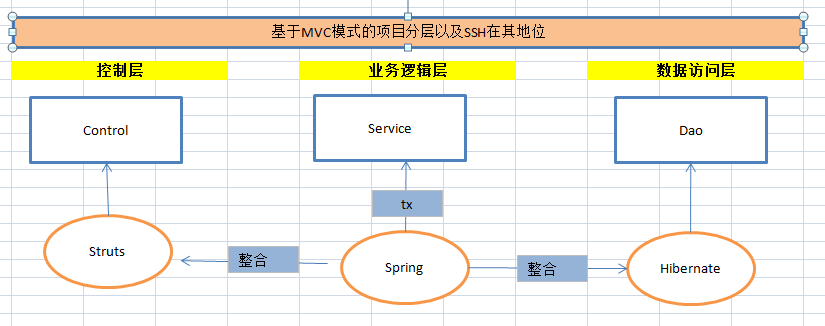
Src/structs.xml

访问路径：localhost:8080/struts2015727/hello

|  |
| --- |
| 1. 引入8个jar文件 |
| 2. web.xml |
| <!-- 引入struts核心过滤器 -->  <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |
| 3. 开发Action |
| // 开发action： 处理请求  **public** **class** HelloAction **extends** ActionSupport {    // 处理请求  **public** String execute() **throws** Exception {  System.*out*.println("访问到了action，正在处理请求");  System.*out*.println("调用service");  **return** "success";//返回跳转的标签  }  } |
|  |
| 4. 配置action : src/struts.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"  "http://struts.apache.org/dtds/struts-2.0.dtd">  <struts>  <package name=*"xxxx"* extends=*"struts-default"*>  <!—action 的访问路径和servelt的访问路径一样-->  <action name=*"hello"* class=*"cn.itcast.action.HelloAction"* method=*"execute"*>  <result name=*"success"*>/success.jsp</result>  </action>  </package>  </struts> |

注：action中的业务方法，必须返回String，方法不能有参数！

SSH框架：



Struts数据的处理（保存数据）：

|  |
| --- |
| 通过ActionContext类：  ActionContext ac = ActionContext.getContext();  Map<String,Object> request = ac.getContextMap();---🡪> HttpServletResult request = ServletActionContext.getRequest();[使用ServletApi，原始的request]  Map<String,Object> session = ac.getSession();  Map<String,Object> application = ac.getApplication();  request.put(“”,””)；  … |

|  |
| --- |
| 实现接口： RequestAware ,SessionAware,ApplicationAware [setRequest] |

**请求数据自动封装：**

Jsp表单数据填充到action 中的属性！ （getXX）

**拦截器的应用（在访问action前必须经过）：**

1. 继承AbstractInterceptor
2. 具体的拦截器业务处理方法：

|  |
| --- |
| /\*\*  \* 拦截器业务处理方法  \*/  public String intercept(ActionInvocation invocation) throws Exception {  // 拿到当前执行的方法名：判断，只有当前方法名不是login,就进行验证    // 获取ActionContext对象  ActionContext ac = invocation.getInvocationContext();    // 获取action的代理对象  ActionProxy proxy = invocation.getProxy();  // 获取当前执行的方法名  String methodName = proxy.getMethod();  // 判断  if (!"login".equals(methodName)) {  // 先获取当前登陆的用户  Object obj = ac.getSession().get("userInfo");  if (obj == null) {  // 没有登陆  return "input";  } else {  // 当前用户有登陆  return invocation.invoke();  }  } else {  // 说明当前用户正在登陆  return invocation.invoke();  }  } |

1. 拦截器配置：

|  |
| --- |
| <!-- 【拦截器配置】 -->  <interceptors>  <interceptor name="loginCheck" class="cn.itcast.interceptor.UserCheckInterceptor"></interceptor>  <interceptor-stack name="myStack">  <interceptor-ref name="defaultStack"></interceptor-ref>  <interceptor-ref name="loginCheck"></interceptor-ref>  </interceptor-stack>  </interceptors> |

1. 执行拦截器：

|  |
| --- |
| <!-- 【执行拦截器:第一种写法： 当前包下所有的acntion都执行myStack栈】(在action前)  <default-interceptor-ref name="myStack"></default-interceptor-ref>  -->  <!--第二种写法： 只是在这一个Action中执行myStack栈 （在action中）  <interceptor-ref name="defaultStackt"></interceptor-ref>  <interceptor-ref name="loginCheck"></interceptor-ref>  -->  <!-- 第三种写法：执行用户栈(与第二种写法一样， 只在当前aciton中执行自定义栈) -->  <interceptor-ref name="myStack"></interceptor-ref> |

**Struts2中国际化：**

1. 写资源文件 (同servlet)

2. 读资源文件

程序：ResourceBundle (同servlet)

JSP:

1）jstl表亲啊 （同servlet）

2）struts标签获取资源文件内容

区别：

Struts2加载资源文件更加简单！通过常量加载即可！再在jsp页面直接使用！

|  |
| --- |
| **🡪1. 写资源文件** |
| **Msg.properties 默认的语言环境； 找不到配置就找它** |
| **Msg\_en\_US.properties 美国** |
| **-🡪2. 加载** |
| **<constant name="struts.custom.i18n.resources" value="cn.itcast.config.msg"></constant>** |
| **🡪3. 使用: 标签name值直接写配置文件中的key** |
| **<s:text name="title"></s:text>** |

另外一点，

(推荐)加载资源文件通过常量加载

还可以在页面加载, 这样用：

<s:i18n name="cn.itcast.config.msg">

<s:text> 标签必须放到标签体中。

</s:i18n>