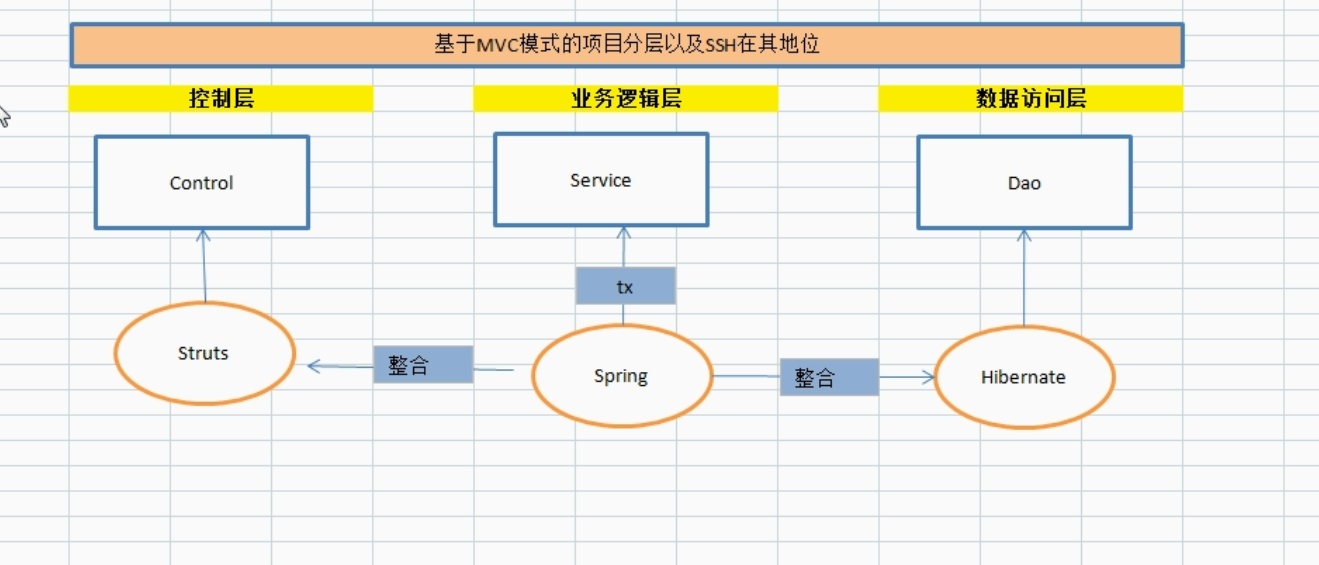
# 关于框架以及自定义struts



2017.4.26

自定义Structs框架的基本思想，更加简化servlet的功能，仅仅只是起到了一个控制的作用。

这就要求：获取请求信息的代码应该封装到一个Action类中，且根据不同的返回值灵活的对处理完请求后，是转发还是重定向，到哪个页面。

1. 创建一个ActionServlet类，让所有的客户端请求都转到这一个Servlet类中来。

这样就出现一个问题：怎么区分各种不同的Action呢？

根据请求获得url，进一步得到是哪个action，如/login.action ,最后获得的就是login。

String url=request.getServletPath(); // login.action;

String actionName=url.subString(string.lastIndexOf("/")+1,string.indexOf("."));

然后再通过配置文件找到相应的 类名 ,以及要使用的方法名。

String className=actionMapping.getClassName(); //actionMapping是一个bean。后面再讲。

String method=actionMapping.getMethod();

再根据反射，我们就可以获得对象，并执行相应的方法。

Class clazz=Class.forName(className);

Method method1=clazz.getDeclaredMethod(method,HttpServletRequest.class, HttpServletRequest.class);

第一个参数为 方法名，后面的参数为方法的 参数.class 参数类的字节码类

Object object=clazz.newInstance();

String returnFlag=(String)method1.invoke(object,req,reps);

第一个参数 为调用方法的对象 后面参数为方法参数。

这样根据配置文件以及反射技术，就可以完美的实现一个类，来处理不同的action。

1. 我们要获得的东西都应该要从配置文件中，弄出来。由上面可知，配置文件是关系型的，所以必须采用xml这种格式的。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <myStructs>  <package>  <action name="login" class="action.LoginAction" method="login">  <result name="success" type="redirect">/index.jsp</result>  <result name="loginFail">/login.jsp</result>  </action>  </package>  </myStructs> |

采用DOM4J.jar包对xml解析。

SAXReader reader=new SAXReader();

Document document=Reader.read(this.getClass().getSourceAsStream(‘mystructs.xml’));

通过解析器获得了Document 对象后，再把所有的元素标签封装到一个bean中，最后存入map集合中。

|  |
| --- |
| InputStream fis = this.getClass().getResourceAsStream("/myStructes.xml");  Document document = saxReader.read(fis);  //3.获取根  Element root = document.getRootElement();  Element Package = root.element("package");  List<Element> listAction = Package.elements("action");  for (Element ele\_action : listAction) {  ActionMapping actionMapping = new ActionMapping();  actionMapping.setName(ele\_action.attributeValue("name"));  actionMapping.setClassName(ele\_action.attributeValue("class"));  actionMapping.setMethod(ele\_action.attributeValue("method"));  Map<String, Result> resultMap = new HashMap<String, Result>();  List<Element> listResult = ele\_action.elements();  for (Element ele\_result : listResult) {  Result result = new Result();  result.setName(ele\_result.attributeValue("name"));  result.setType(ele\_result.attributeValue("type"));  result.setPage(ele\_result.getTextTrim());  resultMap.put(result.getName(), result);  }  actionMapping.setResultMap(resultMap);  allActions.put(actionMapping.getName(), actionMapping);  } |

# Struts全部内容：

1. **struts的开发环境搭建:**
   1. 引入jar包，
   2. 配置核心过滤器
   3. Action开发
   4. Struts.xml的配置。
2. **Struts的执行流程：**
   1. 服务器启动的时候：加载web项目，加载核心过滤器并初始化它的方法：

Struts-default.xml 核心功能初始化

Struts-plugin.xml struts相关插件

Struts.xml 用户编写的配置文件

同时，创建默认拦截器的对象，并对其初始化。

* 1. 第一次访问的时候：创建action实例，初始化，执行拦截器对象的intercept方法，执行就代表通过走向下一个拦截器。然后，再执行action的业务方法。

1. **Struts的核心功能：**
   1. 数据处理（用servlet的方式处理 或者 用action的方法处理）
   2. 数据自动封装（对象需要 set、get方法都要写）
   3. 文件上传与下载

上传：struts可以自动获取file文件，FileName文件名，FileContentType，

然后通过FileUtiles直接把上传的文件存储起来就好了。

|  |
| --- |
| private File file1;  private String file1FileName;  private String file1ContentType;  public void setFile1(File file1) {  this.file1 = file1;  }  public void setFile1FileName(String file1FileName) {  this.file1FileName = file1FileName;  }  public void setFile1ContentType(String file1ContentType) {  this.file1ContentType = file1ContentType;  }  @Override  public String execute() throws Exception {  String path = ServletActionContext.getServletContext().getRealPath("/upload");  File destFile = new File(path,file1FileName);  FileUtils.copyFile(file1, destFile);  return SUCCESS;  } |

同时，可以通过注入参数，来设置上传文件的大小，以及类型。

|  |
| --- |
| <interceptor-ref name="defaultStack">  <param name="fileUpload.maximumSize" >300000000</param>  <!-- 限制运行的文件的扩展名 -->  <param name="fileUpload.allowedExtensions">txt,jpg,jar</param>    </interceptor-ref> |

下载：

文件下载，就是通过在action中配置result的type为stream来实现的。 需要配置：文件类型（统一用二进制的下载类型），文件下载流、文件下载头、以及缓冲区类型。

|  |
| --- |
| <result name="download" type="stream">  <!-- 运行下载的文件的类型:指定为所有的二进制文件类型 -->  <param name="contentType">application/octet-stream</param>  <!-- 对应的是Action中属性： 返回流的属性【其实就是getAttrInputStream()】 -->  <param name="inputName">attrInputStream</param>    <!-- 下载头，包括：浏览器显示的文件名 -->  <param name="contentDisposition">attachment;filename=${downFileName}</param>    <!-- 缓冲区大小设置 -->  <param name="bufferSize">1024</param>  </result> |

* 1. 自定义拦截器实现校验

自定义拦截器：定义一个类，实现interceptor接口 ，并重写方法。

|  |
| --- |
| public String intercept(ActionInvocation actionInvocation) throws Exception {  actionInvocation.invoke();  ActionProxy actionProxy=actionInvocation.getProxy();  actionProxy.getMethod();  return null;  } |

这个方法intercept是在访问后，在action实力创建并初始化后，执行的。执行ActionInvocation对象的invoke(),就代表不拦截了，执行下一个拦截器。

将自定义的拦截器，配置：



注意：通过ActionInvocation对象可以获取当前action的代理对象，通过代理对象，可以获取当前执行的action类名，以及当前要执行的方法名。 这样就可以进行控制。

* 1. Ognl表达式，以及jsp页面上取值，struts标签的使用。

根元素：

在action中的全局变量；

非根元素：

一些存在在域对象中的元素；

根元素取值，不用加# ，非根元素需要。

**为什么可以在jsp中取到值呢？**

因为struts把所有的数据都放到了valuStack值栈对象中了。ValueStack中分根元素，以及非根元素，取法不同。

* 1. 数据校验

通过xml来配置数据校验信息。

Struts中的拦截器已经配置了相应的拦截器。

1.Jsp中的要写：

<s:fielderror fieldName="user.userName"></s:fielderror>

2.xml配置

Xml名字：ActionClassName-actionName(user\_login)-validation.xml

Xml里面的细节：

引入约束：

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE validators PUBLIC  "-//Apache Struts//XWork Validator 1.0.3//EN"  "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd"> |

|  |
| --- |
| <!--需要验证的参数--> <**field** name="user.password">  <!--选择验证的校验器-->  <**field-validator** type="requiredstring">  <!--错误信息-->  <**message**>密码不能为空</**message**>  </**field-validator**>  <**field-validator** type="stringlength">  <!--参数-->  <**param** name="minLength">3</**param**>  <**param** name="maxLength">8</**param**>  <**message**>密码长度必须为3-8位</**message**>  </**field-validator**> </**field**> |

1. **Struts的其他细节**

数据回显，防止表单重复提交，模型驱动