[Struts2使用26 -- Convention插件](http://blog.csdn.net/xiaoping8411/article/details/5641575)

**1.1.** **设置结果页面路径**

默认所有的结果页面都存储在WEB-INF/content下，你可以通过设置struts.convention.result.path这个属性的值来改变到其他路径。如：

Xml代码：

|  |
| --- |
| <constant name="struts.convention.result.path" value="/WEB-INF/page" /> |

则将路径配置到了WEB-INF/page 下。

**1.2.** **设置Convention搜索包**

默认包路径包含action,actions,struts,struts2的所有包都会被struts作为含有Action类的路径来搜索。你可以通过设置struts.convention.package.locators属性来修改这个配置。如：

|  |
| --- |
| <constant name="struts.convention.package.locators" value="web,action" /> |

则定义了在项目中，包路径包含web和action的将被视为Action存在的路径来进行搜索。

Com.ustb.web.\*/com.ustb.action.\*都将被视为含有Action的包路径而被搜索。

接着，Convention从前一步找到的package以及其子package中寻找 com.opensymphony.xwork2.Action 的实现以及以Action结尾的类：

|  |
| --- |
| com.example.actions.MainAction  com.example.actions.products.Display (implements com.opensymphony.xwork2.Action)  com.example.struts.company.details.ShowCompanyDetailsAction |

**1.3.** **命名空间**

命名空间。从定义的.package.locators标示开始到包结束的部分，就是命名空间。举个例子：

|  |
| --- |
| Com.ustb.web.user.userAction的命名空间是：”/user”。  Com.ustb.web.user.detail.UserAction的命名空间是：”/user/detail” |

**1.4.** **Actin类名路径分割**

Convention通过如下规则确定URL的具体资源部分:去掉类名的Action部分。然后将将每个分部的首字母转为小写，用’-’分割，你可以设置struts.convention.action.name.separator 如

<constant name="struts.convention.action.name.separator" value="-" />

还是举个例子：

UserAction->user  UserDetailAction ->user-detail。

结合上面的。

对于com.ustb.web.user.detail.UserDetailAction，映射的url就是/WEB-INF/content/user/detail/user-detail.jsp

**1.5.** **支持jsp、html、htm、vm等格式**

struts支持.jsp .html .htm .vm格式的文件。

下面是actiong和结果模版的映射关系：

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Result | File that could match | Result Type |
| /hello | success | /WEB-INF/content/hello.jsp | Dispatcher |
| /hello | success | /WEB-INF/content/hello-success.htm | Dispatcher |
| /hello | success | /WEB-INF/content/hello.ftl | FreeMarker |
| /hello | input | /WEB-INF/content/hello-world-input.vm | Velocity |
| /hello | error | /WEB-INF/content/test/test2/hello-error.html | Dispatcher |

以上的内容来自struts2的文档http://struts.apache.org/2.1.6/docs/convention-plugin.html

当然，简单的通过默认的方式来进行配置不能完全满足实际项目的需要。所幸，convention的零配置是非常灵活的。

**1.6.** **@Action注解**

通过@Action注释

对如下例子：

|  |
| --- |
| Java代码  package com.example.web;    import com.opensymphony.xwork2.Action;  import com.opensymphony.xwork2.ActionSupport;    public class HelloAction extends ActionSupport {      @Action("action1")      public String method1() {          return SUCCESS;      }        @Action("/user/action2")      public String method2() {          return SUCCESS;  }  }  方法名    默认调用路径       默认映射路径  method1  /hello!method1.action .   /WEB-INF/content/hello.jsp  method2  /hello!method2.action.    /WEB-INF/content/hello.jsp  通过@Action注释后  方法名    @Action注释后调用路径    @Action注释 后映射路径  method1  /action1!method1.action.       /WEB-INF/content/action1.jsp  method1  /user/action2!method2.action /WEB-INF/content/user/action2.jsp |

**1.7.** **@Actions注解**

通过@Actions注释,例子：

|  |
| --- |
| Java代码  package com.example.web;    import com.opensymphony.xwork2.ActionSupport;  import org.apache.struts2.convention.annotation.Action;  import org.apache.struts2.convention.annotation.Actions;    public class HelloAction extends ActionSupport {    @Actions({      @Action("/different/url"),      @Action("/another/url")    })    public String method1() {      return “error”;    } |

我们可以通过：/different/url!method1.action 或 /another/url!method1.action 来调用method1 方法。

对应的映射路径分别是

/WEB-INF/content/different/url-error.jsp; /WEB-INF/content/another/url-error.jsp

可能误导了大家，一个方法被@Action注释后，只是多了一种调用方式，而不是说覆盖了原来的调用方式。比如对于如下例子：

|  |
| --- |
| Java代码  com.example.web;    import com.opensymphony.xwork2.ActionSupport;  import org.apache.convention.annotation.Action;  import org.apache.convention.annotation.Actions;    public class HelloAction extends ActionSupport {    @Action("/another/url")    public String method1() {      return “error”;    } |

我们调用method1方法可以通过两种方式：

1  /hello!method1.action 映射 url：/WEB-INF/content/hello-error.jsp

2 /another/url!method1.action 映射 url：/WEB-INF/content/another/url-error.jsp

可见，两种方式均可对method1方法进行调用，唯一的区别就是，两种调用的映射是不一样的，所以，想跳转到不同的界面，这是一个非常好的选择。

**1.8.** **@Namespace注解**

通过@Namespace 注释

|  |
| --- |
| package com.example.web;    import com.opensymphony.xwork2.ActionSupport;  import org.apache.struts2.convention.annotation.Action;  import org.apache.struts2.convention.annotation.Actions;  @Namespace("/other")  public class HelloWorld extends ActionSupport {      public String method1() {      return “error”;    }      @Action("url")    public String method2() {  return “error”;    }        @Action("/different/url")    public String method3() {  return “error”;    }  } |

通过 /other/hello-world!method1.action 访问method1 方法。

通过 /other/url!method2.action 访问method2 方法

通过 /different /url!method3.action 访问method3 方法

与@Action 注释不同的是，该注释覆盖了默认的namespace(这里是’/’)，此时再用hello!method1.action 已经不能访问method1 了.

**1.9.** **@Results和@Result注解**

@Results和@Result

**1.9.1.全局的（global）。**

全局results可以被action类中所有的action分享，这种results在action类上使用注解进行声明。

|  |
| --- |
| package com.example.actions;    import com.opensymphony.xwork2.ActionSupport;  import org.apache.struts2.convention.annotation.Action;  import org.apache.struts2.convention.annotation.Actions;  import org.apache.struts2.convention.annotation.Result;  import org.apache.struts2.convention.annotation.Results;    @Results({    @Result(name="failure", location="/WEB-INF/fail.jsp")  })  public class HelloWorld extends ActionSupport {    public String method1() {      return “failure”;    }      @Action("/different/url")    public String method2() {      return “failure”;    }  } |

当我们访问 /hello -world !method1.action 时，返回 /WEB-INF/fail.jsp

当我们访问 /hello -world !method2.action 时，返回 /WEB-INF/fail.jsp

当我们访问 /different/url!method2.action 时，返回 /WEB-INF/fail.jsp

**1.9.2.本地的（local）。**

本地results只能在action方法上进行声明。

|  |
| --- |
| Java代码  package com.example.actions;    import com.opensymphony.xwork2.ActionSupport;  import org.apache.struts2.convention.annotation.Action;  import org.apache.struts2.convention.annotation.Actions;  import org.apache.convention.annotation.Result;  import org.apache.convention.annotation.Results;    public class HelloWorld extends ActionSupport {      @Action(value="/other/bar",results={@Result(name = "error", location = "www.baidu.com",type="redirect")})    public String method1() {      return “error”;    }  } |

当我们调用 /hello -world !method1.action 时，返回 /WEB-INF/content/hello-error.jsp

当我们调用 /other/bar!method1.action 时，返回 www.baidu.com

**1.10.**            **@ParentPackage 注解**

ParentPackage注解用来定义具体action类的父XWork包或java包，下面例子演示了在action类上使用本注解：

|  |
| --- |
| package com.example.actions;  import com.opensymphony.xwork2.ActionSupport;  import org.apache.struts2.convention.annotation.Action;  import org.apache.struts2.convention.annotation.ParentPackage;  @ParentPackage("customXWorkPackage")  public class HelloWorld extends ActionSupport {      public String execute() {          return SUCCESS;      }  } |

**1.11.**            **异常注解配置**

ExceptionMapping 注解用来影射action抛出的异常。可以参考exception mapping documentation 获得详细信息。注解用类级别，在这种情况下，注解会应用到类里面的所有action

|  |
| --- |
| @ExceptionMappings({  @ExceptionMapping(exception = "java.lang.NullPointerException",  result = "success", params = {"param1", "val1"})  })  public class ExceptionsActionLevelAction {        public String execute() throws Exception {          return null;      }  } |

可以在ExceptionMapping注解中使用params 属性来传递具体值给结果渲染页。ExceptionMapping注解同样可以在action级别进行设置：

|  |
| --- |
| public class ExceptionsMethodLevelAction {     @Action(value = "exception1", exceptionMappings = {                 @ExceptionMapping(exception = "java.lang.NullPointerException",  result = "success", params = {"param1", "val1"})      })      public String run1() throws Exception {          return null;      }  } |

**1.12.**            **自动加载无需启动服务**

Convention插件可以自动重新加载配置的功能，无需重启容器，就可以刷新类中包含的action。这自动加载automatic xml配置文件类似。你必须在struts.xml 中添加以下代码来启用本功能：

|  |
| --- |
| <constant name="struts.devMode" value="true"/>  <constant name="struts.convention.classes.reload" value="true" /> |

此功能没有在所有容器中进行过测试，强力建议不要在生产环境中使用。

**1.13.**            **扫描Action的Jar包**

默认情况下，Convention 插件不会从jar文件中寻找action。如果想实现这一功能，jar文件必须被struts.convention.action.includeJars 所定义的正则 匹配到。在例子中 myjar1.jar和 myjar2.jar 将被插件检测到：

|  |
| --- |
| <constant  name="struts.convention.action.includeJars"  value=".\*/myjar1.\*?jar(!/)?,.\*/myjar2\*?jar(!/)?" /> |

提示：正则表达式只针对jar文件的路径进行匹配，而不是文件名。jar的URL应该包含jar文件的路径并以"!/"结尾。