# 合作开发

本文档主要对阶段性说明文档1之后开发的部分予以说明。

## 关于拦截器

在本系统中，拦截器是重要的组成部分。主要使用的拦截器是一个自定义的session检查拦截器以及一个token拦截器。

Session拦截器（club.istc.validation.SessionCheck）用于检查登录状态。即，登录状态的保持是由session来维持，在点击一个个的链接时，session也不断地进行传递。每次请求一个URL，都会通过该拦截器的检查。该拦截器对于登录、注册、首页访问和登出等请求不予拦截，但对其他的页面全部检查session。一旦session为null或者session内的personInfo信息为空，那么即判定登录失效，系统自动引导至登出界面。另外，只要不主动销毁session，那么在一个有效期内使用action请求页面时，session会传递下去。有效期的设定在web.xml中。

Token拦截器用于防止表单重复提交。在每个表单中加入<s:token/>标签，每次提交都会随机生成一个token，系统会自动查验该token是否重复，一旦重复则返回错误。而这里使用的tokenSession拦截器则更方便，当重复提交时该请求会被直接阻塞掉，就好像该提交从来没有发生过一样。

在struts.xml中，全局拦截器的配置方法如下：

在package开头，加上如下代码：

<interceptors>

<interceptor name="loginInteceptor" class="club.istc.validation.SessionCheck" />

<interceptor-stack name="loginStack">

<interceptor-ref name="loginInteceptor"></interceptor-ref>

<interceptor-ref name="defaultStack"></interceptor-ref>

</interceptor-stack>

</interceptors>

Interceptor name 代表自定义拦截器的名称，后面是对应的类名。

后面的interceptor-stack代表自定义的拦截器栈。从上到下的顺序决定了拦截器的执行顺序。

Action中的局部拦截器只要将<interceptors/>包含的去掉，剩下的加进去即可。

注意：使用拦截器时，必须在最后加上<interceptor-ref name="defaultStack"></interceptor-ref>，这代表拦截器缺省时使用默认拦截器。不然会报错

另外，一般重复提交很多情况下是发生在登录成功后进入个人主页时，用户重新加载页面的时候。解决方法如下：重定向至action，这个在后文有述。

## 关于动态action

因为很多action将导致管理混乱，因此很多同一类型的操作可以编写在一个action中。比如，对于例会的信息管理有四种：获取例会记录，增加例会，修改例会和删除例会，这些操作都是关于例会的操作。因此我们可以将其全部放在一个action，即MeetingAction中。每个操作对应一个方法，并且涉及表单数据获取的方法再加上对应的validate方法。

在struts.xml中，可以用通配符的方法配置动态action。方法如下：

<action name="interviewee\*" class="club.istc.action.InterviewAction" method="{1}">

<!-- 面试测试页面 -->

<result name="input">interview.jsp</result>

</action>

name这里填写的是URL中将来会填写的内容，当URL中出现intervieweeget时，会自动匹配到该action上，而name后面有一个\*，代表着去掉interviewee，get将会等价为\*。

Method=”{1}”即代表这里的{1}会被\*所取代，即前文中的get。Method即“方法”，意味着会使用该action中对应名称的方法。

注意：一定在所属package中加上这句话（<global-allowed-methods>regex:.\*</global-allowed-methods>）代表开启通配符的使用，否则会报错。

## 关于使用Redirect类进行重定向

我们在跳转的时候，如果直接在URL中写出jsp完整文件名，不仅不安全，还不能传递session。因此使用Redirect类进行重定向。具体的体现是，在URL中不直接写jsp完整文件名，而是写出对应该文件的一个名字（比如去掉扩展名后的jsp文件名），之后系统解析该URL，会引导至Redirect类中的一个同名方法，该方法返回一个同名的result，struts获取该result后再跳转到此页面。

由于经过了一次action，系统自动帮我们传递了前一个页面的session，也不至于暴露真实文件名（只要把action名修改成与页面不一样即可），也满足动态跳转的要求。

实现方式如下：

比如我们想要访问login页面，我们在首页不写<a href=”login.jsp”/>而是写<a href=”login”>。

Struts.xml对应配置为：

<action name="\*" class="club.istc.action.Redirect" method="{1}">

<result name="{1}">{1}.jsp</result>

</action>

login就会被自动拿去匹配Redirect中的方法。

Redirect中对应的方法是：

public String login() {

return "login";

}

这次返回一个名为login的result，然后在result中，<result name="{1}">{1}.jsp</result>

login就会替换所有的{1}，最终我们找到了login.jsp，但URL是http://\*\*\*/login而不是http://\*\*\*/login.jsp

## 关于重定向至action

在提交表单时（以登录为例）如果提交成功，那么进入欢迎页面时，url会显示http://\*\*\*/login。这是因为strut.xml中是这么配置的：

<result name="success" >welcome.jsp</result>

即，登陆成功后会自动引导至welcome.jsp上。但是这个时候用户如果点刷新的话，因为url没有变，因此还会执行login这个action，尽管此时我们并没有提交任何数据。浏览器就会提示“重复提交表单”。改进方法如下：

将上面的配置改为：

<result name="success" type="redirectAction">welcome</result>

这样，当登录成功时，引导目标并不是一个jsp页面，而是一个action。于是将会执行action，也就是进行一遍Redirect重定向至目标页面的过程。即便用户刷新页面，也只是重复重定向的过程而已，并没有重复提交表单的过程。因此就避免了刷新会产生重复提交表单提示的问题。

## 关于ajax

Ajax即“异步js和xml”，即通过点击按钮等行为触发js函数，在js函数中会有一些与后端请求数据的过程以及其他的信息过程，之后通过操作DOM对象对html进行局部重绘。这期间需要与服务器沟通获取数据的话，一般使用的是XMLHttpRequest。但由于struts原生对json的支持，以及json的轻量化，因此我们也可以使用json进行数据的传递。

这里可能使用ajax的部分包括表单数据验证，面试人员加载等。

目前实现了表单数据验证的ajax化，实现原理如下：

这里的表单提交与一般的表单提交不太一样，不能通过form的原生提交方式，即action=”login”,method=”post”，而是只有一个表单。按钮写在表单外部，当按下按钮时，触发一个jQuery函数。该函数首先调用serialize()函数将表单数据序列化，之后用post函数传到action中。如果验证不通过，会返回一个json字符串，返回json字符串需要struts.xml按如下方式配置：

<package name="needajax" extends="json-default,struts-default">

<action name="Login" class="club.istc.action.Login">

<interceptor-ref name="jsonValidationWorkflowStack"></interceptor-ref>

<result name="input" type="json">

<param name="ignoreHierarchy">false</param>

<param name="includeProperties">actionErrors\[\d+\], fieldErrors\..+$, actionMessages\[\d+\]</param>

</result>

</action>

</package>

Extends这里必须继承json-default。

includeProperties即格式化json字符串。

之后获得了json字符串，再通过jQuery函数解析并通过操作DOM对象写入合适的位置。

由于ajax的实现方式较为复杂，目前考虑的是仅实现登录和注册验证的ajax，其余部分全部使用非ajax进行开发。