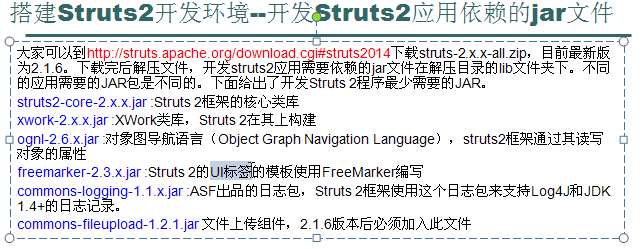


# 1.需要的jar包



# 2.配置文件struts.xml

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

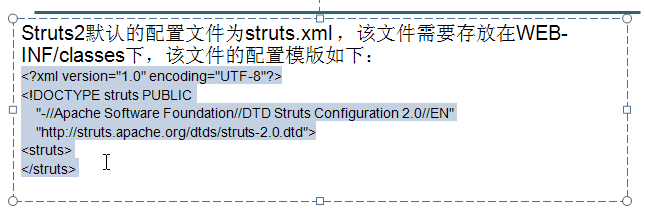
<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"

"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

</struts>



# 3.web中的配置1是servlet启动的

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

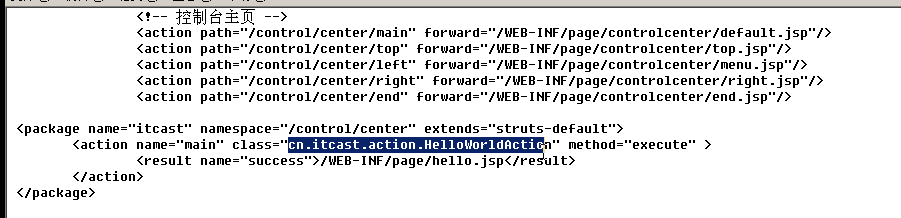
<filter-mapping>

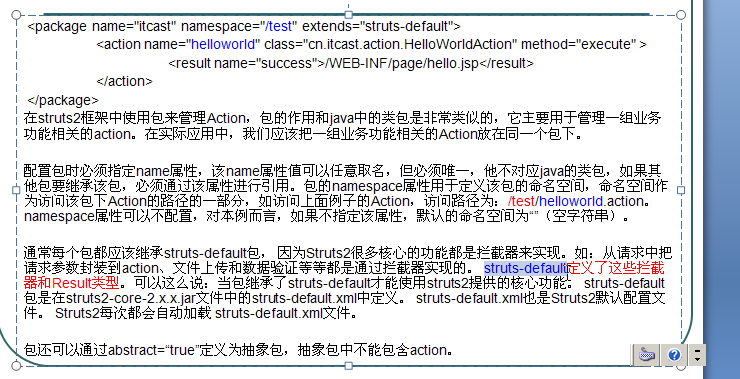
<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

# 4.配置action





# 5.第一个Struts 2应用

## 5.1配置文件

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">

<struts>

下面这个主要是提供继承使用，下面这个是包的命名空间，

<package name=*"itcast"* namespace=*"/test"* extends=*"struts-default"*>

<action name=*"helloworld"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"* >

<!-- 这个相当于是forward -->

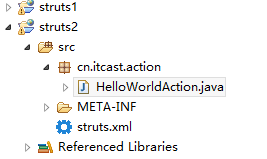
<result name=*"success"*>/WEB-INF/page/hello.jsp</result>

</action>

</package>

</struts>

## 5.2 action，位置及代码，这里的getMessage 就会添加到$中



**package** cn.itcast.action;

**public** **class** HelloWorldAction {

**private** String msg;

**public** String getMessage() {

**return** msg;

}

**public** String execute(){

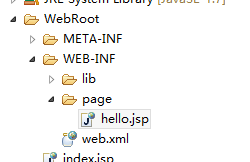
msg = "这是我们的第一个 Struts 2 应用";

**return** "success";

}

}

## 5.3进行测试，



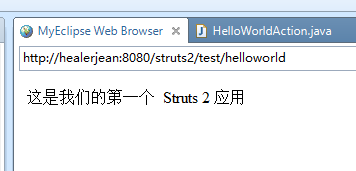
<body>

${message }

</body>

## 5.4测试链接 包的命名空间和action

<http://healerjean:8080/struts2/test/helloworld>



# 6.默认命名空间

## 6.1对于上面的，我们输入也会进入，然后我们也会到了上面的界面。它会自动寻找test以下的包，依次找，直到找到位置。

<http://healerjean:8080/struts2/test/path1path2/helloworld>

## 6.2 如果test 包下面没有的时候，会进入默认命名空间namespace为空，或者不设置

<package name=*"itcast"* namespace=*"/test"* extends=*"struts-default"*>

</package>

<package name=*"it"* extends=*"struts-default"*>

<action name=*"helloworld"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"* >

<result name=*"success"*>/WEB-INF/page/hello.jsp</result>

</action>

</package>

# 7.地址后缀.action

<http://healerjean:8080/struts2/test/path1path2/helloworld>.action

# 8.转发action class不需要配置class直接就可以

## 8.1配置文件

<package name=*"itcast"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"*>/WEB-INF/page/hello.jsp</result>

</action>

<action name=*"addUI"*>

<result>/WEB-INF/page/employeeAdd.jsp</result>

</action>

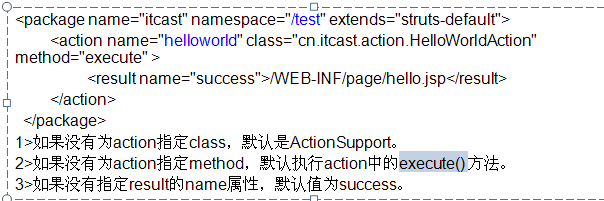
</package>

## 8.2测试

<http://healerjean:8080/struts2/control/employee/addUI.action>

## 8.3解释：如果没有为action 配置class 属性，默认就会

# 9.action配置中的各项默认值



## 9.1如果没有为action 配置class 属性，默认是ActionSupprot ，没有method 默认执行execute（）方法，没有name值，默认为 success

# 10定向转发webroot

## 10.1将employeeAdd.jsp放到 webroot下面（其实就是）

<http://healerjean:8080/struts2/employeeAdd.jsp>

## 10.2 action 中name相同，然后result 中type 和它是一样的

<package name=*"itcast"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"*>/WEB-INF/page/hello.jsp</result>

</action>

<action name=*"addUI"*>

<result>/WEB-INF/page/employeeAdd.jsp</result>

</action>

<action name=*"redirect"*>

<result type=*"redirect"*>/employeeAdd.jsp</result>

</action>

## 10.3测试然后就到了一般情况下的直

<http://healerjean:8080/struts2/control/employee/redirect>

<http://healerjean:8080/struts2/employeeAdd.jsp>

# 11定向传参数*redirect*

## 1.改变action java文件中的内容，设置接收表单

**public** **class** HelloWorldAction {

**private** String msg;

**private** String username;

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

**public** String getMessage() {

**return** msg;

}

**public** String execute() **throws** Exception{

**this**.username = URLEncoder.*encode*("传智播客", "UTF-8");

**this**.msg = "我的第一个struts2应用";

**return** "success";

}

}

## 2.模拟发送

<package name=*"itcast"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"* type=*"redirect"*>

/employeeAdd.jsp?username = ${username}</result>

</action>

## 3.接收

<%= URLDecoder.decode(**new** String(request.getParameter("username").getBytes("ISO8859-1"),"UTF-8"),"UTF-8") %>

<form action=*"/xxx"*>

姓名:<input type=*"text"* name=*"xxx"*>

</form>

# 12.定向到action

<package name=*"itcast"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"* type=*"redirect"*>/employeeAdd.jsp?username = ${username}</result>

</action>

<action name=*"redirect"*>

<result type=*"redirect"*>/employeeAdd.jsp</result>

</action>

<action name=*"redirectAction"*>

<result type=*"redirectAction"*>list</result>

</action>

# 13.定向到其他的包的action，实际上是注入

<package name=*"itcast"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"* type=*"redirect"*>/employeeAdd.jsp?username = ${username}</result>

</action>

<action name=*"redirect"*>

<result type=*"redirect"*>/employeeAdd.jsp</result>

</action>

<action name=*"redirectAction"*>

<result type=*"redirectAction"*>

<param name=*"actionName"*>xxx</param>

<param name=*"namespace"*>/control/department/xdsfdsfds/435435</param>

</result>

</action>

</package>

<package name=*"other"* namespace=*"/control/department"* extends=*"base"*>

<action name=*"manage"* class=*"cn.itcast.action.HelloWorldAction"* method=*"add"*>

</action>

<action name=*"xxx"*>

<result>/WEB-INF/page/hello.jsp</result>

</action>

</package>

# 14.输出源代码*plainText*

<!-- 从定向到 jsp -->

<action name=*"plainText"*>

<result type=*"plainText"*>

<param name=*"location"*>/index.jsp</param>

<param name=*"charSet"*>UTF-8</param><!-- 指定读取文件的编码 -->

</result>

</action>



# 15全局跳转

<!-- 全局跳转 -->

<global-results>

<result name=*"message"*>/WEB-INF/page/message.jsp</result>

</global-results>

<action name=*"manage"* class=*"cn.itcast.action.HelloWorldAction"* method=*"add"*>

</action>

**public** String add(){

**return** "message";

}

## 2.如果希望所有的包里面都可以全局跳转，用到继承

<package name=*"base"* extends=*"struts-default"*>

<global-results>

<result name=*"message"*>/WEB-INF/page/message.jsp</result>

</global-results>

</package>

<package name=*"other"* namespace=*"/control/department"* extends=*"base"*>

<action name=*"manage"* class=*"cn.itcast.action.HelloWorldAction"* method=*"add"*>

</action>

<action name=*"xxx"*>

<result>/WEB-INF/page/hello.jsp</result>

</action>

</package>

# 16.为action 注入值

## 1.依赖注入

<!-- 依赖注入 -->

<package name=*"other"* namespace=*"/control/department"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<!-- 下面这个就是注入 的值 有set方法才可以 -->

<param name=*"pathString"*>/image</param>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

## 2. Action中东西

**public** **class** HelloWorldAction {

String pathString;

**public** String getPathString() {

**return** pathString;

}

**public** **void** setPathString(String pathString) {

**this**.pathString = pathString;

}

**public** String execute() **throws** Exception{

**return** "success";

}

}

## 3.测试

<http://healerjean:8080/struts2/control/department/list>

<body>

<!-- 结果 -->

${pathString }

</body>

# 17.制定请求后缀，使用do或者action

<!-- 制定请求后缀 -->

<constant name=*"struts.action.extension"* value=*"do,action"*/>

<!-- 依赖注入 -->

<constant name=*""* value=*""*></constant>

<package name=*"other"* namespace=*"/control/department"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<!-- 下面这个就是注入 的值 有set方法才可以 -->

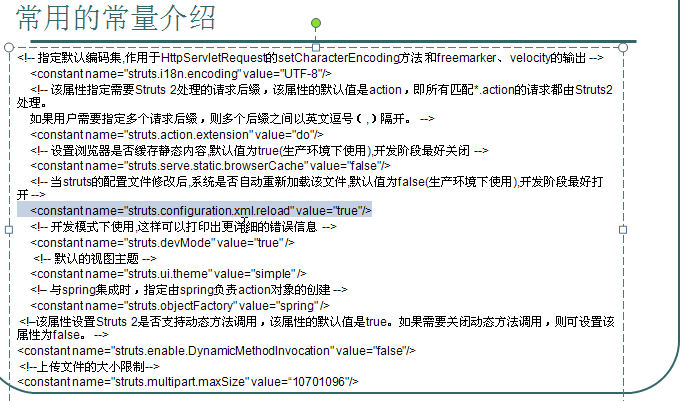
<param name=*"pathString"*>/image</param>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

# 18常用的常亮介绍



# 19为应用指定多个配置文件

## 1.Struts。xml中

<include file=*"department.xml"*></include>

<include file=*"employee.xml"*></include>

## 2. employee.xml

<struts>

<package name=*"employee"* namespace=*"/control/employee"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<param name=*"pathString"*>employee</param>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

</struts>

## 3.测试

<http://healerjean:8080/struts2/control/department/list.do>

# 20.动态方法调用类似于1中的分派action

## 1.action中的两个方法

**public** **class** HelloWorldAction {

**private** String msg;

**public** String getMsg() {

**return** msg;

}

**public** String addUI(){

msg = "addUI";

**return** "success";

}

**public** String execute() **throws** Exception{

msg = "execute";

**return** "success";

}

}

## 2.配置文件

<struts>

<package name=*"department"* namespace=*"/control/department"* extends=*"struts-default"*>

<action name=*"list"* class=*"cn.itcast.action.HelloWorldAction"* method=*"execute"*>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

</struts>

## 2.测试只要加一个！action就可以了

<http://healerjean:8080/struts2/control/department/list!addUI.do>

${msg }

## 3.禁止动态方法调用

<constant name="struts.enable.DynamicMethodInvocation" value="false"/>

# 21通配符定义,下划线可有可无

## 1.使用 action\_\* method = {1，2}

### 1.代表有几个 \*

<!-- 通配符 定义 -->

<package name=*"department"* namespace=*"/control/department"* extends=*"struts-default"*>

<action name=*"list\_\*"* class=*"cn.itcast.action.HelloWorldAction"* method=*"{1}"*>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

## 2.测试/list\_addUI.d

<http://healerjean:8080/struts2/control/department/list_addUI.do>

<http://healerjean:8080/struts2/control/department/list_execute.do>

# 22.接收请求参数 重要

## 1.必须提供 set方法



## 2.action代码

**public** **class** HelloWorldAction {//?id=23&name=xxx

**private** Integer id;

**private** String name;

**private** Person person;

**public** Person getPerson() {

**return** person;

}

**public** **void** setPerson(Person person) {

**this**.person = person;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String addUI(){

**return** "success";

}

**public** String execute() **throws** Exception{

**return** "success";

}

}

## 3.Struts 配置文件中

<package name=*"department"* namespace=*"/control/department"* extends=*"struts-default"*>

<action name=*"list\_\*"* class=*"cn.itcast.action.HelloWorldAction"* method=*"{1}"*>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</action>

</package>

## 4. 测试向action提供参数，然调用

http://healerjean:8080/struts2/control/department/list\_addUI.action?id=123$name=%2212312%22

# 23.post 提供 参数

<form action=*"*<%=request.getContextPath()%>*/control/department/list\_execute.action"* method=*"post"*>

id:<input type=*"text"* name=*"id"*><br/>

name:<input type=*"text"* name=*"name"*><br/>

<input type=*"submit"* value=*"发送"*/>

</form>

# 24通过复合类型(对象)来提供参数

## 1..person.java,这个person 是有一个默认的构造器的，

**public** **class** Person {

**private** String name;

**private** Integer id;

public Person(){

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

}

## 2.form

<form action=*"*<%=request.getContextPath()%>*/control/department/list\_execute.action"* method=*"post"*>

id:<input type=*"text"* name=*"person.id"*><br/>

name:<input type=*"text"* name=*"person.name"*><br/>

<input type=*"submit"* value=*"发送"*/>

</form>

# 25.中文乱码问题

## 一个构造器，这个和Struts 1 是不一样的

# 26 自定义类型转换器

## 1.action 代码

**public** **class** HelloWorldAction {//?birthday=2008-10-10

**private** Date birthday;

**public** Date getBirthday() {

**return** birthday;

}

**public** **void** setBirthday(Date birthday) {

System.*out*.println(birthday);

**this**.birthday = birthday;

}

**public** String addUI(){

**return** "success";

}

**public** String execute() **throws** Exception{

**return** "success";

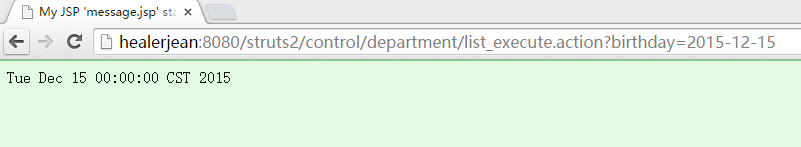
}

}

## 2.然后进行测试

<http://healerjean:8080/struts2/control/department/list_execute.action>

?birthday=2015-12-15



## 3.解释：我们发现这个自动转化了

## 4.但是我们希望 输入的是 20150617 发现控制台报错了，这个时候就是需要我们的 类型转换器（分为局部和全局）

### 4.1局部类型转化器，这个action中让一个类继承DefaultTypeConverter

**public** **class** DateTypeConverter **extends** DefaultTypeConverter {

/\*\*

\* value 这个是请求参数

\* toType 是转化成的类型 就是下面的birthday

\*/

@Override

**public** Object convertValue(Map<String, Object> context, Object value, Class toType) {

//这个是格式，将字符串转化时间

SimpleDateFormat dateFormat = **new** SimpleDateFormat("yyyyMMdd");

**try** {

**if**(toType == Date.**class**){

//字符串转化为时间

String[] params = (String[]) value;// request.getParameterValues()所以这个是数组

**return** dateFormat.parse(params[0]);

}**else** **if**(toType == String.**class**){

//时间转化为字符串

Date date = (Date) value;

**return** dateFormat.format(date);

}

} **catch** (ParseException e) {}

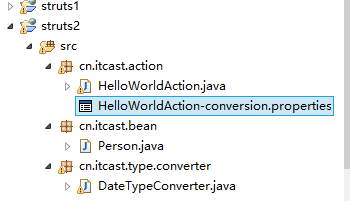
**return** **null**;

}

}

### 4.2注册局部类型转化器

#### 4.2.1HelloWorldAction-conversion.properties



birthday=cn.itcast.type.converter.DateTypeConverter

### 4.3测试成功

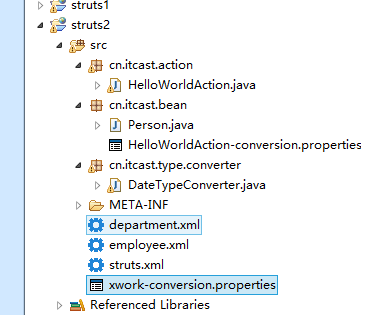
<http://healerjean:8080/struts2/control/department/list_execute.action>

?birthday=20140516

## 5.全局类型转换器



### 5.1将上面的xwork-conversion.properties放到src 目录下面



### 5.2然后打开修改

java.util.Date=cn.itcast.type.converter.DateTypeConverter

# 27setvlet访问或者添加几个属性

**public** **class** HelloWorldAction {

//getApplication

//一般情况下使用下面这种方法

**public** String execute(){

ActionContext ctx = ActionContext.*getContext*();

ctx.getApplication().put("app", "应用范围");//往ServletContext里放入app

ctx.getSession().put("ses", "session范围");//往session里放入ses

ctx.put("req", "request范围");//往request里放入req

ctx.put("names", Arrays.*asList*("老张", "老黎", "老方"));

**return** "success";

}

**public** String rsa() **throws** Exception{

//要得到摸一个路径

//servletContext

HttpServletRequest request = ServletActionContext.*getRequest*();

ServletContext servletContext = ServletActionContext.*getServletContext*();

request.setAttribute("req", "请求范围属性");

request.getSession().setAttribute("ses", "会话范围属性");

servletContext.setAttribute("app", "应用范围属性");

//HttpServletResponse response = ServletActionContext.getResponse();

**return** "success";

}

}

## 2.接收

${app}

${applicationScope.app} <br>

${sessionScope.ses}<br>

${requestScope.req}<br>

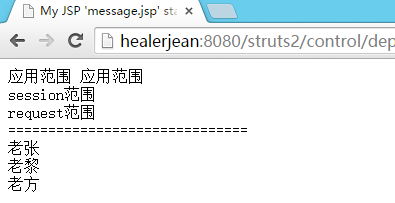
==============================<br/>

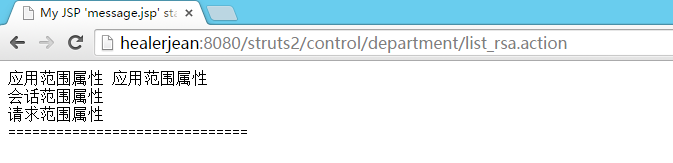
<c:forEach items=*"*${names}*"* var=*"name"*>

${name }<br/>

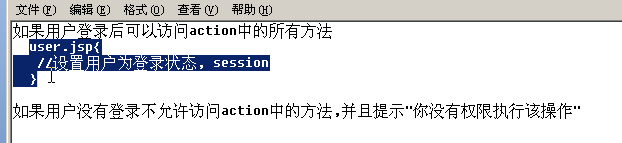
</c:forEach>

## 3.结果





# 28.用户登录后才可以使用action功能，自定义拦截器



## 1.实现Interceptor 接口

**public** **class** PermissionInterceptor **implements** Interceptor {

**public** **void** destroy() {

}

**public** **void** init() {

}

**public** String intercept(ActionInvocation invocation) **throws** Exception {

Object user = ActionContext.*getContext*().getSession().get("user");

**if**(user!=**null**) **return**

//用户名为空的时候 就是环绕执行

invocation.invoke();

//如果user不为null,代表用户已经登录,允许执行action中的方法

ActionContext.*getContext*().put("message", "你没有权限执行该操作");

**return** "success";

}

}

## 2.配置拦截器

<!-- 自定义拦截器 -->

<package name=*"employee"* namespace=*"/control/employee"* extends=*"struts-default"*>

<interceptors>

<interceptor name=*"permission"* class=*"cn.itcast.interceptor.PermissionInterceptor"*/>

<!-- 使用Struts 2 中的默认的拦截器stack 一堆拦截器的组成，然后定义所使用的功能 -->

<interceptor-stack name=*"permissionStack"*>

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

<interceptor-ref name=*"permission"* />

</interceptor-stack>

</interceptors>

<global-results>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</global-results>

<action name=*"list\_\*"* class=*"cn.itcast.action.HelloWorldAction"* method=*"{1}"*>

<!-- 使用上面的拦截器 进行注入 action-->

<interceptor-ref name=*"permissionStack"* />

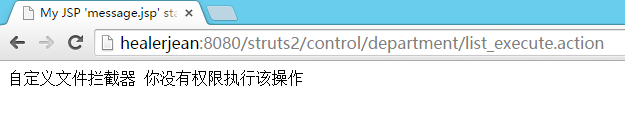
</action>

</package>

## 3.开始测试

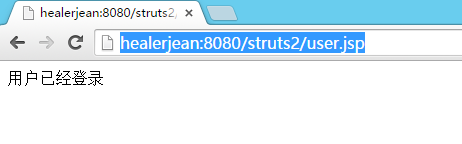
### 3.1直接登录这个action

<http://healerjean:8080/struts2/control/department/list_execute.action>

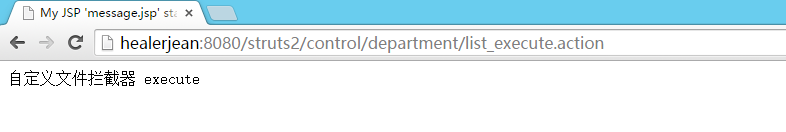


### 2.先登录

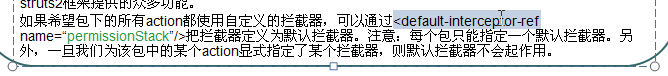
<http://healerjean:8080/struts2/user.jsp>



### 3.然后登录action



## 4.包下面的所有的acting都要使用这个action



### 4.1但是如果单独为某个action提供了action，默认的就会消失，所以就将两个都放到action下面

<default-interceptor-ref name=*"permissionStack"*></default-interceptor-ref>

<package name=*"employee"* namespace=*"/control/department"* extends=*"struts-default"*>

<interceptors>

<interceptor name=*"permission"* class=*"cn.itcast.interceptor.PermissionInterceptor"*/>

<!-- 使用Struts 2 中的默认的拦截器stack 一堆拦截器的组成，然后定义所使用的功能 -->

<interceptor-stack name=*"permissionStack"*>

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

<interceptor-ref name=*"permission"* />

</interceptor-stack>

</interceptors>

<default-interceptor-ref name=*"permissionStack"*></default-interceptor-ref>

<global-results>

<result name=*"success"*>/WEB-INF/page/message.jsp</result>

</global-results>

<action name=*"list\_\*"* class=*"cn.itcast.action.HelloWorldAction"* method=*"{1}"*>

</action>

</package>

# 29.简单问题

struts2中form的theme解释:  
struts2中theme属性包括xhtml,html,simple,ajax,默认是xhtml.  
theme:设置struts2标签的主题.  
theme=xhtml:会额外生成tr,td.  
theme=simple:生成所对应的html标签形式.

二、struts2中form的theme修改:  
html页面中使用struts2的ui tag的时候一般都要把theme设置为simple,把每个tag都设置比较麻烦.  
我们可以使用如下两种方法修改默认值xhtml:  
1、在struts.xml文件中修改默认值:<constant name="struts.ui.theme" value="simple"/>;  
2、在struts.properties文件中,设置以下语句:struts.ui.theme=simple;