# logo欢迎大家进入就业班进行JavaEE培训

*姓名:杜宏*

*电话:18600774790*

*QQ :22597927*

*EMAIL:duhong@itcast.cn*

# 第一天

## Struts2框架介绍

1、三大框架 ： 是企业主流 JavaEE 开发的一套架构

Struts2 + Spring + Hibernate

2、 什么是框架？为什么要学框架 ？

框架 是 实现部分功能的代码 （半成品），使用框架简化企业级软件开发

学习框架 ，清楚的知道框架能做什么？ 还有哪些工作需要自己编码实现 ？

3、 什么是Struts2 ？

Struts 2是Struts的下一代产品，是在 struts 1和WebWork的技术基础上进行了合并的全新的Struts 2框架。

其全新的Struts 2的体系结构与Struts 1的体系结构差别巨大。Struts 2以WebWork为核心

struts2 = struts1 + webwork;

struts2框架是apache产品。

struts2是一个标准的mvc框架。 javaweb中的model2模式就是一个mvc模式。

model2 = servlet + jsp + javaBean

struts2框架是在javaweb开发中使用的。

使用struts2框架，可以简化我们的web开发，并且降低程序的耦合度。

MVC：是一种思想，是一种模式，将软件分为 Model模型、View视图、Controller控制器

\* MVC由来是web开发

JavaEE软件三层结构 ： web层（表现层）、业务逻辑层、数据持久层 （sun提供JavaEE开发规范）

JavaEE开发更强调三层结构， web层开发注重MVC

struts2 就是 web层开发框架，符合MVC模式

\* struts1 、webwork 、jsf 、SpringMVC 都是MVC

4、 Struts2 和 Struts1 关系

没有关系， Struts2 全新框架，引入WebWork很多技术和思想，Struts2 保留Struts1 类似开发流程

\* Struts2 内核 webwork

struts2 = struts1 + webwork;

Xwork提供了很多核心功能：前端拦截机（interceptor），运行时表单属性验证，类型转换，强大的表达式语言（OGNL – the Object Graph Navigation Language），IoC（Inversion of Control反转控制）容器等

5、类似于struts2框架的产品 :

struts1 webwork jsf springmvc

ssh --- struts2 spring hibernate

ssi --- springmvc spring ibatis

## Struts2快速入门

1、 下载开发包

课程 以 struts2 3.15.1 讲解

2、 目录结构

apps ： struts2官方demo

docs ： 文档

lib ： jar包

src ： 源码

core 它是struts2的源代码

work-core struts2底层使用了xwork,xwork的源代码

3、 导入jar包到开发工程

只需要导入 apps/struts2-blank.war 中所有jar包 ---- 13 个jar包

4、 编写页面

hello.jsp 请求页面

<a href="${pageContext.request.contextPath }/hello.action">访问struts2入门</a>

success.jsp 结果页面

5、在web.xml 配置struts2 前端控制器 （Filter）

<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>

6、执行struts2过滤器后，读取struts2配置文件，将请求分发

在src下创建struts.xml

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

<!-- <a href="${pageContext.request.contextPath }/hello.action">访问struts2入门</a> -->

<!-- 将请求 分发给一个Action -->

<!-- action的name 就是hello.action 去掉扩展名 -->

<action name="hello" class="cn.itcast.struts2.demo1.HelloAction"></action>

</package>

7、执行目标Action中execute方法

8、在Action的execute方法中返回 字符串，在struts.xml中配置字符串与页面对应关系

<result name="executesuccess">/demo1/success.jsp</result> 完成结果页面跳转

## Struts2流程分析与工具配置

1、 运行流程

请求 ---- StrutsPrepareAndExecuteFilter 核心控制器 ----- Interceptors 拦截器（实现代码功能 ） ----- Action 的execuute --- 结果页面 Result

\* 拦截器 在 struts-default.xml定义

\* 执行拦截器 是 defaultStack 中引用拦截器

---- 通过源代码级别断点调试，证明拦截器是执行

2、 配置struts.xml 提示问题

如果安装Aptana编辑器 ，请不要用Aptana自带xml编辑器 编写struts2配置文件

struts.xml提示来自于 DTD约束，

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

"http://struts.apache.org/dtds/struts-2.3.dtd">

如果可以上网，自动缓存dtd，提供提示功能

如果不能上网，也可以配置本地DTD提示

\*\*\* 导入DTD时，应该和配置DTD版本一致

3、 关联struts2源码

关联 zip包

4、 Config Brower 插件使用

提供在浏览器中查看 struts2 配置加载情况

将解压struts2/lib/struts2-config-browser-plugin-2.3.7.jar 复制WEB-INF/lib下

访问 http://localhost:8080/struts2\_day1/config-browser/index.action 查看 struts2配置加载情况

## Struts2配置

学习路径

1)、 struts.xml常量配置（配置文件顺序）、Action访问（Servlet API）、结果集 （使用Struts2 编写简单案例）

2)、 请求数据

3)、 响应页面生成

### Struts2配置文件加载顺序

struts2 配置文件 由核心控制器加载 StrutsPrepareAndExecuteFilter (预处理，执行过滤)

nit\_DefaultProperties(); // [1] --- org/apache/struts2/default.properties

init\_TraditionalXmlConfigurations(); // [2] --- struts-default.xml, struts-plugin.xml, struts.xml

init\_LegacyStrutsProperties(); // [3] --- 自定义struts.properties

init\_CustomConfigurationProviders(); // [5] --- 自定义配置提供

init\_FilterInitParameters() ; // [6] --- web.xml

init\_AliasStandardObjects() ; // [7] --- Bean加载

结论 ：

default.properties 该文件保存在 struts2-core-2.3.7.jar 中 org.apache.struts2包里面 （常量的默认值）

struts-default.xml 该文件保存在 struts2-core-2.3.7.jar （Bean、拦截器、结果类型 ）

struts-plugin.xml 该文件保存在struts-Xxx-2.3.7.jar （在插件包中存在 ，配置插件信息 ）

struts.xml 该文件是web应用默认的struts配置文件 （实际开发中，通常写struts.xml ） \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

struts.properties 该文件是Struts的默认配置文件 (配置常量 )

web.xml 该文件是Web应用的配置文件 (配置常量 )

\* 后加载文件中struts2 常量会覆盖之前加载文件 常量内容

### Struts2的Action配置

1）必须要为<action>元素 配置<package>元素 （struts2 围绕package进行Action的相关配置 ）

配置package 三个常用属性

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

name 包名称，在struts2的配置文件文件中 包名不能重复 ，name并不是真正包名，只是为了管理Action

namespace 和 <action>的name属性，决定 Action的访问路径 （以/开始 ）

extends 继承哪个包，通常开发中继承 struts-default 包 （struts-default包在 struts-default.xml定义 ）

\* 继承struts-default包后，可以使用包中定义拦截器和结果类型

2）Action的通过<action>元素配置

<action name="hello" class="cn.itcast.struts2.demo1.HelloAction">

<action>的name 和 <package>的namespace属性 共同决定 Action的访问路径 ！！！！！！！！

例如 ：

<package name="default" namespace="/user" extends="struts-default">

<action name="hello" class="cn.itcast.struts2.demo1.HelloAction">

访问路径 /user/hello.action

3) <action> 元素配置默认值

<package> 的namespace 默认值 “”

<action> 的class 默认值 ActionSupport 类

<result> 的 name 默认值 success

默认Action 和 Action的默认处理类

1） 默认Action ， 解决客户端访问Action不存在的问题 ，客户端访问Action， Action找不到，默认Action 就会执行

<default-action-ref name="action元素的name" />

2） 默认处理类 ，客户端访问Action，已经找到匹配<action>元素，但是<action>元素没有class属性，执行默认处理类

<default-class-ref class="完全类名" />

\* 在struts-default.xml 配置默认处理类 ActionSupport

### Struts2的常量配置

1） struts2 默认常量 在 default.properties 中配置

2） 开发者自定义常量

struts.xml （要求）

格式 ： <constant name="struts.devMode" value="true" />

struts.properties （要求）

格式 ： struts.devMode = true

web.xml

格式 ：

<filter>

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

<init-param>

<param-name>struts.devMode</param-name>

<param-value>true</param-value>

</init-param>

</filter>

3） 常用常量

<constant name="struts.i18n.encoding" value="UTF-8"/> --- 相当于request.setCharacterEncoding("UTF-8"); 解决post请求乱码

<constant name="struts.action.extension" value="action"/> --- 访问struts2框架Action访问路径 扩展名 （要求）

\* struts.action.extension=action,, 默认以.action结尾扩展名 和 不写扩展名 都会分发给 Action

**<constant name="struts.serve.static.browserCache" value="false"/> false不缓存，true浏览器会缓存静态内容，产品环境设置true、开发环境设置false**

**<constant name="struts.devMode" value="true" /> 提供详细报错页面，修改struts.xml后不需要重启服务器 （开发要求）**

### struts2 配置文件分离

通过 <include file="struts-part1.xml"/> 将struts2 配置文件 拆分

## Action

HTTP请求 提交 Struts2 StrutsPrepareAndExecuteFilter 核心控制器 ------ 请求分发给不同Action

### Action书写的的三种格式

第一种 Action可以是 POJO （（PlainOldJavaObjects）简单的Java对象） ---- 不需要继承任何父类，实现任何接口

\* struts2框架 读取struts.xml 获得 完整Action类名

\* obj = Class.forName("完整类名").newInstance();

\* Method m = Class.forName("完整类名").getMethod("execute"); m.invoke(obj); 通过反射 执行 execute方法

第二种 编写Action 实现Action 接口

Action接口中，定义默认五种 逻辑视图名称

public static final String SUCCESS = "success"; // 数据处理成功 （成功页面）

public static final String NONE = "none"; // 页面不跳转 return null; 效果一样

public static final String ERROR = "error"; // 数据处理发送错误 (错误页面)

public static final String INPUT = "input"; // 用户输入数据有误，通常用于表单数据校验 （输入页面）

public static final String LOGIN = "login"; // 主要权限认证 (登陆页面)

\* 五种逻辑视图，解决Action处理数据后，跳转页面

第三种 编写Action 继承ActionSupport **（推荐）**

在Action中使用 表单校验、错误信息设置、读取国际化信息 三个功能

### Action的配置method(通配符)

1） 在配置 <action> 元素时，没有指定method属性， 默认执行 Action类中 execute方法

<action name="request1" class="cn.itcast.struts2.demo3.RequestAction1" />

2） 在<action> 元素内部 添加 method属性，指定执行Action中哪个方法

<action name="regist" class="cn.itcast.struts2.demo4.RegistAction" method="regist"/> 执行 RegistAction 的regist方法

\* 将多个请求 业务方法 写入到一个Action 类中

<action name="addBook" class="cn.itcast.struts2.demo4.BookAction" method="addBook" ></action>

<action name="delBook" class="cn.itcast.struts2.demo4.BookAction" method="delBook" ></action>

3） 使用通配符\* ，简化struts.xml配置

<a href="${pageContext.request.contextPath }/user/customer\_add.action">添加客户</a>

<a href="${pageContext.request.contextPath }/user/customer\_del.action">删除客户</a>

struts.xml

<action name="customer\_\*" class="cn.itcast.struts2.demo4.CustomerAction" method="{1}"></action> --- {1}就是第一个\* 匹配内容

### 动态方法调用

访问Action中指定方法，不进行配置

1） 在工程中使用 动态方法调用 ，必须保证 struts.enable.DynamicMethodInvocation = true 常量值 为true

2） 在action的访问路径 中 使用 "!方法名"

页面

<a href="${pageContext.request.contextPath }/user/product!add.action">添加商品</a>

配置

<action name="product" class="cn.itcast.struts2.demo4.ProductAction"></action>

执行 ProductAction 中的 add方法

### Action访问Servlet

1、 在Action 中解耦合方式 间接访问 Servlet API --------- 使用 ActionContext 对象

在struts2 中 Action API 已经与 Servlet API 解耦合 （没有依赖关系 ）

\* Servlet API 常见操作 ： 表单提交请求参数获取，向request、session、application三个范围存取数据

actionContext = ActionContext.getContext();

1) actionContext.getParameters(); 获得所有请求参数Map集合

2) actionContext.put("company", "传智播客"); / actionContext.get("company") 对request范围存取数据

3) actionContext.getSession(); 获得session数据Map，对Session范围存取数据

4) actionContext.getApplication(); 获得ServletContext数据Map，对应用访问存取数据

2、 使用接口注入的方式，操作Servlet API （耦合）

ServletContextAware ： 注入ServletContext对象

ServletRequestAware ： 注入 request对象

ServletResponseAware ： 注入response对象

\* 程序要使用哪个Servlet的对象，实现对应接口

3、 在Action中直接通过 ServletActionContext 获得Servlet API

ServletActionContext.getRequest() : 获得request对象 （session）

ServletActionContext.getResponse() : 获得response 对象

ServletActionContext.getServletContext() ： 获得ServletContext对象

\* 静态方法没有线程问题，ThreadLocal

## Result结果类型

Action处理请求后， 返回字符串(逻辑视图名), 需要在struts.xml 提供 <result>元素定义结果页面

1、 局部结果页面 和 全局结果页面

<action name="result" class="cn.itcast.struts2.demo6.ResultAction">

<!-- 局部结果 当前Action使用 -->

<result name="success">/demo6/result.jsp</result>

</action>

<global-results>

<!-- 全局结果 当前包中 所有Action都可以用-->

<result name="success">/demo6/result.jsp</result>

</global-results>

2、 结果页面跳转类型

\* 在struts-default.xml 定义了 一些结果页面类型

\* 使用默认type 是 dispatcher 转发 （request.getRequestDispatcher.forward）

1) dispatcher ： Action 转发给 JSP

2) chain ： Action调用另一个Action （同一次请求）

<result name="success" type="chain">hello</result> hello是一个Action的name

3） redirect ： Action重定向到 JSP

4） redirectAction ： Action重定向到另一个Action

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

## 总结

1、 struts2 环境搭建 （导入jar包、web.xml、 struts.xml ）

2、 struts2 运行流程

3、 配置文件加载顺序

4、 <package> <action> <result> 元素配置

5、 Action书写三种方式

6、 指定method方法调用、通配符、动态方法调用

7、 Action访问Servlet API

8.关于result标签的 type属性取值.

## 作业

登陆练习完成

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(20) NOT NULL,

`password` varchar(20) NOT NULL,

PRIMARY KEY (`id`)

)

INSERT INTO `user` VALUES ('1', 'admin', '123');

# 第二天

## Action处理请求参数

struts2 和 MVC 定义关系

StrutsPrepareAndExecuteFilter : 控制器

JSP : 视图

Action : 可以作为模型，也可以是控制器

struts2 Action 接受请求参数 ：属性驱动 和 模型驱动

### Action处理请求参数三种方式(两大种三小种)

第一种 ：Action 本身作为model对象，通过成员setter封装 （属性驱动 ）

页面：

用户名 <input type="text" name="username" /> <br/>

Action ：

public class RegistAction1 extends ActionSupport {

private String username;

public void setUsername(String username) {

this.username = username;

}

}

问题一： Action封装数据，会不会有线程问题 ？

\* 不会，struts2 Action 是多实例 ，为了在Action封装数据 （struts1 Action 是单例的 ）

问题二： 在使用第一种数据封装方式，数据封装到Action属性中，不可能将Action对象传递给 业务层

\* 需要再定义单独JavaBean ，将Action属性封装到 JavaBean

缺点：需要单独定义javaBean,将action中属性copy到javaBean中。

优点：简单。

这种方式 ，底层是通过反射来实现的。

第二种 ：创建独立model对象，页面通过ognl表达式封装 （属性驱动）

页面：

用户名 <input type="text" name="user.username" /> <br/> ----- 基于OGNL表达式的写法

Action：

public class RegistAction2 extends ActionSupport {

private User user;

public void setUser(User user) {

this.user = user;

}

public User getUser() {

return user;

}

}

问题： 谁来完成的参数封装

<interceptor name="params" class="com.opensymphony.xwork2.interceptor.ParametersInterceptor"/>

缺点：在页面上使用了ognl表达式，页面不通用了。

优点：简单，解决了第一种封装的问题。

第三种 ：使用ModelDriven接口，对请求数据进行封装 （模型驱动 ） ----- **主流**

页面：

用户名 <input type="text" name="username" /> <br/>

Action ：

public class RegistAction3 extends ActionSupport implements ModelDriven<User> {

private User user = new User(); // 必须手动实例化

public User getModel() {

return user;

}

}

\* struts2 有很多围绕模型驱动的特性

\* <interceptor name="modelDriven" class="com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor"/> 为模型驱动提供了更多特性

优点：解决了属性驱动存在的问题。

缺点：一次只能封装一个model对象。

对比第二种、第三种 ： 第三种只能在Action中指定一个model对象，第二种可以在Action中定义多个model对象

<input type="text" name="user.username" />

<input type="text" name="product.info" />

### 封装数据到Collection（List）和Map

1） 封装数据到Collection 对象

页面：

产品名称 <input type="text" name="products[0].name" /><br/>

产品名称 <input type="text" name="products[1].name" /><br/>

Action ：

public class ProductAction extends ActionSupport {

private List<Product> products;

public List<Product> getProducts() {

return products;

}

public void setProducts(List<Product> products) {

this.products = products;

}

}

2） 封装数据到Map 对象

页面：

产品名称 <input type="text" name="map['one'].name" /><br/> == one是map的键值

产品名称 <input type="text" name="map['two'].name" /><br/> == one是map的键值

Action :

public class ProductAction2 extends ActionSupport {

private Map<String, Product> map;

public Map<String, Product> getMap() {

return map;

}

public void setMap(Map<String, Product> map) {

this.map = map;

}

}

## Struts2类型转换（了解）

1、 struts2 内部提供大量类型转换器，用来完成数据类型转换问题

boolean 和 Boolean

char和 Character

int 和 Integer

long 和 Long

float 和 Float

double 和 Double

Date 可以接收 yyyy-MM-dd格式字符串

数组 可以将多个同名参数，转换到数组中

集合 支持将数据保存到 List 或者 Map 集合

案例： 输入合法年龄和生日可以自动转换

例如:日期类型，我们传递 yyyy-MM-dd yyyy年MM月dd日格式都可以，但是如果是yyyy/MM/d就会出现问题.

当输入abc 转换为 int类型 age时

Caused by: java.lang.NoSuchMethodException: cn.itcast.struts2.demo3.CustomerAction.setAge([Ljava.lang.String;

分析: 输入20 ，转换 int类型20 --- setAge(int)

输入abc，转换int 出错 ---- setAge(String) ----- 报错方法不存在异常

### 自定义类型转换器

步骤:

1.创建一个类实现TypeConverter接口.

2.重写接口中方法，实现类型转换操作.

3.注册类型转换器.

详解说明：

1.创建一个自定义类型转换器

第一种 实现TypeConverter接口

public Object convertValue(java.util.Map<java.lang.String,java.lang.Object> context, java.lang.Object target, java.lang.reflect.Member member, java.lang.String propertyName, java.lang.Object value, java.lang.Class toType)

如果实现接口，这个方法参数太多(6个)

第二种 不推荐实现接口，可以继承DefaultTypeConverter

public Object convertValue(java.util.Map<java.lang.String,java.lang.Object> context, java.lang.Object value, java.lang.Class toType)

第三种 个人推荐使用 继承DefaultTypeConverter类的一个子类StrutsTypeConverter.

原因:在这个类中将从页面传递的数据怎样封装，以及action中的数据怎样在页面上显示做了分离。

convertFromString(java.util.Map context, java.lang.String[] values, java.lang.Class toClass) --- 请求封装

convertToString(java.util.Map context, java.lang.Object o) --- 数据回显

\* 类型转换器 一直都是双向转换

页面提交请求参数，封装到model --- 需要转换

model数据 需要在页面 回显 ---- 需要转换

2.怎样注册一个自定义类型转换器.

1.局部--针对于action （针对属性）

配置文件所在位置以及名称：在Action类所在包 创建 Action类-conversion.properties

配置文件书写： 格式：属性名称=类型转换器的全类名

2.局部--针对于model （针对属性）

配置文件所在位置以及名称：在model类所在包 创建 model类名-conversion.properties

配置文件书写： 格式：属性名称=类型转换器的全类名

3.全局 （针对类型 ）

配置文件所在位置以及名称：在src下创建一个xwork-conversion.properties

配置文件书写： 格式：要转换的类型全名=类型转换器的全类名

例：以 1990/10/10 为例，自定义日期转换器，完成转换

public Object convertValue(Map<String, Object> context, Object value,

Class toType) {

// 根据toType判断 是请求封装 还是 数据回显

DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");

if (toType == Date.class) {

// 请求参数封装 （value是字符串）

String[] params = (String[]) value;

String strVal = params[0]; // 转换为 日期类型

try {

return dateFormat.parse(strVal);

} catch (ParseException e) {

e.printStackTrace();

}

} else {

// 回显(value是 Date)

Date date = (Date) value;

return dateFormat.format(date);

}

return null;

}

### 类型转换错误处理

注意:

对于struts2中类型转换器，如果表单数据提交时，将数据向model封装，出现了问题，会报错:

No result defined for action cn.itcast.action.RegistAction and result input

上面的意思是说，在RegistAction的配置中没有配置input结果视图.

<action name="regist" class="cn.itcast.action.RegistAction">

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

</action>

如果配置了，出现类型转换问题，就会跳转到input指定的视图。

问题：为什么会向input视图跳转?

是因为struts2中的拦截器(interceptor).

在struts2中的

<interceptor name="conversionError" class="org.apache.struts2.interceptor.StrutsConversionErrorInterceptor"/>

用于记录类型转换问题

在struts2中

<interceptor name="workflow" class="com.opensymphony.xwork2.interceptor.DefaultWorkflowInterceptor"/>

用于得到问题，向input视图跳转。

关于错误信息展示（错误信息变成中文）：

通过分析拦截器作用，得知当类型转换出错时，自动跳转input视图 ，在input视图页面中 <s:fieldError/> 显示错误信息

\* 在Action所在包中，创建 ActionName.properties，在局部资源文件中配置提示信息 ： invalid.fieldvalue.属性名=错误信息

例如：invalid.fieldvalue.birthday=日期格式要求yyyy/MM/dd

**如果是自定义类型转换器，出现类型转换问题，要跳转到input视图，在类型转换器中，必须抛出异常才可以。**

## 请求参数校验

校验的分类 ： 客户端数据校验 和 服务器端数据校验

客户端数据校验 ，通过JavaScript 完成校验 （改善用户体验，使用户减少出错 ）

服务器数据校验 ，使用框架内置校验功能 （struts2 内置校验功能 ） ----- 必须的

struts2 支持校验方式

代码校验 ：在服务器端通过编写java代码，完成数据校验

配置校验 ：XML配置校验（主流） 和 注解（annotation）配置校验

### 手工代码校验请求参数

步骤一： 封装数据

步骤二： 实现校验Action ，必须继承ActionSupport 类

步骤三： 覆盖validate方法，完成对Action的业务方法 数据校验

通过代码逻辑判断参数是否有效，如果参数非法 ， this.addFieldError (ActionSupport提供)

workflow拦截器跳转回input页面

步骤四： 在jsp中 通过 <s:fieldError/> 显示错误信息

<s:fielderror> 展示所有错误信息

<s:fielderror fieldName="">展示特定名称的错误信息.

\* validate方法会对Action中所有业务方法进行校验，如果只想校验某一个方法：

public void validate方法名()

### Xml配置方式数据校验

XML配置方式 数据校验 （企业主流校验）

代码校验不适用于大型项目，流程数据复杂时，开发量和维护量都会很大。

xml配置校验原理 ： 将很多校验规则代码已经写好，只需要在xml中定义数据所使用校验规则就可以了

步骤一 ：编写jsp

步骤二 ：编写Action 继承ActionSupport 或者 实现 Validateable 接口

步骤三 ：封装请求参数

\* 使用xml校验 必须提供get方法

步骤四 ：编写校验规则xml文件

在Action所在包 编写 Action类名-validation.xml 对Action所有业务方法进行校验

引入DTD

------ xwork-core-2.3.7.jar 中 xwork-validator-1.0.3.dtd

<!DOCTYPE validators PUBLIC

"-//Apache Struts//XWork Validator 1.0.3//EN"

"http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">

书写：

1.根元素

<validators>

2.子元素

<field name="属性名称"></field>

3.<field>子元素

<field-validator type="校验器"> 这个是指定校验器

4.<field-validator>子元素

<message>错误信息</message>

5.<field-validator>子元素

<param name="">值</param>

用于指定校验器中的参数

内置校验器定义文件

xwork-core-2.3.7.jar 中 /com/opensymphony/xwork2/validator/validators/default.xml

内建校验器

\* required (必填校验器,要求被校验的属性值不能为null)

\* requiredstring (必填字符串校验器,要求被校验的属性值不能为null，并且长度大于0,默认情况下会对字符串去前后空格)

\* stringlength (字符串长度校验器，要求被校验的属性值必须在指定的范围内，否则校验失败,minLength参数指定最小长度，maxLength参数指定最大长度，trim参数指定校验field之前是否去除字符串前后的空格)

\* regex (正则表达式校验器，检查被校验的属性值是否匹配一个正则表达式，expression参数指定正则表达式，caseSensitive参数指定进行正则表达式匹配时，是否区分大小写,默认值为true)

\* int (整数校验器，要求field的整数值必须在指定范围内，min指定最小值，max指定最大值)

\* double (双精度浮点数校验器,要求field的双精度浮点数必须在指定范围内,min指定最小值,max指定最大值)

\* fieldexpression (字段OGNL表达式校验器,要求field满足一个ognl表达式，expression参数指定ognl表达式,该逻辑表达式基于ValueStack进行求值,返回true时校验通过，否则不通过)

\* email (邮件地址校验器，要求如果被校验的属性值非空，则必须是合法的邮件地址)

\* url (网址校验器,要求如果被校验的属性值非空,则必须是合法的url地址)

\* date (日期校验器,要求field的日期值必须在指定范围内,min指定最小值,max指定最大值)

案例

required ：必填校验器

<field-validator type="required">

<message>性别不能为空!</message>

</field-validator>

requiredstring ：必填字符串校验器

<field-validator type="requiredstring">

<param name="trim">true</param>

<message>用户名不能为空!</message>

</field-validator>

stringlength ：字符串长度校验器

<field-validator type="stringlength">

<param name="maxLength">10</param>

<param name="minLength">2</param>

<param name="trim">true</param>

<message><![CDATA[产品名称应在2-10个字符之间]]></message>

</field-validator>

int ：整数校验器

<field-validator type="int">

<param name="min">1</param>

<param name="max">150</param>

<message>年龄必须在1-150之间</message>

</field-validator>

date : 日期校验器

<field-validator type="date">

<param name="min">1900-01-01</param>

<param name="max">2050-02-21</param>

<message>生日必须在${min}到${max}之间</message>

</field-validator>

url : 网络路径校验器

<field-validator type="url">

<message>传智播客的主页地址必须是一个有效网址</message>

</field-validator>

email ：邮件地址校验器

<field-validator type="email">

<message>电子邮件地址无效</message>

</field-validator>

regex ：正则表达式校验器

<field-validator type="regex">

<param name="regexExpression"><![CDATA[^13\d{9}$]]></param>

<message>手机号格式不正确!</message>

</field-validator>

fieldexpression : 字段表达式校验

<field-validator type="fieldexpression">

<param name="expression"><![CDATA[(password==repassword)]]></param>

<message>两次密码输入不一致</message>

</field-validator>

============================= 如何对指定的方法校验 ？？？

格式：Action类名-ActionName(<action>元素name属性)-validation.xml

例如：校验AddCustomerAction中execute方法

配置 <action name="addcustomer" .../> 校验文件名字： AddCusotmerAction-addcustomer-validation.xml

### 自定义校验规则(了解一下就行)

步骤一： 自定义校验器 必须实现 Validator 接口

通常自定义校验器 继承 ValidatorSupport 和 FieldValidatorSupport

\* ValidatorSupport 针对不是一个输入字段 (两个密码一致)

\* FieldValidatorSupport 针对是一个输入字段 (用户名非空)

步骤二： 注册校验器

在工程的src下新建validators.xml文件

引入 xwork-core-2.3.7.jar 中 xwork-validator-config-1.0.dtd

步骤三 ：使用校验器

在Action所有包 创建Action类名-validation.xml

\*\*\*\* 实际开发中很少用到自定义校验器

# 第三天

## Struts2国际化（了解）

1、 国际化原理 ？ 什么是国际化 ？

同一款软件 可以为不同用户，提供不同语言界面 ---- 国际化软件

需要一个语言资源包（很多properties文件，每个properties文件 针对一个国家或者语言 ，通过java程序根据来访者国家语言，自动读取不同properties文件 ）

2、 资源包编写

properties文件命名 ： 基本名称\_语言（小写）\_国家（大写）.properties

例如 ：

messages\_zh\_CN.properties 中国中文

messages\_en\_US.properties 美国英文

3、 ResourceBundle 根据不同Locale（地域信息），读取不同国家 properties文件

ResourceBundle bundle = ResourceBundle.getBundle("messages", Locale.US);

### 国际化配置

1.properties文件怎样声明

1.全局

位置:任意

需要在struts.xml文件中声明常量。

2.局部

1.针对于action

位置:与action类在同一个包下

名称:actionClassNam.properties

2.针对于包

位置:在指定的包下.

称:package.properties

3.在jsp页面临时使用

位置:任意

在jsp页面上使用<s:i18n name="">

2.在struts2中什么位置使用国际化

1.action类中

2.validation配置文件

3.jsp页面

3.在struts2中怎样使用国际化

getText();

<message key="">

<s:text>

第一种 全局国际化信息文件 （所有Action都可以使用 ） ------- 最常用

\* properties文件可以在任何包中

\* 需要在struts.xml 中配置全局信息文件位置

struts.xml

<constant name="struts.custom.i18n.resources" value="messages"></constant> messages.properties 在src根目录

<constant name="struts.custom.i18n.resources" value="cn.itcast.resources.messages"></constant> messages.properties 在 cn.itcast.resources 包

国际化信息

在Action中使用 ：this.getText("msg");

在jsp中使用 ：<s:text name="msg" />

在配置文件中（校验xml） ： <message key="agemsg"></message>

第二种 Action范围信息文件 （只能在某个Action中使用 ）

数据只能在对应Action中使用，在Action类所在包 创建 Action类名.properties --------- 无需配置

第三种 package范围信息文件 （package中所有Action都可以使用 ）

数据对包 （包括子包）中的所有Action 都有效 ， 在包中创建 package.properties ----- 无需配置

第四种 临时信息文件 （主要在jsp中 引入国际化信息 ）

在jsp指定读取 哪个properties文件

<s:i18n name="cn.itcast.struts2.demo7.package">

<s:text name="customer"></s:text>

</s:i18n>

怎样在struts2中操作国际化?(怎样使用)

1.在action类中使用

前提：action类要继承ActionSupport类。

getText(String name)就可以获取配置文件中对应名称的值。

2.在validation.xml文件中使用

<message key="名称"/>

3.在jsp页面上使用

<s:text name="名称"> 如果没有使用<s:i18n name="">来指定，会从全局配置文件中获取。

如果要从某一个配置文件中获取，通过<s:i18n name="">来指定，包名.配置文件名称。

例如：

<s:i18n name="cn.itcast.action.package">

<s:text name="nameerror" />

</s:i18n>

\*\*\* 向信息中传递参数 {0} {1} ------------ MessageFormat 动态消息文本

在struts2中国际化配置文件中使用动态文本

1.action中怎样使用

msg=hello world {0}

this.getText("msg",new String[]{"tom"})

结果就是 hello world tom

2.jsp页面上怎样使用

msg=hello world {0}

<s:i18n name="cn.itcast.action.I18nDemo1Action">

<s:text name="msg">

<s:param>张三</s:param>

</s:text>

</s:i18n>

结果就是 hello world 张三.

## Struts2拦截器

### 拦截器介绍

拦截器 的使用 ，源自Spring AOP（面向切面编程）思想

拦截器 采用 责任链 模式

\* 在责任链模式里,很多对象由每一个对象对其下家的引用而连接起来形成一条链。

\* 责任链每一个节点，都可以继续调用下一个节点，也可以阻止流程继续执行

在struts2 中可以定义很多个拦截器，将多个拦截器按照特定顺序 组成拦截器栈 （顺序调用 栈中的每一个拦截器 ）

1、 struts2 所有拦截器 都必须实现 Interceptor 接口

2、 AbstractInterceptor 类实现了 Interceptor 接口. 并为 init, destroy 提供了一个空白的实现

详细使用请看课堂笔记和源代码！

所有实际开发中，自定义拦截器 只需要 继承 AbstractInterceptor类， 提供 intercept 方法实现

3、 常用struts2 拦截器

<interceptor-ref name="modelDriven"/> 模型驱动

<interceptor-ref name="fileUpload"/> 文件上传

<interceptor-ref name="params"> 参数解析封装

<interceptor-ref name="conversionError"/> 类型转换错误

<interceptor-ref name="validation"> 请求参数校验

<interceptor-ref name="workflow"> 拦截跳转 input 视图

### 自定义拦截器案例

案例 ： 登陆，对其它Action访问 通过自定义拦截器 进行权限控制

导入jar包 （struts2 jar、c3p0、 dbutils、 mysql驱动）

web.xml

struts.xml

JDBCUtils 工具类

第一步 ： 编写index.jsp 提供 图书增删改查 四个功能

编写BookAction ，提供四个业务方法

第二步： 完成登陆功能

第三步 ：必须要登陆 才能进行图书管理

使用Filter 进行权限控制 ---- 过滤所有web请求 （所有web资源访问）

使用拦截器 进行权限控制 ---- 主要拦截对Action访问 （不能拦截JSP）

定义拦截器 继承AbstractInterceptor

配置拦截器

**方式一**

<!-- 注册拦截器，拦截add, update, delete，不拦截search -->

<interceptors>

<interceptor name="bookInterceptor" class="cn.itcast.interceotor.BookInterceptor">

<param name="includeMethods">add, update, delete</param>

<param name="excludeMethods">search</param>

</interceptor>

<interceptor-stack name="BookStack">

<interceptor-ref name="defaultStack" />

<interceptor-ref name="bookInterceptor" />

</interceptor-stack>

</interceptors>

<action name="book\_\*" class="cn.itcast.action.BookAction"

method="{1}">

<interceptor-ref name="BookStack" />

</action>

方式二

<!-- 注册拦截器 -->

<interceptors>

<interceptor name="privilege" class="cn.itcast.interceptor.PrivilegeInterceptor">

<param name="includeMethods">add, update, delete</param>

<param name="excludeMethods">search</param>

</interceptor>

<!-- 自定义拦截器栈 -->

<interceptor-stack name="privilegeStack">

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

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

</interceptor-stack>

</interceptors>

<!-- 设置当前包 所有Action 都使用 自定义拦截器栈 -->

<default-interceptor-ref name="privilegeStack"></default-interceptor-ref>

## Struts2文件上传下载

### Struts2文件上传

提供 FileUpload 拦截器，用于解析 multipart/form-data 编码格式请求，解析上传文件的内容

fileUpload拦截器 默认在 defaultStack 栈中， 默认会执行的

在Action需要对上传文件内容进行接收

页面：

<input type="file" name="upload" />

Action　：

public class UploadAction extends ActionSupport {

// 接收上传内容

// <input type="file" name="upload" />

private File upload; // 这里变量名 和 页面表单元素 name 属性一致

private String uploadContentType;

private String uploadFileName;

}

\* 格式 ： 上传表单项name属性 + ContentType 、 上传表单项name属性 + FileName

\* 为三个对象 提供 setter 方法

通过FileUtils 提供 copyFile 进行文件复制，将上传文件 保存到服务器端

### Struts2文件上传问题解决

配置 input 视图 ，作为上传出错后 跳转页面

在文件上传时，如果发生错误 ，fileUpload拦截器 会设置错误信息，workflow拦截器 跳转到 input 视图

struts.multipart.parser=jakarta 定义文件上传，采用 commons-fileupload 技术

\* 同时支持 cos 、pell 上传技术 （如果使用其它上传技术，单独下载jar包 ）

通过 struts.multipart.maxSize 常量设置文件上传总大小限制

\* struts.multipart.maxSize=2097152 默认上传文件总大小 2MB

\* 超过文件总大小，跳转input 视图， 通过 <s:actionError /> 回显错误信息

在struts.xml 设置上传总大小

<constant name="struts.multipart.maxSize" value="20971520"></constant>

设置上传文件总大小，对所有上传form有效，只想对当前form进行设置，可以设置fileUpload拦截器属性

FileUpload 拦截器有 3 个属性可以设置.

\* maximumSize: 上传文件的最大长度(以字节为单位), 默认值为 2 MB

\* allowedTypes: 允许上传文件的类型, 各类型之间以逗号分隔

\* allowedExtensions: 允许上传文件扩展名, 各扩展名之间以逗号分隔

如果针对fileUpload 进行参数设置，当出错时，在页面通过 <s:fieldError /> 回显错误信息

例如：

<action name="upload" class="cn.itcast.action.UploadAction">

<interceptor-ref name="defaultStack">

<param name="fileUpload.maximumSize"></param>

<param name="fileUpload.allowedTypes"></param>

<param name="fileUpload.allowedExtensions">text, mp3, avi</param>

</interceptor-ref>

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

</action>

struts-messages.properties 文件里预定义 上传错误信息，通过覆盖对应key 显示中文信息

struts.messages.error.uploading=Error uploading: {0}

struts.messages.error.file.too.large=The file is to large to be uploaded: {0} "{1}" "{2}" {3}

struts.messages.error.content.type.not.allowed=Content-Type not allowed: {0} "{1}" "{2}" {3}

struts.messages.error.file.extension.not.allowed=File extension not allowed: {0} "{1}" "{2}" {3}

修改为

struts.messages.error.uploading=上传错误: {0}

struts.messages.error.file.too.large=上传文件太大: {0} "{1}" "{2}" {3}

struts.messages.error.content.type.not.allowed=上传文件的类型不允许: {0} "{1}" "{2}" {3}

struts.messages.error.file.extension.not.allowed=上传文件的后缀名不允许: {0} "{1}" "{2}" {3}

{0}：<input type=“file” name=“uploadImage”>中name属性的值

{1}：上传文件的真实名称

{2}：上传文件保存到临时目录的名称

{3}：上传文件的类型(对struts.messages.error.file.too.large是上传文件的大小)

### 多文件上传

**第一步：在WEB-INF/lib下加入commons-fileupload-1.2.1.jar、commons-io-1.3.2.jar。这两个文件可以从http://commons.apache.org/下载。**

**第二步：把form表的enctype设置为：“multipart/form-data“，如下：**

<form enctype="multipart/form-data" action="${pageContext.request.contextPath}/xxx.action" method="post">

<input type="file" name="uploadImages">

<input type="file" name="uploadImages">

</form>

**第三步：在Action类中添加以下属性，属性红色部分对应于表单中文件字段的名称：**

public class uploadAction{

private File[] uploadImages;//得到上传的文件

private String[] uploadImagesContentType;//得到文件的类型

private String[] uploadImagesFileName;//得到文件的名称

//这里略省了属性的getter/setter方法

public String saveFiles() throws Exception{

ServletContext sc = ServletActionContext.getServletContext();

String realpath = sc.getRealPath("/uploadfile");

try {

if(uploadImages!=null&&uploadImages.length>0){

for(int i=0;i<uploadImages.length;i++){

File destFile = new File(realpath,uploadImageFileNames[i]);

FileUtils.copyFile(uploadImages[i], destFile);

}

}

} catch (IOException e) {

e.printStackTrace();}return "success";}}

### Struts2文件下载

1） struts2 完成文件下载，通过 结果集类型 （Result Type） stream 来完成的

struts-default.xml 定义 <result-type name="stream" class="org.apache.struts2.dispatcher.StreamResult"/>

2） 使用Stream结果集 完成文件下载

文件下载原理： 服务器读取下载文件内容，通过Response响应流写回， 设置 ContentType、 ContentDisposition 头信息

public class StreamResult extends StrutsResultSupport {

protected String contentType = "text/plain"; // contentType头信息 （下载文件对应 MIME协议规定类型 ）

\* .html --- text/html . txt--- text/plain

protected String contentDisposition = "inline"; // ContentDisposition头信息 (下载文件打开方式 inline浏览器内部打开, attachment 以附件形式打开)

protected String inputStream= "inputStream"; // 需要Action中 提供 getInputStream 方法 返回 InputStream 提供下载文件 内容

}

Action 提供 InputStream 返回值 getInputStream 方法 ------- 指定下载文件流

配置 stream 结果集 参数 <param name="contentType">${contentType}</param> ---- 在Action 中提供 getContentType

\* ServletActionContext.getServletContext().getMimeType(filename);

配置 stream 结果集 参数 <param name="contentDisposition">attachment;filename=${filename}</param> ---- 在Action 提供 getFilename

\* 下载附件名乱码问题 ， IE和火狐 解决不同

public String encodeDownloadFilename(String filename, String agent)

throws IOException {

if (agent.contains("Firefox")) { // 火狐浏览器

filename = "=?UTF-8?B?"

+ new BASE64Encoder().encode(filename.getBytes("utf-8"))

+ "?=";

} else { // IE及其他浏览器

filename = URLEncoder.encode(filename, "utf-8");

}

return filename;

}

## OGNL表示式使用 和 值栈

OGNL是Object Graphic Navigation Language（对象图导航语言）的缩写，它是一个开源项目。 Struts2框架使用OGNL作为默认的表达式语言。

\* xwork 提供 OGNL表达式

\* ognl-3.0.5.jar

OGNL 是一种比EL 强大很多倍的语言

OGNL 提供五大类功能

1、支持对象方法调用，如xxx.doSomeSpecial()；

2、支持类静态的方法调用和值访问

3、访问OGNL上下文（OGNL context）和ActionContext； （重点 操作ValueStack值栈 ）

4、支持赋值操作和表达式串联

5、操作集合对象。

### 1、 使用OGNL访问 对象方法 和 静态方法

\* OGNL 在jsp 结合 struts2 标签库 使用 ， <s:property value="ognl表达式" /> 执行 ognl表达式

调用 实例方法 ： 对象.方法() ---- <s:property value="'hello,world'.length()"/>

调用 静态方法 ： @[类全名（包括包路径）]@[方法名] --- <s:property value="@java.lang.String@format('您好,%s','小明')"/>

\* 使用 静态方法调用 必须 设置 struts.ognl.allowStaticMethodAccess=true

### 2、 访问OGNL上下文（OGNL context）和ActionContext

OGNL上下文（OGNL context） 对象 ----- 值栈 ValueStack

问题一 ： 什么是值栈 ValueStack ?

ValueStack 是 struts2 提供一个接口，实现类 OgnlValueStack ---- 值栈对象 （OGNL是从值栈中获取数据的 ）

每个Action实例都有一个ValueStack对象 （一个请求 对应 一个ValueStack对象 ）

在其中保存当前Action 对象和其他相关对象 （值栈中 是有Action 引用的 ）

Struts 框架把 ValueStack 对象保存在名为 “struts.valueStack” 的请求属性中,request中 （值栈对象 是 request一个属性）

问题二 ： 值栈的内部结构 ？

值栈由两部分组成

ObjectStack: Struts 把动作和相关对象压入 ObjectStack 中--List

ContextMap: Struts 把各种各样的映射关系(一些 Map 类型的对象) 压入 ContextMap 中

Struts 会把下面这些映射压入 ContextMap 中

parameters: 该 Map 中包含当前请求的请求参数

request: 该 Map 中包含当前 request 对象中的所有属性

session: 该 Map 中包含当前 session 对象中的所有属性

application:该 Map 中包含当前 application 对象中的所有属性

attr: 该 Map 按如下顺序来检索某个属性: request, session, application

ValueStack中 存在root属性 (CompoundRoot) 、 context 属性 （OgnlContext ）

\* CompoundRoot 就是ArrayList

\* OgnlContext 就是 Map

context 对应Map 引入 root对象

\* context中还存在 request、 session、application、 attr、 parameters 对象引用

\* OGNL表达式，访问root中数据时 不需要 #, 访问 request、 session、application、 attr、 parameters 对象数据 必须写 #

\* 操作值栈 默认指 操作 root 元素

问题三 ： 值栈对象的创建 ，ValueStack 和 ActionContext 是什么关系 ？

值栈对象 是请求时 创建的

doFilter中 prepare.createActionContext(request, response);

\* 创建ActionContext 对象过程中，创建 值栈对象ValueStack

\* ActionContext对象 对 ValueStack对象 有引用的 （在程序中 通过 ActionContext 获得 值栈对象 ）

Dispatcher类 serviceAction 方法中 将值栈对象保存到 request范围

request.setAttribute(ServletActionContext.STRUTS\_VALUESTACK\_KEY, proxy.getInvocation().getStack());

问题四 ： 如何获得值栈对象

获得值栈对象 有两种方法

ValueStack valueStack = (ValueStack) ServletActionContext.getRequest().getAttribute(ServletActionContext.STRUTS\_VALUESTACK\_KEY);

ValueStack valueStack2 = ActionContext.getContext().getValueStack();

问题五： 向值栈保存数据 （主要针对 root）

两种方式

// 将数据保存root的索引0位置，放置到第一个元素 ArrayList add(0,element);

valueStack.push("itcast");

// 在值栈创建参数map， 将数据保存到map中

valueStack.set("company", "传智播客");

在jsp中 通过 <s:debug /> 查看值栈的内容

问题六： 在JSP中获取值栈的数据

访问root中数据 不需要#

访问 其它对象数据 加 #

通过下标获取root中对象

<s:property value="[0].top"/> //取值栈顶对象

直接在root中查找对象属性 （自上而下自动查找）

valueStack:<s:property value="username"/>

在OgnlContext中获取数据

request:<s:property value="#request.username"/>

session:<s:property value="#session.username"/>

application:<s:property value="#application.username"/>

attr:<s:property value="#attr.username"/>

parameters:<s:property value="#parameters.cid[0]"/>

# 第四天

## OGNL表达式使用和值栈

### 1、 值栈在开发中应用

主流应用 ： 值栈 解决 Action 向 JSP 传递 数据问题

Action 向JSP 传递数据处理结果 ，结果数据有两种形式

1） 消息 String类型数据

this.addFieldError("msg", "字段错误信息");

this.addActionError("Action全局错误信息");

this.addActionMessage("Action的消息信息");

\* fieldError 针对某一个字段错误信息 （常用于表单校验）、actionError (普通错误信息，不针对某一个字段 登陆失败)、 actionMessage 通用消息

在jsp中使用 struts2提供标签 显示消息信息

<s:fielderror fieldName="msg"/>

<s:actionerror/>

<s:actionmessage/>

2） 数据 （复杂类型数据）

使用值栈 valueStack.push(products);

哪些数据默认会放入到值栈 ？？？

1）每次请求，访问Action对象 会被压入值栈 ------- DefaultActionInvocation 的 init方法 stack.push(action);

\* Action如果想传递数据给 JSP，只有将数据保存到成员变量，并且提供get方法就可以了

2）ModelDriven 接口 有一个单独拦截器

<interceptor name="modelDriven" class="com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor"/>

在拦截器中 ，将model对象 压入了 值栈 stack.push(model);

\* 如果Action 实现ModelDriven接口，值栈默认栈顶对象 就是model对象

### 2、 值栈的数据 通过EL访问

问题七：为什么 EL也能访问值栈中的数据

StrutsPreparedAndExecuteFilter的doFilter代码中 request = prepare.wrapRequest(request);

\* 对Request对象进行了包装 ，StrutsRequestWrapper

\* 重写request的 getAttribute

Object attribute = super.getAttribute(s);

if (attribute == null) {

attribute = stack.findValue(s);

}

访问request范围的数据时，如果数据找不到，去值栈中找

* request对象 具备访问值栈数据的能力 （查找root的数据）

### 3、 OGNL表达式 常见使用

#、 % 、$ 符号使用

**1） # 的 使用**

用法一 # 代表 ActionContext.getContext() 上下文

<s:property value="#request.name" /> ---->ActionContext().getContext().getRequest().get("name");

#request

#session

#application

#attr

#parameters

用法二 ： 不写# 默认在 值栈中root中进行查找

<s:property value="name" /> 在root中查找name属性

\* 查询元素时，从root的栈顶元素 开始查找， 如果访问指定栈中元素 <s:property value="[1].name" /> 访问栈中第二个元素name属性

\* 访问第二个元素对象 <s:property value="[1].top" />

用法三 ：进行投影映射 （结合复杂对象遍历 ）

1）集合的投影(只输出部分属性

<h1>遍历集合只要name属性</h1>

<s:iterator value="products.{name}" var="pname">

<s:property value="#pname"/>

</s:iterator>

2）遍历时，对数据设置条件

<h1>遍历集合只要price大于1500商品</h1>

<s:iterator value="products.{?#this.price>1500}" var="product">

<s:property value="#product.name"/> --- <s:property value="#product.price"/>

</s:iterator>

3）综合

<h1>只显示价格大于1500 商品名称</h1>

<s:iterator value="products.{?#this.price>1500}.{name}" var="pname">

<s:property value="#pname"/>

</s:iterator>

用法四： 使用#构造map集合

经常结合 struts2 标签用来生成 select、checkbox、radio

<h1>使用#构造map集合 遍历</h1>

<s:iterator value="#{'name':'aaa','age':'20', 'hobby':'sport' }" var="entry">

key : <s:property value="#entry.key"/> , value: <s:property value="#entry.value"/> <br/>

</s:iterator>

**2) %的使用**

用法一： 结合struts2 表单表单使用， 通过%通知struts， %{}中内容是一个OGNL表达式，进行解析

<s:textfield name="username" value="%{#request.username}"/>

用法二： 设置ognl表达式不解析 %{'ognl表达式'}

<s:property value="%{'#request.username'}"/>

**3）$的使用**

用法一 ：用于在国际化资源文件中，引用OGNL表达式

在properties文件 msg=欢迎您, ${#request.username}

在页面

<s:i18n name="messages">

<s:text name="username"></s:text>

</s:i18n>

\* 自动将值栈的username 结合国际化配置信息显示

用法二 ：在Struts 2配置文件中，引用OGNL表达式

<!-- 在Action 提供 getContentType方法 -->

<param name="contentType">${contentType}</param>

\* ${contentType} 读取值栈中contentType数据，在Action提供 getContentType 因为Action对象会被压入值栈，

contentType是Action属性，从值栈获得

结论： #使用ognl表达式获取数据，% 控制ognl表达式是否解析 ，$ 用于配置文件获取值栈的数据

## struts2 标签库

tag-reference.html 就是 struts2 标签规范

### 1、 通用标签库 的学习

<s:property> 解析ognl表达式，设置默认值，设置内容是否HTML转义

<s:set> 向四个数据范围保存数据

<s:iterator> 遍历值栈中数据

<s:if> <s:elseif> <s:else> 进行条件判断 -------- elseif 可以有多个

<s:url> 进行URL重写（追踪Session ） ，结合s:param 进行参数编码

\* <s:url action="download" namespace="/" var="myurl">

<s:param name="filename" value="%{'MIME协议简介.txt'}"></s:param>

</s:url>

<s:property value="#myurl"/>

<s:a> 对一个链接 进行参数编码

\* <s:a action="download" namespace="/" >下载MIME协议简介.txt

<s:param name="filename" value="%{'MIME协议简介.txt'}"></s:param>

</s:a>

OGNL 了解部分 ： 支持赋值操作和表达式串联 、 操作集合对象

1） 在值栈中保存一个对象

<s:property value="price=1000,name='冰箱',getPrice()"/> 自动查找值栈中 price 和 name 属性 为其赋值

2） ognl操作集合

<s:property value="products[0].name"/> 访问集合第一个元素name属性

<s:property value="map['name']"/> 访问map中key为name的值

{} 直接构造List 元素、 #{} 直接构造 Map元素

<s:iterator value="{'aaa','bbb'}" var="s">

<s:property value="#s"/>

</s:iterator>

<s:iterator value="#{'ccc':'111','ddd':'222' }" var="entry">

<s:property value="#entry.key"/>

</s:iterator>

### 2、 UI标签库的学习 （Form标签）

使用struts2 form标签 好处 ： 支持数据回显 ， 布局排班（基于Freemarker 模板定义 ）

struts2 表单标签 value属性。 必须写 %{} 进行设值

\*\*\*\*\*\*\* 使用struts2表单标签前， 必须配置 StrutsPrepareAndExecuteFilter

The Struts dispatcher cannot be found. This is usually caused by using Struts tags without the associated filter. Struts tags are only usable when the request has passed through its servlet filter, which initializes the Struts dispatcher needed for this tag

<s:form> 表单标签

<s:form action="regist" namespace="/" method="post" theme="xhtml"> --- theme="xhtml" 默认布局样式

<s:textfield> 生成 <input type="text" >

<s:password > 生成 <input type="password" >

<s:submit type="submit" value="注册"/> 生成 <input type="submit" >

<s:reset type="reset" value="重置" /> 生成 <input type="reset" >

\* struts2 除了支持JSP之外，支持两种模板技术 Velocity (扩展名 .vm ) 、 Freemarker （扩展名 .ftl）

<s:textarea> 生成 <textarea> 多行文本框

<s:checkboxlist> 生成一组checkbox

\* 使用ognl构造 Map （看到值和提交值 不同时）

\* <s:checkboxlist list="#{'sport':'体育','read':'读书','music':'音乐' }" name="hobby"></s:checkboxlist>

<s:radio> 生成一组radio

\* 使用 ognl构造 List （看到内容和提交值 相同时）

\* <s:radio list="{'男','女'}" name="gender"></s:radio>

<s:select> 生成一个<select>

\* <s:select list="{'北京','上海','南京','广州'}" name="city"></s:select>

============= struts2 开发 密码框 默认不回显

<s:password name="password" id="password" showPassword="true"/>

3、 页面元素主题设置

default.properties ---- struts.ui.theme=xhtml 设置struts2 页面元素使用默认主题

struts.ui.templateSuffix=ftl 默认模板引擎 Freemarker

修改主题

方式一 ：<s:textfield name="username" label="用户名“ theme="simple"></s:textfield> 只对当前元素有效

方式二 ：<s:form action="" method="post" namespace="/ui“ theme="simple"> 对form中所有元素有效

方式三 ： struts.xml

<constant name="struts.ui.theme" value="simple"></constant> 修改默认主题样式，页面所有元素都有效

优先级 ： 方式一 > 方式二 > 方式三

## 防止表单重复提交原理

表单防止重复提交

表单重复提交 危害： 刷票、 重复注册、带来服务器访问压力（拒绝服务）

1、 在jsp 通过 <s:token /> 生成令牌号

生成表单隐藏域

将令牌号保存到Session

2、 通过struts2 提供 tokenIntercetor 拦截器 完成请求中令牌号 和 session中令牌号 比较

<interceptor name="token" class="org.apache.struts2.interceptor.TokenInterceptor"/>

<action name="token" class="cn.itcast.struts2.TokenAction">

<result>/index.jsp</result>

<!-- 重新定义拦截器 -->

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

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

</action>

3、 当表单重复提交时，token拦截器自动跳转 result name="invalid.token"

通过 <s:actionError/> 显示错误信息

覆盖重复提交信息 struts.messages.invalid.token=您已经重复提交表单，请刷新后重试

## Struts2 内置json插件

知识点 ：struts2 的 Ajax 开发

Ajax开发客户端 和 服务器交互数据格式 --------------- json

\* json 是最轻量级，体积最小

服务器将程序处理结果，转换为json格式发送给 客户端

\* json-lib 、 flexjson 工具类库

\* struts2-json-plugin-2.3.7.jar

案例一： 输入用户名，鼠标点击密码（触发用户名元素离焦事件），使用Ajax 将用户名发送到服务器 判断是否存在

jquery 1.4 、 1.6 新特性比较多 （企业主流 1.4 ）

使用struts2 json插件

要点1 ： <package> extends 继承 json-default

要点2 ： <result> type 类型写 json

\*\*\*\*\*\*\*\*\*\* struts2 json插件 ，默认将值栈root顶端对象 所有属性返回（get方法）

不想将company属性返回 ，在get方法上 @JSON(serialize=false)

案例二 ：服务器将 商品对象 List列表返回

如果Action 实现ModelDriven ， model对象就是值栈栈顶对象，struts2 json插件默认 将model返回

通过设置root属性，修改插件返回 根对象

\* <param name="root">action</param> 将Action作为根对象返回

只想要每个商品的 name 属性

方案一： 在pnum 、price 的get方法上 添加 @JSON(serialize=false) ========= 只要@JSON注解，属性将永远不能参与 json返回

方案二： 设置 includeProperties 属性

* <param name="includeProperties">products\[\d+\]\.name</param>

# 第五天

## 一、 项目本地导入

MyEclipse 新建 web project ， 覆盖对应 src 和 WebRoot

Eclipse 新建 Dynamic web project ， 覆盖src ， 将WebRoot 中内容复制 WebContent 目录

JavaEE 企业级应用软件，布局经常采用 Frameset ， 左侧菜单树使用Dztree js组件制作的

目标功能 ：

**1、 登陆**

**2、 添加用户 （简历上传）**

**3、 组合条件 员工信息列表查询**

**4、 员工信息详情查看（简历下载）**

**5、 员工信息删除**

**6、 员工信息编辑**

## 二、 数据库设计

#新建数据库

create database struts2exec;

#创建用户

create user struts2@localhost identified by 'struts2';

#授权

grant all on struts2exec.\* to struts2@localhost;

\*\*\*\*\* Oracle和MySQL 作为应用数据库区别

mysql存在数据库概念，在企业开发中，针对一个项目创建一个单独数据库，创建单独用户， 为用户授予数据库权限 ，

oracle 一个数据库就是一个服务，在这个库中可以存在很多用户，每个用户有单独表空间 ，针对一个项目，只需要创建一个用户

#用户表

CREATE TABLE S\_User(

userID INT NOT NULL AUTO\_INCREMENT, #主键ID

userName VARCHAR(50) NULL, #用户姓名

logonName VARCHAR(50) NULL, #登录名

logonPwd VARCHAR(50) NULL, #密码#

sex VARCHAR(10) NULL, #性别（例如：男，女）

birthday VARCHAR(50) NULL, #出生日期

education VARCHAR(20) NULL, #学历（例如：研究生、本科、专科、高中）

telephone VARCHAR(50) NULL, #电话

interest VARCHAR(20) NULL, #兴趣爱好（例如：体育、旅游、逛街）

path VARCHAR(500) NULL, #上传路径（path路径）

filename VARCHAR(100) NULL, #上传文件名称（文件名）

remark VARCHAR(500) NULL, #备注

PRIMARY KEY (userID)

);

#初始化数据：默认用户名和密码是admin

INSERT INTO s\_user (userID,userName,logonName,logonPwd) VALUES (1,'超级管理员','admin','admin');

## 三、 搭建开发环境

struts2 + javabean + DAO + C3P0 + DBUtils + MySQL

导入jar包 和 配置文件

创建包结构

cn.itcast.user.domain

cn.itcast.user.dao

cn.itcast.user.service

cn.itcast.user.web.action

cn.itcast.user.utils

## 四、 功能开发 ---- 登陆功能

内网应用系统通常不需要验证码技术

1、登陆页面 WebRoot/login/login.jsp

form元素 改造 <s:form>

\* form的action 提交路径 怎么写 ？

企业中为了Action 代码可维护性，通常一个业务模块（Model）使用 一个Action

将登陆、 员工增删改查 对应业务方法 写入到 同一个Action

2、登陆表单数据校验

用户名 非空，3-12位

密码 非空

3、完成登陆逻辑，session保存用户， 跳转主页

top.jsp ${user.userName } 显示当前登陆用户

## 五、 功能开发 ---- 员工添加

1、 员工添加页面 WebRoot/user/add.jsp

form元素 改造 <s:form>

日期输入，使用 jquery datapicker控件

\* $('#birthday').datepick({dateFormat: 'yy-mm-dd'});

<s:form action="user\_add"> 访问 UserAction 中 add方法

2、 数据校验

校验出现input跳转问题，登陆页面校验失败跳转 login.jsp , 员工添加校验失败跳转 add.jsp ---- 只能配置一个input

\* <interceptor name="workflow" class="com.opensymphony.xwork2.interceptor.DefaultWorkflowInterceptor"/> 完成跳转

可以 通过 @InputConfig注解，改为校验失败后 跳转视图

3、 上传简历

如果真实文件名保存，出现覆盖问题 ------------ 唯一文件名 UUID

\* 在DB保存 uuid文件名路径 和 真实文件名

## 六、 组合条件查询 -- 查询员工列表

1、 查询页面 WebRoot/user/list.jsp

form 改造 <s:form>

<s:form action="user\_list" > 提交请求 UserAction 的 list方法

2、 条件组合查询

SQL语句 动态拼接

String sql = "select \* from s\_user where 1=1 ";

List<String> argList = new ArrayList<String>(); // 参数列表

if (user.getUserName() != null

&& user.getUserName().trim().length() > 0) {

sql += "and userName like ? ";

argList.add("%" + user.getUserName() + "%");

}

if (user.getSex() != null && user.getSex().trim().length() > 0) {

sql += "and sex = ? ";

argList.add(user.getSex());

}

if (user.getEducation() != null

&& user.getEducation().trim().length() > 0) {

sql += "and education = ? ";

argList.add(user.getEducation());

}

if (user.getIsUpload() != null

&& user.getIsUpload().trim().length() > 0) {

if (user.getIsUpload().equals("1")) {

// 上传简历

sql += "and filename is not null";

} else if (user.getIsUpload().equals("2")) {

// 没有上传简历

sql += "and filename is null";

}

}

3、 Action将查询结果 定义成员变量，提供getter方法 （值栈）

4、 JSP 通过struts2 标签 显示数据

\* 修改指向条件查询链接

修改 left.jsp

d.add(3,2,'用户管理','${pageContext.request.contextPath}/user\_list.action','','mainFrame');

修改 struts.xml

<result name="addSUCCESS" type="redirectAction">user\_list</result>

<result name="listSUCCESS">/user/list.jsp</result>

## 七、 功能实现 --- 员工删除功能

list.jsp 点击每行数据后 删除图标，删除该行数据

删除原理： 根据id 删除

<s:a action="user\_delete" namespace="/">

<s:param name="userID" value="#user.userID"></s:param>

<img src="${pageContext.request.contextPath}/images/i\_del.gif" width="16" height="16" border="0" style="CURSOR: hand">

</s:a>

服务器根据 id 删除数据表记录

\*\* 同时删除员工的简历

JavaScript 确认删除效果

$(".delLink").click(function(event){

var isConfirm = window.confirm("想好了吗？");

if(!isConfirm){

// 阻止提交

event.preventDefault();

}

});

## 八、 员工信息详情查看

原理 ： 点击按钮，传递员工id，在服务器根据id查询， 将查询结果保存值栈，页面回显

1、查询按钮的改写

<s:a action="user\_view" namespace="/">

<s:param name="userID" value="#user.userID"></s:param>

<img src="${pageContext.request.contextPath}/images/button\_view.gif" border="0" style="CURSOR: hand">

</s:a>

2、 Action查询数据结果，将结果被 getModel 引用

\* 通过getModel将查询结果 传递jsp

3、 使用model传递结果数据给jsp ，并不是栈顶model对象

## 九、 员工简历下载

1、 改造页面链接

<s:a action="user\_download" namespace="/" cssClass="cl\_01">

<s:param name="userID" value="model.userID"></s:param>

<s:property value="model.filename"/>

</s:a>

传递下载简历员工id 给服务器

2、 服务器接收id，查询员工简历文件，提供下载

\* 一个流 ，两个头信息

bug ： 查看无简历 员工信息时，出现异常

org.apache.jasper.JasperException: Caught an exception while getting the property values of cn.itcast.user.web.action.UserAction@1264e3b - Class: ognl.OgnlRuntime File: OgnlRuntime.java Method: getMethodValue Line: 1440 - ognl/OgnlRuntime.java:1440:-1

原因 ：<s:debug/> ----- 调用值栈中所有 getXXX 方法

\*\*\*\*\*\*\*\*\*\* 解决：1） 不用 <s:debug/> 2）处理所有getXXX方法可能出错的情况

## 十、 员工修改功能

原理 ：

流程一 根据id查询员工信息，通过值栈将员工信息 显示form表单中

1） 修改编辑按钮

<s:a action="user\_editview" namespace="/">

<s:param name="userID" value="#user.userID"></s:param>

<img src="${pageContext.request.contextPath}/images/i\_edit.gif" border="0" style="CURSOR: hand">

</s:a>

2） 通过model引用，将结果数据传递jsp

\*\*\*\*\* 编辑form 和 添加 form 基本一样， 区别编辑form 含有隐藏域，保存编辑员工id

流程二 提交修改form表单，完成员工信息修改

1） 修改员工信息，form表单 是 上传表单， 用户可以选择简历

如果用户选择了简历，在服务器端替换原有简历

如果用户没选择简历，忽略对简历的修改

2） 上传新简历，替换原有简历

存在简历无法删除问题 （因为有时简历被占用的 ）

解决 ： 如果不能及时删除简历， 可以在文件夹简历 简历文件删除标志文件

3） 校验

@InputConfig 定义返回 不同视图

## 十一、登陆校验拦截器

1、 自定义拦截器

获取Action对象添加错误信息

ActionSupport action = (ActionSupport) actionInvocation.getAction();

action.addActionError(action.getText("nologin"));

2、 配置拦截器

<interceptors>

<!-- 注册 -->

<interceptor name="login" class="cn.itcast.user.web.interceptor.LoginInterceptor"></interceptor>

<interceptor-stack name="myStack">

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

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

</interceptor-stack>

</interceptors>

<!-- 设置默认拦截器栈 -->

<default-interceptor-ref name="myStack"></default-interceptor-ref>

<!-- 如果未登陆 定义全局结果集 -->

<global-results>

<result name="login">/login/login.jsp</result>

</global-results>

\*\*\*\*\* 元素有顺序的！！！！！！！！

问题：永远无法登陆！！！！！

\*\*\*\* 通过拦截器参数设置，控制哪些方法 不被拦截 ---- 对user\_login 不进行拦截

在login.jsp

<s:form action="user\_login" method="post" theme="simple" namespace="/" id="loginAction\_home" name="form1" target="\_top">

控制登陆后，整个页面跳转

## 十二、 异常处理

如果发生错误，企业中 会第一时间 将错误转换为 自定义异常 错误 ---- 抛出

1、 自定义异常

将出现通用异常，转换为 自定义异常

try {

...

} catch(){

throw 自定义异常;

}

2、 通常在拦截器 抓取自定义异常，记录日志， 提供友好页面

3、 自定义异常拦截，通常不拦截所有异常 ---- 通用异常页面

<global-exception-mappings>

<!-- 全局错误页面 -->

<exception-mapping result="error" exception="java.lang.Exception"></exception-mapping>

</global-exception-mappings>

\*\*\* 通用错误页面 可以 与 自定义异常错误页面 不同