Tarena Technology
达内科技

Tarena Teaching System

Struts 核心

学员用书

Java企业应用及互联网
高级工程师培训课程

达内集团教学研发部 编著

目　录

Unit01 1
1. Struts2 简介 3
1.1. 什么是 Struts2 3
1.1.1. Struts2 的概念 3
1.1.2. 什么是 MVC 3
1.2. 为什么用 Struts2 3
1.2.1. Struts2 与 Servlet 对比 3
1.2.2. Struts2 自身的优势 4
1.3. Struts2 发展史 4
1.3.1. Struts1 4
1.3.2. WebWork 4
1.3.3. Struts2 5
2. 使用 Struts2 5
2.1. Struts2 与 MVC 5
2.1.1. Servlet 如何实现 MVC 5
2.1.2. Spring 如何实现 MVC 5
2.1.3. Struts2 如何实现 MVC 6
2.2. Struts2 的使用步骤 6
2.2.1. 大致步骤 6
2.2.2. 1、创建 WEB 项目 6
2.2.3. 2、导入 Struts2 核心包 7
2.2.4. 3、配置前端控制器 7
2.2.5. 4、创建 struts.xml 7
2.2.6. 5、编写业务控制器 Action 8
2.2.7. 6、编写 JSP 页面 8
2.2.8. 7、配置 struts.xml 8
3. 参数传递 9
3.1. 页面向 Action 传参 9
3.1.1. 基本属性注入 9
3.1.2. 域模型注入 9
3.2. 页面从 Action 取值 10
3.2.1. 使用 EL 表达式显示 Action 值 10
3.2.2. 使用 OGNL 表达式显示 Action 值 10
4. 资费列表 11
4.1. 需求讲解 11
4.1.1. 需求描述 11

达内IT培训集团

4.1.2. 查询功能要求 11
4.2. 开发思路 11
4.2.1. 查询功能需要几种请求 11
4.2.2. 查询功能请求的过程 12
4.3. 开发步骤 12
4.3.1. 1、Entity 12
4.3.2. 2、DAO 12
4.3.3. 3、Action 13
4.3.4. 4、struts.xml 13
4.3.5. 5、JSP 13
经典案例 14
1. Struts2 之 HelloWorld，Step1-4 14
2. Struts2 之 HelloWorld，Step5-7 18
3. 将页面表单中的数据提交给 Action 23
4. 使用 EL 表达式，显示 Action 中的数据 31
5. NetCTOSS 资费列表，Step1-3 35
6. NetCTOSS 资费列表，Step4-6 44
课后作业 61
Unit02 62
1. OGNL 64
1.1. OGNL 介绍 64
1.1.1. 什么是 OGNL 64
1.1.2. 为什么用 OGNL 64
1.1.3. OGNL 原理 64
1.2. OGNL 用法 65
1.2.1. Struts2 显示标签 65
1.2.2. 2 个常用的 OGNL 表达式 65
1.2.3. 6 个需要了解的 OGNL 表达式 65
2. ValueStack 66
2.1. ValueStack 介绍 66
2.1.1. 什么是 ValueStack 66
2.1.2. ValueStack 原理 66
2.2. 访问 ValueStack 66
2.2.1. 1、通过 debug 标签观察其结构 66
2.2.2. 2、输出栈顶 67
2.2.3. 3、访问 Context 对象 67
2.2.4. 4、迭代集合 67
2.2.5. 5、按数字迭代 68
2.2.6. 总结：ValueStack 栈顶的变化 69

2.3. Struts2 对 EL 的支持 69
2.3.1. EL 表达式如何访问 ValueStack 69
3. 重构资费列表 70
3.1. 重构资费列表 70
3.1.1. 用 Struts2 标签+OGNL 重构资费列表 70
4. Action 基本原理 70
4.1. 6 大核心组件 70
4.1.1. 6 大核心组件关系 70
4.1.2. 6 大核心组件作用 71
经典案例 72
1. 使用 OGNL 表达式-1 72
2. 使用 OGNL 表达式-2 79
3. 访问 ValueStack-1 89
4. 访问 ValueStack-2 93
5. 重构资费列表 104
课后作业 112
Unit03 113
1. 登录 115
1.1. 如何获取 Session 115
1.1.1. 获取 Session 的方式 115
1.1.2. 各种方式的对比 115
1.2. 登录的需求 115
1.2.1. 登录的需求 115
1.3. 开发思路 116
1.3.1. 请求 1： 打开登录页面 116
1.3.2. 请求 2：登录验证 116
1.4. 开发步骤 116
1.4.1. 请求 1 116
1.4.2. 请求 2 117
2. Result 原理 117
2.1. Result 介绍 117
2.1.1. 什么是 Result 组件 117
2.2. 常用类型的 Result 118
2.2.1. dispatcher 118
2.2.2. stream 118
2.2.3. redirectAction 118
2.2.4. json 119
3. stream Result 119
3.1. 作用 119
3.1.1. stream Result 的作用 119

达内IT培训集团

3.2. 使用方式 119
3.2.1. 语法 119
3.2.2. 使用步骤 120
3.3. 登录验证码 120
3.3.1. 需求描述 120
3.3.2. 开发思路 120
3.3.3. 开发步骤-请求 1 121
3.3.4. 开发步骤-请求 2 121
4. redirectAction Result 122
4.1. 作用 122
4.1.1. redirectAction Result 的作用 122
4.2. 使用方式 122
4.2.1. 语法 122
4.2.2. 使用步骤 123
4.3. 资费删除 123
4.3.1. 需求描述 123
4.3.2. 开发思路 124
4.3.3. 开发步骤 124
经典案例 125
1. NetCTOSS 登录，请求 1 125
2. NetCTOSS 登录，请求 2 129
3. 登录验证码-1 147
4. 登录验证码-2 156
5. 资费删除 163
课后作业 176
Unit04 177
1. json Result 178
1.1. 作用 178
1.1.1. json Result 的作用 178
1.2. 使用方式 178
1.2.1. 语法 178
1.2.2. 使用步骤 179
1.3. 名称唯一性校验 179
1.3.1. 需求描述 179
1.3.2. 开发思路 180
1.3.3. 开发步骤 180
2. 资费修改 181
2.1. 开发思路 181
2.1.1. 需要几种请求 181

2.1.2. 打开修改页面的请求过程 182
2.2. 开发步骤 182
2.2.1. 打开修改页面的开发步骤 182
3. UI 标签 182
3.1. 作用 182
3.1.1. UI 标签的作用 182
3.2. 简单的 UI 标签 183
3.2.1. 表单 183
3.2.2. 文本框、密码框、文本域 183
3.2.3. 布尔框 183
3.3. 复杂的 UI 标签 184
3.3.1. 单选框 184
3.3.2. 多选框 184
3.3.3. 下拉选 185
经典案例 186
1. 资费名唯一性校验-1 186
2. 资费名唯一性校验-2 206
3. 资费修改 214
4. 使用简单的 UI 标签 237
5. 使用复杂的 UI 标签 248
课后作业 266
Unit05 267
1. 拦截器 268
1.1. 作用 268
1.1.1. 拦截器的用途 268
1.2. 使用步骤 268
1.2.1. 1、创建拦截器组件 268
1.2.2. 2、注册拦截器 268
1.2.3. 3、引用拦截器 269
1.3. 扩展 269
1.3.1. 拦截器栈 269
1.3.2. 预置拦截器 269
1.3.3. 默认引用拦截器 270
1.3.4. 拦截器调用顺序 270
2. 登录检查 270
2.1. 需求描述 270
2.1.1. 为什么要做登录检查 270
2.2. 开发思路 271

达内IT培训集团

2.2.1. 如何做登录检查 271
2.3. 开发步骤 271
2.3.1. 登录检查的开发步骤 271
3. 上传文件拦截器 272
3.1. FileUpload 介绍 272
3.1.1. FileUpload 拦截器介绍 272
3.1.2. FileUpload 拦截器原理 272
3.2. FileUpload 使用 272
3.2.1. FileUpload 拦截器使用步骤 272
3.2.2. 注意事项 273
经典案例 274
1. 拦截器 HelloWorld 274
2. 扩展拦截器 HelloWorld 279
3. NetCTOSS 登录检查 283
4. 上传文件 290
课后作业 300

Struts 核心

Unit01

知识体系.....Page 3

Struts2 简介	什么是 Struts2	Struts2 的概念
		什么是 MVC
	为什么用 Struts2	Struts2 与 Servlet 对比
		Struts2 自身的优势
	Struts2 发展史	Struts1
		WebWork
		Struts2
使用 Struts2	Struts2 与 MVC	Servlet 如何实现 MVC
		Spring 如何实现 MVC
		Struts2 如何实现 MVC
	Struts2 的使用步骤	大致步骤
		1、创建 WEB 项目
		2、导入 Struts2 核心包
		3、配置前端控制器
		4、创建 struts.xml
		5、编写业务控制器 Action
		6、编写 JSP 页面
		7、配置 struts.xml
参数传递	页面向 Action 传参	基本属性注入
		域模型注入
	页面从 Action 取值	使用 EL 表达式显示 Action 值
		使用 OGNL 表达式显示 Action 值
资费列表	需求讲解	需求描述
		查询功能要求
	开发思路	查询功能需要几种请求
		查询功能请求的过程
	开发步骤	1、Entity
		2、DAO
		3、Action

达内IT培训集团

		4、struts.xml
		5、JSP

经典案例.....Page 14

Struts2 之 HelloWorld，Step1-4	1、创建 WEB 项目
	2、导入 Struts2 核心包
	3、配置前端控制器
	4、创建 struts.xml
Struts2 之 HelloWorld，Step5-7	5、编写业务控制器 Action
	6、编写 JSP 页面
	7、配置 struts.xml
将页面表单中的数据提交给 Action	基本属性注入
	域模型注入
使用 EL 表达式，显示 Action 中的数据	使用 EL 表达式显示 Action 值
NetCTOSS 资费列表，Step1-3	1、Entity
	2、DAO
NetCTOSS 资费列表，Step4-6	3、Action
	4、struts.xml
	5、JSP

课后作业.....Page 61

1. Struts2 简介

1.1. 什么是 Struts2

1.1.1. 【什么是 Struts2】Struts2 的概念

Struts2的概念

解说识知

- Struts2
  是轻量级的MVC框架，主要解决了请求分发的问题，重心在控制层和表现层。
- 轻量级
  低侵入性，与业务代码的耦合度很低。即业务代码中基本不需要import它的包。
- MVC框架
  Struts2实现了MVC，并提供一系列API，采用模式化方式简化业务开发过程。

1.1.2. 【什么是 Struts2】什么是 MVC

什么是MVC

解说识知

- **M-Model 模型**
  模型（Model）的职责是负责业务逻辑。包含两层：业务数据和业务处理逻辑。比如实体类、DAO、Service都属于模型层。
- **V-View 视图**
  视图（View）的职责是负责显示界面和用户交互（收集用户信息）。属于视图的组件是不包含业务逻辑和控制逻辑的JSP。
- **C-Controller 控制器**
  控制器是模型层M和视图层V之间的桥梁，用于控制流程比如：在Servlet项目中的单一控制器ActionServlet。

1.2. 为什么用 Struts2

1.2.1. 【为什么用 Struts2】Struts2 与 Servlet 对比

Struts2与Servlet对比

解说识知

- 优点
  - 业务代码解耦，适合团队开发
    将请求分发给不同的处理类，从而降低了业务代码耦合度。
  - 提升开发效率
    提供了一系列API，可以大大提升项目的开发效率。如：使用拦截器自动给请求参数转型。
- 缺点
  - 执行效率偏低
    需要使用反射、解析XML等技术手段，会降低执行效率。
  - 结构复杂，有学习成本
    需要花一定成本学习Struts2的API及使用步骤。

1.2.2. 【为什么用 Struts2】Struts2 自身的优势

Struts2自身的优势

- 健壮性（4★）
  Struts2是一个成熟稳定的框架，目前最稳定的版本是2.1.8。
- 易用性（4★）
  Struts2易学好用，几天即可上手。
- 扩展性（5★）
  Struts2运用AOP的思想，使用拦截器来扩展业务控制器Action。
- 侵入性（4★）
  Struts2对业务代码依赖性很低，基本不需要import它的包。

1.3. Struts2 发展史

1.3.1. 【Struts2 发展史】Struts1

Struts1

- Struts1是Apache软件基金会（ASF）赞助的一个开源项目。它通过采用JavaServlet/JSP技术，实现了基于Java EEWeb应用的MVC设计模式的应用框架，是MVC经典设计模式中的一个经典产品。
- Struts1结构简单小巧，十分易用，一度市场占有率超过20%。
- Struts1框架，与JSP/Servlet耦合非常紧密，这制约了它的发展，以至于被后来的框架陆续赶超。

1.3.2. 【Struts2 发展史】WebWork

WebWork

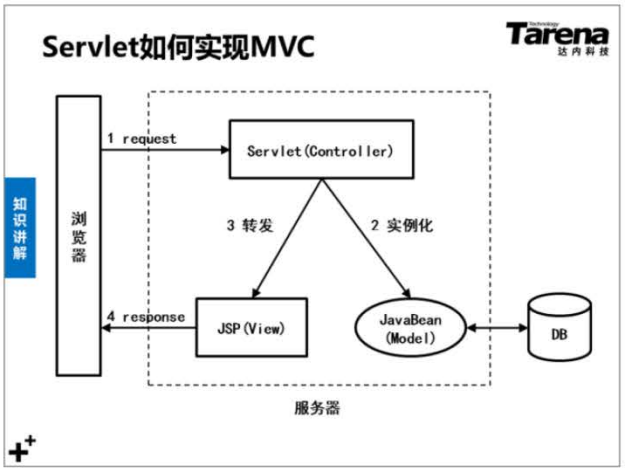
- WebWork是由OpenSymphony组织开发的，是建立在称为XWork的Command模式框架之上的强大的MVC框架。
- WebWork晚于Struts1，技术上更为先进。
- 由于组织知名度、人们的习惯等原因，WebWork市场反响远不及Struts1.

1.3.3. 【Struts2 发展史】Struts2

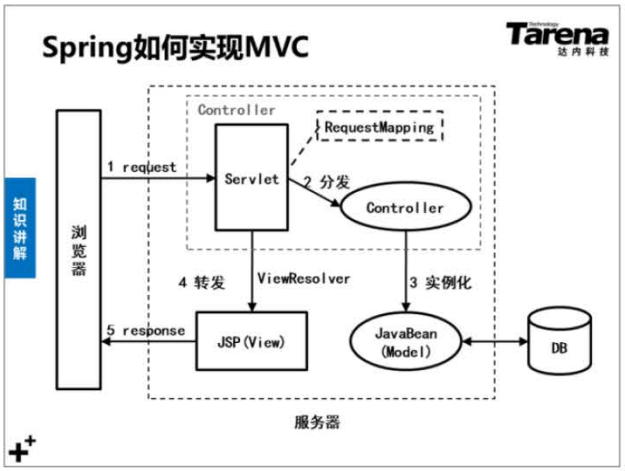
2. 使用 Struts2

2.1. Struts2 与 MVC

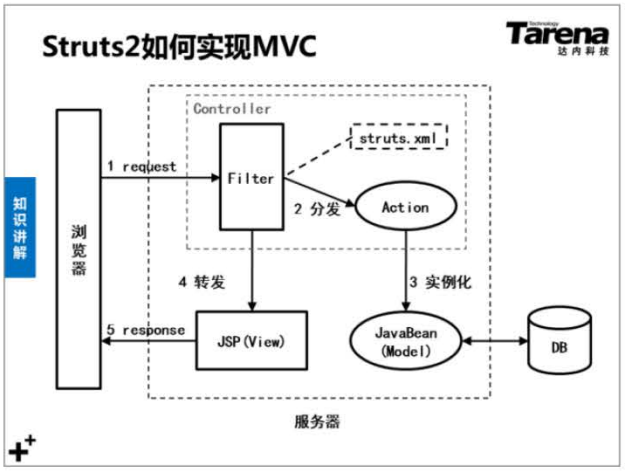
2.1.1. 【Struts2 与 MVC】Servlet 如何实现 MVC



2.1.2. 【Struts2 与 MVC】Spring 如何实现 MVC

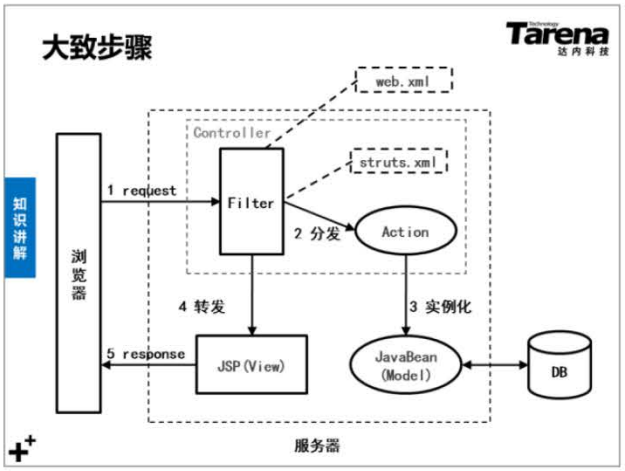


2.1.3. 【Struts2 与 MVC】Struts2 如何实现 MVC

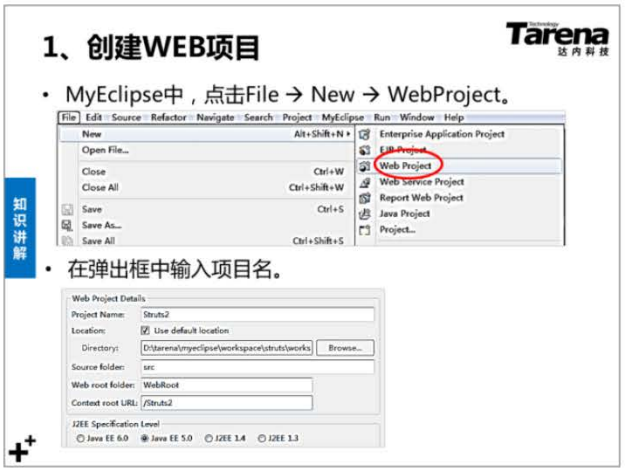


2.2. Struts2 的使用步骤

2.2.1. 【Struts2 的使用步骤】大致步骤



2.2.2. 【Struts2 的使用步骤】1、创建 WEB 项目



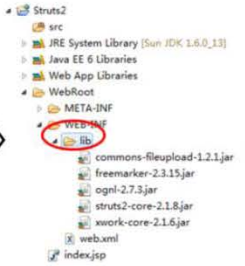
2.2.3. 【Struts2 的使用步骤】2、导入 Struts2 核心包

2、导入Struts2核心包

- 导入Struts2最小范围核心包（5个）。
- 将这些包复制到新建项目的/WEB-INF/lib目录下。

解释说明

commons-fileupload-1.2.1.jar
freemarker-2.3.15.jar
ognl-2.7.3.jar
struts2-core-2.1.8.jar
xwork-core-2.1.6.jar



2.2.4. 【Struts2 的使用步骤】3、配置前端控制器

3、配置前端控制器

- Struts2使用filter来充当前端控制器，因此在web.xml中配置一个filter即可。
- Struts2预置了该filter的实现类，名为org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter。
- 指定该filter处理所有的请求。

解释说明

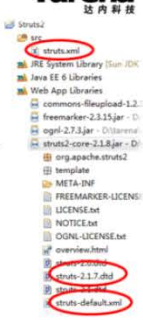
```
<!-- Struts2前端控制器 -->
<filter>
    <filter-name>Struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>Struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

2.2.5. 【Struts2 的使用步骤】4、创建 struts.xml

4、创建struts.xml

- 在src下，创建名为struts.xml的配置文件。
  - Struts2配置文件默认要放在src下。
  - Struts2配置文件默认名称为struts.xml。
- struts.xml的格式
  - 可以参考DTD文件，位于核心包根路径下。
  - 可以参考默认配置文件struts-default.xml，也位于核心包根路径下。
- 配置struts.xml的版本信息及DTD引用。

解释说明




```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE struts PUBLIC
3     "-//Apache Software Foundation//DTD Struts Configuration 2.1.7
4     "http://struts.apache.org/dtds/struts-2.1.7.dtd">
```

2.2.6. 【Struts2 的使用步骤】5、编写业务控制器 Action

5、编写业务控制器Action

解说识知

- 创建业务控制器组件，通常命名为XxxAction，该组件是一个满足JavaBean规范的类。
- 在Action中定义业务方法，要满足下列条件
  - 方法是public的
  - 返回值为String类型
  - 参数列表为空
- 编写业务方法
  - 方法内编写业务逻辑代码
  - 返回的字符串与struts.xml→action→result的name属性匹配，即根据此返回值可以找到对应的result。

2.2.7. 【Struts2 的使用步骤】6、编写 JSP 页面

6、编写JSP页面

解说识知

- 创建JSP页面
- 在页面上显示" Hello,Struts2."

2.2.8. 【Struts2 的使用步骤】7、配置 struts.xml

7、配置struts.xml

解说识知

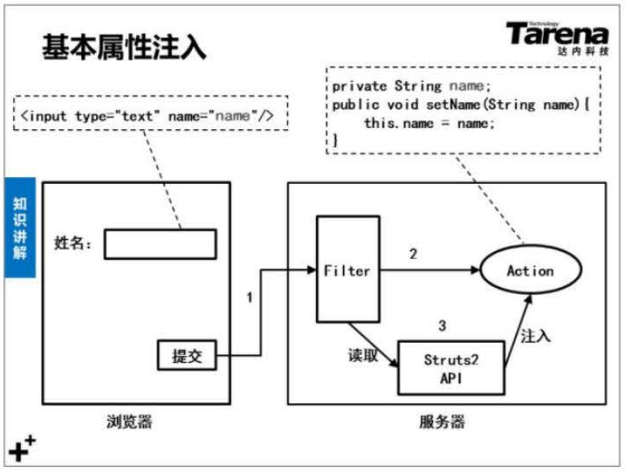
- 在struts.xml中配置请求与Action的关系。
- 在action下，通过result设置转发的页面。

```
<struts>
    <!-- package：包，用于对Action进行封装。
        1、name：包名，根元素下可以有多个包，彼此之间不能重名。
        2、extends：继承，用于指定继承的包，相当于将继承包下的配置信息复制到了当前包下。
        3、namespace：命名空间，用于规定Action的访问路径，必须以"/"开头。-->
    <package name="day01" namespace="/day01" extends="struts-default">
        <!-- action：业务控制器，用于注册业务控制器组件（类）。
            1、name：action名称，用于规定Action的访问路径。
              一个包下可以有多个Action，彼此之间不能重名。
            2、class：业务控制器组件，用于指定业务控制器对应的类。
            3、method：方法，用于指定访问当前action时要调用的方法。
            *4、请求URL：http://IP:PORT/ProjectName/NAMESPACE/ACTIONNAME.action-->
        <action name="hello" class="day01.action.HelloAction" method="execute">
            <!-- result：输出组件，用于转发、重定向、直接输出。
                1、name：名称，一个action下可以由多个result，彼此之间不能重名。
                2、默认像转发，元素内设置转发的页面。-->
            <result name="success">
                /hello.jsp
            </result>
        </action>
    </package>
</struts>
```

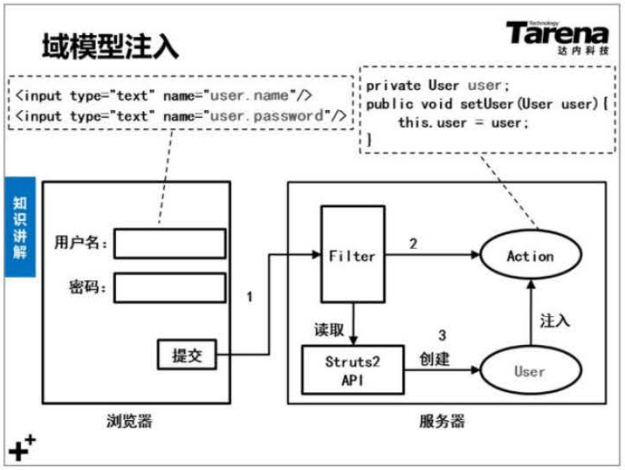
3. 参数传递

3.1. 页面向 Action 传参

3.1.1. 【页面向 Action 传参】基本属性注入



3.1.2. 【页面向 Action 传参】域模型注入



域模型注入（续1）

- 在Action中定义实体对象属性，并提供set、get方法

```
public class User {

    private String userName;
    private String password;

    public String getPassword() {
        return password;
    }

    public String getUserName() {
        return userName;
```

```
/**
 * 接收注入的域模型对象
 * 1、Struts2会自动实例化该对象
 * 2、Struts2会自动调用set方法为该对象设置值
 * 3、上述行为发生在Action实例化之后，业务方法调用之前
 */
private User user;

public void setUser(User user) {
    System.out.println("注入域模型对象...");
    this.user = user;
}
```

- 表单中文本框指定表达式“对象名.属性名”

```
<!--
    演示域模型注入
    1、name中的表达式要写"对象名.属性名"。
    2、Struts2会自动调用对象属性的set方法为其赋值。
-->
用户名：<input type="text" name="user.userName"/><br/>
密码：<input type="password" name="user.password"/><br/>
```

3.2. 页面从 Action 取值

3.2.1. 【页面从 Action 取值】使用 EL 表达式显示 Action 值

使用EL表达式显示Action值

- 取基本属性值
  ${ 属性名 }
- 取域模型对象值
  ${ 对象名.属性名 }
- 结论
  取值时EL表达式的写法，与注入时表达式的写法一致。

```
<!-- 显示基本属性值 -->
<h1>Hello,Struts2.</h1>
<h1>姓名：${name }</h1>

<!-- 显示域模型对象值 -->
<h1>用户名：${user.userName }</h1>
<h1>密码：${user.password }</h1>
```

3.2.2. 【页面从 Action 取值】使用 OGNL 表达式显示 Action 值

使用OGNL表达式显示Action值

- Struts2虽然支持EL表达式，但实际上OGNL表达式才是其默认使用的表达式。
- OGNL表达式功能更强大，后面会讲述其原理及使用。

4. 资费列表

4.1. 需求讲解

4.1.1. 【需求讲解】需求描述

需求描述

- 资费管理模块，维护的是电信业务中资费的标准，类似于办理手机号时选择的套餐。
- 查询功能，是使用列表的方式将维护好的资费数据进行展现。

解说须知

4.1.2. 【需求讲解】查询功能要求

查询功能要求

- 按列表形式展现数据，只显示一部分列。
- 状态字段数据库中存的是char(1)，即0(开通)，1(暂停)，页面上需要显示出中文含义。
- 具有分页功能，该功能暂不考虑。

解说须知

4.2. 开发思路

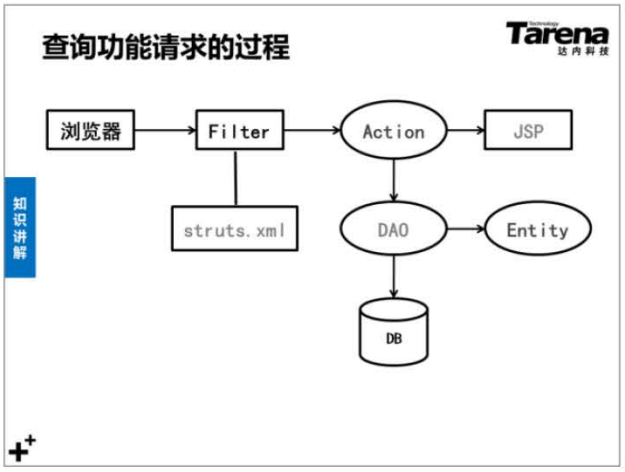
4.2.1. 【开发思路】查询功能需要几种请求

查询功能需要几种请求

- 在浏览器地址栏输入URL可以查询资费。
- 点击页码也可以查询资费。
- 上述方式都是针对资费的查询，只是条件有变化，因此可以使用一种请求来处理查询功能。

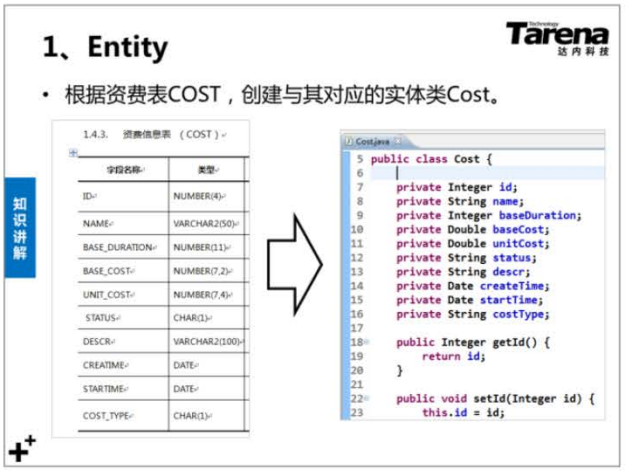
解说须知

4.2.2. 【开发思路】查询功能请求的过程



4.3. 开发步骤

4.3.1. 【开发步骤】1、Entity



4.3.2. 【开发步骤】2、DAO

4.3.3. 【开发步骤】3、Action

3、Action

- 创建处理查询请求的Action，即FindCostAction
- 定义输入属性
  - Action需要JSP传入的参数，称为输入属性
  - 当前案例中不需要输入属性
- 定义输出属性
  - Action需要传出给JSP的参数，称为输出属性
  - 当前案例中输出属性为 List<Cost> costs
- 根据输入算输出
  - Action是业务控制器，其本质就是根据输入算输出
  - 当前案例中，只需要调用DAO查询出资费数据，并且赋值给输出属性即可

4.3.4. 【开发步骤】4、struts.xml

4、struts.xml

- 在struts.xml中，配置一个新的package，专门用于封装资费模块中的Action
  - 该包以模块命名，即name="cost"
  - 命名空间以模块命名，即namespace="/cost"
- 在当前package下，注册查询的Action

4.3.5. 【开发步骤】5、JSP

5、JSP

- 创建资费查询页面，如find_cost.jsp
- 将资费查询静态HTML代码复制到该JSP中
- 将table中的行删除，保留标题行（第一行）和内容首行（第二行）
- 使用JSTL循环标签，循环输出内容首行（第二行）
- 在内容首行中，使用EL表达式输出各列的值

经典案例

1. Struts2 之 HelloWorld，Step1-4

- **问题**

使用 Struts2 框架，写一个 HelloWorld 示例。其中，在 Action 的业务方法中输出“Hello,Action.”，在 JSP 页面上显示“Hello,Struts2.”。

本案例要求完成 Struts2 使用步骤 1-4，来构建起 Struts2 使用的准备环境。

- **方案**

Struts2 使用步骤：

1) 创建 WEB 项目
2) 导入 Struts2 核心包
3) 配置前端控制器
4) 创建 struts.xml

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建 WEB 项目

在 MyEclipse 中，点击 File → New → WebProject，如下图：

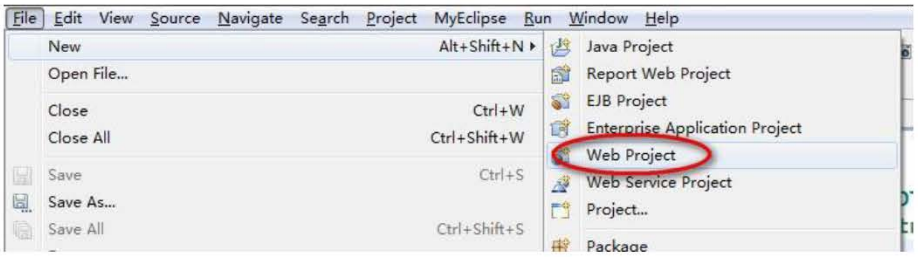


图-1

在弹出框中输入项目名，如下图：

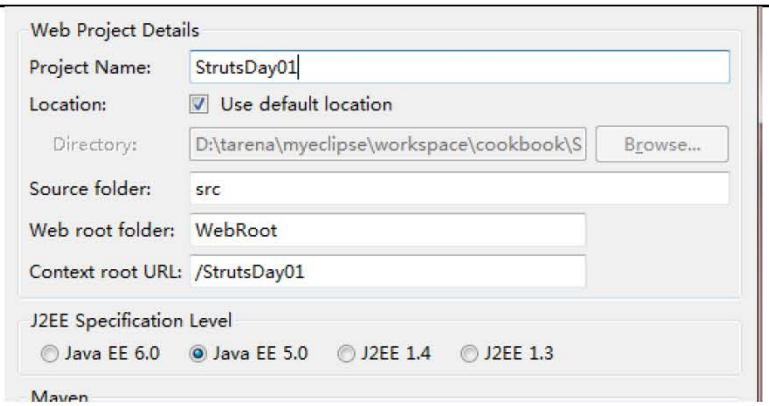


图-2

步骤二：导入 Struts2 核心包

要想使用 Struts2，至少需要导入 5 个核心包，如下图：

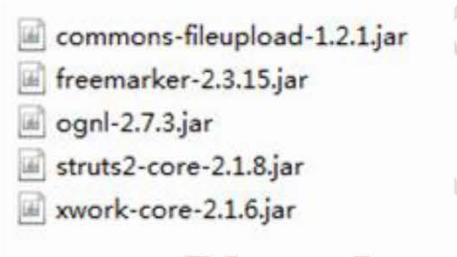


图-3

将这 5 个包复制到新创建项目的/WEB-INF/lib 目录下，如下图：

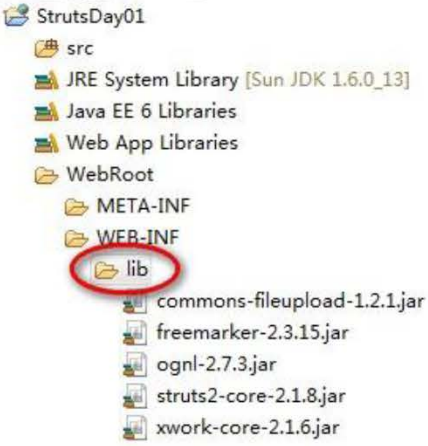


图-4

步骤三：配置前端控制器

在 web.xml 中，配置前端控制器 filter，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
```

```
http://java.sun.com/xml/ns/javaee/web-app 3 0.xsd">
  <display-name></display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>


  <!-- 前端控制器 -->
  <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>


</web-app>
```

步骤四：创建 struts.xml

在 src 下，创建名为 struts.xml 的配置文件，如下图：

图-5

注意：该配置文件必须放于 src 下，且名称必须为 struts.xml。

打开 struts.xml，设置其版本信息及 DTD 引用。这些内容可以从 Struts2 默认配置文件中直接复制过来，该默认配置文件名为 struts-default.xml，位于核心包 struts2-core-2.1.8.jar 根路径下，如下图：

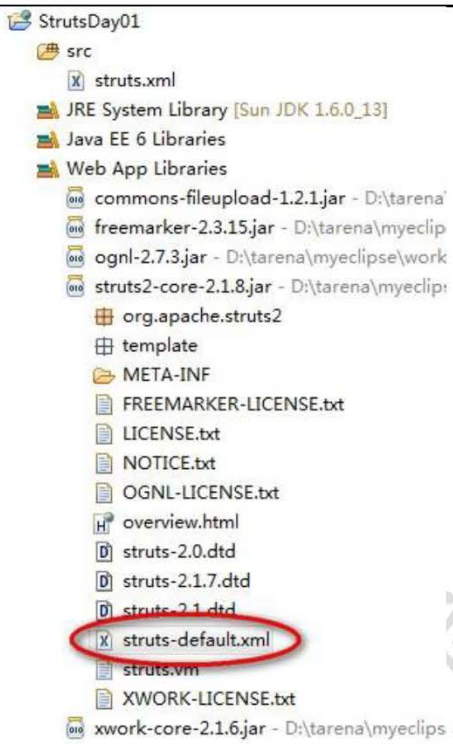


图-6

打开 struts-default.xml，从中复制出该 XML 的版本信息及 DTD，如下图：

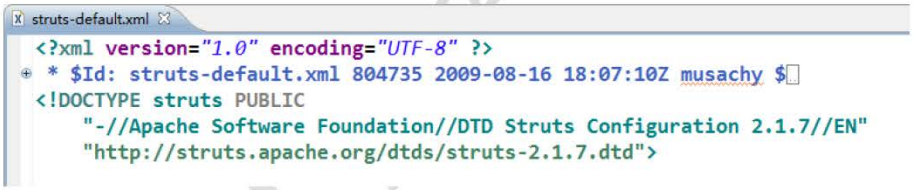


图-7

将这些复制好的内容，粘贴到刚才创建的 struts.xml 中。其中注释部分可以删除，完成后 struts.xml 的代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
```

- **完整代码**

本案例 web.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
  <display-name></display-name>
  <welcome-file-list>
```

```
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <!-- 前端控制器 -->
  <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>
</web-app>
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
```

2. Struts2 之 HelloWorld，Step5-7

- **问题**

继续实现 Struts2 的 HelloWorld 示例，完成在 Action 的业务方法中输出“Hello,Action.”，在 JSP 页面上显示“Hello,Struts2.”。

- **方案**

Struts2 使用步骤：

5) 编写业务控制器 Action
6) 编写 JSP 页面
7) 配置 struts.xml

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：编写业务控制器 Action

创建名为 action 的包，并在该包下创建类 HelloAction，如下图：

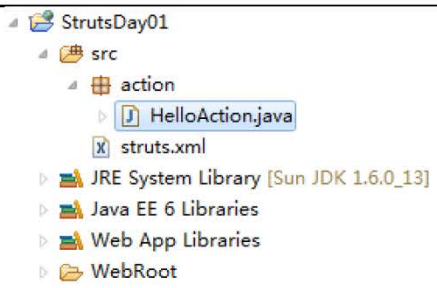


图-8

在 HelloAction 中，创建方法 sayHello，该方法必须满足如下条件：

- 方法是 public 的
- 返回值为 String 类型
- 参数列表为空

在该方法中输出“Hello，Action.”，并返回字符串”success”，代码如下：

```
package action;

public class HelloAction {

 /**
  * 在业务方法中输出"Hello，Action."
  */
 public String sayHello() {
     System.out.println("Hello,Action.");
     return "success";
 }

}
```

步骤二：编写 JSP 页面

在 WebRoot 下，创建页面 hello.jsp，如下图：

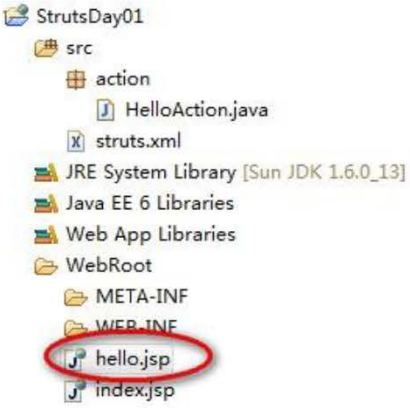


图-9

在页面上用标题显示出“Hello，Struts2.”。实现代码如下：

```
<%@page pageEncoding="utf-8" %>
<html>
<head>
</head>
<body>
 <h1>Hello,Struts2.</h1>
</body>
</html>
```

步骤三：配置 struts.xml

在 struts.xml 中，配置 Action 及 JSP。主要是配置请求找到 Action 的途径，以及 Action 跳转到 JSP 的途径，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     package：包，用于对 Action 进行封装。
     1、name：包名，根元素下可以有多个包，彼此之间不能重名。
     2、extends：继承，用于指定继承的包，相当于将继承包下的配置信息复制到了当前包下。
     3、namespace：命名空间，用于规定 Action 的访问路径，必须以"/"开头。
     4、请求 Action 时，按照如下格式拼写 URL：
        http://IP:PORT/ProjectName/NAMESPACE/ACTIONNAME.action
 -->
 <package name="day01" namespace="/demo" extends="struts-default">
     <!--
         action：业务控制器，用于注册业务控制器组件（类）。
         1、name：action 名称，用于规定 Action 的访问路径。
             一个包下可以有多个 action，彼此之间不能重名。
         2、class：业务控制器组件，用于指定业务控制器对应的类。
         3、method：方法，用于指定访问当前 action 时要调用的方法。
      -->
     <action name="hello" class="action.HelloAction" method="sayHello">
         <!--
             result：输出组件，用于转发、重定向、直接输出。
             1、name：名称，一个 action 下可以有多个 result，彼此之间不能重名。
             2、默认做转发，标记内容设置成转发的页面。
          -->
         <result name="success">
             /hello.jsp
         </result>
     </action>
 </package>

</struts>
```

步骤四：测试

部署该项目并启动 tomcat，打开浏览器，针对当前的案例，在地址栏中输入如下访问地址 http://localhost:8088/StrutsDay01/demo/hello.action。其中 demo 对应的是

配置文件中的 namespace 值，hello 对应的是配置文件中的 action 名称，“.action”是固定的后缀，也可以省略。

点击回车键，浏览器显示效果如下图：

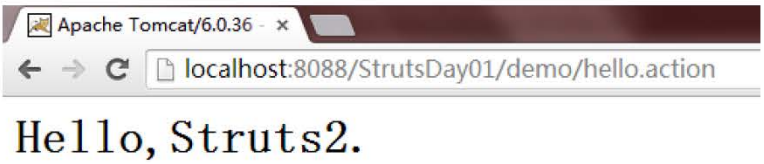


图-10

同时，控制台输出内容如下图：

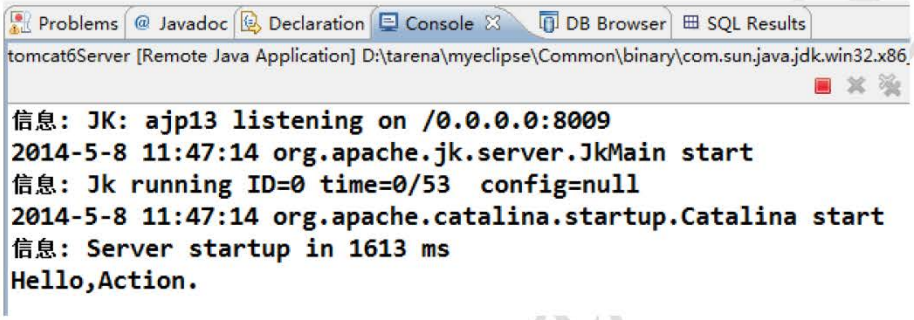


图-11

- **完整代码**

本案例的 web.xml 完整代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app 3 0.xsd">
  <display-name></display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <!-- 前端控制器 -->
  <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>
</web-app>
```

HelloAction 完整代码：

```
package action;
```

```
public class HelloAction {

 /**
  * 在业务方法中输出"Hello，Action."
  */
 public String sayHello() {
     System.out.println("Hello,Action.");
     return "success";
 }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
 <!--
     package：包，用于对 Action 进行封装。
     1、name：包名，根元素下可以有多个包，彼此之间不能重名。
     2、extends：继承，用于指定继承的包，相当于将继承包下的配置信息复制到了当前包下。
     3、namespace：命名空间，用于规定 Action 的访问路径，必须以"/"开头。
     4、请求 Action 时，按照如下格式拼写 URL：
        http://IP:PORT/ProjectName/NAMESPACE/ACTIONNAME.action
 -->
 <package name="day01" namespace="/demo" extends="struts-default">
     <!--
         action：业务控制器，用于注册业务控制器组件（类）。
         1、name：action 名称，用于规定 Action 的访问路径。
             一个包下可以有多个 action，彼此之间不能重名。
         2、class：业务控制器组件，用于指定业务控制器对应的类。
         3、method：方法，用于指定访问当前 action 时要调用的方法。
      -->
     <action name="hello" class="action.HelloAction" method="sayHello">
         <!--
             result：输出组件，用于转发、重定向、直接输出。
             1、name：名称，一个 action 下可以有多个 result，彼此之间不能重名。
             2、默认做转发，标记内容设置成转发的页面。
          -->
         <result name="success">
             /hello.jsp
         </result>
     </action>
 </package>
</struts>
```

hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<html>
<head>
</head>
<body>
 <h1>Hello,Struts2.</h1>
</body>
</html>
```

3. 将页面表单中的数据提交给 Action

- **问题**

在 Struts2 框架下，如何将表单数据传递给业务控制器 Action。

- **方案**

Struts2 中，表单向 Action 传递参数的方式有 2 种，并且这 2 种传参方式都是 Struts2 默认实现的，他们分别是基本属性注入、域模型注入。其中：

- 基本属性注入，是将表单的数据项分别传入给 Action 中的一些基本类型。
- 域模型注入，是将表单的数据项打包传入给 Action 中的一个实体对象。

我们可以使用项目 StrutsDay01 的 HelloWorld 示例，在其基础上练习使用这 2 种方式完成页面向 Action 的参数传递。具体的，我们可以在项目首页 index.jsp 上追加表单，并在表单中模拟一些数据，将这些数据提交给 HelloAction，最后在 HelloAction 中将接收的参数输出到控制台。

- **步骤**

1）基本属性注入

步骤一：追加表单，用于输入数据

在 StrutsDay01 项目的 index.jsp 中，追加表单，并将该表单设置提交给 HelloAction，即将 form 的 action 属性设置为"/StrutsDay01/demo/hello.action"。

在表单中增加一个文本框，用于输入一个姓名，该文本框的 name 属性值为 name。代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String                                basePath                                =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort(
)+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
 <meta http-equiv="pragma" content="no-cache">
 <meta http-equiv="cache-control" content="no-cache">
 <meta http-equiv="expires" content="0">
 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
 <meta http-equiv="description" content="This is my page">
 <!--
 <link rel="stylesheet" type="text/css" href="styles.css">
 -->
  </head>
```

```
  <body>
    This is my JSP page. <br><br>

    <form action="/StrutsDay01/demo/hello.action" method="post">
     <!-- 演示基本属性注入 -->
     姓名：<input type="text" name="realName"/><br/><br/>

     <input type="submit" value="提交"/>
    </form>

  </body>
</html>
```

步骤二：HelloAction 中，接收表单传入的参数

在 HelloAction 中，追加属性用于接收表单传入的姓名参数，该属性的名称要求与文本框的 name 值相同(realName)，并且该属性需要具备 set 方法。

在业务方法中输出属性 realName 的值。同时为了方便观察代码执行的顺序，建议在 Action 默认构造器中，输出任意的文字，代码如下：

```
package action;

public class HelloAction {

 public HelloAction() {
     System.out.println("实例化 Action...");
 }

 // 定义基本类型属性，接收表单参数：姓名
 private String realName;

 public void setRealName(String realName) {
     System.out.println("注入参数 realName...");
     this.realName = realName;
 }

 /**
  * 在业务方法中输出"Hello，Action."
  */
 public String sayHello() {
     System.out.println("Hello,Action.");

     // 输出基本类型数据
     System.out.println("姓名：" + realName);

     return "success";
 }
}
```

步骤三：测试

重新部署该项目并启动 tomcat，打开浏览器，针对当前的案例，在地址栏中输入如下访问地址 http://localhost:8088/StrutsDay01。点击回车，页面显示出项目首页的内

容，如下图：

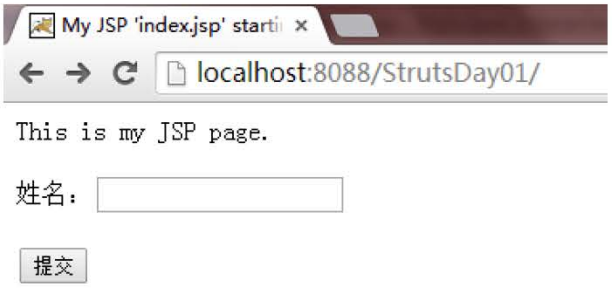


图-12

在姓名文本框中输入任意内容（如 Tarena），点击提交按钮，然后观察 MyEclipse 控制台输出的内容，如下图：

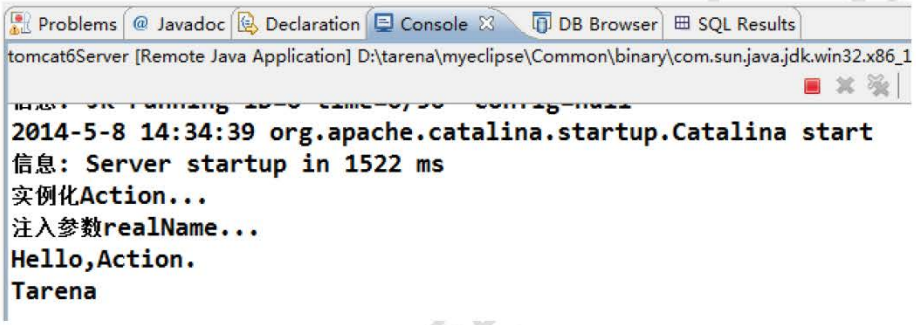


图-13

控制台输出的顺序可以证明代码的执行顺序为：实例化 Action→调用 set 方法注入参数→调用业务方法，当然这个过程是 Struts2 的 API 自行实现的，我们只需要在写代码时满足上述步骤中的要求即可。

由于 index.jsp 中的表单将请求提交给 HelloAction，而 HelloAction 又会跳转到 hello.jsp 页面，因此最终浏览器显示的效果如下图：

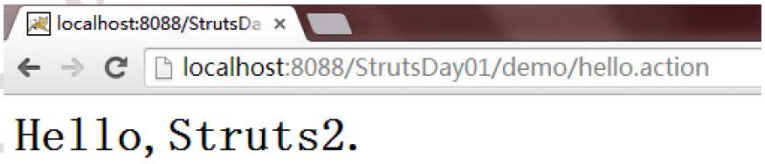


图-14

2）域模型注入

步骤一：修改表单，追加演示数据

修改表单，追加用户名、密码两个文本框，模拟输入用户的相关信息，代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String                              basePath                              =
```

```
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort(
)+path+"/";
   %>

   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
   <html>
     <head>
       <base href="<%=basePath%>">

       <title>My JSP 'index.jsp' starting page</title>
    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->
     </head>

     <body>
       This is my JSP page. <br><br>

       <form action="/StrutsDay01/demo/hello.action" method="post">
        <!-- 演示基本属性注入 -->
        姓名：<input type="text" name="realName"/><br/><br/>


        <!-- 演示域模型注入 -->
        用户名：<input type="text" /><br/><br/>
        密码：<input type="password" /><br/><br/>


        <input type="submit" value="提交"/>
       </form>
     </body>
   </html>
```

步骤二：创建实体类

创建包 entity，用于存放实体类。在 entity 包下创建实体类 User，用于封装表单中追加的数据，即用户名、密码。User 中要包含 2 个属性，用于封装用户名、密码，并给属性提供 get 和 set 方法，代码如下：

```
   package entity;

   public class User {

    private String userName;// 用户名
    private String password;// 密码

    public String getPassword() {
        return password;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
```

```
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

步骤三：修改 HelloAction,接收表单传入的参数

在 HelloAction 中，追加属性用于接收表单传入的用户名、密码参数，该属性的类型为 User 类型，名称为 user，并为该属性提供 get 和 set 方法。

在业务方法中输出属性 user 的值，代码如下：

```
package action;

import entity.User;

public class HelloAction {

    public HelloAction() {
        System.out.println("实例化Action...");
    }

    // 定义基本类型属性，接收表单参数：姓名
    private String realName;

    public void setRealName(String realName) {
        System.out.println("注入参数realName...");
        this.realName = realName;
    }

    //定义实体对象属性，接收表单参数：用户名、密码
    private User user;

    public void setUser(User user) {
        System.out.println("注入对象user...");
        this.user = user;
    }

    public User getUser() {
        return this.user;
    }

    /**
     * 在业务方法中输出"Hello，Action."
     */
    public String sayHello() {
        System.out.println("Hello,Action.");

        // 输出基本类型数据
        System.out.println("姓名：" + realName);

        // 输出域模型方式注入的参数
        System.out.println("用户名：" + user.getUserName());
        System.out.println("密码：" + user.getPassword());
```

```
        return "success";
    }
}
```

步骤四：修改表单，设置文本框属性

在 index.jsp 中，修改表单里新增的 2 个文本框的 name 属性值。对于域模型注入的方式，文本框 name 属性值应该是具有“对象名.属性名”格式的表达式。其中对象名指的是 Action 中的实体类型属性名，属性名指的是实体类型中的属性名，代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
String path = request.getContextPath();
String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort(
)+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->
  </head>

  <body>
    This is my JSP page. <br><br>

    <form action="/StrutsDay01/demo/hello.action" method="post">
     <!-- 演示基本属性注入 -->
     姓名：<input type="text" name="realName"/><br/><br/>


     <!-- 演示域模型注入 -->
     用户名：<input type="text" name="user.userName"/><br/><br/>
     密码：<input type="text" name="user.password"/><br/><br/>


     <input type="submit" value="提交"/>
    </form>
  </body>
</html>
```

步骤五：测试

重新部署项目并启动 tomcat，在浏览器中刷新地址 http://localhost:8088/StrutsDay01，效果如下图所示：

图-15

在文本框中输入任意的内容，点击提交，查看 MyEclipse 的控制台，输出结果如下图：

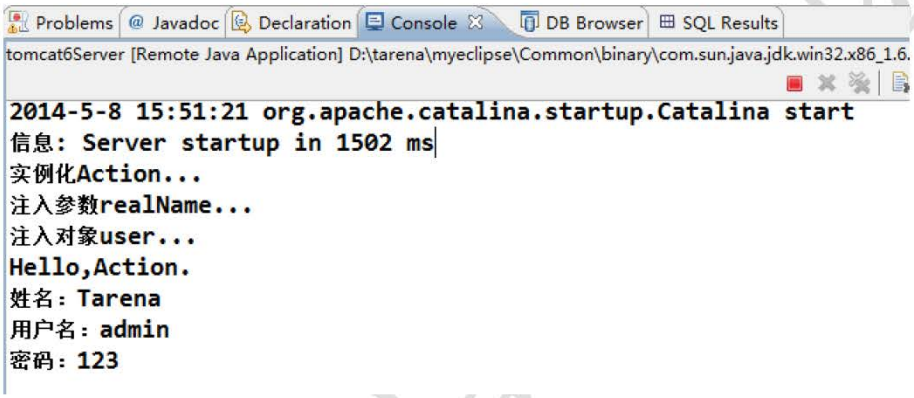


图-16

控制台输出的顺序可以证明代码的执行顺序为：实例化 Action→实例化 User 并注入参数→调用 set 方法注入 User 对象→调用业务方法。

最终浏览器显示的效果如下图：

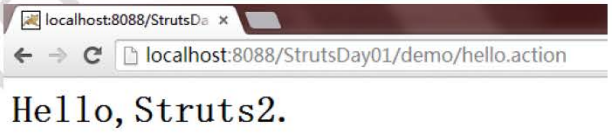


图-17

- **完整代码**

本案例的 index.jsp 完整代码：

```
    <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
    <%
    String path = request.getContextPath();
    String basePath =
request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort(
)+path+"/";
    %>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <base href="<%=basePath%>">

    <title>My JSP 'index.jsp' starting page</title>
 <meta http-equiv="pragma" content="no-cache">
 <meta http-equiv="cache-control" content="no-cache">
 <meta http-equiv="expires" content="0">
 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
 <meta http-equiv="description" content="This is my page">
 <!--
 <link rel="stylesheet" type="text/css" href="styles.css">
 -->
  </head>

  <body>
    This is my JSP page. <br><br>

    <form action="/StrutsDay01/demo/hello.action" method="post">
     <!-- 演示基本属性注入 -->
     姓名：<input type="text" name="realName"/><br/><br/>
     <!-- 演示域模型注入 -->
     用户名：<input type="text" name="user.userName"/><br/><br/>
     密码：<input type="text" name="user.password"/><br/><br/>

     <input type="submit" value="提交"/>
    </form>

  </body>
</html>
```

HelloAction 完整代码：

```
package action;

import com.opensymphony.xwork2.ModelDriven;

import entity.User;

public class HelloAction {

 public HelloAction() {
     System.out.println("实例化 Action...");
 }

 // 定义基本类型属性，接收表单参数：姓名
 private String realName;

 public void setRealName(String realName) {
     System.out.println("注入参数 realName...");
     this.realName = realName;
 }

 //定义实体对象属性，接收表单参数：用户名、密码
 private User user;

 public void setUser(User user) {
     System.out.println("注入对象 user...");
     this.user = user;
 }

 public User getUser() {
     return this.user;
```

```
    }

    /**
     * 在业务方法中输出"Hello, Action."
     */
    public String sayHello() {
        System.out.println("Hello,Action.");

        // 输出基本类型数据
        System.out.println("姓名: " + realName);
        // 输出域模型方式注入的参数
        System.out.println("用户名: " + user.getUserName());
        System.out.println("密码: " + user.getPassword());

        return "success";
    }

}
```

User 完整代码：

```
package entity;

public class User {

 private String userName;// 用户名
 private String password;// 密码

 public String getPassword() {
     return password;
 }

 public String getUserName() {
     return userName;
 }

 public void setUserName(String userName) {
     this.userName = userName;
 }

 public void setPassword(String password) {
     this.password = password;
 }

}
```

其他代码没有变化，不再重复。

4. 使用 EL 表达式，显示 Action 中的数据

- **问题**

在 Struts2 框架下，如何将业务控制器 Action 的数据传递给 JSP，并在 JSP 上显示出这些数据。

- **方案**

Struts2 会自动的将 Action 的数据传递给 JSP，并且对传递方式进行了封装，在使用时变得十分方便，甚至不需要使用 request 对象。它仅仅要求我们在 Action 中定义属性，并为属性提供 get 方法，那么从 Action 跳转到 JSP 时 Struts2 会自动的通过这些 get 方法将这些属性的值传递给 JSP。最终在 JSP 上我们可以使用 EL 表达式来显示 Action 的属性值。

我们还是利用 StrutsDay01 项目的 HelloWorld 示例，目前 HelloAction 中已经有了 2 个属性，即 realName、user，而该 Action 最终跳转的页面为 hello.jsp，我们的目标是在页面 hello.jsp 上使用 EL 表达式输出这些属性值。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：在 Action 中，给属性追加 get 方法

在 HelloAction 中，给 realName 属性追加 get 方法，用于页面的 EL 表达式访问并取值。user 属性已经有了 get 方法，不再需要追加。代码如下：

```
package action;

import com.opensymphony.xwork2.ModelDriven;

import entity.User;

public class HelloAction {

 public HelloAction() {
     System.out.println("实例化Action...");
 }

 // 定义基本类型属性，接收表单参数：姓名
 private String realName;

 public void setRealName(String realName) {
     System.out.println("注入参数realName...");
     this.realName = realName;
 }

 public String getRealName() {
     return this.realName;
 }

 //定义实体对象属性，接收表单参数：用户名、密码
 private User user;

 public void setUser(User user) {
     System.out.println("注入对象user...");
     this.user = user;
 }

 public User getUser() {
     return this.user;
 }
```

```
    /**
     * 在业务方法中输出"Hello，Action."
     */
    public String sayHello() {
        System.out.println("Hello,Action.");
        // 输出基本类型数据
        System.out.println("姓名：" + realName);
        // 输出域模型方式注入的参数
        System.out.println("用户名：" + user.getUserName());
        System.out.println("密码：" + user.getPassword());

        return "success";
    }
}
```

步骤二：在 JSP 上，使用 EL 表达式显示 Action 的属性值，代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<html>
<head>
</head>
<body>
 <h1>Hello,Struts2.</h1>

 <h1>姓名：${realName }</h1>
 <h1>用户名：${user.userName }</h1>
 <h1>密码：${user.password }</h1>

</body>
</html>
```

注意：如果使用的是 tomcat5，那么它默认会忽略 EL 表达式，需要在 page 指令中设置不忽略，即 isELIgnored="false"。如果使用的是 tomcat6 或者更高版本，可以不进行此项设置。

步骤三：测试

重新部署项目并启动 tomcat，在浏览器中刷新地址 http://localhost:8088/StrutsDay01，效果如下图所示：

图-18

在文本框中输入任意的内容，点击提交，此时表单数据提交给了 HelloAction，HelloAction 接收到了表单数据后，跳转到了 hello.jsp，我们在 hello.jsp 上使用了 EL 表达式来输出 HelloAction 的属性值，效果如下图：

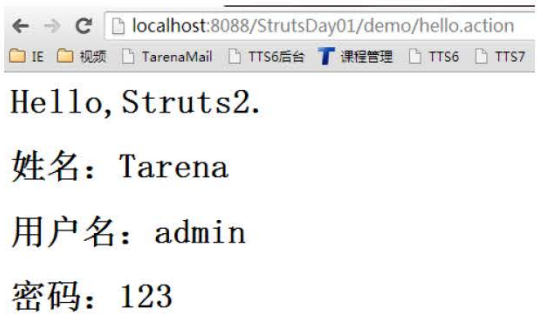


图-19

小结：

- Action 中追加属性时，不必区别何时加 get 方法，何时加 set 方法，通常每个属性都有 get 和 set 方法。
- 页面上写的 EL 表达式，实际上与二种注入方式中，对应的表单文本框的 name 表达式写法一致。

- **完整代码**

本案例的 HelloAction 完整代码：

```
package action;

import com.opensymphony.xwork2.ModelDriven;

import entity.User;

public class HelloAction {

 public HelloAction() {
     System.out.println("实例化 Action...");
 }

 // 定义基本类型属性，接收表单参数：姓名
 private String realName;

 public void setRealName(String realName) {
     System.out.println("注入参数 realName...");
     this.realName = realName;
 }

 public String getRealName() {
     return this.realName;
 }

 //定义实体对象属性，接收表单参数：用户名、密码
 private User user;

 public void setUser(User user) {
     System.out.println("注入对象 user...");
```

```
        this.user = user;
    }

    public User getUser() {
        return this.user;
    }


    /**
     * 在业务方法中输出"Hello, Action."
     */
    public String sayHello() {
        System.out.println("Hello,Action.");

        // 输出基本类型数据
        System.out.println("姓名: " + realName);
        // 输出域模型方式注入的参数
        System.out.println("用户名: " + user.getUserName());
        System.out.println("密码: " + user.getPassword());
        //输出模型驱动方式注入的参数
        System.out.println("员工名: " + emp.getEmpName());
        System.out.println("工资: " + emp.getSalary());

        return "success";
    }

    }
```

hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<html>
<head>
</head>
<body>
 <h1>Hello,Struts2.</h1>

 <h1>姓名: ${realName }</h1>

 <h1>用户名: ${user.userName }</h1>
 <h1>密码: ${user.password }</h1>

</body>
</html>
```

其他代码没有变化，不再重复。

5. NetCTOSS 资费列表，Step1-3

- **问题**

开发 NetCTOSS 项目中，资费模块的查询功能。要求查询全部的资费表数据并以列表的方式显示在页面上，其中分页功能先不实现。具体效果如下图：

图-20

- **方案**

查询功能只包含一个请求，即对资费数据的查询请求。在这次请求中，我们需要查询出资费所有的数据，然后将数据传递给查询列表页面，在页面上用表格显示出来。

使用 Struts2 处理这种请求的过程如下图：

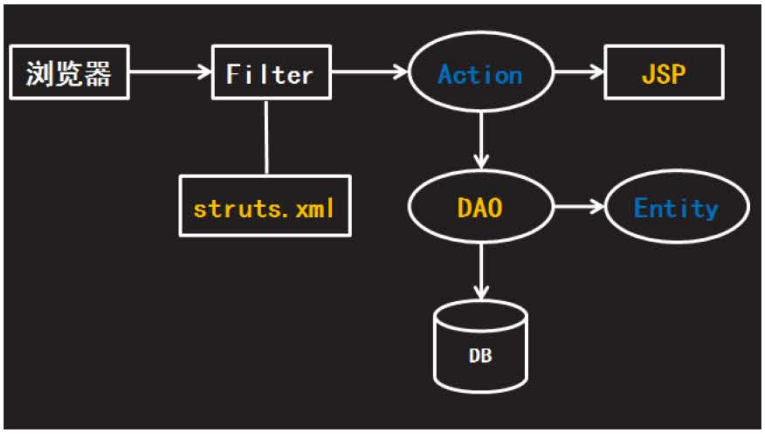


图-21

在这个请求的过程中，我们需要编写的代码有：

- Entity，封装资费表的实体类。
- DAO，数据库访问代码的组件。
- Action，业务流程控制组件。
- struts.xml，Struts2 的配置文件。
- JSP，负责内容的展现。

这些代码的编写顺序，往往是按照他们的相互依赖关系来进行的，优先编写被依赖的代

码，因此推荐代码编写的顺序是 Entity→DAO→Action→struts.xml→JSP。由于步骤比较多，我们分 2 次来完成。本案例中我们先完成 Entity、DAO 代码的编写。

注意：

- struts.xml 写在 JSP 前面的原因是，开发完该配置文件，可以部署并启动项目，而在开发 JSP 时可以一边写一边刷新看效果，不必再重新启动服务了。
- NetCTOSS 项目我们采用 ORACLE 数据库开发。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：项目搭建

项目搭建包括项目的创建以及引入框架等操作，目前 NetCTOSS 项目仅采用了 Struts2 的框架，搭建过程只包含 Struts2 框架的引入即可，因此这一步可以参考今天的第一个案例，即“Struts2 之 HelloWorld，Step1-4”。

项目名称定为 NETCTOSS，后续我们会陆续完成项目的各个功能，代码都将写在这个项目下。项目搭建后结构如下图：

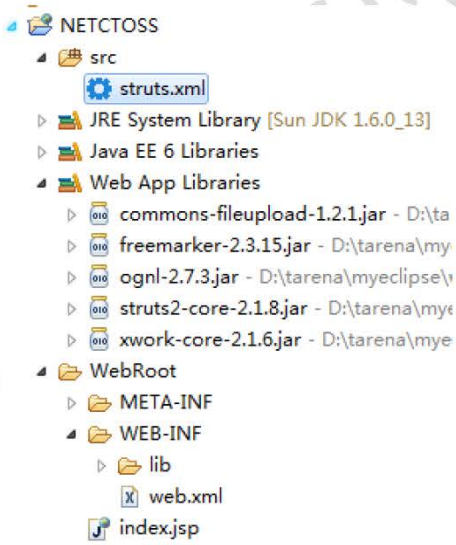


图-22

步骤二：Entity

创建 com.netctoss.entity 包，用于封装实体类。然后创建资费表对应的实体类 Cost，代码如下：

```
package com.netctoss.entity;

import java.sql.Date;

/**
 * 资费实体类
 */
public class Cost {
```

```
    private Integer id;// 主键
    private String name;// 资费名称
    private Integer baseDuration;// 在线时长
    private Double baseCost;// 基本费用
    private Double unitCost;// 单位费用
    private String status;// 状态
    private String descr;// 资费说明
    private Date createTime;// 创建日期
    private Date startTime;// 启用日期
    private String costType;// 资费类型

    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 Integer getBaseDuration() {
        return baseDuration;
    }

    public void setBaseDuration(Integer baseDuration) {
        this.baseDuration = baseDuration;
    }

    public Double getBaseCost() {
        return baseCost;
    }

    public void setBaseCost(Double baseCost) {
        this.baseCost = baseCost;
    }

    public Double getUnitCost() {
        return unitCost;
    }

    public void setUnitCost(Double unitCost) {
        this.unitCost = unitCost;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getDescr() {
        return descr;
    }
```

```
    public void setDescr(String descr) {
        this.descr = descr;
    }

    public Date getCreateTime() {
        return createTime;
    }

    public Date getStartTime() {
        return startTime;
    }

    public void setStartTime(Date startTime) {
        this.startTime = startTime;
    }

    public void setCreateTime(Date createTime) {
        this.createTime = createTime;
    }

    public String getCostType() {
        return costType;
    }

    public void setCostType(String costType) {
        this.costType = costType;
    }

}
```

步骤三：DAO

DAO 是数据库访问组件，用于封装对数据库的增、删、改、查操作。当前请求中查询全部资费数据的方法，需要写在 DAO 中。

首先，我们要创建 DAO 组件并编写本案例中需要使用的查询全部资费数据的代码。但是通常情况下，我们并不是直接写一个 DAO 的类，在类中直接编写这样的代码，而是先定义接口。

定义接口的原因在于，一般情况下业务是比较固定的，针对这些业务我们需要提供的数据库访问方法也相对固定，而通过接口我们是可以描述出这些固定的业务。那么定义接口的目的在于，实际项目中，随着技术的革新和性能上的要求，我们可能会改变实现这种业务的手段，比如当前我们使用 JDBC 来实现 DAO，可能过了一段时间，我们又想使用 Hibernate 替换 JDBC 来实现 DAO。

对于这种场景，如果 DAO 仅仅是一个类，直接实现业务，那么业务代码中会大量的调用它，我们很难再去进行修改了。而使用接口的话，我们业务代码中可以通过工厂（后面会说明如何使用工厂实例化接口）调用接口，修改时只需要重新写个实现类，来替换工厂中原有的实现类即可，对业务代码没有任何的影响。简单来说，定义接口是为了提高代码的复用性，降低代码维护的成本。

因此我们需要创建一个 DAO 接口，并定义查询全部资费的方法，代码如下：

```
package com.netctoss.dao;

import java.util.List;

import com.netctoss.entity.Cost;
```

```
/**
 * 资费 DAO 接口
 */
public interface ICostDao {

 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();

}
```

然后，要实现这个接口，目前我们可以使用 JDBC/MyBatis 来访问数据库，实现这个接口，由于本案例侧重点在 Struts2 的使用，因此就简化 DAO 的实现，这里简单的模拟一下这个查询方法的实现。即创建一个实现类 CostDaoImpl 实现接口 ICostDao，并使用模拟的方式来实现查询方法，代码如下：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;

import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

 @Override
 public List<Cost> findAll() {
     // 模拟查询全部资费数据
     List<Cost> list = new ArrayList<Cost>();

     Cost c1 = new Cost();
     c1.setId(95);
     c1.setName("6 元套餐");
     c1.setBaseDuration(66);
     c1.setBaseCost(6.6);
     c1.setUnitCost(0.6);
     c1.setDescr("6 元套餐");
     c1.setStatus("0");
     c1.setCostType("2");
     list.add(c1);

     Cost c2 = new Cost();
     c2.setId(96);
     c2.setName("8 元套餐");
     c2.setBaseDuration(88);
     c2.setBaseCost(8.8);
     c2.setUnitCost(0.8);
     c2.setDescr("8 元套餐");
     c2.setStatus("0");
     c2.setCostType("2");
     list.add(c2);

     Cost c3 = new Cost();
     c3.setId(97);
```

```
        c3.setName("tarena");
        c3.setBaseDuration(99);
        c3.setBaseCost(9.9);
        c3.setUnitCost(0.9);
        c3.setDescr("tarena 套餐");
        c3.setStatus("0");
        c3.setCostType("2");
        list.add(c3);

        return list;
    }
```

- **完整代码**

本案例的 Entity 完整代码：

```
package com.netctoss.entity;

import java.sql.Date;

/**
 * 资费实体类
 */
public class Cost {

 private Integer id;// 主键
 private String name;// 资费名称
 private Integer baseDuration;// 在线时长
 private Double baseCost;// 基本费用
 private Double unitCost;// 单位费用
 private String status;// 状态
 private String descr;// 资费说明
 private Date createTime;// 创建日期
 private Date startTime;// 启用日期
 private String costType;// 资费类型

 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 Integer getBaseDuration() {
     return baseDuration;
 }

 public void setBaseDuration(Integer baseDuration) {
     this.baseDuration = baseDuration;
 }

 public Double getBaseCost() {
     return baseCost;
 }
```

```
    public void setBaseCost(Double baseCost) {
        this.baseCost = baseCost;
    }

    public Double getUnitCost() {
        return unitCost;
    }

    public void setUnitCost(Double unitCost) {
        this.unitCost = unitCost;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getDescr() {
        return descr;
    }

    public void setDescr(String descr) {
        this.descr = descr;
    }

    public Date getCreateTime() {
        return createTime;
    }

    public Date getStartTime() {
        return startTime;
    }

    public void setStartTime(Date startTime) {
        this.startTime = startTime;
    }

    public void setCreateTime(Date createTime) {
        this.createTime = createTime;
    }

    public String getCostType() {
        return costType;
    }

    public void setCostType(String costType) {
        this.costType = costType;
    }

}
```

DAO 接口完整代码：

```
package com.netctoss.dao;

import java.util.List;

import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {
```

```
 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();

}
```

DAO 实现类完整代码：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;

import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

 @Override
 public List<Cost> findAll() {
     // 模拟查询全部资费数据
     List<Cost> list = new ArrayList<Cost>();

     Cost c1 = new Cost();
     c1.setId(95);
     c1.setName("6 元套餐");
     c1.setBaseDuration(66);
     c1.setBaseCost(6.6);
     c1.setUnitCost(0.6);
     c1.setDescr("6 元套餐");
     c1.setStatus("0");
     c1.setCostType("2");
     list.add(c1);

     Cost c2 = new Cost();
     c2.setId(96);
     c2.setName("8 元套餐");
     c2.setBaseDuration(88);
     c2.setBaseCost(8.8);
     c2.setUnitCost(0.8);
     c2.setDescr("8 元套餐");
     c2.setStatus("0");
     c2.setCostType("2");
     list.add(c2);

     Cost c3 = new Cost();
     c3.setId(97);
     c3.setName("tarena");
     c3.setBaseDuration(99);
     c3.setBaseCost(9.9);
     c3.setUnitCost(0.9);
     c3.setDescr("tarena 套餐");
     c3.setStatus("0");
     c3.setCostType("2");
     list.add(c3);

     return list;
 }
```

6. NetCTOSS 资费列表，Step4-6

- **问题**

继续第 5 个案例，NetCTOSS 资费列表。

- **方案**

本案例中，我们要完成 Action、struts.xml、JSP 代码的编写。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：Action

Action 是业务控制器，需要调用 DAO 组织业务流程，并将计算出的结果传递给页面显示。其中对 DAO 的调用，不要直接在 Action 中实例化 CostDaoImpl，避免业务代码中大量的引用该实现类，如将来更换实现 DAO 方式时不好修改。

这种情况，我们通常情况是使用工厂类来创建 DAO 的实例，并返回其接口类型即可。那么 Action 中通过工厂获取到接口，其依赖的仅仅是这个接口，当 DAO 实现方式变更时，只需要修改工厂即可，项目中所有的 Action 类都不需要做任何的修改。

因此，首先我们创建这个工厂类，使用工厂方法来创建 DAO 的实例，即在 com.netctoss.dao 的包下创建工厂类 DAOFactory，代码如下：

```
package com.netctoss.dao;

/**
 * DAO 工厂，负责统一的创建 DAO 实例。
 */
public class DAOFactory {

 /**
  * 资费 DAO 实例
  */
 private static ICostDao costDao = new CostDaoImpl();

 /**
  * 返回资费 DAO 实例
  */
 public static ICostDao getCostDAO() {
     return costDao;
 }
}
```

接下来，我们要创建业务控制器 Action，并在其中处理查询请求。即创建包 com.netctoss.action，用于封装所有的 Action 类，然后在此包下创建类 FindCostAction，并在 Action 的业务方法中调用 DAO 查询出所有的资费，然后通过属性将结果传递给 JSP。代码如下：

```
package com.netctoss.action;

import java.util.List;
import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;

/**
 *   资费查询Action
 */
public class FindCostAction {

 // output
 private List<Cost> costs;

 /**
  * 业务方法，配置action时如果不指定method属性，
  * 那么Struts2会自动调用execute方法。
  */
 public String execute() {
     // 获取DAO实例
     ICostDao dao = DAOFactory.getCostDAO();
     // 调用DAO，计算输出
     try {
         costs = dao.findAll();
     } catch (Exception e) {
         e.printStackTrace();
         // 捕获到异常时，跳转到错误页面
         return "error";
     }
     // 正常执行时，跳转到资费列表页面
     return "success";
 }

 public List<Cost> getCosts() {
     return costs;
 }

 public void setCosts(List<Cost> costs) {
     this.costs = costs;
 }

}
```

需要注意的是，Action 是业务控制器，它的作用是根据用户的输入来计算输出，并将结果返回给页面显示给用户看，简单说就是根据输入计算输出。因此在实现 Action 时，我们首先分析当前请求的输入和输出分别是什么，然后在 Action 中定义成员变量来封装输入和输出，之后在业务方法中根据输入属性计算输出属性的值即可。

步骤二：struts.xml

在 struts.xml 下，配置 action 以及 JSP。由于目前还没有创建 JSP，因此我们先预想好 JSP 的名称加以配置，后面创建 JSP 时要使用配置中指定的名称，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
```

```
    <!--
        资费模块配置信息：
        一般情况下，一个模块的配置单独封装在一个 package 下，
        并且以模块名来命名 package 的 name 和 namespace。
     -->
    <package name="cost" namespace="/cost" extends="struts-default">
        <!-- 查询资费数据 -->
        <action name="findCost" class="com.netctoss.action.FindCostAction">
            <!--
                正常情况下跳转到资费列表页面。
                一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
             -->
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

   </struts>
```

步骤三：JSP

资费列表页面需要依赖样式文件，因此首先将样式文件及图片复制到项目中来，即将静态项目 NETCTOSS_V02 中 images 和 styles 文件夹复制到 NetCTOSS 项目的 WebRoot 文件夹下，完成后 WebRoot 结构如下图：

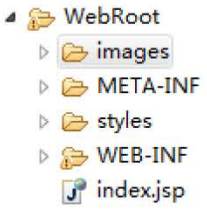


图-23

接下来，在 WEB-INF 下创建文件夹 cost，并在该文件夹下创建 find_cost.jsp。在该 JSP 中，我们先指定文件的编码为 utf-8，然后从静态代码 NETCTOSS_V02 中，将资费列表的 HTML 代码（/fee/ fee_list.html）复制到 JSP 内，完成后代码如下：

```
   <%@page pageEncoding="utf-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内 - NetCTOSS</title>
         <link type="text/css" rel="stylesheet" media="all"
```

```
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort desc")
                    btnObj.className = "sort asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee() {
                var r = window.confirm("确定要删除此资费吗？");
                document.getElementById("operate_result_info").style.display
= "block";
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index off"></a></li>
                <li><a                      href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                      href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                  href="../account/account list.html"
class="account_off"></a></li>
                <li><a                  href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                      href="../bill/bill list.html"
class="bill off"></a></li>
                <li><a                  href="../report/report list.html"
class="report_off"></a></li>
                <li><a                      href="../user/user_info.html"
class="information_off"></a></li>
                <li><a                  href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
```

```
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input type="button" value="基费" class="sort_asc"
onclick="sort(this);" />
                        <input type="button" value="时长" class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input type="button" value="增加" class="btn_add"
onclick="location.href='fee_add.html';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>
                        <tr>
                            <td>1</td>
                            <td><a href="fee_detail.html">包 20 小时</a></td>
                            <td>20 小时</td>
                            <td>24.50 元</td>
                            <td>3.00 元/小时</td>
                            <td>2013/01/01 00:00:00</td>
                            <td></td>
                            <td>暂停</td>
                            <td>
                                <input type="button" value="启用"
class="btn_start" onclick="startFee();" />
                                <input type="button" value="修改"
class="btn_modify" onclick="location.href='fee_modi.html';" />
                                <input type="button" value="删除"
class="btn_delete" onclick="deleteFee();" />
                            </td>
                        </tr>
                        <tr>
                            <td>2</td>
                            <td><a href="fee_detail.html">包 40 小时</a></td>
```

```
                        <td>40 小时</td>
                        <td>40.50 元</td>
                        <td>3.00 元/小时</td>
                        <td>2013/01/21 00:00:00</td>
                        <td>2013/01/23 00:00:00</td>
                        <td>开通</td>
                        <td>
                        </td>
                    </tr>
                </table>
                <p>业务说明：<br />
                1、创建资费时，状态为暂停，记载创建时间；<br />
                2、暂停状态下，可修改，可删除；<br />
                3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                </p>
            </div>
            <!--分页-->
            <div id="pages">
              <a href="#">上一页</a>
                <a href="#" class="current_page">1</a>
                <a href="#">2</a>
                <a href="#">3</a>
                <a href="#">4</a>
                <a href="#">5</a>
                <a href="#">下一页</a>
            </div>
        </form>
    </div>
    <!--主要区域结束-->
    <div id="footer">
        <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
        <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
    </div>
  </body>
</html>
```

然后，在 WEB-INF 下创建文件夹 main，并在该文件夹下创建公用的错误页面 error.jsp。在该 JSP 中，我们先指定文件的编码为 utf-8，然后从静态项目 NETCTOSS_V02 中，将错误页面的 HTML 代码（error.html）复制到 JSP 内，完成后代码如下：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>达内 - NetCTOSS</title>
    <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
    <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
    <script language="javascript" type="text/javascript">
      var timer;
```

```
            //启动跳转的定时器
            function startTimes() {
                timer = window.setInterval(showSecondes,1000);
            }

            var i = 5;
            function showSecondes() {
                if (i > 0) {
                    i--;
                    document.getElementById("secondes").innerHTML = i;
                }
                else {
                    window.clearInterval(timer);
                    location.href = "login.html";
                }
            }

            //取消跳转
            function resetTimer() {
                if (timer != null && timer != undefined) {
                    window.clearInterval(timer);
                    location.href = "login.html";
                }
            }
        </script>
    </head>
    <body class="error page" onload="startTimes();">
        <h1 id="error">
         遇到错误， <span id="secondes">5</span> 秒后将自动跳转，立即跳转请点击 
            <a  href="javascript:resetTimer();">返回</a>
        </h1>
    </body>
</html>
```

完成这一步后，我们先做一下阶段性测试，以验证是否能通过 FindCostAction 访问到 find_cost.jsp。即部署 NetCTOSS 项目，然后启动 tomcat，在浏览器地址栏中输入 http://localhost:8088/NETCTOSS/cost/findCost，点击回车，效果如下图：

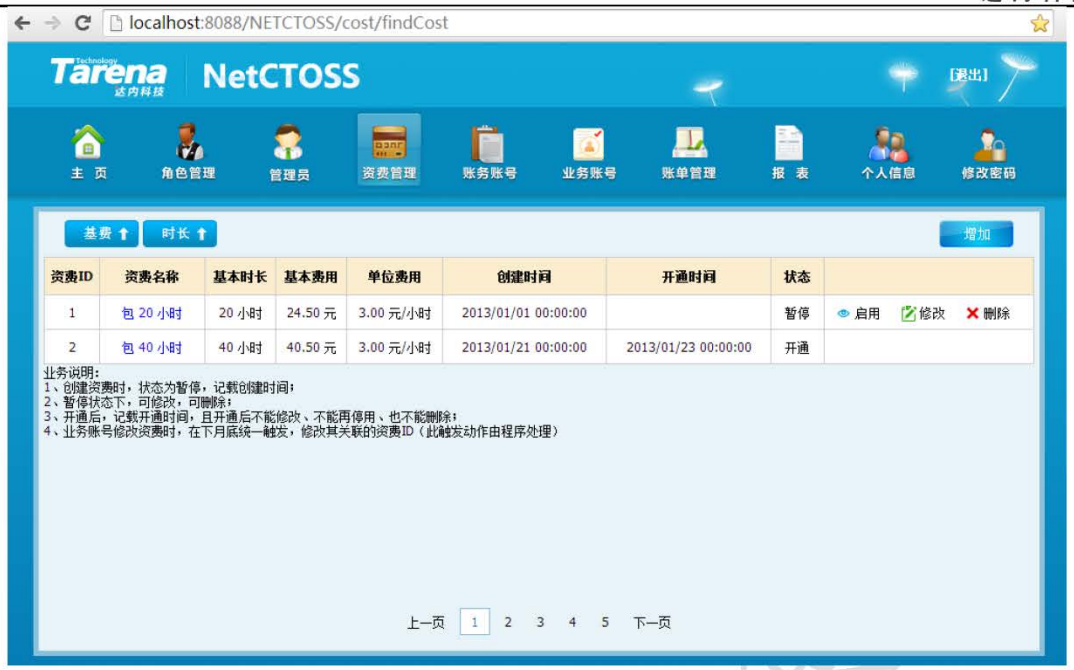


图-24

最后，我们需要修改 find_cost.jsp，将中间的表格区域静态的数据改为动态的数据显示，其数据来源于 FindCostAction。

对于页面上这份集合数据的动态显示，我们需要使用 JSTL 标签+EL 表达式来实现，因此需要先引入 JSTL 的包，即 jstl.jar、standard.jar。引入他们之后，项目中的 lib 包结构如下图：

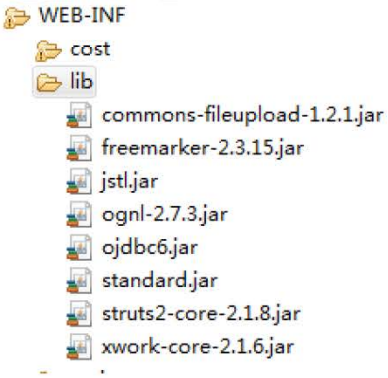


图-25

然后，我们要在 find_cost.jsp 中通过 taglib 指令引入 JSTL 标签，并设置不忽略 EL 表达式（tomcat6 及以上版本不需要设置），相关代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

最后，我们使用 JSTL 标签和 EL 表达式来动态显示页面表格中的数据，代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort_desc")
                    btnObj.className = "sort_asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee() {
                var r = window.confirm("确定要删除此资费吗？");
                document.getElementById("operate_result_info").style.display
= "block";
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a href="../role/role_list.html"
class="role_off"></a></li>
                <li><a href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a href="../account/account_list.html"
class="account_off"></a></li>
                <li><a href="../service/service_list.html"
class="service_off"></a></li>
                <li><a href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a href="../report/report_list.html"
class="report_off"></a></li>
                <li><a href="../user/user_info.html"
class="information_off"></a></li>
                <li><a href="../user/user_modi_pwd.html"
class="password_off"></a></li>
```

```
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input type="button" value="基费" class="sort_asc"
onclick="sort(this);" />
                        <input type="button" value="时长" class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input type="button" value="增加" class="btn_add"
onclick="location.href='fee_add.html';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>


                        <!-- 使用JSTL标签遍历集合，使用EL表达式输出内容。 -->
                        <c:forEach items="${costs}" var="cost">
                         <tr>
                            <td>${cost.id}</td>
                            <td><a href="fee_detail.html">${cost.name}</a></td>
                            <td>${cost.baseDuration}</td>
                            <td>${cost.baseCost}</td>
                            <td>${cost.unitCost}</td>
                            <td>${cost.createTime}</td>
                            <td>${cost.startTime}</td>
                            <td>
                                <c:choose>
                                    <c:when test="${cost.status==0}">开通</c:when>
                                    <c:otherwise>暂停</c:otherwise>
                                </c:choose>
```

```
                            </td>
                            <td>
                                <input type="button" value="启用" class="btn_start"
onclick="startFee();" />
                                <input type="button" value="修改" class="btn_modify"
onclick="location.href='fee_modi.html';" />
                                <input type="button" value="删除" class="btn_delete"
onclick="deleteFee();" />
                            </td>
                        </tr>
                      </c:forEach>


                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                    <a href="#" class="current_page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
  </html>
```

步骤四：单元测试

重新访问此功能，根据页面显示的数据测试代码是否正确。即在浏览器中刷新地址 http://localhost:8088/NETCTOSS/cost/findCost，页面效果如下图：

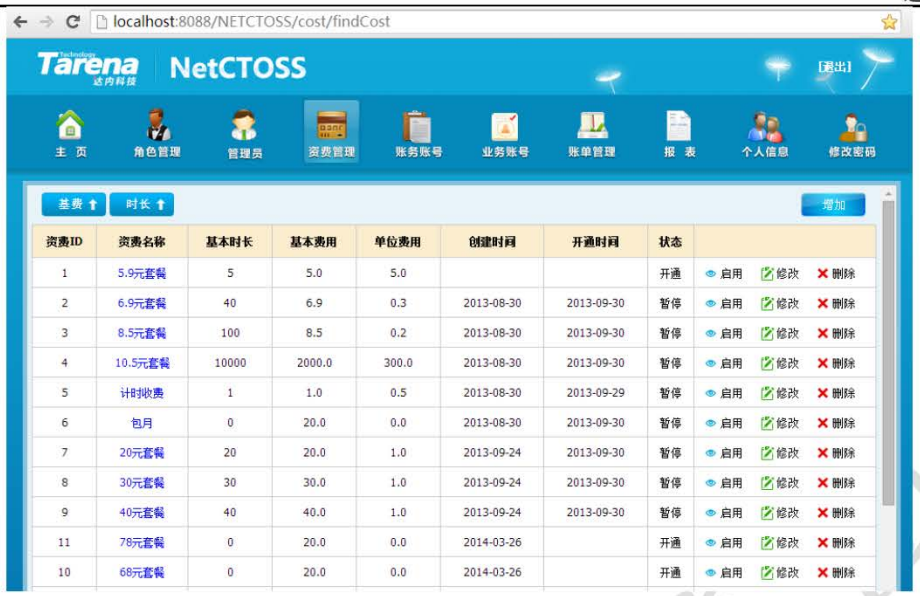


图-26

- **完整代码**

本案例的 DAOFactory 完整代码：

```
package com.netctoss.dao;

/**
 * DAO 工厂，负责统一的创建 DAO 实例。
 */
public class DAOFactory {

 /**
  * 资费 DAO 实例
  */
 private static ICostDao costDao = new CostDaoImpl();

 /**
  * 返回资费 DAO 实例
  */
 public static ICostDao getCostDAO() {
     return costDao;
 }

}
```

FindCostAction 完整代码：

```
package com.netctoss.action;

import java.util.List;
import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;

/**
 *   资费查询 Action
 */
public class FindCostAction {
```

```
    // output
    private List<Cost> costs;

    /**
     * 业务方法，配置 action 时如果不指定 method 属性，
     * 那么 Struts2 会自动调用 execute 方法。
     */
    public String execute() {
        // 获取 DAO 实例
        ICostDao dao = DAOFactory.getCostDAO();
        // 调用 DAO，计算输出
        try {
            costs = dao.findAll();
        } catch (Exception e) {
            e.printStackTrace();
            // 捕获到异常时，跳转到错误页面
            return "error";
        }
        // 正常执行时，跳转到资费列表页面
        return "success";
    }

    public List<Cost> getCosts() {
        return costs;
    }

    public void setCosts(List<Cost> costs) {
        this.costs = costs;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="struts-default">
     <!-- 查询资费数据 -->
     <action name="findCost" class="com.netctoss.action.FindCostAction">
         <!--
             正常情况下跳转到资费列表页面。
             一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
          -->
         <result name="success">
             /WEB-INF/cost/find_cost.jsp
         </result>
         <!--
             错误情况下，跳转到错误页面。
             错误页面可以被所有模块复用，因此放在 main 下，
             该文件夹用于存放公用的页面。
          -->
         <result name="error">
             /WEB-INF/main/error.jsp
         </result>
     </action>
```

```
    </package>

   </struts>
```

find_cost.jsp 完整代码：

```
   <%@page pageEncoding="utf-8" isELIgnored="false"%>
   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
         <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
               if (btnObj.className == "sort_desc")
                  btnObj.className = "sort_asc";
               else
                  btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
               var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
               document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee() {
               var r = window.confirm("确定要删除此资费吗？");
               document.getElementById("operate_result_info").style.display
= "block";
            }
         </script>
      </head>
      <body>
         <!--Logo 区域开始-->
         <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
         </div>
         <!--Logo 区域结束-->
         <!--导航区域开始-->
         <div id="navi">
            <ul id="menu">
               <li><a href="../index.html" class="index_off"></a></li>
               <li><a                        href="../role/role_list.html"
class="role_off"></a></li>
               <li><a                       href="../admin/admin_list.html"
class="admin_off"></a></li>
               <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
               <li><a                   href="../account/account_list.html"
class="account_off"></a></li>
               <li><a                   href="../service/service_list.html"
class="service_off"></a></li>
               <li><a                        href="../bill/bill_list.html"
class="bill_off"></a></li>
```

```
                <li><a                      href="../report/report list.html"
class="report off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information_off"></a></li>
                <li><a                      href="../user/user_modi_pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基费"  class="sort asc"
onclick="sort(this);" />
                        <input  type="button"  value="时长"  class="sort asc"
onclick="sort(this);" />
                    </div>
                    <input  type="button"  value="增加"  class="btn_add"
onclick="location.href='fee_add.html';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                            src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功!
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>

                        <!-- 使用JSTL标签遍历集合，使用EL表达式输出内容。 -->
                        <c:forEach items="${costs}" var="cost">
                         <tr>
                            <td>${cost.id}</td>
                            <td><a
href="fee_detail.html">${cost.name}</a></td>
                            <td>${cost.baseDuration}</td>
                            <td>${cost.baseCost}</td>
                            <td>${cost.unitCost}</td>
                            <td>${cost.createTime}</td>
                            <td>${cost.startTime}</td>
                            <td>
                                <c:choose>
                                    <c:when   test="${cost.status==0}">  开  通
</c:when>
                                    <c:otherwise>暂停</c:otherwise>
                                </c:choose>
                            </td>
                            <td>
```

```
                                <input    type="button"    value="   启   用   "
class="btn start" onclick="startFee();" />
                                <input    type="button"    value="   修   改   "
class="btn modify" onclick="location.href='fee modi.html';" />
                                <input    type="button"    value="   删   除   "
class="btn delete" onclick="deleteFee();" />
                            </td>
                        </tr>
                    </c:forEach>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                    <a href="#" class="current_page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
</html>
```

error.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内－NetCTOSS</title>
        <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
        <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            var timer;
            //启动跳转的定时器
            function startTimes() {
                timer = window.setInterval(showSecondes,1000);
            }

            var i = 5;
            function showSecondes() {
                if (i > 0) {
```

```
                    i--;
                    document.getElementById("secondes").innerHTML = i;
                }
                else {
                    window.clearInterval(timer);
                    location.href = "login.html";
                }
            }

            //取消跳转
            function resetTimer() {
                if (timer != null && timer != undefined) {
                    window.clearInterval(timer);
                    location.href = "login.html";
                }
            }
        </script>
    </head>
    <body class="error_page" onload="startTimes();">
        <h1 id="error">
         遇到错误， <span id="secondes">5</span> 秒后将自动跳转，立即跳
转请点击 
            <a  href="javascript:resetTimer();">返回</a>
        </h1>
    </body>
</html>
```

课后作业

1. 请简述 Struts2 与 Struts1 的区别和联系

2. Struts2 如何实现 MVC，与 Spring MVC 有什么不同？

3. 在 Struts2 中页面如何向 Action 传参

达内IT培训集团

Struts 核心

Unit02

知识体系.....Page 64

OGNL	OGNL 介绍	什么是 OGNL
		为什么用 OGNL
		OGNL 原理
	OGNL 用法	Struts2 显示标签
		2 个常用的 OGNL 表达式
		6 个需要了解的 OGNL 表达式
ValueStack	ValueStack 介绍	什么是 ValueStack
		ValueStack 原理
	访问 ValueStack	1、通过 debug 标签观察其结构
		2、输出栈顶
		3、访问 Context 对象
		4、迭代集合
		5、按数字迭代
		总结：ValueStack 栈顶的变化
	Struts2 对 EL 的支持	EL 表达式如何访问 ValueStack
重构资费列表	重构资费列表	用 Struts2 标签+OGNL 重构资费列表
Action 基本原理	6 大核心组件	6 大核心组件关系
		6 大核心组件作用

经典案例.....Page 72

使用 OGNL 表达式-1	2 个常用的 OGNL 表达式
使用 OGNL 表达式-2	6 个需要了解的 OGNL 表达式
访问 ValueStack-1	1、通过 debug 标签观察其结构
	2、输出栈顶
	3、访问 context 对象
访问 ValueStack-2	4、迭代集合
	5、按数字迭代
重构资费列表	用 Struts2 标签+OGNL 重构资费列表

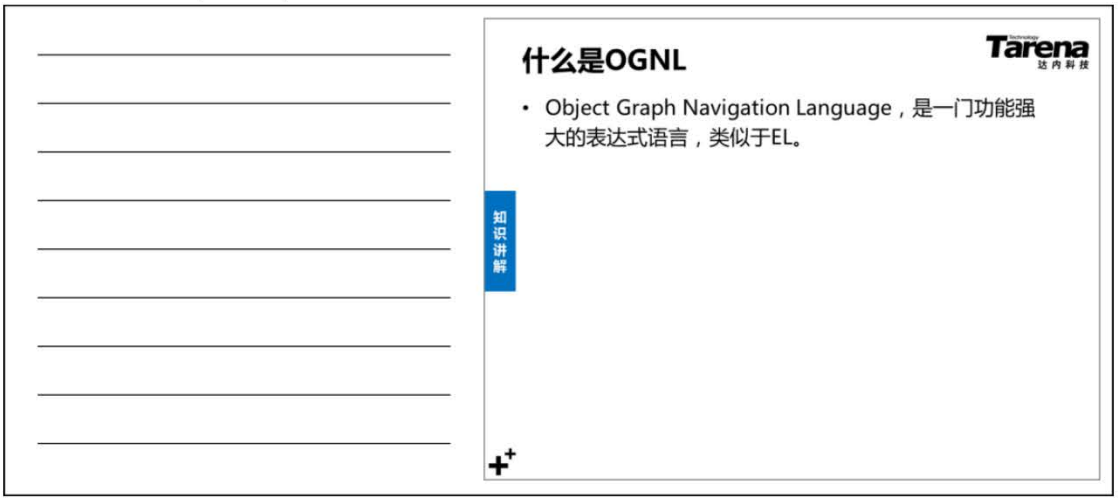
课后作业.....Page 112

达内IT培训集团

1. OGNL

1.1. OGNL 介绍

1.1.1. 【OGNL 介绍】什么是 OGNL

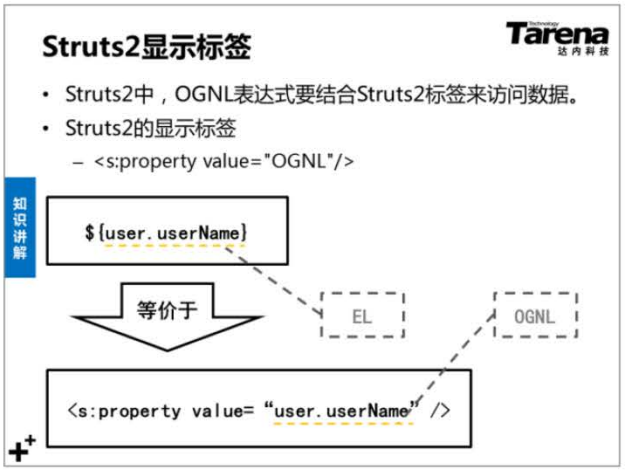


1.1.2. 【OGNL 介绍】为什么用 OGNL

1.1.3. 【OGNL 介绍】OGNL 原理

1.2. OGNL 用法

1.2.1. 【OGNL 用法】Struts2 显示标签



1.2.2. 【OGNL 用法】2 个常用的 OGNL 表达式

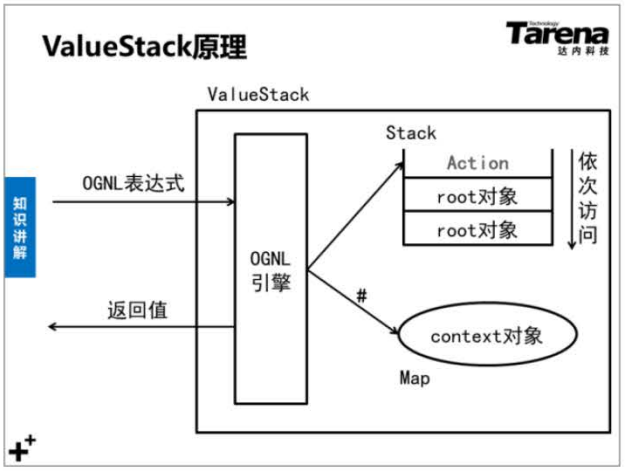
1.2.3. 【OGNL 用法】6 个需要了解的 OGNL 表达式

2. ValueStack

2.1. ValueStack 介绍

2.1.1. 【ValueStack 介绍】什么是 ValueStack

2.1.2. 【ValueStack 介绍】ValueStack 原理



2.2. 访问 ValueStack

2.2.1. 【访问 ValueStack】1、通过 debug 标签观察其结构

2.2.2. 【访问 ValueStack】2、输出栈顶

2、输出栈顶

- 直接输出栈顶
  - <s:property/>
  - 看输出的内容，是否与通过debug标签观察到的一致

解说识知

2.2.3. 【访问 ValueStack】3、访问 Context 对象

3、访问Context对象

- 输出context对象数据
  - OGNL表达式以“#”开头
  - 以key来访问context对象的值
  - 即通过OGNL表达式“#key”，得到context中某属性的值

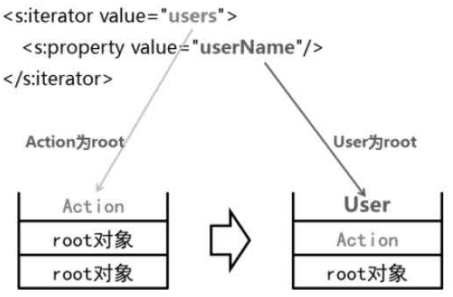
解说识知

2.2.4. 【访问 ValueStack】4、迭代集合

4、迭代集合

- 语法

```
<s:iterator value="users">
  <s:property value="userName"/>
</s:iterator>
```



解说识知

4、迭代集合（续1）

解详识知

- 解释
  - users是OGNL表达式，自顶向下访问ValueStack栈中root对象的users属性，这里会从栈顶的Action对象取到该集合属性（List<User> users）值
  - 在循环过程中，循环变量User会被压入栈顶，此时Action被压到栈的第二位
  - userName是OGNL表达式，自顶向下访问ValueStack栈中root对象的userName属性，这里会从栈顶的User对象取到userName属性值

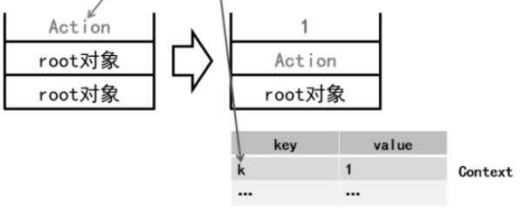
2.2.5. 【访问 ValueStack】5、按数字迭代

5、按数字迭代

解详识知

- 语法

```
<s:iterator begin="from" end="to"  var="k">
  <s:property value="#k"/>
</s:iterator>
```



5、按数字迭代（续1）

解详识知

- 解释
  - from/to是OGNL表达式，自顶向下访问ValueStack栈中root对象的from/to属性，这里会从栈顶的Action对象取到属性（int from=1, to=3）值
  - 在循环过程中，循环变量会被压入栈顶，此时Action被压到栈的第二位
  - 我们不能以数字做root对象，因此无法写OGNL访问栈顶的整数，此时可以声明循环变量k，该声明会将循环变量加入context对象中，如k=2
  - 此时，我们可以写#k这样的OGNL表达式来访问context对象，从而得到循环变量的值
  - 其实，我们也可以不写OGNL表达式，而是直接输出栈顶的值，即<s:property/>

2.2.6. 【访问 ValueStack】总结：ValueStack 栈顶的变化

总结：ValueStack栈顶的变化

解说识知

- 默认情况下栈顶为Action
- 循环过程中，栈顶为循环变量
  - 迭代集合时，循环变量是集合中的对象，通常都是实体对象，即栈顶为实体对象，我们可以以实体对象为root来写OGNL表达式
  - 按数字迭代时，循环变量是数字，我们不能以数字为实体对象写OGNL，如果需要引用该数字，需要通过var声明变量名，然后以“#变量名”来引用它，这种情况下，我们是从context对象中取出的值
- 循环结束后，栈顶变回Action

2.3. Struts2 对 EL 的支持

2.3.1. 【Struts2 对 EL 的支持】EL 表达式如何访问 ValueStack

EL表达式如何访问ValueStack

解说识知

- Struts2将数据封装于ValueStack，默认使用OGNL取值。
- Struts2也支持使用EL表达式取值，那么EL是从哪里取的值，如何取的值呢？
  - 实际上，EL也是从ValueStack中取的值
  - EL默认的取值范围是page, request, session, application
  - Struts2重写了request的getAttribute方法，先试图从原始request中取值，如果没取到再从ValueStack中取值

EL表达式如何访问ValueStack（续1）

解说识知

```
public Object getAttribute(String s) {
    ActionContext ctx = ActionContext.getContext();
    Object attribute = super.getAttribute(s);
    if (ctx != null) {
        if (attribute == null) {
            boolean alreadyIn = false;
            Boolean b = (Boolean) ctx.get("__requestWrapper.getAttribute");
            if (b != null) {
                alreadyIn = b.booleanValue();
            }
            if (!alreadyIn && s.indexOf("#") == -1) {
                try {
                    ctx.put("__requestWrapper.getAttribute", Boolean.TRUE);
                    ValueStack stack = ctx.getValueStack();
                    if (stack != null) {
                        attribute = stack.findValue(s);
                    }
                } finally {
                    ctx.put("__requestWrapper.getAttribute", Boolean.FALSE);
                }
            }
        }
    }
    return attribute;
}
```

3. 重构资费列表

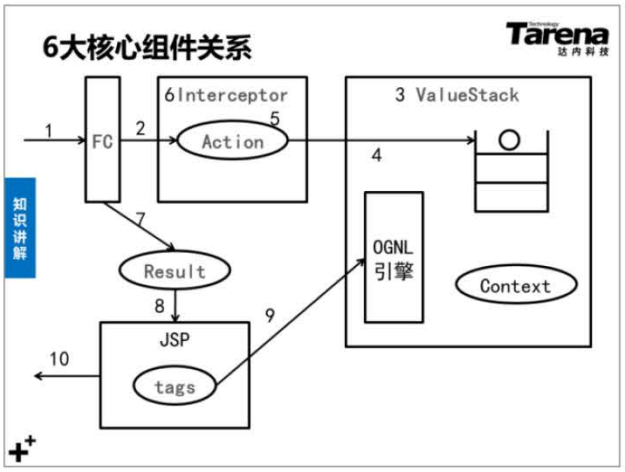
3.1. 重构资费列表

3.1.1. 【重构资费列表】用 Struts2 标签+OGNL 重构资费列表

4. Action 基本原理

4.1. 6 大核心组件

4.1.1. 【6 大核心组件】6 大核心组件关系



4.1.2. 【6 大核心组件】6 大核心组件作用

达内IT培训集团

经典案例

1. 使用 OGNL 表达式-1

- **问题**

练习 OGNL 表达式的 8 种用法，即

1) 访问基本属性
2) 访问实体对象
3) 访问集合/数组
4) 访问 Map
5) 访问时运算
6) 访问时调用方法
7) 创建集合
8) 创建 Map

本案例中，先练习前 2 种用法，他们是最常用的。

- **方案**

在一次请求当中，我们可以在页面上使用 OGNL 访问 Action 中的基本属性和实体对象属性，这里重点是关注 OGNL 的写法，因此案例中 Action 和 JSP 可以简化处理。

我们可以复用 StrutsDay01 中的 HelloWorld 示例，在该示例的 Action 中追加基本属性和实体对象，在 JSP 中用 OGNL 进行访问。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一： 准备项目

选中项目 StrutsDay01，右键复制，如下图：

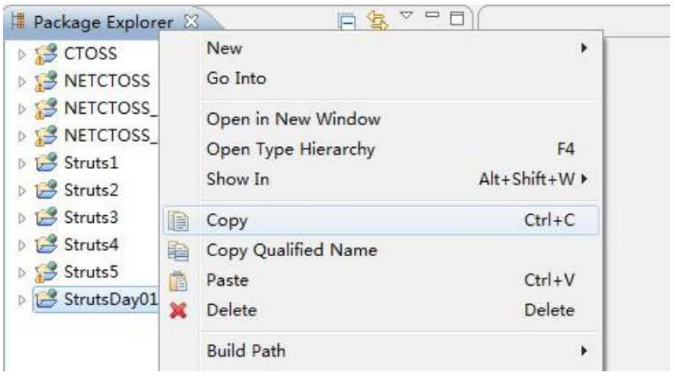


图-1

粘贴项目 StrutsDay01，如下图：

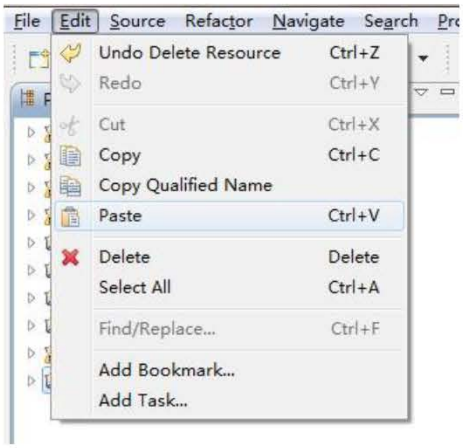


图-2

在弹出框中，将项目名改为 StrutsDay02，如下图：

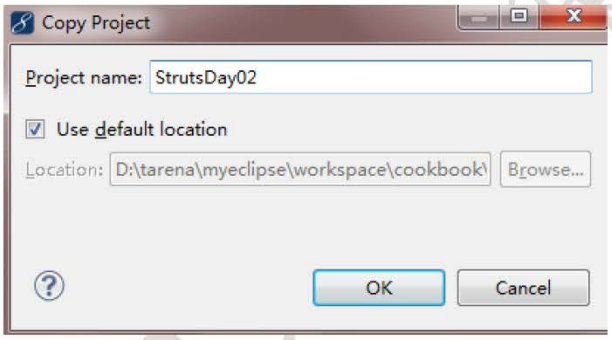


图-3

选中 StrutsDay02，右键选择 Properties，如下图：

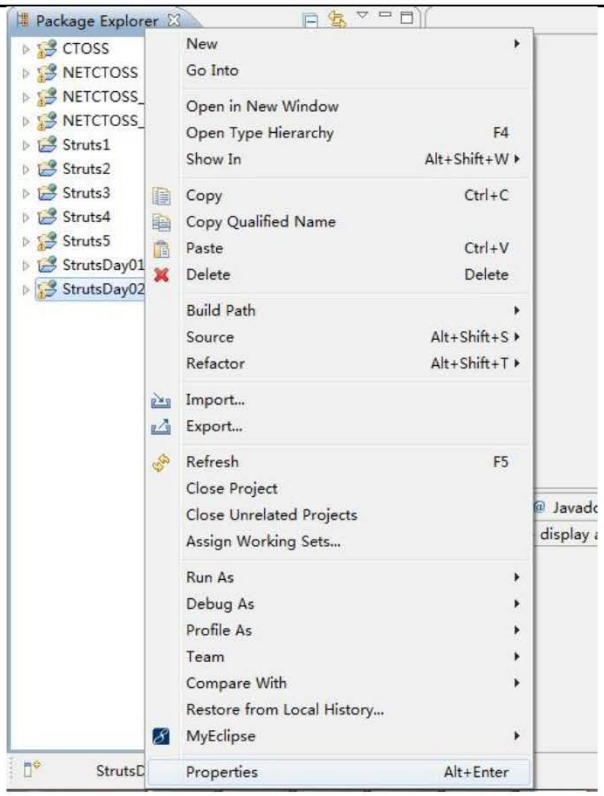


图-4

在弹出框中，将此项目的 web 访问地址由 StrutsDay01 改为 StrutsDay02，如下图：

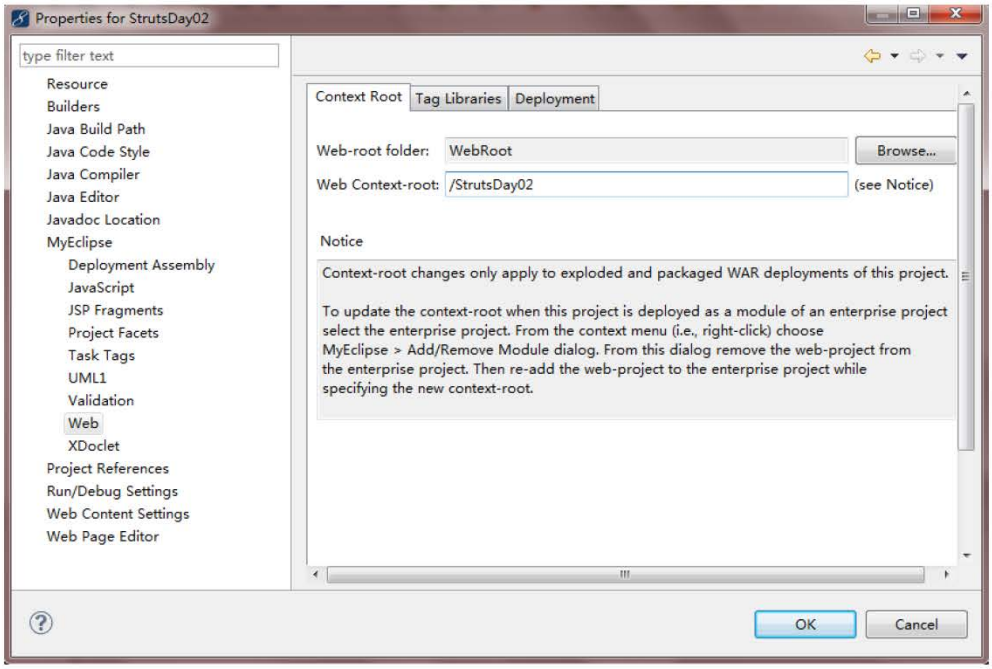


图-5

修改 HelloAction，删除其他演示代码，只保留业务方法即可，代码如下：

```
package action;

public class HelloAction {
```

```
    public String sayHello() {
        return "success";
    }
}
```

修改 hello.jsp，删除其他演示代码，只保留基本的 HTML 结构即可，代码如下：

```
<%@page pageEncoding="utf-8"%>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>
</body>
</html>
```

此时，项目 StrutsDay02 的结构如下图：

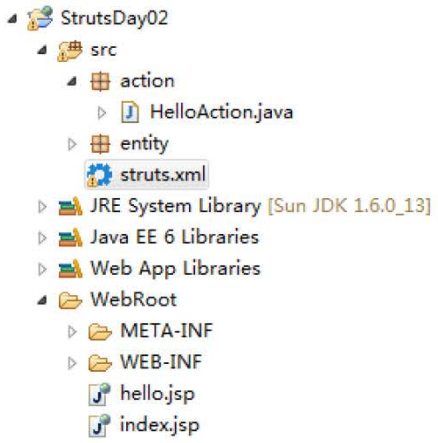


图-6

部署项目，启动 tomcat 并访问 http://localhost:8088/StrutsDay02/demo/hello，效果如下图则项目准备完毕：

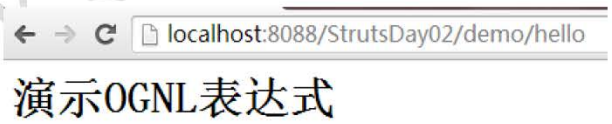


图-7

步骤二：使用 OGNL 访问 Action 基本属性

在 HelloAction 中追加任意基本属性并初始化，同时提供 get、set 方法，代码如下：

```
package action;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
```

```
    public String sayHello() {
        return "success";
    }

    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;
    }

}
```

在 hello.jsp 中，先通过 taglib 指令引入 Struts2 标签，然后在 Struts2 的显示标签上，使用 OGNL 表达式输出 HelloAction 中的基本属性值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示OGNL表达式</h1>

 <h1>1.基本属性</h1>
 <h2>ID:<s:property value="id"/></h2>
 <h2>Name:<s:property value="name"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-8

步骤三：使用 OGNL 访问 Action 实体对象属性

在 HelloAction 中追加实体属性并初始化，在业务方法中初始化实体对象的数据，同

时提供 get、set 方法，代码如下：

```
package action;

import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";


 // 实体对象
 private User user = new User();


 public String sayHello() {


     // 初始化实体对象数据
     user.setUserName("zhangsan");
     user.setPassword("123456");


     return "success";
 }

 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 User getUser() {
     return user;
 }

 public void setUser(User user) {
     this.user = user;
 }


}
```

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式输出 HelloAction 中的实体对象值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
```

```
<body>
 <h1>演示 OGNL 表达式</h1>

 <h1>1.基本属性</h1>
 <h2>ID:<s:property value="id"/></h2>
 <h2>Name:<s:property value="name"/></h2>

 <h1>2.实体对象</h1>
 <h2>用户名:<s:property value="user.userName"/></h2>
 <h2>密码:<s:property value="user.password"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-9

- **完整代码**

本案例的 HelloAction 完整代码：

```
package action;

import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";

 // 实体对象
 private User user = new User();

 public String sayHello() {
     // 初始化实体对象数据
     user.setUserName("zhangsan");
     user.setPassword("123456");
     return "success";
 }
```

```
    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 User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

}
```

Hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <h1>1.基本属性</h1>
 <h2>ID:<s:property value="id"/></h2>
 <h2>Name:<s:property value="name"/></h2>

 <h1>2.实体对象</h1>
 <h2>用户名:<s:property value="user.userName"/></h2>
 <h2>密码:<s:property value="user.password"/></h2>

</body>
</html>
```

2. 使用 OGNL 表达式-2

- **问题**

练习使用 OGNL 表达式的后 6 种用法，即

3) 访问集合/数组
4) 访问 Map
5) 访问时运算
6) 访问时调用方法
7) 创建集合

8) 创建 Map

相对于前 2 种用法，这些用法不是很常用，同学们以了解为主。

- **方案**

在项目 StrutsDay02 基础上，使用 OGNL 表达式的后 6 种用法访问 HelloAction 中的属性值。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：Action 中准备数据

由于用法较多，比较繁琐，我们一次性在 Action 中准备好页面要访问的全部数据，这样就不用反复重启 tomcat 了。即在 HelloAction 中准备如下类型的数据：集合、数组、Map，代码如下：

```
package action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
 // 实体对象
 private User user = new User();


 // 数组和集合
 private String[] cityArray = new String[] { "beijing", "shanghai", "guangzhou" };
 private List<String> cityList = new ArrayList<String>();
 // Map
 private Map<String, String> cityMap = new HashMap<String, String>();


 public String sayHello() {
     // 初始化实体对象数据
     user.setUserName("zhangsan");
     user.setPassword("123456");


     // 初始化集合数据
     cityList.add("shenzhen");
     cityList.add("chongqing");
     cityList.add("qingdao");
     // 初始化 Map 数据
     cityMap.put("beijing", "2300 万人口");
```

```
        cityMap.put("shanghai", "2000万人口");
        cityMap.put("guangzhou", "1800万人口");

        return "success";
    }

    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 User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }


    public String[] getCityArray() {
        return cityArray;
    }

    public void setCityArray(String[] cityArray) {
        this.cityArray = cityArray;
    }

    public List<String> getCityList() {
        return cityList;
    }

    public void setCityList(List<String> cityList) {
        this.cityList = cityList;
    }

    public Map<String, String> getCityMap() {
        return cityMap;
    }

    public void setCityMap(Map<String, String> cityMap) {
        this.cityMap = cityMap;
    }

}
```

步骤二：访问集合/数组

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式输出 HelloAction 中的数组和集合属性值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

<!—此处略去其他表达式的演示... -->

 <h1>3.数组和集合</h1>
 <h2>数组中的城市:<s:property value="cityArray[1]"/></h2>
 <h2>集合中的城市:<s:property value="cityList[1]"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

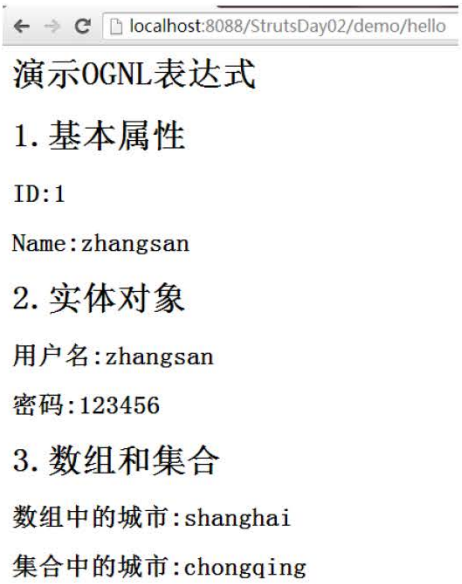


图-10

步骤三：访问 Map

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式输出 HelloAction 中的 Map 属性值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去其他表达式的演示... -->
```

```
<h1>4.Map</h1>
<h2>Map 中的城市人口:<s:property value="cityMap.beijing"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-11

步骤四：访问时进行运算

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式输出 HelloAction 中的任意属性值，同时可以在 OGNL 表达式中对返回结果进行运算，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去其他表达式的演示... -->


 <h1>5.访问时运算</h1>
 <h2>My Name:<s:property value="'My name is '+name"/></h2>
 <h2>去哪:<s:property value="cityArray[1]+'，我来了！'"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-12

步骤五：访问时调用方法

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式输出 HelloAction 中的任意属性值，同时可以在 OGNL 中调用返回值的方法，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去其他表达式的演示... -->


 <h1>6.访问时调用方法</h1>
 <h2>MY NAME:<s:property value="name.toUpperCase()"/></h2>


</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

localhost:8088/StrutsDay02/demo/hello

2. 实体对象

用户名:zhangsan

密码:123456

3. 数组和集合

数组中的城市:shanghai

集合中的城市:chongqing

4. Map

Map中的城市人口:2300万人口

5. 访问时运算

My Name:My name is zhangsan

去哪:shanghai，我来了！

6. 访问时调用方法

MY NAME:ZHANGSAN

图-13

步骤六：创建集合

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式创建一个临时的集合并输出，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去其他表达式的演示... -->


 <h1>7.创建集合</h1>
 <h2>创建集合:<s:property value="{'a','b','c'}"/></h2>
 <h2>集合类型:<s:property value="{'a','b','c'}.getClass().getName()"/></h2>


</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-14

步骤七：创建 Map

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 表达式创建一个临时的 Map 并输出，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去其他表达式的演示... -->


 <h1>8.创建 Map</h1>
 <h2>创建 Map:<s:property value="#{'mm':'MM','nn':'NN'}"/></h2>
 <h2>
     Map 类型:<s:property value="#{'mm':'MM','nn':'NN'}.getClass().getName()"/>
 </h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

localhost:8088/StrutsDay02/demo/hello

4. Map

Map中的城市人口:2300万人口

5. 访问时运算

My Name:My name is zhangsan

去哪:shanghai，我来了！

6. 访问时调用方法

MY NAME:ZHANGSAN

7. 创建集合

创建集合:[a, b, c]

集合类型:java.util.ArrayList

8. 创建Map

创建Map:{mm=MM, nn=NN}

Map类型:java.util.LinkedHashMap

图-15

- **完整代码**

本案例的 HelloAction 完整代码：

```
package action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
 // 实体对象
 private User user = new User();
 // 数组和集合
 private String[] cityArray = new String[] { "beijing", "shanghai",
"guangzhou" };
 private List<String> cityList = new ArrayList<String>();
 // Map
 private Map<String, String> cityMap = new HashMap<String, String>();

 public String sayHello() {
     // 初始化实体对象数据
     user.setUserName("zhangsan");
     user.setPassword("123456");
     // 初始化集合数据
     cityList.add("shenzhen");
     cityList.add("chongqing");
     cityList.add("qingdao");
```

```
        // 初始化Map数据
        cityMap.put("beijing", "2300万人口");
        cityMap.put("shanghai", "2000万人口");
        cityMap.put("guangzhou", "1800万人口");

        return "success";
    }

    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 User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String[] getCityArray() {
        return cityArray;
    }

    public void setCityArray(String[] cityArray) {
        this.cityArray = cityArray;
    }

    public List<String> getCityList() {
        return cityList;
    }

    public void setCityList(List<String> cityList) {
        this.cityList = cityList;
    }

    public Map<String, String> getCityMap() {
        return cityMap;
    }

    public void setCityMap(Map<String, String> cityMap) {
        this.cityMap = cityMap;
    }

}
```

hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
```

```
    <body>
     <h1>演示 OGNL 表达式</h1>

     <h1>1.基本属性</h1>
     <h2>ID:<s:property value="id"/></h2>
     <h2>Name:<s:property value="name"/></h2>

     <h1>2.实体对象</h1>
     <h2>用户名:<s:property value="user.userName"/></h2>
     <h2>密码:<s:property value="user.password"/></h2>

     <h1>3.数组和集合</h1>
     <h2>数组中的城市:<s:property value="cityArray[1]"/></h2>
     <h2>集合中的城市:<s:property value="cityList[1]"/></h2>

     <h1>4.Map</h1>
     <h2>Map 中的城市人口:<s:property value="cityMap.beijing"/></h2>

     <h1>5.访问时运算</h1>
     <h2>My Name:<s:property value="'My name is '+name"/></h2>
     <h2>去哪:<s:property value="cityArray[1]+', 我来了！'"/></h2>

     <h1>6.访问时调用方法</h1>
     <h2>MY NAME:<s:property value="name.toUpperCase()"/></h2>

     <h1>7.创建集合</h1>
     <h2>创建集合:<s:property value="{'a','b','c'}"/></h2>
     <h2>集合类型:<s:property value="{'a','b','c'}.getClass().getName()"/></h2>

     <h1>8.创建 Map</h1>
     <h2>创建 Map:<s:property value="#{'mm':'MM','nn':'NN'}"/></h2>
     <h2>
         Map                    类                    型                    :<s:property
value="#{'mm':'MM','nn':'NN'}.getClass().getName()"/>
     </h2>

    </body>
    </html>
```

其他代码不变，不再重复。

3. 访问 ValueStack-1

- **问题**

在页面上，使用 Struts2 标签和 OGNL 表达式观察 ValueStack 的结构，并访问 ValueStack 中栈对象和 Context 对象的数据，包括如下内容

1) 观察 ValueStack 结构
2) 输出栈顶内容
3) 输出 Context 对象中的数据
4) 遍历集合
5) 按数字迭代

本案例中，我们先练习前 3 个知识点。

- **方案**

复用项目 StrutsDay02，在 hello.jsp 上写 Struts2 标签和 OGNL 表达式观察并访问 ValueStack。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：观察 ValueStack 结构

在 hello.jsp 上增加<s:debug/>标签，通过该标签来观察 ValueStack 结构，代码如下图：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去 OGNL 表达式的演示... -->


 <!-- ValueStack 示例 -->
 <h1>1.观察 ValueStack 结构</h1>
 <h2><s:debug/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-16

步骤二：输出栈顶内容

在 hello.jsp 上，使用 Struts2 标签输出栈顶的内容，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去 OGNL 表达式的演示... -->

 <!-- ValueStack 示例 -->
 <h1>1.观察 ValueStack 结构</h1>
 <h2><s:debug/></h2>


 <h1>2.栈顶</h1>
 <h2>Stack Top:<s:property/></h2>


</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-17

步骤三：输出 Context 对象中的数据

hello.jsp 中，在 Struts2 显示标签上使用 OGNL 输出 ValueStack 中 Context 对象的值，表达式格式为“#key”，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去 OGNL 表达式的演示... -->

 <!-- ValueStack 示例 -->
 <h1>1.观察 ValueStack 结构</h1>
 <h2><s:debug/></h2>

 <h1>2.栈顶</h1>
 <h2>Stack Top:<s:property/></h2>


 <h1>3.输出 context 对象中的数据</h1>
 <h2>context:<s:property value="#action.user.userName"/></h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

图-18

- **完整代码**

本案例的 hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
```

```
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <h1>1.基本属性</h1>
 <h2>ID:<s:property value="id"/></h2>
 <h2>Name:<s:property value="name"/></h2>

 <h1>2.实体对象</h1>
 <h2>用户名:<s:property value="user.userName"/></h2>
 <h2>密码:<s:property value="user.password"/></h2>

 <h1>3.数组和集合</h1>
 <h2>数组中的城市:<s:property value="cityArray[1]"/></h2>
 <h2>集合中的城市:<s:property value="cityList[1]"/></h2>

 <h1>4.Map</h1>
 <h2>Map 中的城市人口:<s:property value="cityMap.beijing"/></h2>

 <h1>5.访问时运算</h1>
 <h2>My Name:<s:property value="'My name is '+name"/></h2>
 <h2>去哪:<s:property value="cityArray[1]+', 我来了！'"/></h2>

 <h1>6.访问时调用方法</h1>
 <h2>MY NAME:<s:property value="name.toUpperCase()"/></h2>

 <h1>7.创建集合</h1>
 <h2>创建集合:<s:property value="{'a','b','c'}"/></h2>
 <h2>集合类型:<s:property value="{'a','b','c'}.getClass().getName()"/></h2>

 <h1>8.创建 Map</h1>
 <h2>创建 Map:<s:property value="#{'mm':'MM','nn':'NN'}"/></h2>
 <h2>
     Map 类 型 :<s:property
value="#{'mm':'MM','nn':'NN'}.getClass().getName()"/>
 </h2>

 <!-- ValueStack 示例 -->
 <h1>1.观察 ValueStack 结构</h1>
 <h2><s:debug/></h2>

 <h1>2.栈顶</h1>
 <h2>Stack Top:<s:property/></h2>

 <h1>3.输出 context 对象中的数据</h1>
 <h2>context:<s:property value="#action.user.userName"/></h2>
</body>
</html>
```

其他代码不变，不再重复。

4. 访问 ValueStack-2

- **问题**

练习使用 ValueStack，即：

6) 遍历集合

7) 按数字迭代

- **方案**

复用项目 StrutsDay02，在 hello.jsp 上写 Struts2 标签和 OGNL 表达式，练习迭代集合以及按数字进行迭代。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：遍历集合

在 HelloAction 中追加集合属性并实例化，同时提供 get、set 方法，在业务方法中初始化集合的数据，代码如下：

```
package action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import entity.Emp;
import entity.User;

public class HelloAction {
 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
 // 实体对象
 private User user = new User();
 // 数组和集合
 private String[] cityArray = new String[] { "beijing", "shanghai",
"guangzhou" };
 private List<String> cityList = new ArrayList<String>();
 // Map
 private Map<String, String> cityMap = new HashMap<String, String>();

 // 员工集合
 private List<Emp> emps = new ArrayList<Emp>();

 public String sayHello() {
     // 初始化实体对象数据
     user.setUserName("zhangsan");
     user.setPassword("123456");
     // 初始化集合数据
     cityList.add("shenzhen");
     cityList.add("chongqing");
     cityList.add("qingdao");
     // 初始化 Map 数据
     cityMap.put("beijing", "2300 万人口");
     cityMap.put("shanghai", "2000 万人口");
```

```
        cityMap.put("guangzhou", "1800万人口");

        // 初始化员工数据
        Emp e1 = new Emp();
        e1.setEmpName("刘备");
        e1.setSalary(8000.0);
        emps.add(e1);
        Emp e2 = new Emp();
        e2.setEmpName("关羽");
        e2.setSalary(6000.0);
        emps.add(e2);
        Emp e3 = new Emp();
        e3.setEmpName("张飞");
        e3.setSalary(5000.0);
        emps.add(e3);

        return "success";
    }

    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 User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String[] getCityArray() {
        return cityArray;
    }

    public void setCityArray(String[] cityArray) {
        this.cityArray = cityArray;
    }

    public List<String> getCityList() {
        return cityList;
    }

    public void setCityList(List<String> cityList) {
        this.cityList = cityList;
    }

    public Map<String, String> getCityMap() {
        return cityMap;
    }

    public void setCityMap(Map<String, String> cityMap) {
```

```
        this.cityMap = cityMap;
    }

    public List<Emp> getEmps() {
        return emps;
    }

    public void setEmps(List<Emp> emps) {
        this.emps = emps;
    }

}
```

在 hello.jsp 中，使用迭代标签遍历上述集合属性，并输出集合中变量 emp 的属性值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去 OGNL 表达式的演示... -->

 <!-- ValueStack 示例 -->

 <h1>1.观察 ValueStack 结构</h1>
 <h2><s:debug/></h2>

 <h1>2.栈顶</h1>
 <h2>Stack Top:<s:property/></h2>

 <h1>3.输出 context 对象中的数据</h1>
 <h2>context:<s:property value="#action.user.userName"/></h2>

 <h1>4.遍历集合</h1>
 <h2>
     <!-- 迭代标签，遍历集合 -->
     <s:iterator value="emps">
         <!--
             迭代过程中，栈顶为 Emp 对象，
             写 OGNL 输出其属性值。
          -->
         员工：<s:property value="empName"/>，
         工资为<s:property value="salary"/><br/>
     </s:iterator>
 </h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

localhost:8088/StrutsDay02/demo/hello

8. 创建Map

创建Map:{mm=MM, nn=NN}

Map类型:java.util.LinkedHashMap

1. 观察ValueStack结构

[Debug]

2. 栈顶

Stack Top:action.HelloAction@555aff

3. 输出context对象中的数据

context:zhangsan

4. 遍历集合

员工：刘备， 工资为8000.0
员工：关羽， 工资为6000.0
员工：张飞， 工资为5000.0

图-19

步骤二：按数字迭代

在 HelloAction 中追加迭代的起止值并实例化，同时提供 get、set 方法，代码如下：

```
package action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import entity.Emp;
import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
 // 实体对象
 private User user = new User();
 // 数组和集合
 private String[] cityArray = new String[] { "beijing", "shanghai",
"guangzhou" };
 private List<String> cityList = new ArrayList<String>();
 // Map
 private Map<String, String> cityMap = new HashMap<String, String>();

 // 员工集合
 private List<Emp> emps = new ArrayList<Emp>();
```

```
//迭代数字的起止值
private Integer from = 1;
private Integer to = 3;

public String sayHello() {
    // 初始化实体对象数据
    user.setUserName("zhangsan");
    user.setPassword("123456");
    // 初始化集合数据
    cityList.add("shenzhen");
    cityList.add("chongqing");
    cityList.add("qingdao");
    // 初始化 Map 数据
    cityMap.put("beijing", "2300 万人口");
    cityMap.put("shanghai", "2000 万人口");
    cityMap.put("guangzhou", "1800 万人口");
    // 初始化员工数据
    Emp e1 = new Emp();
    e1.setEmpName("刘备");
    e1.setSalary(8000.0);
    emps.add(e1);
    Emp e2 = new Emp();
    e2.setEmpName("关羽");
    e2.setSalary(6000.0);
    emps.add(e2);
    Emp e3 = new Emp();
    e3.setEmpName("张飞");
    e3.setSalary(5000.0);
    emps.add(e3);

    return "success";
}

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 User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public String[] getCityArray() {
    return cityArray;
}
```

```
    public void setCityArray(String[] cityArray) {
        this.cityArray = cityArray;
    }

    public List<String> getCityList() {
        return cityList;
    }

    public void setCityList(List<String> cityList) {
        this.cityList = cityList;
    }

    public Map<String, String> getCityMap() {
        return cityMap;
    }

    public void setCityMap(Map<String, String> cityMap) {
        this.cityMap = cityMap;
    }

    public List<Emp> getEmps() {
        return emps;
    }

    public void setEmps(List<Emp> emps) {
        this.emps = emps;
    }


    public Integer getFrom() {
        return from;
    }

    public void setFrom(Integer from) {
        this.from = from;
    }

    public Integer getTo() {
        return to;
    }

    public void setTo(Integer to) {
        this.to = to;
    }

}
```

在 hello.jsp 中，使用迭代标签按照 HelloAction 中的起止值循环，并输出循环变量的值，代码如下：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <!—此处略去 OGNL 表达式的演示... -->

 <!-- ValueStack 示例 -->
```

```
<h1>1.观察 ValueStack 结构</h1>
<h2><s:debug/></h2>

<h1>2.栈顶</h1>
<h2>Stack Top:<s:property/></h2>

<h1>3.输出 context 对象中的数据</h1>
<h2>context:<s:property value="#action.user.userName"/></h2>

<h1>4.遍历集合</h1>
<h2>
    <!-- 迭代标签，遍历集合 -->
    <s:iterator value="emps">
        <!--
            迭代过程中，栈顶为 Emp 对象，
            写 OGNL 输出其属性值。
         -->
        员工：<s:property value="empName"/>，
        工资为<s:property value="salary"/><br/>
    </s:iterator>
</h2>

<h1>5.循环数字</h1>
<h2>
    <!-- 迭代标签，按数字迭代 -->
    <s:iterator begin="from" end="to" var="p">
        <!--
            迭代过程中，栈顶为数字，但不能以数字为 root 写 OGNL。
            此时可以声明变量 p，通过 Context 对象实现对变量的访问。
         -->
        <s:property value="#p"/>
    </s:iterator>
</h2>

</body>
</html>
```

重新部署 StrutsDay02 并启动 tomcat 后，访问 StrutsDay02，效果如下图：

localhost:8088/StrutsDay02/demo/hello

Map类型:java.util.LinkedHashMap

1. 观察ValueStack结构

[Debug]

2. 栈顶

Stack Top:action.HelloAction@12b0f8a

3. 输出context对象中的数据

context:zhangsan

4. 遍历集合

员工：刘备， 工资为8000.0
员工：关羽， 工资为6000.0
员工：张飞， 工资为5000.0

5. 循环数字

1 2 3

图-20

- **完整代码**

本案例的 HelloAction 完整代码：

```
package action;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import entity.Emp;
import entity.User;

public class HelloAction {

 // 基本类型
 private Integer id = 1;
 private String name = "zhangsan";
 // 实体对象
 private User user = new User();
 // 数组和集合
 private String[] cityArray = new String[] { "beijing", "shanghai",
"guangzhou" };
 private List<String> cityList = new ArrayList<String>();
 // Map
 private Map<String, String> cityMap = new HashMap<String, String>();

 // 员工集合
 private List<Emp> emps = new ArrayList<Emp>();
 //迭代数字的起止值
 private Integer from = 1;
 private Integer to = 3;
```

```
    public String sayHello() {
        // 初始化实体对象数据
        user.setUserName("zhangsan");
        user.setPassword("123456");
        // 初始化集合数据
        cityList.add("shenzhen");
        cityList.add("chongqing");
        cityList.add("qingdao");
        // 初始化 Map 数据
        cityMap.put("beijing", "2300 万人口");
        cityMap.put("shanghai", "2000 万人口");
        cityMap.put("guangzhou", "1800 万人口");
        // 初始化员工数据
        Emp e1 = new Emp();
        e1.setEmpName("刘备");
        e1.setSalary(8000.0);
        emps.add(e1);
        Emp e2 = new Emp();
        e2.setEmpName("关羽");
        e2.setSalary(6000.0);
        emps.add(e2);
        Emp e3 = new Emp();
        e3.setEmpName("张飞");
        e3.setSalary(5000.0);
        emps.add(e3);

        return "success";
    }

    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 User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String[] getCityArray() {
        return cityArray;
    }

    public void setCityArray(String[] cityArray) {
        this.cityArray = cityArray;
    }

    public List<String> getCityList() {
        return cityList;
    }

    public void setCityList(List<String> cityList) {
```

```
        this.cityList = cityList;
    }

    public Map<String, String> getCityMap() {
        return cityMap;
    }

    public void setCityMap(Map<String, String> cityMap) {
        this.cityMap = cityMap;
    }

    public List<Emp> getEmps() {
        return emps;
    }

    public void setEmps(List<Emp> emps) {
        this.emps = emps;
    }

    public Integer getFrom() {
        return from;
    }

    public void setFrom(Integer from) {
        this.from = from;
    }

    public Integer getTo() {
        return to;
    }

    public void setTo(Integer to) {
        this.to = to;
    }

}
```

hello.jsp 完整代码：

```
<%@page pageEncoding="utf-8" %>
<%@taglib uri="/struts-tags" prefix="s" %>
<html>
<head>
</head>
<body>
 <h1>演示 OGNL 表达式</h1>

 <h1>1.基本属性</h1>
 <h2>ID:<s:property value="id"/></h2>
 <h2>Name:<s:property value="name"/></h2>

 <h1>2.实体对象</h1>
 <h2>用户名:<s:property value="user.userName"/></h2>
 <h2>密码:<s:property value="user.password"/></h2>

 <h1>3.数组和集合</h1>
 <h2>数组中的城市:<s:property value="cityArray[1]"/></h2>
 <h2>集合中的城市:<s:property value="cityList[1]"/></h2>

 <h1>4.Map</h1>
 <h2>Map 中的城市人口:<s:property value="cityMap.beijing"/></h2>

 <h1>5.访问时运算</h1>
 <h2>My Name:<s:property value="'My name is '+name"/></h2>
```

```
    <h2>去哪:<s:property value="cityArray[1]+', 我来了! '"/></h2>

    <h1>6.访问时调用方法</h1>
    <h2>MY NAME:<s:property value="name.toUpperCase()"/></h2>

    <h1>7.创建集合</h1>
    <h2>创建集合:<s:property value="{'a','b','c'}"/></h2>
    <h2>集合类型:<s:property value="{'a','b','c'}.getClass().getName()"/></h2>

    <h1>8.创建 Map</h1>
    <h2>创建 Map:<s:property value="#{'mm':'MM','nn':'NN'}"/></h2>
    <h2>
        Map 类 型 :<s:property
value="#{'mm':'MM','nn':'NN'}.getClass().getName()"/>
    </h2>

    <!-- ValueStack 示例 -->
    <h1>1.观察 ValueStack 结构</h1>
    <h2><s:debug/></h2>

    <h1>2.栈顶</h1>
    <h2>Stack Top:<s:property/></h2>

    <h1>3.输出 context 对象中的数据</h1>
    <h2>context:<s:property value="#action.user.userName"/></h2>

    <h1>4.遍历集合</h1>
    <h2>
        <!-- 迭代标签, 遍历集合 -->
        <s:iterator value="emps">
            <!--
                迭代过程中, 栈顶为 Emp 对象,
                写 OGNL 输出其属性值。
             -->
            员工: <s:property value="empName"/>,
            工资为<s:property value="salary"/><br/>
        </s:iterator>
    </h2>

    <h1>5.循环数字</h1>
    <h2>
        <!-- 迭代标签, 按数字迭代 -->
        <s:iterator begin="from" end="to" var="p">
            <!--
                迭代过程中, 栈顶为数字, 但不能以数字为 root 写 OGNL。
                此时可以声明变量 p, 通过 Context 对象实现对变量的访问。
             -->
            <s:property value="#p"/>
        </s:iterator>
    </h2>

   </body>
   </html>
```

5. 重构资费列表

- **问题**

我们已经学会了使用Struts2标签+OGNL表达式访问ValueStack ,从而访问到Action中的数据，那么请使用该手段重构 NetCTOSS 资费列表页面 find_cost.jsp，将该页面上

的 JSTL 标签+EL 表达式重构为 Struts2 标签+OGNL 表达式。

- **方案**

使用 Struts2 的迭代标签及 OGNL 表达式重构 find_cost.jsp。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：重构资费列表页面

重构 find_cost.jsp，将其中的 JSTL 标签+EL 表达式替换成 Struts2 标签+OGNL 表达式，代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global color.css" />
        <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort desc")
                    btnObj.className = "sort asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate result info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
```

```
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                       href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                    href="../account/account list.html"
class="account_off"></a></li>
                <li><a                    href="../service/service list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report list.html"
class="report off"></a></li>
                <li><a                          href="../user/user info.html"
class="information_off"></a></li>
                <li><a                      href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基 费 "  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value="时 长 "  class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input   type="button"   value="  增  加  "   class="btn_add"
onclick="location.href='toAddCost';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                            src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
```

```
                        <th>创建时间</th>
                        <th>开通时间</th>
                        <th class="width50">状态</th>
                        <th class="width200"></th>
                    </tr>

                    <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                    <s:iterator value="costs">
                     <tr>
                        <td><s:property value="id"/></td>
                        <td><a              href="fee_detail.html"><s:property
value="name"/></a></td>
                        <td><s:property value="baseDuration"/></td>
                        <td><s:property value="baseCost"/></td>
                        <td><s:property value="unitCost"/></td>
                        <td><s:property value="createTime"/></td>
                        <td><s:property value="startTime"/></td>
                        <td>
                            <s:if test="status==0">开通</s:if>
                            <s:else>暂停</s:else>
                        </td>
                        <td>
                            <input type="button" value="启用" class="btn_start"
onclick="startFee();" />
                            <input type="button" value="修改" class="btn_modify"
onclick="location.href='toUpdateCost?id=<s:property value="id"/>';" />
                            <input type="button" value="删除" class="btn_delete"
onclick="deleteFee(<s:property value="id"/>);" />
                        </td>
                     </tr>
                   </s:iterator>

                </table>
                <p>业务说明：<br />
                1、创建资费时，状态为暂停，记载创建时间；<br />
                2、暂停状态下，可修改，可删除；<br />
                3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                </p>
            </div>
            <!--分页-->
            <div id="pages">
              <a href="#">上一页</a>
                <a href="#" class="current page">1</a>
                <a href="#">2</a>
                <a href="#">3</a>
                <a href="#">4</a>
                <a href="#">5</a>
                <a href="#">下一页</a>
            </div>
        </form>
    </div>
    <!--主要区域结束-->
    <div id="footer">
```

```
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
</html>
```

重新部署并启动 tomcat，访问资费列表页面，效果如下图：

图-21

- **完整代码**

本案例的 find_cost.jsp 完整代码：

```
    <%@page pageEncoding="utf-8" isELIgnored="false"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>达内－NetCTOSS</title>
            <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
            <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
            <script language="javascript" type="text/javascript">
                //排序按钮的点击事件
                function sort(btnObj) {
                    if (btnObj.className == "sort desc")
                        btnObj.className = "sort asc";
                    else
                        btnObj.className = "sort_desc";
                }
```

```
            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate result info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                         href="../admin/admin list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                   href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                   href="../service/service list.html"
class="service off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill_off"></a></li>
                <li><a                     href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user info.html"
class="information off"></a></li>
                <li><a                     href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基费"  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value="时长"  class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input  type="button"  value="增加"  class="btn add"
onclick="location.href='toAddCost';" />
                </div>
                <!--启用操作的操作提示-->
```

```
                <div id="operate result info" class="operate success">
                   <img                          src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                   删除成功!
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                   <table id="datalist">
                      <tr>
                         <th>资费 ID</th>
                         <th class="width100">资费名称</th>
                         <th>基本时长</th>
                         <th>基本费用</th>
                         <th>单位费用</th>
                         <th>创建时间</th>
                         <th>开通时间</th>
                         <th class="width50">状态</th>
                         <th class="width200"></th>
                      </tr>

                      <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                      <s:iterator value="costs">
                       <tr>
                          <td><s:property value="id"/></td>
                          <td><a          href="fee_detail.html"><s:property
value="name"/></a></td>
                          <td><s:property value="baseDuration"/></td>
                          <td><s:property value="baseCost"/></td>
                          <td><s:property value="unitCost"/></td>
                          <td><s:property value="createTime"/></td>
                          <td><s:property value="startTime"/></td>
                          <td>
                              <s:if test="status==0">开通</s:if>
                              <s:else>暂停</s:else>
                          </td>
                          <td>
                             <input    type="button"    value="   启   用   "
class="btn_start" onclick="startFee();" />
                             <input    type="button"    value="   修   改   "
class="btn_modify"        onclick="location.href='toUpdateCost?id=<s:property
value="id"/>';" />
                             <input    type="button"    value="   删   除   "
class="btn_delete" onclick="deleteFee(<s:property value="id"/>);" />
                          </td>
                       </tr>
                     </s:iterator>

                   </table>
                   <p>业务说明：<br />
                   1、创建资费时，状态为暂停，记载创建时间；<br />
                   2、暂停状态下，可修改，可删除；<br />
                   3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                   4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                   </p>
                </div>
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                   <a href="#" class="current_page">1</a>
                   <a href="#">2</a>
                   <a href="#">3</a>
                   <a href="#">4</a>
```

```
                <a href="#">5</a>
                <a href="#">下一页</a>
            </div>
        </form>
    </div>
    <!--主要区域结束-->
    <div id="footer">
        <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
        <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
    </div>
  </body>
</html>
```

其他代码不变，不再重复。

达内IT培训集团

课后作业

1. OGNL 表达式有哪些用法，你熟悉其中哪几种

2. 请简述 Struts2 中 ValueStack 组件运行原理

3. 请简述 Struts2 中各组件的作用及调用顺序

Struts 核心

Unit03

知识体系.....Page 115

登录	如何获取 Session	获取 Session 的方式
		各种方式的对比
	登录的需求	登录的需求
	开发思路	请求 1： 打开登录页面
		请求 2：登录验证
	开发步骤	请求 1
		请求 2
Result 原理	Result 介绍	什么是 Result 组件
	常用类型的 Result	dispatcher
		stream
		redirectAction
		json
stream Result	作用	stream Result 的作用
	使用方式	语法
		使用步骤
	登录验证码	需求描述
		开发思路
		开发步骤-请求 1
		开发步骤-请求 2
redirectAction Result	作用	redirectAction Result 的作用
	使用方式	语法
		使用步骤
	资费删除	需求描述
		开发思路
		开发步骤

经典案例.....Page 125

NetCTOSS 登录，请求 1	请求 1
NetCTOSS 登录，请求 2	请求 2

达内IT培训集团

登录验证码-1	开发步骤-请求 1
登录验证码-2	开发步骤-请求 2
资费删除	开发步骤

课后作业……Page 176

1. 登录

1.1. 如何获取 Session

1.1.1. 【如何获取 Session】获取 Session 的方式

获取Session的方式

- ActionContext
  - ActionContext.getContext().getSession()
  - 返回类型为Map<String,Object>
- ServletActionContext
  - ServletActionContext.getRequest().getSession()
  - 返回类型为HttpSession
- SessionAware
  - 让Action实现SessionAware接口
  - 实现void setSession(Map<String,Object> session)方法，Struts2会在实例化Action后调用该方法，通过方法参数将Session对象注入进来
  - 定义成员变量，接收注入进来的Session对象

1.1.2. 【如何获取 Session】各种方式的对比

各种方式的对比

- 返回类型对比
  - 第1、3种方式，获取的Session是Map<String,Object>类型，Struts2采用该类型的目的是简化Session对象，而Session的存储结构和Map是一致的
  - 第2种方式，获取的Session是HttpSession，为了保持兼容性，Struts2提供了获取该类型的方式
- 获取方式对比
  - 第1、2种方式，是我们主动的获取Session
  - 第3种方式，是采用注入的方式自动注入Session，这种方式是被动的
- 推荐使用第3种方式
  - 采用注入思想，更为灵活
  - 面向接口编程，符合主流规范

1.2. 登录的需求

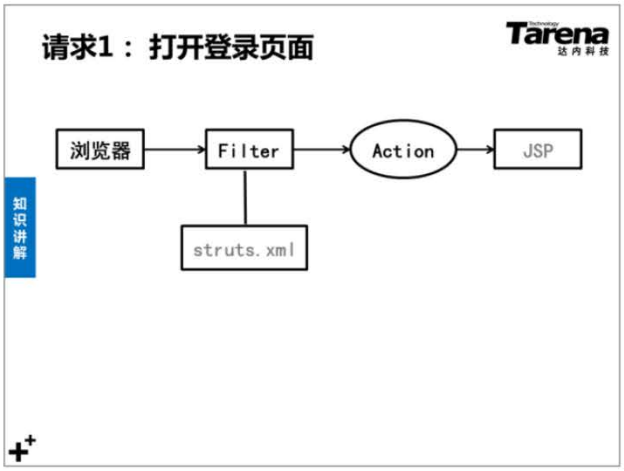
1.2.1. 【登录的需求】登录的需求

登录的需求

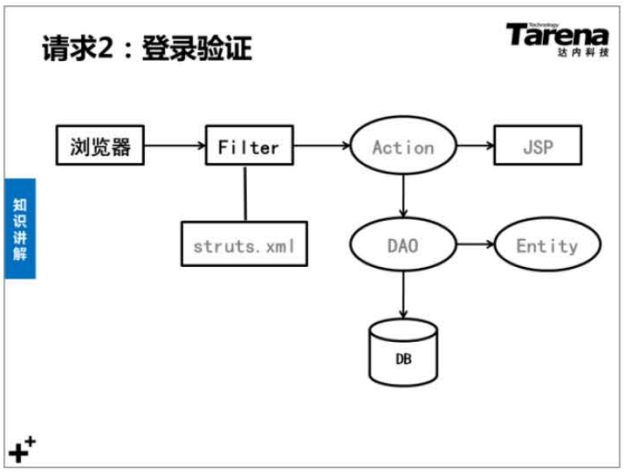
- 打开登录页面
- 输入账号、密码
- 点击登录，进行校验
  - 账号、密码正确，跳转至系统首页
  - 账号或密码错误，刷新登录页，并给与提示
- 暂不考虑验证码

1.3. 开发思路

1.3.1. 【开发思路】请求 1： 打开登录页面



1.3.2. 【开发思路】请求 2：登录验证



1.4. 开发步骤

1.4.1. 【开发步骤】请求 1

1.4.2. 【开发步骤】请求 2

请求2

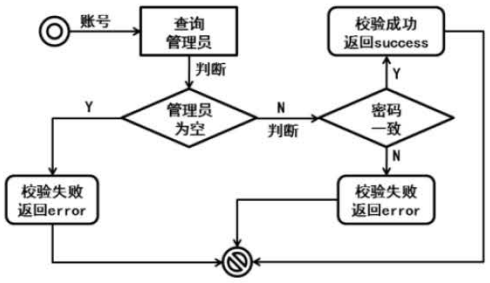
1、创建管理员实体类Admin.java
2、创建登录模块DAO，增加根据账号查询管理员的方法
3、创建登录校验Action
- 输入属性：账号、密码
- 输出属性：提示信息
- 创建业务方法，校验

4、struts.xml中配置action
5、登录页面上，设置表单项

解说知识

请求2（续1）

- 校验逻辑



解说知识

2. Result 原理

2.1. Result 介绍

2.1.1. 【Result 介绍】什么是 Result 组件

什么是Result组件

- Result是用于做输出的组件，用于向页面输出一些内容，转发、重定向可以理解为特殊方式的输出
- 每一个Result实际上就是一个类，这些类都实现了共同的接口Result
- Struts2预置了10种类型的Result，他们被定义在struts-default.xml中

```
<result-types>
    <result-type name="chain" class="com.opensymphony.xwork2.ActionChainResult"/>
    <result-type name="dispatcher" class="org.apache.struts2.dispatcher.ServletDispatche
    <result-type name="freemarker" class="org.apache.struts2.views.freemarker.Freemarker
    <result-type name="httpheader" class="org.apache.struts2.dispatcher.HttpHeaderResult
    <result-type name="redirect" class="org.apache.struts2.dispatcher.ServletRedirectRes
    <result-type name="redirectAction" class="org.apache.struts2.dispatcher.ServletActio
    <result-type name="stream" class="org.apache.struts2.dispatcher.StreamResult"/>
    <result-type name="velocity" class="org.apache.struts2.dispatcher.VelocityResult"/>
    <result-type name="xslt" class="org.apache.struts2.views.xslt.XSLTResult"/>
    <result-type name="plainText" class="org.apache.struts2.dispatcher.PlainTextResult"
</result-types>
```

解说知识

2.2. 常用类型的 Result

2.2.1. 【常用类型的 Result】dispatcher

dispatcher

- 用于转发的Result，可以将请求转发给JSP
- 此种类型Result对应的类为ServletDispatcherResult
- 此类Result是Struts2默认的Result类型，该默认行为是定义该Result时，通过default="true"指定的
- 在此之前，我们使用的Result都是这种默认的Result

解详识知

2.2.2. 【常用类型的 Result】stream

stream

- 用于向页面输出二进制数据，此种类型的Result，可以将二进制数据输出到请求发起端
- 此种类型Result对应的类为StreamResult

解详识知

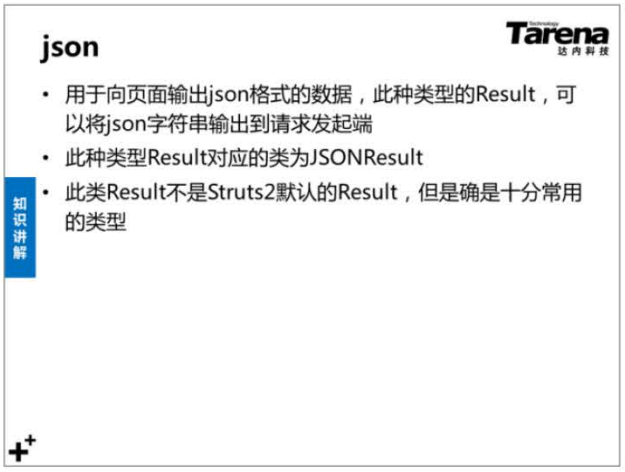
2.2.3. 【常用类型的 Result】redirectAction

redirectAction

- 用于将请求重定向给另外一个Action
- 此种类型Result对应的类为ServletActionRedirectResult

解详识知

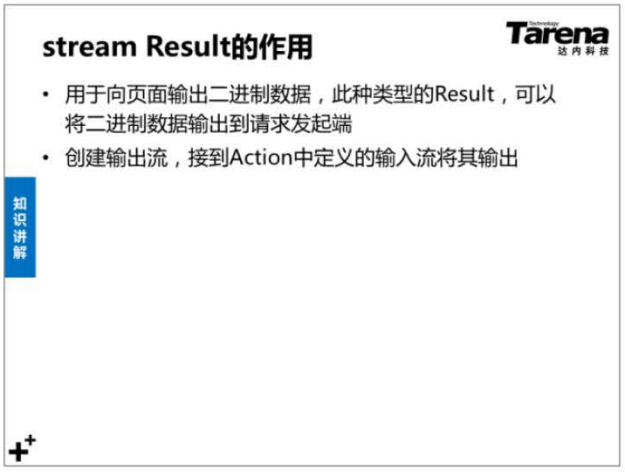
2.2.4. 【常用类型的 Result】json



3. stream Result

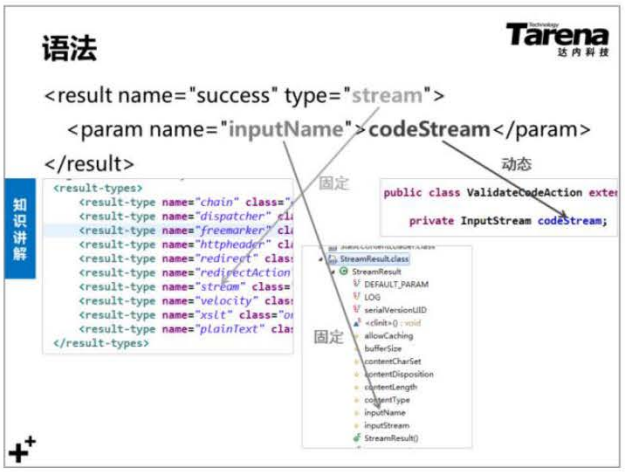
3.1. 作用

3.1.1. 【作用】stream Result 的作用

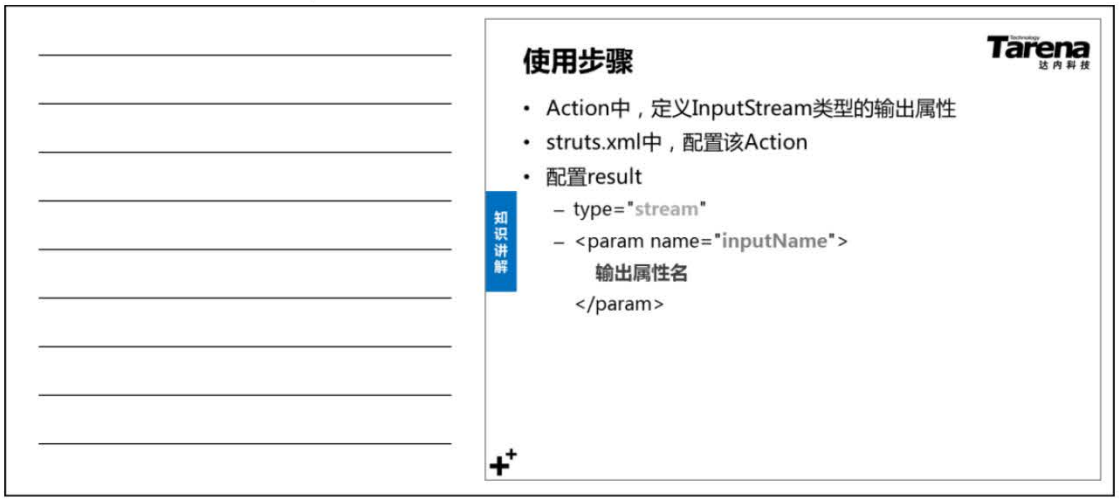


3.2. 使用方式

3.2.1. 【使用方式】语法

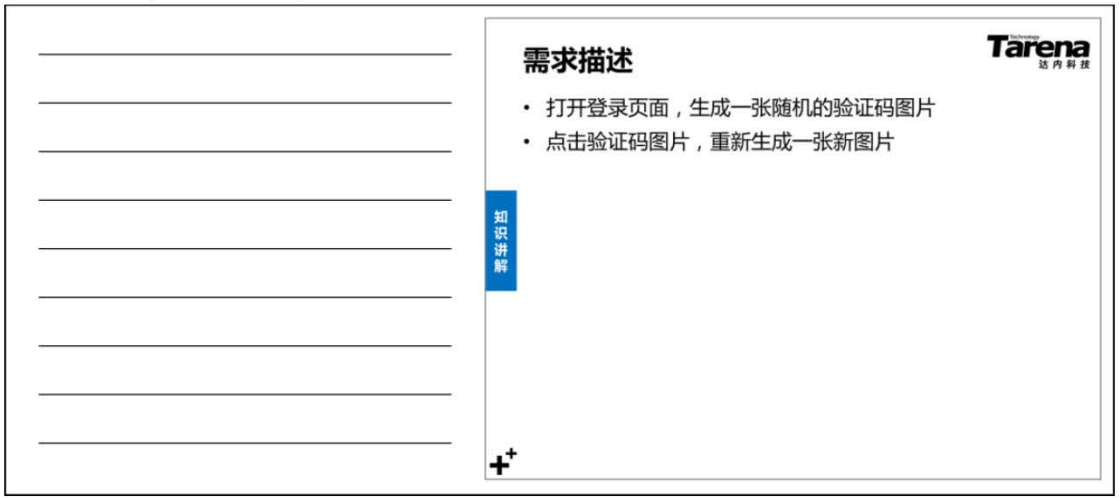


3.2.2. 【使用方式】使用步骤

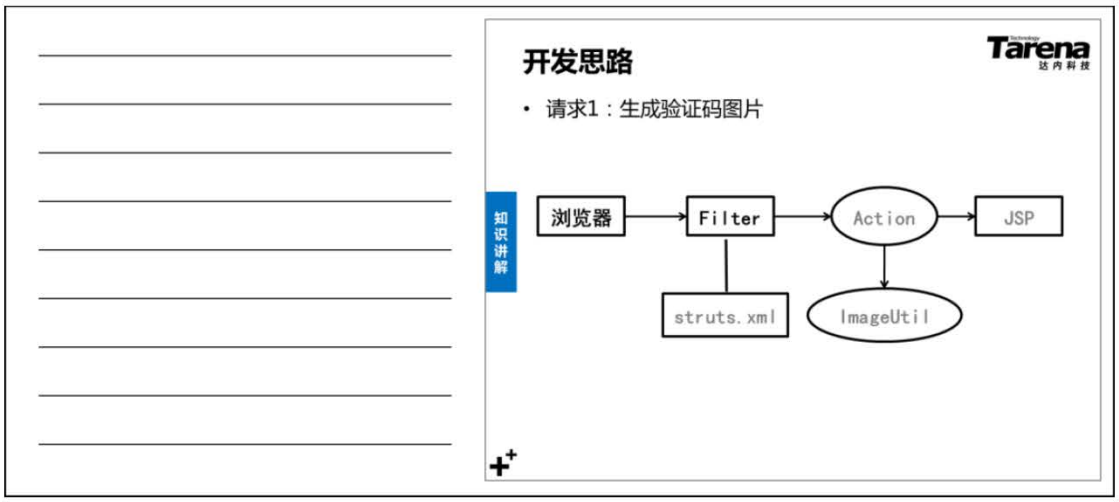


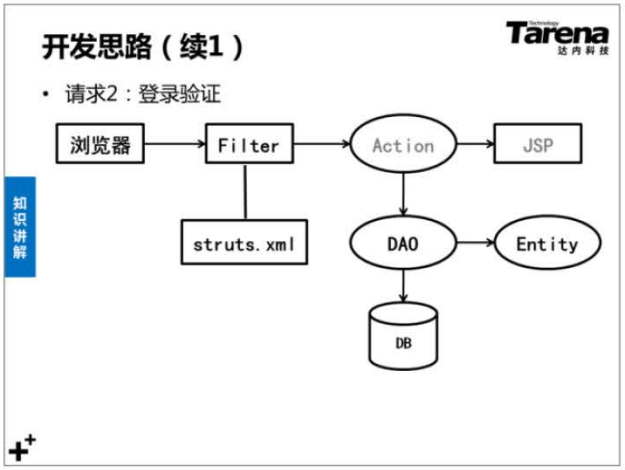
3.3. 登录验证码

3.3.1. 【登录验证码】需求描述



3.3.2. 【登录验证码】开发思路





3.3.3. 【登录验证码】开发步骤-请求 1

3.3.4. 【登录验证码】开发步骤-请求 2

4. redirectAction Result

4.1. 作用

4.1.1. 【作用】redirectAction Result 的作用

redirectAction Result的作用

- 用于将请求重定向给另外一个Action

解详识知

4.2. 使用方式

4.2.1. 【使用方式】语法

语法

```
<result name="login" type="redirectAction">
  <param name="namespace">
    /命名空间
  </param>
  <param name="actionName">
    action名
  </param>
</result>
```

解详识知

语法（续1）

- 如果重定向的Action与当前Action在同一个namespace下，则可以省略

  `<param name="namespace">/login</param>`

- 省略后代码如下

```
<result name="login" type="redirectAction">
    <param name="actionName">action名</param>
</result>
```

解详识知

语法（续2）

- 如果只需要给result的actionName注入值，配置可以进一步的简化，即

```
<result name="login" type="redirectAction">
    action名
</result>
```

解详识知

4.2.2. 【使用方式】使用步骤

使用步骤

- struts.xml中，配置Action
- 配置result
  - type="redirectAction "
  - <param name="namespace">/命名空间</param>
    <param name="actionName">action名</param>

解详识知

4.3. 资费删除

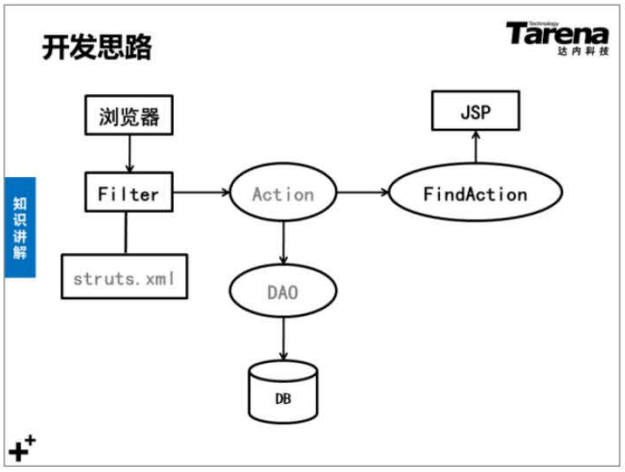
4.3.1. 【资费删除】需求描述

需求描述

- 点击删除按钮，删除当前行的资费数据
- 删除之后，刷新资费列表页面

解详识知

4.3.2. 【资费删除】开发思路



4.3.3. 【资费删除】开发步骤

经典案例

1. NetCTOSS 登录，请求 1

- **问题**

实现 NetCTOSS 项目的登录功能，要求在登录页面上输入账号、密码后，点击登录则自动校验登录信息是否正确，若校验成功则跳转至系统首页 index.jsp，否则跳转回登录页并给出错误的提示信息。

本案例中，只要求打开登录页面即可。

- **方案**

由于打开登录页面不需要任何业务，只是把请求转发给 JSP 即可，因此无需增加新的 Action，只需要在 struts.xml 中对请求加以配置即可。

- **步骤**

步骤一：struts.xml 中，配置访问登录页的请求 action

由于登录功能与资费功能无关，是一个新的模块，因此在 struts.xml 中追加一个新的 package 用于封装登录模块的配置信息，该 package 以模块名 login 命名。在该 package 下，配置当前请求对应的 action，此 action 由于没有业务逻辑，因此不必指定 class 属性，Struts2 会自动实例化默认的 Action 类 ActionSupport，该类中存在方法 execute，该方法返回 success。另外此 action 也不需要配置 method，当不配置此属性时，Struts2 会默认调用 execute 方法。代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="struts-default">
     <!—此处略去其他 Action 的配置... -->
 </package>


 <!-- 登录模块 -->
 <package name="login" namespace="/login" extends="struts-default">
     <!--
         打开登录页面：
```

```
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
               该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
               会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>

    </package>
   </struts>
```

步骤二：增加登录页面

在 WEB-INF/main/下创建登录页面 login.jsp，先在该页面上指定其编码类型为 utf-8，然后将静态页面中的代码复制到此页面中。再修改页面上引用的样式文件以及图片的路径，最后将页面默认的一些错误提示信息删除。代码如下：

```
   <%@page pageEncoding="utf-8"%>
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内 - NetCTOSS</title>
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global_color.css" />
      </head>
      <body class="index">
         <div class="login box">
            <table>
               <tr>
                  <td class="login_info">账号：</td>
                  <td colspan="2"><input name="" type="text" class="width150"
/></td>
                  <td class="login_error_info"><span class="required">30 长度
的字母、数字和下划线</span></td>
               </tr>
               <tr>
                  <td class="login_info">密码：</td>
                  <td     colspan="2"><input     name=""     type="password"
class="width150" /></td>
                  <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
               </tr>
               <tr>
                  <td class="login_info">验证码：</td>
                  <td     class="width70"><input     name=""     type="text"
class="width70" /></td>
                  <td><img src="../images/valicode.jpg" alt="验证码" title="
点击更换" /></td>
```

```
                    <td><span class="required"></span></td>
                </tr>
                <tr>
                    <td></td>
                    <td class="login_button" colspan="2">
                        <a                                  href="index.html"><img
src="../images/login_btn.png" /></a>
                    </td>
                    <td><span class="required"></span></td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

步骤三：测试

重新部署项目 NETCTOSS 并启动 tomcat，然后访问 NETCTOSS，路径为 http://localhost:8088/NETCTOSS/login/toLogin，效果如下：

图-1

- **完整代码**

本案例的 struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
```

```
        并且以模块名来命名package的name和namespace。
     -->
    <package name="cost" namespace="/cost" extends="struts-default">
        <!-- 查询资费数据 -->
        <action name="findCost" class="com.netctoss.action.FindCostAction">
            <!--
                正常情况下跳转到资费列表页面。
                一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
             -->
            <result name="success">
                /WEB-INF/cost/find cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在main下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action的class属性可以省略，省略时Struts2
               会自动实例化默认的Action类ActionSupport，
               该类中有默认业务方法execute，返回success。
            2、action的method属性可以省略，省略时Struts2
               会自动调用execute方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
    </package>

   </struts>
```

login.jsp 完整代码：

```
   <%@page pageEncoding="utf-8"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
         <link type="text/css" rel="stylesheet" media="all"
href="../styles/global color.css" />
      </head>
      <body class="index">
         <div class="login_box">
            <table>
               <tr>
                  <td class="login_info">账号：</td>
                  <td colspan="2"><input name="" type="text" class="width150"
/></td>
```

```
                    <td class="login_error_info"><span class="required">30 长度的字母、数字和下划线</span></td>
                </tr>
                <tr>
                    <td class="login_info">密码：</td>
                    <td colspan="2"><input name="" type="password" class="width150" /></td>
                    <td><span class="required">30 长度的字母、数字和下划线</span></td>
                </tr>
                <tr>
                    <td class="login_info">验证码：</td>
                    <td class="width70"><input name="" type="text" class="width70" /></td>
                    <td><img src="../images/valicode.jpg" alt="验证码" title="点击更换" /></td>
                    <td><span class="required"></span></td>
                </tr>
                <tr>
                    <td></td>
                    <td class="login_button" colspan="2">
                        <a href="index.html"><img src="../images/login_btn.png" /></a>
                    </td>
                    <td><span class="required"></span></td>
                </tr>
            </table>
        </div>
    </body>
</html>
```

2. NetCTOSS 登录，请求 2

- **问题**

完成 NetCTOSS 登录验证。

- **方案**

登录页面上，输入完账号、密码后，点击登录按钮时触发登录验证，该请求的过程如下图：

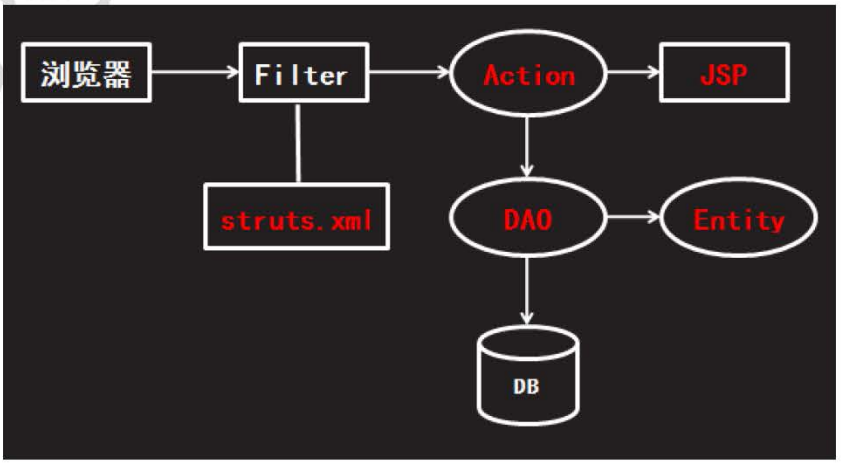


图-2

图中红色字体为需要开发的代码，其中最重要的是要在 Action 中写校验逻辑，具体校验逻辑如下图：

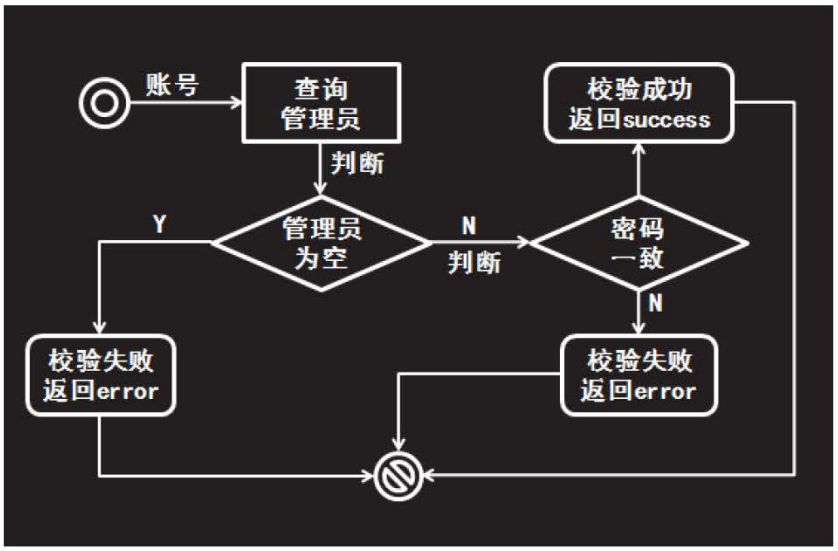


图-3

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建管理员实体类

在 com.netctoss.entity 包下，创建管理员实体类 Admin，用于封装管理员信息表 admin_info 中的数据，代码如下：

```
package com.netctoss.entity;

import java.sql.Date;

/**
 * 管理员实体类
 */
public class Admin {

 private Integer id;// 主键
 private String adminCode;// 账号
 private String password;// 密码
 private String name;// 姓名
 private String telephone;// 电话
 private String email;// 邮件
 private Date enrollDate;// 创建日期

 public Integer getId() {
     return id;
 }

 public void setId(Integer id) {
     this.id = id;
 }
```

```
    public String getAdminCode() {
        return adminCode;
    }

    public void setAdminCode(String adminCode) {
        this.adminCode = adminCode;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getTelephone() {
        return telephone;
    }

    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Date getEnrollDate() {
        return enrollDate;
    }

    public void setEnrollDate(Date enrollDate) {
        this.enrollDate = enrollDate;
    }

}
```

步骤二：创建登录模块 DAO，追加根据账号查询管理员的方法

在 com.netctoss.dao 包下，创建登录 DAO 的接口 ILoginDao，并增加根据账号查询管理员的方法，代码如下：

```
package com.netctoss.dao;

import com.netctoss.entity.Admin;

/**
 * 登录模块 DAO
 */
public interface ILoginDao {
```

```
    /**
     * 根据管理员账号查询管理员
     * @param code 账号
     * @return
     */
    Admin findByCode(String code);

    }
```

在 com.netctoss.dao 包下，创建 ILoginDao 接口的实现类 LoginDaoImpl，并实现根据账号查询管理员的方法，代码如下：

```
package com.netctoss.dao;

import com.netctoss.entity.Admin;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class LoginDaoImpl implements ILoginDao {

 @Override
 public Admin findByCode(String code) {
     // 模拟根据账号查询管理员信息
     Admin a = new Admin();
     a.setId(1);
     a.setAdminCode("tarena");
     a.setPassword("123");
     a.setName("达内");
     a.setTelephone("110");
     a.setEmail("tarena@tarena.com.cn");
     return a;
 }
}
```

为了方便后续Action 访问此DAO 将该DAO的实例化封装在DAOFactory 中代码如下：

```
package com.netctoss.dao;

/**
 * DAO 工厂，负责统一的创建 DAO 示例。
 */
public class DAOFactory {

 /**
  * 资费 DAO 实例
  */
 private static ICostDao costDao = new CostDaoImpl();


 /**
  * 登录 DAO 实例
  */
 private static ILoginDao loginDao = new LoginDaoImpl();
```

```
    /**
     * 返回资费 DAO 实例
     */
    public static ICostDao getCostDAO() {
        return costDao;
    }

    /**
     * 返回登录 DAO 实例
     */
    public static ILoginDao getLoginDAO() {
        return loginDao;
    }

}
```

步骤四：创建登录校验 Action

在 com.netctoss.action 包下，创建 Action 的父类 BaseAction，用于封装 Action 公用的方法，当前要将获取 session 的方法封装在该类中，以便于复用。代码如下：

```
package com.netctoss.action;

import java.util.Map;

import org.apache.struts2.interceptor.SessionAware;

/**
 * Action 基类，用于封装 Action 通用的方法。
 */
public class BaseAction implements SessionAware {

 protected Map<String, Object> session;

 /*
  * 采用接口注入的方式统一获取 session
  */
 @Override
 public void setSession(Map<String, Object> arg0) {
     session = arg0;
 }
}
```

在 com.netctoss.action 包下，创建登录校验 Action 类 LoginAction，继承于 BaseAction。在该 Action 的业务方法中，根据传入的账号及密码，调用登录 DAO 校验用户登录是否成功，成功时将登录信息记录到 session 中，并跳转至系统首页，否则跳转回登录页，并给予错误提示。代码如下：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ILoginDao;
import com.netctoss.entity.Admin;

/**
 *   登录校验 Action
 */
```

```
public class LoginAction extends BaseAction {

    // input
    private String adminCode;// 账号
    private String password;// 密码

    // output
    private String errorMsg;// 错误信息

    public String execute() {
        ILoginDao dao = DAOFactory.getLoginDAO();
        Admin admin = null;
        try {
            // 根据账号查询管理员
            admin = dao.findByCode(adminCode);
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }

        if (admin == null) {
            // 如果管理员为空，则说明账号有误，校验失败
            errorMsg = "账号不存在.";
            return "fail";
        } else {
            // 如果管理员不为空，进一步校验密码
            if (password != null
                    && password.equals(admin.getPassword())) {
                // 如果密码一致，校验成功
                session.put("admin", admin);
                return "success";
            } else {
                // 如果密码不一致，校验失败
                errorMsg = "密码有误.";
                return "fail";
            }
        }
    }

    public String getAdminCode() {
        return adminCode;
    }

    public void setAdminCode(String adminCode) {
        this.adminCode = adminCode;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getErrorMsg() {
        return errorMsg;
    }

    public void setErrorMsg(String errorMsg) {
        this.errorMsg = errorMsg;
    }
}
```

步骤五：在 struts.xml 中配置校验 Action

在 struts.xml 中配置登录校验 action，登录成功则转至系统首页，否则转至登录页，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="struts-default">
     <!-- 查询资费数据 -->
     <action name="findCost" class="com.netctoss.action.FindCostAction">
         <!—此处略去其他 Action 的配置... -->
     </action>
 </package>

 <!-- 登录模块 -->
 <package name="login" namespace="/login" extends="struts-default">
     <!—此处略去其他 Action 的配置... -->


     <!-- 登录校验 -->
     <action name="login" class="com.netctoss.action.LoginAction">
         <!-- 校验成功，跳转到系统首页 -->
         <result name="success">
             /WEB-INF/main/index.jsp
         </result>
         <!-- 登录失败，跳转回登录页面 -->
         <result name="fail">
             /WEB-INF/main/login.jsp
         </result>
         <!-- 报错，跳转到错误页面 -->
         <result name="error">
             /WEB-INF/main/error.jsp
         </result>
     </action>


 </package>
</struts>
```

步骤六：创建系统首页

在 WEB-INF/main 下创建系统首页 index.jsp，首先设置其编码为 utf-8，然后从静态页面中将 HTML 代码复制到此页面上，最后修改样式引用的路径。代码如下：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
    </head>
    <body class="index">
        <!--导航区域开始-->
        <div id="index_navi">
            <ul id="menu">
                <li><a href="index.html" class="index on"></a></li>
                <li><a href="role/role list.html" class="role off"></a></li>
                <li><a                        href="admin/admin list.html"
class="admin_off"></a></li>
                <li><a href="fee/fee_list.html" class="fee_off"></a></li>
                <li><a                     href="account/account_list.html"
class="account off"></a></li>
                <li><a                     href="service/service list.html"
class="service_off"></a></li>
                <li><a href="bill/bill_list.html" class="bill_off"></a></li>
                <li><a                       href="report/report_list.html"
class="report_off"></a></li>
                <li><a                           href="user/user_info.html"
class="information_off"></a></li>
                <li><a                       href="user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
    </body>
</html>
```

步骤七：在登录页面上设置表单项

在 login.jsp 中，追加表单设置，将账号及密码放在表单内，并且表单提交路径设置为登录 action。然后对账号、密码文本框的 name 进行设置，使用基本属性注入的方式将其值注入给登录 Action,并给登录按钮图片设置提交事件。最后使用 Struts2 显示标签+OGNL 表达式，在页面上输出错误信息。代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global color.css" />
    </head>
    <body class="index">
        <div class="login_box">


          <form action="login" method="post">


            <table>
                <tr>
                    <td class="login_info">账号：</td>
```

```
                    <td    colspan="2"><input    name="adminCode"    type="text"
class="width150" /></td>

                    <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
                </tr>
                <tr>
                    <td class="login_info">密码：</td>

                    <td   colspan="2"><input   name="password"   type="password"
class="width150" /></td>

                    <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
                </tr>
                <tr>
                    <td class="login_info">验证码：</td>
                    <td    class="width70"><input    name=""    type="text"
class="width70" /></td>
                    <td><img src="../images/valicode.jpg" alt="验证码" title="
点击更换" /></td>
                    <td><span class="required"></span></td>
                </tr>
                <tr>
                    <td></td>
                    <td class="login_button" colspan="2">

                        <a     href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>

                    </td>

                    <td><span                        class="required"><s:property
value="errorMsg"/></span></td>

                </tr>
            </table>
            </form>
        </div>
    </body>
</html>
```

步骤八：测试

部署项目 NETCTOSS 并重新启动 tomcat，访问 NETCTOSS 登录页，在页面上输入正确的账号及密码，点击登录后，效果如下图：

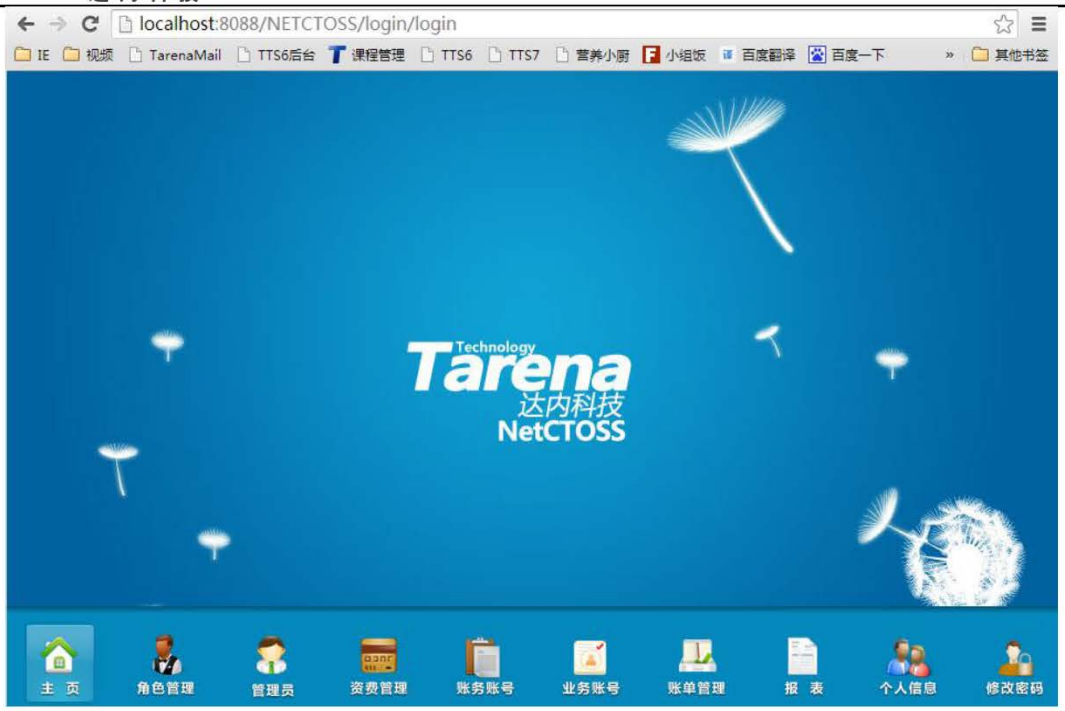


图-4

后退，在页面上输入错误的账号或密码，点击登录后，效果如下图：

图-5

从上图可以看出，校验的结果满足要求，但是由于页面的刷新，导致文本框中原先输入的内容丢失了，应该予以保留才对。实际上只需要在文本框中追加 value 属性，并将 Action 中的数据设置给它既可，因此需要对 login.jsp 进行完善。

步骤九：完善

在 login.jsp 上，给文本框设置默认值，以便登录失败回到登录页时，无法保留住刚刚输入的账号及密码，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
    </head>
    <body class="index">
        <div class="login box">
          <form action="login" method="post">
            <table>
                <tr>
                    <td class="login_info">账号：</td>
                    <td colspan="2">

                        <input name="adminCode" type="text" class="width150"
                            value="<s:property value="adminCode"/>"/>

                    </td>
                    <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
                </tr>
                <tr>
                    <td class="login_info">密码：</td>
                    <td colspan="2">

                        <input name="password" type="password" class="width150"
                            value="<s:property value="password"/>"/>

                    </td>
                    <td><span class="required">30 长度的字母、数字和下划线
</span></td>
                </tr>
                <tr>
                    <td class="login_info">验证码：</td>
                    <td class="width70"><input name="" type="text"
class="width70" /></td>
                    <td><img src="../images/valicode.jpg" alt="验证码" title="
点击更换" /></td>
                    <td><span class="required"></span></td>
                </tr>
                <tr>
                    <td></td>
                    <td class="login_button" colspan="2">
                        <a href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>
                    </td>
```

```
                    <td><span                    class="required"><s:property
value="errorMsg"/></span></td>
                </tr>
            </table>
            </form>
        </div>
    </body>
</html>
```

完善后，刷新地址栏，重新输入错误的账号或密码后，点击登录，效果如下图：

图-6

注：实际上 Struts2 有更好的办法实现上述显示默认值（步骤九）的逻辑，即使用 Struts2 的 UI 标签。但目前还没有讲到这些内容，因此暂且这样写，以后可以进行重构。希望大家理解这一点，实际项目中不要写出这样设置默认值的代码。

- **完整代码**

本案例的实体类 Admin 完整代码：

```
package com.netctoss.entity;

import java.sql.Date;

/**
 * 管理员实体类
 */
public class Admin {

 private Integer id;// 主键
 private String adminCode;// 账号
 private String password;// 密码
 private String name;// 姓名
```

```
    private String telephone;// 电话
    private String email;// 邮件
    private Date enrollDate;// 创建日期

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getAdminCode() {
        return adminCode;
    }

    public void setAdminCode(String adminCode) {
        this.adminCode = adminCode;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getTelephone() {
        return telephone;
    }

    public void setTelephone(String telephone) {
        this.telephone = telephone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public Date getEnrollDate() {
        return enrollDate;
    }

    public void setEnrollDate(Date enrollDate) {
        this.enrollDate = enrollDate;
    }
}
```

ILoginDao 完整代码：

```
package com.netctoss.dao;
```

```
import com.netctoss.entity.Admin;

/**
 * 登录模块 DAO
 */
public interface ILoginDao {

 /**
  * 根据管理员账号查询管理员
  * @param code 账号
  * @return
  */
 Admin findByCode(String code);

}
```

LoginDaoImpl 完整代码：

```
package com.netctoss.dao;

import com.netctoss.entity.Admin;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class LoginDaoImpl implements ILoginDao {

 @Override
 public Admin findByCode(String code) {
     // 模拟根据账号查询管理员信息
     Admin a = new Admin();
     a.setId(1);
     a.setAdminCode("tarena");
     a.setPassword("123");
     a.setName("达内");
     a.setTelephone("110");
     a.setEmail("tarena@tarena.com.cn");
     return a;
 }
}
```

DAOFactory 完整代码：

```
package com.netctoss.dao;

/**
 * DAO 工厂，负责统一的创建 DAO 示例。
 */
public class DAOFactory {

 /**
  * 资费 DAO 实例
  */
 private static ICostDao costDao = new CostDaoImpl();

 /**
  * 登录 DAO 实例
  */
 private static ILoginDao loginDao = new LoginDaoImpl();

 /**
```

```
 * 返回资费 DAO 实例
 */
public static ICostDao getCostDAO() {
    return costDao;
}

/**
 * 返回登录 DAO 实例
 */
public static ILoginDao getLoginDAO() {
    return loginDao;
}

}
```

BaseAction 完整代码：

```
package com.netctoss.action;

import java.util.Map;

import org.apache.struts2.interceptor.SessionAware;

/**
 * Action 基类，用于封装 Action 通用的方法。
 */
public class BaseAction implements SessionAware {

protected Map<String, Object> session;

/*
 * 采用接口注入的方式统一获取 session
 */
@Override
public void setSession(Map<String, Object> arg0) {
    session = arg0;
}
}
```

LoginAction 完整代码：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ILoginDao;
import com.netctoss.entity.Admin;

/**
 *   登录校验 Action
 */
public class LoginAction extends BaseAction {

// input
private String adminCode;// 账号
private String password;// 密码

// output
private String errorMsg;// 错误信息

public String execute() {
    ILoginDao dao = DAOFactory.getLoginDAO();
    Admin admin = null;
    try {
```

```
            // 根据账号查询管理员
            admin = dao.findByCode(adminCode);
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }

        if (admin == null) {
            // 如果管理员为空，则说明账号有误，校验失败
            errorMsg = "账号不存在.";
            return "fail";
        } else {
            // 如果管理员不为空，进一步校验密码
            if (password != null
                    && password.equals(admin.getPassword())) {
                // 如果密码一致，校验成功
                session.put("admin", admin);
                return "success";
            } else {
                // 如果密码不一致，校验失败
                errorMsg = "密码有误.";
                return "fail";
            }
        }
    }

    public String getAdminCode() {
        return adminCode;
    }

    public void setAdminCode(String adminCode) {
        this.adminCode = adminCode;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getErrorMsg() {
        return errorMsg;
    }

    public void setErrorMsg(String errorMsg) {
        this.errorMsg = errorMsg;
    }
}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
```

```
    <package name="cost" namespace="/cost" extends="struts-default">
        <!-- 查询资费数据 -->
        <action name="findCost" class="com.netctoss.action.FindCostAction">
            <!--
                正常情况下跳转到资费列表页面。
                一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
             -->
            <result name="success">
                /WEB-INF/cost/find cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
               该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
               会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    </struts>
```

index.jsp 完整代码：

```
    <%@page pageEncoding="utf-8"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
        <title>达内－NetCTOSS</title>
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
      </head>
      <body class="index">
        <!--导航区域开始-->
        <div id="index navi">
          <ul id="menu">
            <li><a href="index.html" class="index_on"></a></li>
            <li><a href="role/role_list.html" class="role_off"></a></li>
            <li><a                        href="admin/admin_list.html"
class="admin_off"></a></li>
            <li><a href="fee/fee list.html" class="fee off"></a></li>
            <li><a                      href="account/account list.html"
class="account_off"></a></li>
            <li><a                      href="service/service_list.html"
class="service_off"></a></li>
            <li><a href="bill/bill list.html" class="bill off"></a></li>
            <li><a                        href="report/report list.html"
class="report off"></a></li>
            <li><a                            href="user/user info.html"
class="information off"></a></li>
            <li><a                        href="user/user_modi_pwd.html"
class="password_off"></a></li>
          </ul>
        </div>
      </body>
    </html>
```

login.jsp 完整代码：

```
    <%@page pageEncoding="utf-8"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内－NetCTOSS</title>
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
      </head>
      <body class="index">
        <div class="login_box">
          <form action="login" method="post">
            <table>
              <tr>
                <td class="login_info">账号: </td>
                <td colspan="2">
                  <input name="adminCode" type="text" class="width150"
                      value="<s:property value="adminCode"/>"/>
                </td>
                <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
              </tr>
              <tr>
                <td class="login_info">密码: </td>
                <td colspan="2">
                  <input name="password" type="password" class="width150"
                      value="<s:property value="password"/>"/>
                </td>
```

```
                    <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
                </tr>
                <tr>
                    <td class="login_info">验证码: </td>
                    <td     class="width70"><input     name=""     type="text"
class="width70" /></td>
                    <td><img src="../images/valicode.jpg" alt="验证码" title="
点击更换" /></td>
                    <td><span class="required"></span></td>
                </tr>
                <tr>
                    <td></td>
                    <td class="login_button" colspan="2">
                        <a href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>
                    </td>
                    <td><span                    class="required"><s:property
value="errorMsg"/></span></td>
                </tr>
            </table>
            </form>
        </div>
    </body>
  </html>
```

3. 登录验证码-1

- **问题**

NetCTOSS 登录的案例中，提交时只考虑了账号、密码，没有考虑验证码，那么本案例中，要求给登录功能追加生成验证码的功能，以及在登录时要对验证码进行校验。

本案例中，完成在登录页面上生成验证码图片即可。

- **方案**

我们可以在服务端动态生成验证码图片，然后使用 stream 类型的 result 将图片输出给页面 img 元素。

在生成图片后，需要将图片中的文本记录到 session，以便后续提交登录时，方便对验证码的校验。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建生成图片工具类

在 com.netctoss.util 包下，创建生成图片工具类 ImageUtil，并提供创建图片的代码，以及将图片转换成输入流的代码，代码如下：

```
package com.netctoss.util;

import java.awt.Color;
import java.awt.Font;
```

```
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;

/**
 *  生成验证码图片工具类
 */
public final class ImageUtil {

 private static final char[] chars = { '0', '1', '2', '3', '4', '5', '6',
        '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I' };
 private static final int SIZE = 4;
 private static final int LINES = 5;
 private static final int WIDTH = 80;
 private static final int HEIGHT = 40;
 private static final int FONT SIZE = 30;

 /**
  * 生成验证码图片，封装与 Map 中。
  * 其中 Map 的 key 是验证码，Map 的 value 是验证码图片。
  */
 public static Map<String, BufferedImage> createImage() {
     StringBuffer sb = new StringBuffer();
     BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
             BufferedImage.TYPE_INT_RGB);
     Graphics graphic = image.getGraphics();
     graphic.setColor(Color.LIGHT GRAY);
     graphic.fillRect(0, 0, WIDTH, HEIGHT);
     Random ran = new Random();
     // 画随机字符
     for (int i = 1; i <= SIZE; i++) {
         int r = ran.nextInt(chars.length);
         graphic.setColor(getRandomColor());
         graphic.setFont(new Font(null, Font.BOLD + Font.ITALIC, FONT SIZE));
         graphic.drawString(chars[r] + "", (i - 1) * WIDTH / SIZE,
                 HEIGHT / 2);
         sb.append(chars[r]);// 将字符保存，存入 Session
     }
     // 画干扰线
     for (int i = 1; i <= LINES; i++) {
         graphic.setColor(getRandomColor());
         graphic.drawLine(ran.nextInt(WIDTH), ran.nextInt(HEIGHT),
                 ran.nextInt(WIDTH), ran.nextInt(HEIGHT));
     }
     Map<String, BufferedImage> map = new HashMap<String, BufferedImage>();
     map.put(sb.toString(), image);
     return map;
 }

 /**
  * 将图片转换为输入流
  */
 public static InputStream getInputStream(BufferedImage image)
         throws IOException {
     ByteArrayOutputStream bos = new ByteArrayOutputStream();
```

```
        JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(bos);
        encoder.encode(image);
        byte[] imageBts = bos.toByteArray();
        InputStream in = new ByteArrayInputStream(imageBts);
        return in;
    }

    private static Color getRandomColor() {
        Random ran = new Random();
        Color color = new Color(ran.nextInt(256), ran.nextInt(256),
                ran.nextInt(256));
        return color;
    }
}
```

步骤二：在 Action 中生成图片

在 com.netctoss.action 包下，创建 CreateImageAction 类，用于处理生成图片的请求，代码如下：

```
package com.netctoss.action;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.util.Map;

import com.netctoss.util.ImageUtil;

/**
 *   生成验证码图片 Action
 */
public class CreateImageAction extends BaseAction {

    /**
     * 验证码图片输入流
     */
    private InputStream imageStream;

    public String execute() {
        // 创建验证码图片
        Map<String, BufferedImage> imageMap = ImageUtil.createImage();
        // 取出验证码，放入 session
        String code = imageMap.keySet().iterator().next();
        session.put("imageCode", code);

        // 取出图片
        BufferedImage image = imageMap.get(code);
        try {
            // 将图片转变为输入流
            imageStream = ImageUtil.getInputStream(image);
        } catch (IOException e) {
            e.printStackTrace();
            return "error";
        }

        return "success";
    }

    public InputStream getImageStream() {
        return imageStream;
    }
```

```
    public void setImageStream(InputStream imageStream) {
        this.imageStream = imageStream;
    }
}
```

步骤三：在 struts.xml 中配置 action

在 struts.xml 中对此 action 进行配置，利用 stream 类型的 result，将 CreateImageAction 中的输入流的内容输出给页面，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

<!--
    资费模块配置信息：
    一般情况下，一个模块的配置单独封装在一个 package 下，
    并且以模块名来命名 package 的 name 和 namespace。
 -->
<package name="cost" namespace="/cost" extends="struts-default">
    <!—此处略去其他 Action 的配置... -->
</package>

<!-- 登录模块 -->
<package name="login" namespace="/login" extends="struts-default">
    <!—此处略去其他 Action 的配置... -->


    <!-- 生成验证码 -->
    <action name="createImage" class="com.netctoss.action.CreateImageAction">
        <!-- 使用 stream 类型的 result -->
        <result name="success" type="stream">
            <!-- 指定输出的内容 -->
            <param name="inputName">imageStream</param>
        </result>
    </action>


</package>
</struts>
```

步骤四：登录页面上，修改图片路径

在登录页面 login.jsp 上，将 img 的 src 属性指定为生成验证码图片的 action，即 createImage。并且为了每次点击该图片时都刷新为新验证码，给该 img 的单击事件绑定函数，在函数中刷新 img 的 src 属性，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
```

```
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
        <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
        <script type="text/javascript" language="javascript">
          //刷新验证码图片
          function change(image){
              //改变 img 的 src 即可，由于该 URL 并没有变化，因此追加动态参数伪装成变化的
URL。
              image.src = "createImage?date=" + new Date().getTime();
          }
       </script>
      </head>
      <body class="index">
        <div class="login box">
          <form action="login" method="post">
            <table>
              <tr>
                <td class="login_info">账号：</td>
                <td colspan="2">
                  <input name="adminCode" type="text" class="width150"
                      value="<s:property value="adminCode"/>"/>
                </td>
                <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
              </tr>
              <tr>
                <td class="login_info">密码：</td>
                <td colspan="2">
                  <input name="password" type="password" class="width150"
                      value="<s:property value="password"/>"/>
                </td>
                <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
              </tr>
              <tr>
                <td class="login_info">验证码：</td>
                <td    class="width70"><input    name=""    type="text"
class="width70" /></td>


                <td><img  src="createImage"  alt="验证码"  title="点击更换"
onclick="change(this);"/></td>


                <td><span class="required"></span></td>
              </tr>
              <tr>
                <td></td>
                <td class="login_button" colspan="2">
                    <a href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>
                </td>
                <td><span                  class="required"><s:property
value="errorMsg"/></span></td>
              </tr>
            </table>
           </form>
        </div>
      </body>
   </html>
```

步骤五：测试

重新部署项目 NETCTOSS 并重启 tomcat，访问登录页面，效果如下图：

图-7

- **完整代码**

本案例的 ImageUtil 完整代码：

```
package com.netctoss.util;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;

import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;

/**
 *   生成验证码图片工具类
 */
public final class ImageUtil {
 private static final char[] chars = { '0', '1', '2', '3', '4', '5', '6',
        '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I' };
 private static final int SIZE = 4;
 private static final int LINES = 5;
 private static final int WIDTH = 80;
 private static final int HEIGHT = 40;
```

```
    private static final int FONT SIZE = 30;

    /**
     * 生成验证码图片，封装与Map中。
     * 其中Map的key是验证码，Map的value是验证码图片。
     */
    public static Map<String, BufferedImage> createImage() {
        StringBuffer sb = new StringBuffer();
        BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
                BufferedImage.TYPE INT RGB);
        Graphics graphic = image.getGraphics();
        graphic.setColor(Color.LIGHT_GRAY);
        graphic.fillRect(0, 0, WIDTH, HEIGHT);
        Random ran = new Random();
        // 画随机字符
        for (int i = 1; i <= SIZE; i++) {
            int r = ran.nextInt(chars.length);
            graphic.setColor(getRandomColor());
            graphic.setFont(new Font(null, Font.BOLD + Font.ITALIC, FONT_SIZE));
            graphic.drawString(chars[r] + "", (i - 1) * WIDTH / SIZE,
                    HEIGHT / 2);
            sb.append(chars[r]);// 将字符保存，存入Session
        }
        // 画干扰线
        for (int i = 1; i <= LINES; i++) {
            graphic.setColor(getRandomColor());
            graphic.drawLine(ran.nextInt(WIDTH), ran.nextInt(HEIGHT),
                    ran.nextInt(WIDTH), ran.nextInt(HEIGHT));
        }
        Map<String, BufferedImage> map = new HashMap<String, BufferedImage>();
        map.put(sb.toString(), image);
        return map;
    }

    /**
     * 将图片转换为输入流
     */
    public static InputStream getInputStream(BufferedImage image)
            throws IOException {
        ByteArrayOutputStream bos = new ByteArrayOutputStream();
        JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(bos);
        encoder.encode(image);
        byte[] imageBts = bos.toByteArray();
        InputStream in = new ByteArrayInputStream(imageBts);
        return in;
    }

    private static Color getRandomColor() {
        Random ran = new Random();
        Color color = new Color(ran.nextInt(256), ran.nextInt(256),
                ran.nextInt(256));
        return color;
    }
}
```

CreateImageAction 完整代码：

```
package com.netctoss.action;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.util.Map;

import com.netctoss.util.ImageUtil;
```

```
/**
 *   生成验证码图片 Action
 */
public class CreateImageAction extends BaseAction {

 /**
  * 验证码图片输入流
  */
 private InputStream imageStream;

 public String execute() {
     // 创建验证码图片
     Map<String, BufferedImage> imageMap = ImageUtil.createImage();
     // 取出验证码，放入 session
     String code = imageMap.keySet().iterator().next();
     session.put("imageCode", code);

     // 取出图片
     BufferedImage image = imageMap.get(code);
     try {
         // 将图片转变为输入流
         imageStream = ImageUtil.getInputStream(image);
     } catch (IOException e) {
         e.printStackTrace();
         return "error";
     }

     return "success";
 }

 public InputStream getImageStream() {
     return imageStream;
 }

 public void setImageStream(InputStream imageStream) {
     this.imageStream = imageStream;
 }
}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="struts-default">
     <!-- 查询资费数据 -->
     <action name="findCost" class="com.netctoss.action.FindCostAction">
         <!--
             正常情况下跳转到资费列表页面。
             一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
          -->
         <result name="success">
             /WEB-INF/cost/find cost.jsp
         </result>
         <!--
             错误情况下，跳转到错误页面。
```

```
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
            -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
               该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
               会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 生成验证码 -->
        <action                                                name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!-- 使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!-- 指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>
   </struts>
```

login.jsp 完整代码：

```
    <%@page pageEncoding="utf-8"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>达内－NetCTOSS</title>
            <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
```

```
          <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global color.css" />
          <script type="text/javascript" language="javascript">
            //刷新验证码图片
            function change(image){
                //改变 img 的 src 即可，由于该 URL 并没有变化，因此追加动态参数伪装成变化的
URL。
                image.src = "createImage?date=" + new Date().getTime();
            }
        </script>
      </head>
      <body class="index">
          <div class="login_box">
            <form action="login" method="post">
              <table>
                  <tr>
                      <td class="login info">账号：</td>
                      <td colspan="2">
                         <input name="adminCode" type="text" class="width150"
                             value="<s:property value="adminCode"/>"/>
                      </td>
                      <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
                  </tr>
                  <tr>
                      <td class="login_info">密码：</td>
                      <td colspan="2">
                         <input name="password" type="password" class="width150"
                             value="<s:property value="password"/>"/>
                      </td>
                      <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
                  </tr>
                  <tr>
                      <td class="login info">验证码：</td>
                      <td     class="width70"><input     name=""     type="text"
class="width70" /></td>
                      <td><img src="createImage" alt="验证码" title="点击更换"
onclick="change(this);"/></td>
                      <td><span class="required"></span></td>
                  </tr>
                  <tr>
                      <td></td>
                      <td class="login_button" colspan="2">
                          <a href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>
                      </td>
                      <td><span                    class="required"><s:property
value="errorMsg"/></span></td>
                  </tr>
              </table>
             </form>
          </div>
      </body>
   </html>
```

4. 登录验证码-2

- **问题**

NetCTOSS 登录时，追加对验证码的校验。

- **方案**

点击登录按钮提交表单时，在 Action 中先对验证码进行校验。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：Action 中对验证码进行校验

在 LoginAction 中，追加输入属性验证码，并在业务方法中校验账号密码前对其进行校验，代码如下：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ILoginDao;
import com.netctoss.entity.Admin;

/**
 *   登录校验 Action
 */
public class LoginAction extends BaseAction {

 // input
 private String adminCode;// 账号
 private String password;// 密码

 private String verifyCode;// 验证码

 // output
 private String errorMsg;// 错误信息

 public String execute() {

     // 从 session 中取出生成的验证码
     String imageCode = (String) session.get("imageCode");
     // 验证用户输入的验证码是否与生成验证码一致
     if(imageCode == null || !imageCode.equalsIgnoreCase(verifyCode)) {
         //如果不一致，提示错误
         errorMsg = "验证码有误.";
         return "fail";
     }

     // 此处略去登录验证逻辑...
 }

 public String getAdminCode() {
     return adminCode;
 }

 public void setAdminCode(String adminCode) {
     this.adminCode = adminCode;
 }
```

```
    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getErrorMsg() {
        return errorMsg;
    }

    public void setErrorMsg(String errorMsg) {
        this.errorMsg = errorMsg;
    }


    public String getVerifyCode() {
        return verifyCode;
    }

    public void setVerifyCode(String verifyCode) {
        this.verifyCode = verifyCode;
    }

   }
```

步骤二：JSP 上设置表单项

在 login.jsp 上，设置验证码文本框的 name 属性值，使其与 LoginAction 中追加的属性名相同，代码如下：

```
   <%@page pageEncoding="utf-8"%>
   <%@taglib uri="/struts-tags" prefix="s"%>
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内 - NetCTOSS</title>
         <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
         <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
         <script type="text/javascript" language="javascript">
           //刷新验证码图片
           function change(image){
               //改变 img 的 src 即可，由于该 URL 并没有变化，因此追加动态参数伪装成变化的
URL。
               image.src = "createImage?date=" + new Date().getTime();
           }
        </script>
      </head>
      <body class="index">
         <div class="login_box">
           <form action="login" method="post">
             <table>
                <tr>
                   <td class="login_info">账号：</td>
                   <td colspan="2">
                      <input name="adminCode" type="text" class="width150"
```

```
                                value="<s:property value="adminCode"/>"/>
                        </td>
                        <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
                    </tr>
                    <tr>
                        <td class="login_info">密码：</td>
                        <td colspan="2">
                           <input name="password" type="password" class="width150"
                               value="<s:property value="password"/>"/>
                        </td>
                        <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
                    </tr>
                    <tr>
                        <td class="login_info">验证码：</td>


                        <td   class="width70"><input   name="verifyCode"   type="text"
class="width70" /></td>


                        <td><img  src="createImage"  alt="验证码"  title="点击更换"
onclick="change(this);"/></td>
                        <td><span class="required"></span></td>
                    </tr>
                    <tr>
                        <td></td>
                        <td class="login button" colspan="2">
                           <a  href="javascript:document.forms[0].submit();"><img
src="../images/login btn.png" /></a>
                        </td>
                        <td><span                    class="required"><s:property
value="errorMsg"/></span></td>
                    </tr>
                </table>
                </form>
            </div>
        </body>
    </html>
```

步骤三：测试

重新部署项目，然后访问登录页，输入错误的验证码后点击登录，效果如下图：

图-8

- **完整代码**

本案例的完整代码如下所示：

```
LoginAction 完整代码
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ILoginDao;
import com.netctoss.entity.Admin;

/**
 *   登录校验 Action
 */
public class LoginAction extends BaseAction {
 // input
 private String adminCode;// 账号
 private String password;// 密码
 private String verifyCode;// 验证码
 // output
 private String errorMsg;// 错误信息

 public String execute() {
     // 从 session 中取出生成的验证码
     String imageCode = (String) session.get("imageCode");
     // 验证用户输入的验证码是否与生成验证码一致
     if(imageCode == null || !imageCode.equalsIgnoreCase(verifyCode)) {
         //如果不一致，提示错误
         errorMsg = "验证码有误.";
         return "fail";
     }

     ILoginDao dao = DAOFactory.getLoginDAO();
     Admin admin = null;
```

```
        try {
            // 根据账号查询管理员
            admin = dao.findByCode(adminCode);
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }

        if (admin == null) {
            // 如果管理员为空，则说明账号有误，校验失败
            errorMsg = "账号不存在.";
            return "fail";
        } else {
            // 如果管理员不为空，进一步校验密码
            if (password != null
                    && password.equals(admin.getPassword())) {
                // 如果密码一致，校验成功
                session.put("admin", admin);
                return "success";
            } else {
                // 如果密码不一致，校验失败
                errorMsg = "密码有误.";
                return "fail";
            }
        }
    }

    public String getAdminCode() {
        return adminCode;
    }

    public void setAdminCode(String adminCode) {
        this.adminCode = adminCode;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getErrorMsg() {
        return errorMsg;
    }

    public void setErrorMsg(String errorMsg) {
        this.errorMsg = errorMsg;
    }

    public String getVerifyCode() {
        return verifyCode;
    }

    public void setVerifyCode(String verifyCode) {
        this.verifyCode = verifyCode;
    }

}
```

login.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
```

```
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
         <script type="text/javascript" language="javascript">
           //刷新验证码图片
           function change(image){
               //改变 img 的 src 即可，由于该 URL 并没有变化，因此追加动态参数伪装成变化的
URL。
               image.src = "createImage?date=" + new Date().getTime();
           }
        </script>
      </head>
      <body class="index">
         <div class="login_box">
           <form action="login" method="post">
             <table>
                <tr>
                   <td class="login_info">账号：</td>
                   <td colspan="2">
                      <input name="adminCode" type="text" class="width150"
                          value="<s:property value="adminCode"/>"/>
                   </td>
                   <td class="login_error_info"><span class="required">30 长
度的字母、数字和下划线</span></td>
                </tr>
                <tr>
                   <td class="login_info">密码：</td>
                   <td colspan="2">
                      <input name="password" type="password" class="width150"
                          value="<s:property value="password"/>"/>
                   </td>
                   <td><span  class="required">30 长度的字母、数字和下划线
</span></td>
                </tr>
                <tr>
                   <td class="login_info">验证码：</td>
                   <td class="width70"><input name="verifyCode" type="text"
class="width70" /></td>
                   <td><img src="createImage" alt="验证码" title="点击更换"
onclick="change(this);"/></td>
                   <td><span class="required"></span></td>
                </tr>
                <tr>
                   <td></td>
                   <td class="login_button" colspan="2">
                      <a href="javascript:document.forms[0].submit();"><img
src="../images/login_btn.png" /></a>
                   </td>
                   <td><span                    class="required"><s:property
value="errorMsg"/></span></td>
                </tr>
             </table>
            </form>
         </div>
      </body>
   </html>
```

5. 资费删除

- **问题**

增加资费的删除功能，用于删除一条资费数据。

- **方案**

点击删除按钮时，发起一次删除的请求，并传入资费 ID。本次请求的 Action 负责调用 DAO 将数据删除，然后再将请求重定向给查询 Action，从而实现列表页面的刷新。

- **步骤**

步骤一：DAO 中追加删除方法

在 ICostDao 中追加删除方法，参数为资费 ID，代码如下：

```
package com.netctoss.dao;

import java.util.List;
import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {

 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();


 /**
  * 删除一条资费数据
  * @param id 主键
  */
 void delete(int id);

}
```

在 CostDaoImpl 中，实现删除方法，代码如下：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;
import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
```

```
public class CostDaoImpl implements ICostDao {

 // 此处略去其他方法的实现...

 @Override
 public void delete(int id) {
     // 模拟根据 id 删除资费数据
     System.out.println("删除 ID 为[" + id + "]的资费数据.");
 }
```

步骤二：Action 中处理删除请求

在 com.netctoss.action 包下，创建删除资费 Action 类 DeleteCostAction，并增加输入属性 id，在业务方法中调用 DAO 的删除方法来删除该资费数据即可，代码如下：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;

/**
 * 删除资费 Action
 */
public class DeleteCostAction {

 // input
 private int id;

 public String execute() {
     ICostDao dao = DAOFactory.getCostDAO();
     try {
         // 删除资费
         dao.delete(id);
     } catch (Exception e) {
         e.printStackTrace();
         return "error";
     }
     return "success";
 }

 public int getId() {
     return id;
 }

 public void setId(int id) {
     this.id = id;
 }
}
```

步骤三：struts.xml 中配置删除 action

在 struts.xml 中，资费 package 下配置删除资费 action，删除成功时用 redirectAction 类型的 result 将请求重定向给资费查询 Action，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
```

```
    <!--
        资费模块配置信息：
        一般情况下，一个模块的配置单独封装在一个 package 下，
        并且以模块名来命名 package 的 name 和 namespace。
     -->
    <package name="cost" namespace="/cost" extends="struts-default">
        <!—此处略去其他 Action 的配置... -->


        <!-- 删除资费 -->
        <action name="deleteCost"
            class="com.netctoss.action.DeleteCostAction">
            <!-- 删除完之后，重定向到查询 action -->
            <result name="success" type="redirectAction">
                findCost
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>


    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!—此处略去其他 Action 的配置... -->
    </package>

   </struts>
```

步骤四：JSP 中发起删除请求

在 find_cost.jsp 中，给删除按钮的事件绑定函数，并传入 id 值，在该函数中请求资费删除 action，代码如下：

```
    <%@page pageEncoding="utf-8" isELIgnored="false"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
       <head>
          <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
          <title>达内 - NetCTOSS</title>
          <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
          <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global color.css" />
          <script language="javascript" type="text/javascript">
             //排序按钮的点击事件
             function sort(btnObj) {
                if (btnObj.className == "sort desc")
                   btnObj.className = "sort asc";
                else
                   btnObj.className = "sort_desc";
             }

             //启用
             function startFee() {
```

```
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                         href="../admin/admin_list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                    href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill off"></a></li>
                <li><a                      href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基费"  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value="时长"  class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input   type="button"   value="增 加 "   class="btn_add"
```

```
onclick="location.href='fee add.html';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                                src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>

                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
                            <td><a          href="fee_detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                                <input    type="button"    value="   启   用   "
class="btn start" onclick="startFee();" />
                                <input    type="button"    value="   修   改   "
class="btn_modify" onclick="location.href='fee_modi.html';" />

                                <input type="button" value="删除" class="btn_delete"
onclick="deleteFee(<s:property value="id"/>);" />

                            </td>
                         </tr>
                        </s:iterator>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
```

```
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                    <a href="#" class="current page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
</html>
```

步骤五：测试

重新部署项目并重启 tomcat，打开资费列表页面，点选任意资费数据的删除按钮，效果如下图：

图-9

点击确定后，该数据被删除，页面被刷新，效果如下图：

图-10

- **完整代码**

本案例的 ICostDao 完整代码：

```
package com.netctoss.dao;

import java.util.List;
import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {

 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();

 /**
  * 删除一条资费数据
  * @param id 主键
  */
 void delete(int id);

}
```

CostDaoImpl 完整代码:

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;
import com.netctoss.entity.Cost;
```

```
/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

 @Override
 public List<Cost> findAll() {
     // 模拟查询全部资费数据
     List<Cost> list = new ArrayList<Cost>();

     Cost c1 = new Cost();
     c1.setId(95);
     c1.setName("6 元套餐");
     c1.setBaseDuration(66);
     c1.setBaseCost(6.6);
     c1.setUnitCost(0.6);
     c1.setDescr("6 元套餐");
     c1.setStatus("0");
     c1.setCostType("2");
     list.add(c1);

     Cost c2 = new Cost();
     c2.setId(96);
     c2.setName("8 元套餐");
     c2.setBaseDuration(88);
     c2.setBaseCost(8.8);
     c2.setUnitCost(0.8);
     c2.setDescr("8 元套餐");
     c2.setStatus("0");
     c2.setCostType("2");
     list.add(c2);

     Cost c3 = new Cost();
     c3.setId(97);
     c3.setName("tarena");
     c3.setBaseDuration(99);
     c3.setBaseCost(9.9);
     c3.setUnitCost(0.9);
     c3.setDescr("tarena 套餐");
     c3.setStatus("0");
     c3.setCostType("2");
     list.add(c3);

     return list;
 }

 @Override
 public void delete(int id) {
     // 模拟根据 id 删除资费数据
     System.out.println("删除 ID 为[" + id + "]的资费数据.");
 }
```

DeleteCostAction 完整代码：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;

/**
 * 删除资费 Action
 */
public class DeleteCostAction {
```

```
    // input
    private int id;

    public String execute() {
        ICostDao dao = DAOFactory.getCostDAO();
        try {
            // 删除资费
            dao.delete(id);
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }
        return "success";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="struts-default">
     <!-- 查询资费数据 -->
     <action name="findCost" class="com.netctoss.action.FindCostAction">
         <!--
             正常情况下跳转到资费列表页面。
             一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
          -->
         <result name="success">
             /WEB-INF/cost/find_cost.jsp
         </result>
         <!--
             错误情况下，跳转到错误页面。
             错误页面可以被所有模块复用，因此放在 main 下，
             该文件夹用于存放公用的页面。
          -->
         <result name="error">
             /WEB-INF/main/error.jsp
         </result>
     </action>
     <!-- 删除资费 -->
     <action name="deleteCost"
         class="com.netctoss.action.DeleteCostAction">
         <!-- 删除完之后，重定向到查询 action -->
         <result name="success" type="redirectAction">
             findCost
         </result>
```

```
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
               该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
               会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 生成验证码 -->
        <action                                          name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!-- 使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!-- 指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>

   </struts>
```

find_cost.jsp 完整代码：

```
   <%@page pageEncoding="utf-8" isELIgnored="false"%>
   <%@taglib uri="/struts-tags" prefix="s"%>
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global_color.css" />
         <script language="javascript" type="text/javascript">
```

```
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort_desc")
                    btnObj.className = "sort_asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                         href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                     href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                     href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                       href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information_off"></a></li>
                <li><a                       href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value=" 基 费 "  class="sort_asc"
```

```
onclick="sort(this);" />
                        <input  type="button"  value=" 时 长 "  class="sort asc"
onclick="sort(this);" />
                    </div>
                    <input  type="button"  value=" 增 加 "  class="btn_add"
onclick="location.href='fee_add.html';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                          src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功!
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>

                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
                            <td><a          href="fee_detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                                <input    type="button"    value="   启   用   "
class="btn_start" onclick="startFee();" />
                                <input    type="button"    value="   修   改   "
class="btn_modify" onclick="location.href='fee_modi.html';" />
                                <input    type="button"    value="   删   除   "
class="btn_delete" onclick="deleteFee(<s:property value="id"/>);" />
                            </td>
                         </tr>
                       </s:iterator>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
```

```
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                    <a href="#" class="current page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
</html>
```

达内IT培训集团

课后作业

1. 请简述 Struts2 中 Result 组件的作用，你了解哪些 Result 组件

2. stream 类型的 Result 有什么作用

3. redirectAction 类型的 Result 有什么作用

Struts 核心

Unit04

知识体系.....Page 178

json Result	作用	json Result 的作用
	使用方式	语法
		使用步骤
	名称唯一性校验	需求描述
		开发思路
		开发步骤
资费修改	开发思路	需要几种请求
		打开修改页面的请求过程
	开发步骤	打开修改页面的开发步骤
UI 标签	作用	UI 标签的作用
	简单的 UI 标签	表单
		文本框、密码框、文本域
		布尔框
	复杂的 UI 标签	单选框
		多选框
		下拉选

经典案例.....Page 186

资费名唯一性校验-1	开发步骤
资费名唯一性校验-2	
资费修改	打开修改页面的开发步骤
使用简单的 UI 标签	表单
	文本框、密码框、文本域
	布尔框
使用复杂的 UI 标签	单选框
	多选框
	下拉选

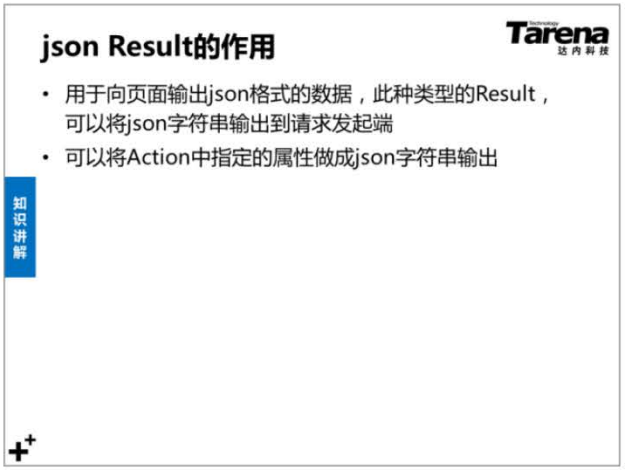
课后作业.....Page 266

达内IT培训集团

1. json Result

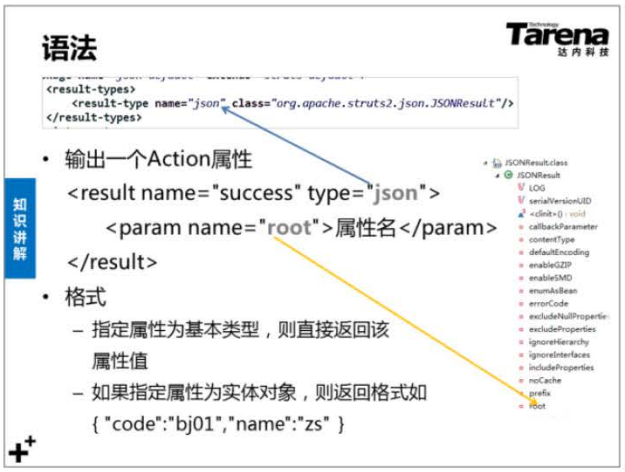
1.1. 作用

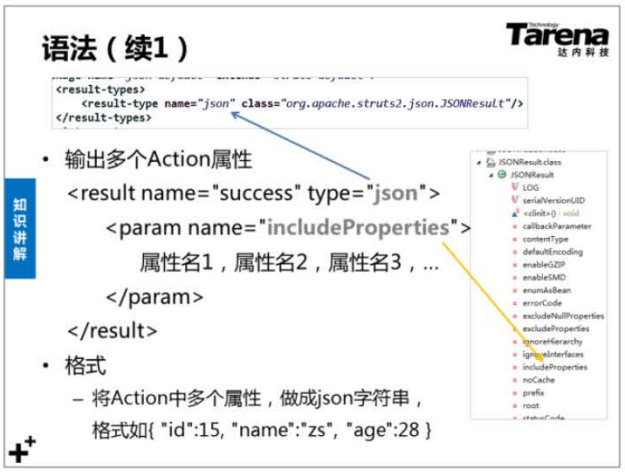
1.1.1. 【作用】json Result 的作用

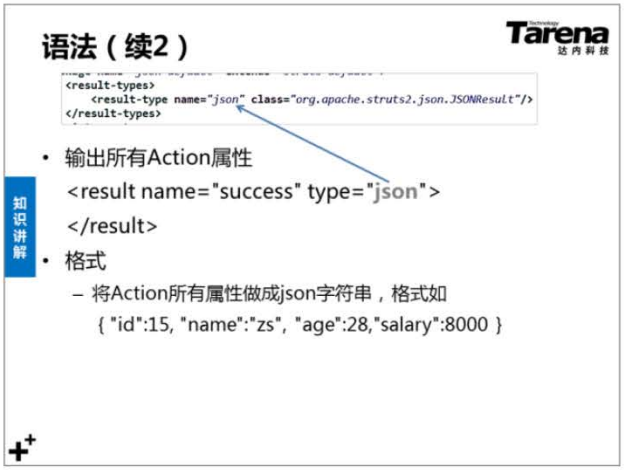


1.2. 使用方式

1.2.1. 【使用方式】语法



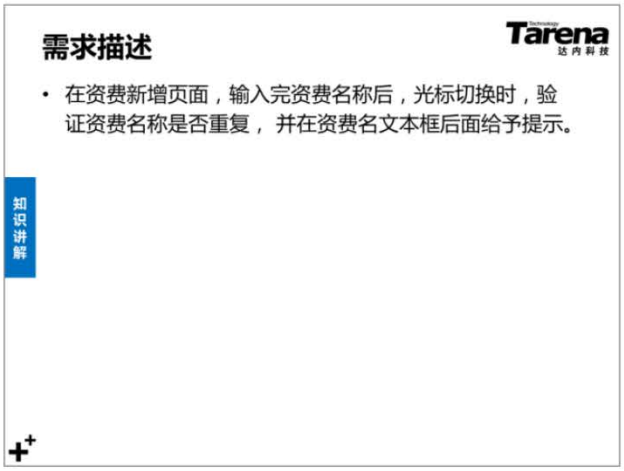




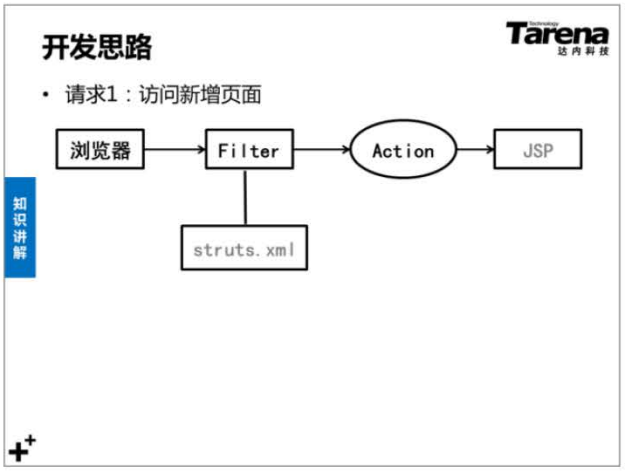
1.2.2. 【使用方式】使用步骤

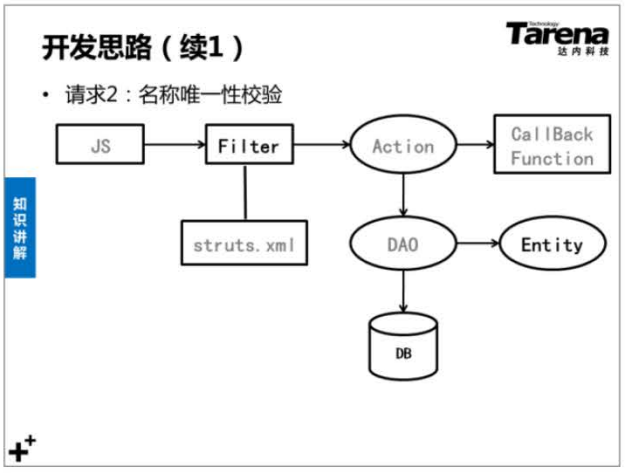
1.3. 名称唯一性校验

1.3.1. 【名称唯一性校验】需求描述



1.3.2. 【名称唯一性校验】开发思路





1.3.3. 【名称唯一性校验】开发步骤

开发步骤（续1）

- 请求2
  - 资费DAO中追加根据名称查询资费方法
  - 创建资费名唯一性校验Action
    1）输入：资费名
    2）输出：是否重复、提示信息
    3）业务方法：验证资费名是否重复
  - 配置该action
    1）struts.xml中配置当前action
    2）单元测试

解详识知

开发步骤（续2）

- 请求2
  - 异步请求
    1）页面上使用JQuery发异步请求访问此action
    2）回调函数中处理请求返回结果

解详识知

2. 资费修改

2.1. 开发思路

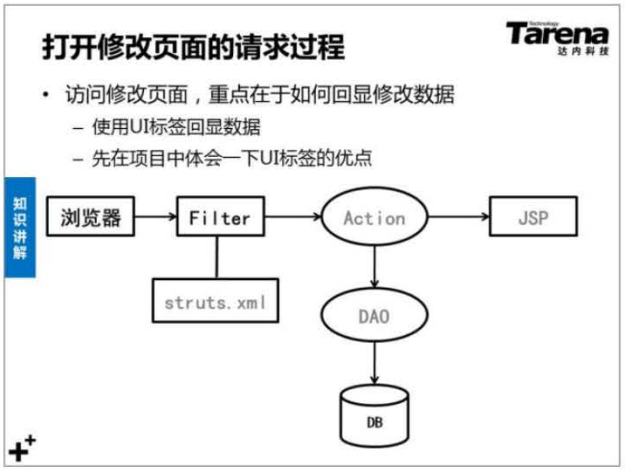
2.1.1. 【开发思路】需要几种请求

需要几种请求

- 点击修改按钮，打开修改页面，将要修改的数据在页面上做默认的显示
- 点击保存按钮，对数据进行修改（扩展）

解详识知

2.1.2. 【开发思路】打开修改页面的请求过程



2.2. 开发步骤

2.2.1. 【开发步骤】打开修改页面的开发步骤

3. UI 标签

3.1. 作用

3.1.1. 【作用】UI 标签的作用

3.2. 简单的 UI 标签

3.2.1. 【简单的 UI 标签】表单

表单

- 表单
  - 语法
    <s:form action="" method="" theme="simple"></s:form>
  - 说明
    1）用于生成HTML表单元素
    2）theme用于指定主题。simple是简约主题，生成时不带样式和表格

解详识知

3.2.2. 【简单的 UI 标签】文本框、密码框、文本域

文本框、密码框、文本域

- 文本框
  - 语法
    <s:textfield name="**userName**" />
  - 说明
    1）首先生成一个文本框
    2）根据OGNL表达式（**userName**）访问ValueStack，并将取得的结果设置为文本框的默认值
- 密码框
  - 用法同文本框，不同的是生成一个密码框
- 文本域
  - 用法同文本框，不同的是生成一个文本域

解详识知

3.2.3. 【简单的 UI 标签】布尔框

布尔框

- 布尔框
  - 语法
    <s:checkbox name="**marry**" />
  - 说明
    1）首先生成一个布尔框，即单个复选框
    2）根据OGNL表达式（**marry**）访问ValueStack，访问的属性应该是布尔类型的数据。根据返回的布尔值，设置该复选框是否勾选。

解详识知

3.3. 复杂的 UI 标签

3.3.1. 【复杂的 UI 标签】单选框

单选框

- 静态初始化
  - 语法

    <s:radio name="***sex***" list="#{'M':'男', 'F':'女'}/>
  - 说明

    1）根据OGNL表达式创建的Map（斜体）生成一组单选框，Map中有几个键值对，就生成几个radio。其中Map的key用于生成radio的value值，Map的value用于生成radio的label显示值。

    2）根据OGNL表达式（**粗体**）访问ValueStack，并将返回的结果与radio的value值比较，哪个radio的value值与返回结果一致，则该radio默认选中。

解详识知

3.3.2. 【复杂的 UI 标签】多选框

多选框

- 静态初始化
  - 语法

    <s:checkboxlist name= "**travelCities**"

    list="#{'01':'北京', '02':'上海', '03':'广州', '04':'深圳'}"/>
  - 说明

    1）根据OGNL表达式创建的Map（*斜体*）生成一组多选框，Map中有几个键值对，就生成几个checkbox。其中Map的key用于生成checkbox的value值，Map的value用于生成checkbox的label显示值。

    2）根据OGNL表达式（**粗体**）访问ValueStack，访问的属性为集合（List<String>），并将返回的结果与checkbox的value值比较，哪个checkbox的value值在返回结果的集合中，则该checkbox默认选中。

解详识知

多选框（续1）

- 动态初始化
  - 语法

```
< s:checkboxlist name= "travelCities" list="cities"
        listKey="cityCode" listValue="cityName"/>
```

  - 说明

    1）根据OGNL表达式（*斜体*）访问ValueStack，访问的属性应为集合（List<City>），并根据返回结果生成一组多选框。集合中有几个值，就生成几个checkbox。期间，会根据listKey指定的实体（City）属性来生成checkbox的value值，根据listValue指定的实体（City）属性来生成checkbox的显示值。

    2）根据OGNL表达式（**粗体**）访问ValueStack，访问的属性为集合（List<String>），并将返回的结果与checkbox的value值比较，哪个checkbox的value值在返回结果的集合中，则该checkbox默认选中。

3.3.3. 【复杂的 UI 标签】下拉选

下拉选

- 静态初始化
  - 语法

```
<s:select name="home"
    list="#{'01':'北京', '02':'上海', '03':'广州', '04':'深圳'}"/>
```

  - 说明

    1）根据OGNL表达式创建的Map（*斜体*）生成一组下拉选，Map中有几个键值对，就生成几个option。其中Map的key用于生成option的value值，Map的value用于生成option的显示值。

    2）根据OGNL表达式（**粗体**）访问ValueStack，并将返回的结果与option的value值比较，哪个option的value值与返回结果一致，则该option默认选中。

下拉选（续1）

- 动态初始化
  - 语法

```
<s:select name="home" list="cities"
        listKey="cityCode" listValue="cityName"/>
```

  - 说明

    1）根据OGNL表达式（*斜体*）访问ValueStack，访问的属性应为集合（List<City>），并根据返回结果生成一组下拉选。集合中有几个值，就生成几个option。期间，会根据listKey指定的实体（City）属性来生成option的value值，根据listValue指定的实体（City）属性来生成option的显示值。

    2）根据OGNL表达式（**粗体**）访问ValueStack，并将返回的结果与option的value值比较，哪个option的value值与返回结果一致，则该option默认选中。

经典案例

1. 资费名唯一性校验-1

- **问题**

在资费新增页面，输入完资费名称后，光标切换时，验证资费名称是否重复，并在资费名文本框后面给与提示。

- **方案**

该需求中包含 2 个请求，一个是打开新增资费页面，另一个是输入完资费名称后进行唯一性校验。

其中打开资费页面很简单，参考打开登录页面完成即可，本案例中要完成该请求。

输入完资费名后在光标切换时验证资费名是否重复，这需要发起一次新请求，并且页面不能刷新，因此需要采用异步的方式来实现。而对于异步请求，我们需要在页面上使用 JQuery 发起请求，在 Action 中处理请求并使用 json 类型的 result 处理请求，将结果发送给异步请求的回调函数。

本案例中要求完成异步请求的服务端代码，包括 DAO、Action、struts.xml，其中

- DAO 中增加根据名称查询资费数据的方法。
- Action 中根据传入的名称查询资费数据，并根据返回结果判断该名称是否重复。
- struts.xml 中配置该校验资费名的 action，并使用 json 类型的 result 将结果发送给页面的异步请求回调函数。

- **步骤**

实现此案例需要按照如下步骤进行。

请求 1：打开新增页面

步骤一：在 struts.xml 中配置请求 action

在 struts.xml 中，配置打开新增资费页面的请求 action，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
```

```
    <package name="cost" namespace="/cost" extends="struts-default">
        <!—此处略去其他 Action 的配置... -->


        <!-- 打开资费新增页 -->
        <action name="toAddCost">
            <result name="success">
                /WEB-INF/cost/add_cost.jsp
            </result>
        </action>

    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!—此处略去其他 Action 的配置... -->
    </package>

   </struts>
```

步骤二：创建新增资费页面

在 WEB-INF/cost 下，创建 add_cost.jsp，并将静态页面的代码复制到该 JSP 中,代码如下：

```
   <%@page pageEncoding="utf-8"%>
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内 - NetCTOSS</title>
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
         <script language="javascript" type="text/javascript">
            //保存结果的提示
            function showResult() {
               showResultDiv(true);
               window.setTimeout("showResultDiv(false);", 3000);
            }
            function showResultDiv(flag) {
               var divResult = document.getElementById("save_result_info");
               if (flag)
                  divResult.style.display = "block";
               else
                  divResult.style.display = "none";
            }

            //切换资费类型
            function feeTypeChange(type) {
               var                      inputArray                      =
document.getElementById("main").getElementsByTagName("input");
               if (type == 1) {
                  inputArray[4].readonly = true;
                  inputArray[4].value = "";
                  inputArray[4].className += " readonly";
                  inputArray[5].readonly = false;
                  inputArray[5].className = "width100";
                  inputArray[6].readonly = true;
```

```
                    inputArray[6].className += " readonly";
                    inputArray[6].value = "";
                }
                else if (type == 2) {
                    inputArray[4].readonly = false;
                    inputArray[4].className = "width100";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
                else if (type == 3) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                         href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                     href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                     href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                       href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information_off"></a></li>
                <li><a                       href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save_result_info" class="save_fail">保存失败，资费名称重复！
</div>
            <form action="" method="" class="main_form">
                <div class="text_info clearfix"><span>资费名称：</span></div>
                <div class="input_info">
                    <input type="text" class="width300" value=""/>
                    <span class="required">*</span>
```

```
                    <div class="validate_msg_short">50 长度的字母、数字、汉字和下划线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">
                    <input type="radio" name="radFeeType" id="monthly" onclick="feeTypeChange(1);" />
                    <label for="monthly">包月</label>
                    <input type="radio" name="radFeeType" checked="checked" id="package" onclick="feeTypeChange(2);" />
                    <label for="package">套餐</label>
                    <input type="radio" name="radFeeType" id="timeBased" onclick="feeTypeChange(3);" />
                    <label for="timeBased">计时</label>
                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>单位费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info high">
                    <textarea class="width300 height70"></textarea>
                    <div class="validate_msg_short error_msg">100 长度的字母、数字、汉字和下划线的组合</div>
                </div>
                <div class="button_info clearfix">
                    <input type="button" value="保存" class="btn_save" onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</span>
```

```
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
    </body>
</html>
```

步骤三：列表页面上给增加按钮指定跳转的 URL

在列表页面 find_cost.jsp 中，给增加按钮设置访问路径，代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort_desc")
                    btnObj.className = "sort_asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
```

```
                <li><a                          href="../role/role list.html"
class="role off"></a></li>
                <li><a                        href="../admin/admin_list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                     href="../account/account list.html"
class="account off"></a></li>
                <li><a                     href="../service/service list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                       href="../report/report list.html"
class="report off"></a></li>
                <li><a                          href="../user/user info.html"
class="information off"></a></li>
                <li><a                       href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基费"  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value="时长"  class="sort_asc"
onclick="sort(this);" />
                    </div>


                    <input   type="button"   value="  增  加  "   class="btn_add"
onclick="location.href='toAddCost';" />


                </div>
                <!--启用操作的操作提示-->
                <div id="operate result info" class="operate success">
                    <img                            src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
```

```
                        </tr>

                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
                            <td><a          href="fee detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                               <input    type="button"    value="   启   用   "
class="btn start" onclick="startFee();" />
                               <input    type="button"    value="   修   改   "
class="btn_modify" onclick="location.href='fee_modi.html';" />
                               <input    type="button"    value="   删   除   "
class="btn_delete" onclick="deleteFee(<s:property value="id"/>);" />
                            </td>
                         </tr>
                       </s:iterator>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                 </div>
                 <!--分页-->
                 <div id="pages">
                   <a href="#">上一页</a>
                    <a href="#" class="current page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                 </div>
              </form>
           </div>
           <!--主要区域结束-->
           <div id="footer">
              <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
              <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
           </div>
        </body>
    </html>
```

步骤四：测试

重新部署项目并重启 tomcat，打开资费列表页面，点击增加按钮，效果如下图：

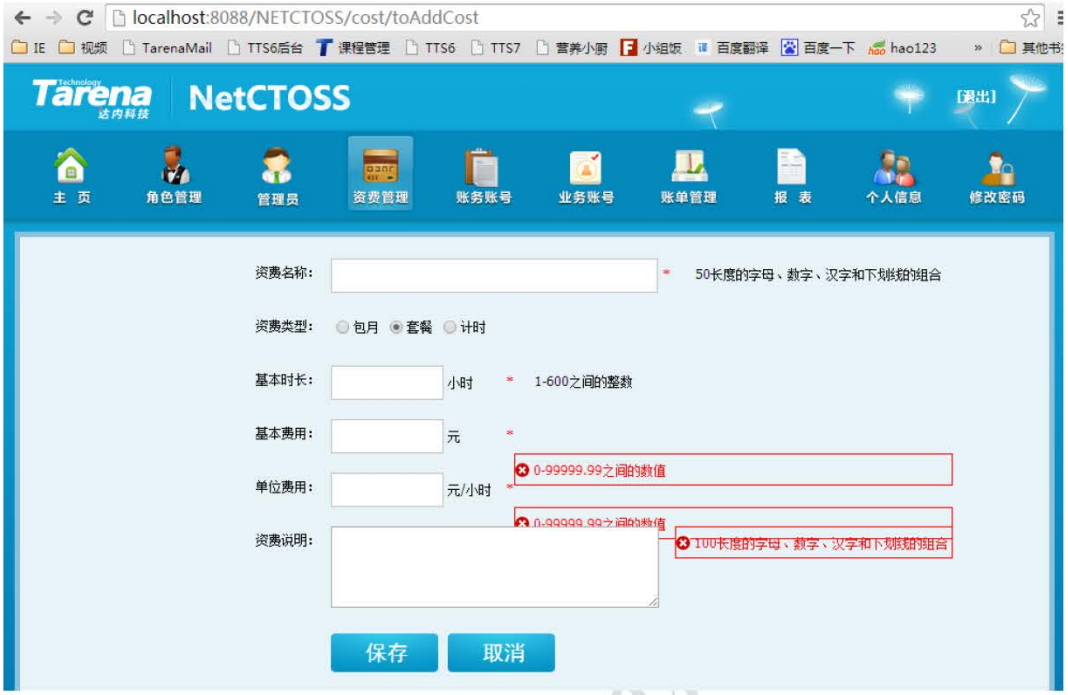


图-1

请求 2：资费名称唯一性校验

步骤一：导包

由于需要使用 json 类型的 result，因此需要导入新的包，名为 struts2-json-plugin-2.1.8.jar。

步骤二：在 DAO 中追加根据名称查询资费的方法

在 ICostDao 中，追加根据名称查询资费的方法，代码如下：

```
package com.netctoss.dao;

import java.util.List;

import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {

// 此处略去其他方法的声明...

 /**
  * 根据名称查询资费
  * @param name 资费名
  * @return
  */
```

```
    Cost findByName(String name);

    }
```

在 CostDaoImpl 类中，实现根据名称查询资费的方法，代码如下：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;

import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

// 此处略去其他方法的实现...

    @Override
    public Cost findByName(String name) {
        // 模拟根据名称查询资费数据，假设资费表中只有一条名为 tarena 的数据
        if("tarena".equals(name)) {
            Cost c = new Cost();
            c.setId(97);
            c.setName("tarena");
            c.setBaseDuration(99);
            c.setBaseCost(9.9);
            c.setUnitCost(0.9);
            c.setDescr("tarena 套餐");
            c.setStatus("0");
            c.setCostType("2");
            return c;
        }
        return null;
    }

}
```

步骤三：追加校验资费名 Action

在 com.netctoss.action 包下，创建资费名称校验 Action 类 CheckCostNameAction，该 Action 的输入属性是资费名，输出属性设置为一个 Map，这样方便封装多个输出内容。在业务方法中根据资费名查询资费数据，并从结果是否为空来判断资费名是否重复，最终将判断结果封装于 Map 中输出给回调函数，代码如下：

```
package com.netctoss.action;

import java.util.HashMap;
import java.util.Map;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;
```

```
public class CheckCostNameAction {

    // input
    private String name; // 资费名

    // output
    private Map<String, Object> info = new HashMap<String, Object>(); // 提示信
息

    public String execute() {
        ICostDao dao = DAOFactory.getCostDAO();
        Cost cost = null;
        try {
            cost = dao.findByName(name);
        } catch (Exception e) {
            e.printStackTrace();
            /*
             * 由于是异步请求，并通过 json 类型的 result 处理结果，
             * 那么该 result 一定会把结构发给回调函数，因此这里
             * 不要返回 error 了，否则 result 会把 error.jsp 中的代码
             * 发给回调函数，因此这里只给出提示信息即可。
             * */
            info.put("success", false);
            info.put("message", "系统发生异常，请联系管理员.");
        }

        if(cost == null) {
            // 没找到资费，说明名称没重复
            info.put("success", true);
            info.put("message", "有效的资费名称.");
        } else {
            //找到了资费数据，说明名称重复了
            info.put("success", false);
            info.put("message", "资费名称已存在.");
        }

        return "success";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Map<String, Object> getInfo() {
        return info;
    }

    public void setInfo(Map<String, Object> info) {
        this.info = info;
    }

}
```

步骤四：在 struts.xml 中配置校验资费名 Action

在 struts.xml 中，配置资费名校验 action。由于该 action 需要使用 json 类型的

result，因此需要先将它所在的 package 继承修改为 json-default，然后再配置资费名校验 action，并使用 json 类型的 result 将结果输出给回调函数，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->

 <package name="cost" namespace="/cost" extends="json-default">

     <!—此处略去其他 Action 的配置... -->

     <!-- 资费名唯一性校验 -->
     <action name="checkCostName"
         class="com.netctoss.action.CheckCostNameAction">
         <!-- 使用 json 类型的 result 把结果输出给回调函数 -->
         <result name="success" type="json">
             <param name="root">info</param>
         </result>
     </action>

 </package>

 <!-- 登录模块 -->
 <package name="login" namespace="/login" extends="struts-default">
     <!—此处略去其他 Action 的配置... -->
 </package>

</struts>
```

步骤五：测试

在浏览器中，直接输入地址访问此 action，并给 name 参数赋值，路径为 http://localhost:8088/NETCTOSS/cost/checkCostName?name=aaa，通过观察浏览器中的内容可以达到测试目的。注意，有些浏览器可能不是直接显示结果，而是将接收的结果保存到文件中，这是浏览器设置的问题，也可以将内容保存后再行浏览。效果如下图：

图-2

- **完整代码**

本案例的 ICostDao 完整代码：

```
package com.netctoss.dao;

import java.util.List;

import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {

 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();

 /**
  * 删除一条资费数据
  * @param id 主键
  */
 void delete(int id);

 /**
  * 根据名称查询资费
  * @param name 资费名
  * @return
  */
 Cost findByName(String name);

}
```

CostDaoImpl 完整代码：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;

import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

 @Override
 public List<Cost> findAll() {
     // 模拟查询全部资费数据
     List<Cost> list = new ArrayList<Cost>();

     Cost c1 = new Cost();
     c1.setId(95);
     c1.setName("6 元套餐");
     c1.setBaseDuration(66);
     c1.setBaseCost(6.6);
     c1.setUnitCost(0.6);
     c1.setDescr("6 元套餐");
```

```
        c1.setStatus("0");
        c1.setCostType("2");
        list.add(c1);

        Cost c2 = new Cost();
        c2.setId(96);
        c2.setName("8 元套餐");
        c2.setBaseDuration(88);
        c2.setBaseCost(8.8);
        c2.setUnitCost(0.8);
        c2.setDescr("8 元套餐");
        c2.setStatus("0");
        c2.setCostType("2");
        list.add(c2);

        Cost c3 = new Cost();
        c3.setId(97);
        c3.setName("tarena");
        c3.setBaseDuration(99);
        c3.setBaseCost(9.9);
        c3.setUnitCost(0.9);
        c3.setDescr("tarena 套餐");
        c3.setStatus("0");
        c3.setCostType("2");
        list.add(c3);

        return list;
    }

    @Override
    public void delete(int id) {
        // 模拟根据 id 删除资费数据
        System.out.println("删除 ID 为[" + id + "]的资费数据.");
    }

    @Override
    public Cost findByName(String name) {
        // 模拟根据名称查询资费数据，假设资费表中只有一条名为 tarena 的数据
        if("tarena".equals(name)) {
            Cost c = new Cost();
            c.setId(97);
            c.setName("tarena");
            c.setBaseDuration(99);
            c.setBaseCost(9.9);
            c.setUnitCost(0.9);
            c.setDescr("tarena 套餐");
            c.setStatus("0");
            c.setCostType("2");
            return c;
        }
        return null;
    }
}
```

CheckCostNameAction 完整代码：

```
package com.netctoss.action;

import java.util.HashMap;
import java.util.Map;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;
```

```
public class CheckCostNameAction {

    // input
    private String name; // 资费名

    // output
    private Map<String, Object> info = new HashMap<String, Object>(); // 提示信
息

    public String execute() {
        ICostDao dao = DAOFactory.getCostDAO();
        Cost cost = null;
        try {
            cost = dao.findByName(name);
        } catch (Exception e) {
            e.printStackTrace();
            /*
             * 由于是异步请求，并通过 json 类型的 result 处理结果，
             * 那么该 result 一定会把结构发给回调函数，因此这里
             * 不要返回 error 了，否则 result 会把 error.jsp 中的代码
             * 发给回调函数，因此这里只给出提示信息即可。
             * */
            info.put("success", false);
            info.put("message", "系统发生异常，请联系管理员.");
        }

        if(cost == null) {
            // 没找到资费，说明名称没重复
            info.put("success", true);
            info.put("message", "有效的资费名称.");
        } else {
            //找到了资费数据，说明名称重复了
            info.put("success", false);
            info.put("message", "资费名称已存在.");
        }

        return "success";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Map<String, Object> getInfo() {
        return info;
    }

    public void setInfo(Map<String, Object> info) {
        this.info = info;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
```

```
<!--
    资费模块配置信息：
    一般情况下，一个模块的配置单独封装在一个 package 下，
    并且以模块名来命名 package 的 name 和 namespace。
 -->
<package name="cost" namespace="/cost" extends="json-default">
    <!-- 查询资费数据 -->
    <action name="findCost" class="com.netctoss.action.FindCostAction">
        <!--
            正常情况下跳转到资费列表页面。
            一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
         -->
        <result name="success">
            /WEB-INF/cost/find cost.jsp
        </result>
        <!--
            错误情况下，跳转到错误页面。
            错误页面可以被所有模块复用，因此放在 main 下，
            该文件夹用于存放公用的页面。
         -->
        <result name="error">
            /WEB-INF/main/error.jsp
        </result>
    </action>
    <!-- 删除资费 -->
    <action name="deleteCost"
        class="com.netctoss.action.DeleteCostAction">
        <!-- 删除完之后，重定向到查询 action -->
        <result name="success" type="redirectAction">
            findCost
        </result>
        <result name="error">
            /WEB-INF/main/error.jsp
        </result>
    </action>
    <!-- 打开资费新增页 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add_cost.jsp
        </result>
    </action>
    <!-- 资费名唯一性校验 -->
    <action name="checkCostName"
        class="com.netctoss.action.CheckCostNameAction">
        <!-- 使用 json 类型的 result 把结果输出给回调函数 -->
        <result name="success" type="json">
            <param name="root">info</param>
        </result>
    </action>
</package>

<!-- 登录模块 -->
<package name="login" namespace="/login" extends="struts-default">
    <!--
        打开登录页面：
        1、action 的 class 属性可以省略，省略时 Struts2
           会自动实例化默认的 Action 类 ActionSupport，
           该类中有默认业务方法 execute，返回 success。
        2、action 的 method 属性可以省略，省略时 Struts2
           会自动调用 execute 方法。
    -->
    <action name="toLogin">
        <result name="success">
            /WEB-INF/main/login.jsp
        </result>
```

```
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 生成验证码 -->
        <action                                                name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!-- 使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!-- 指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>

   </struts>
```

find_cost.jsp 完整代码：

```
   <%@page pageEncoding="utf-8" isELIgnored="false"%>
   <%@taglib uri="/struts-tags" prefix="s"%>
   <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
   <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global.css" />
         <link      type="text/css"      rel="stylesheet"      media="all"
href="../styles/global_color.css" />
         <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
               if (btnObj.className == "sort desc")
                  btnObj.className = "sort asc";
               else
                  btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
               var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
               document.getElementById("operate result info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
               var r = window.confirm("确定要删除此资费吗？");
               if(r) {
```

```
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role list.html"
class="role off"></a></li>
                <li><a                        href="../admin/admin list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account off"></a></li>
                <li><a                    href="../service/service list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report_list.html"
class="report off"></a></li>
                <li><a                          href="../user/user info.html"
class="information off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value=" 基 费 "  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value=" 时 长 "  class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input   type="button"   value=" 增  加  "   class="btn add"
onclick="location.href='toAddCost';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                            src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功!
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
```

```
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>

                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
                            <td><a          href="fee detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                                <input    type="button"    value="   启   用   "
class="btn_start" onclick="startFee();" />
                                <input    type="button"    value="   修   改   "
class="btn_modify" onclick="location.href='fee_modi.html';" />
                                <input    type="button"    value="   删   除   "
class="btn_delete" onclick="deleteFee(<s:property value="id"/>);" />
                            </td>
                        </tr>
                       </s:iterator>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
                <!--分页-->
                <div id="pages">
                  <a href="#">上一页</a>
                    <a href="#" class="current page">1</a>
                    <a href="#">2</a>
                    <a href="#">3</a>
                    <a href="#">4</a>
                    <a href="#">5</a>
                    <a href="#">下一页</a>
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
            <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
        </div>
    </body>
  </html>
```

add_cost.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内－NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            //保存结果的提示
            function showResult() {
                showResultDiv(true);
                window.setTimeout("showResultDiv(false);", 3000);
            }
            function showResultDiv(flag) {
                var divResult = document.getElementById("save_result_info");
                if (flag)
                    divResult.style.display = "block";
                else
                    divResult.style.display = "none";
            }

            //切换资费类型
            function feeTypeChange(type) {
                var inputArray =
document.getElementById("main").getElementsByTagName("input");
                if (type == 1) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = true;
                    inputArray[6].className += " readonly";
                    inputArray[6].value = "";
                }
                else if (type == 2) {
                    inputArray[4].readonly = false;
                    inputArray[4].className = "width100";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
                else if (type == 3) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
```

```
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                       href="../admin/admin list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                    href="../service/service_list.html"
class="service off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user info.html"
class="information off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save_result_info" class="save_fail">保存失败，资费名称重复！
</div>
            <form action="" method="" class="main form">
                <div class="text info clearfix"><span>资费名称：</span></div>
                <div class="input info">
                    <input type="text" class="width300" value=""/>
                    <span class="required">*</span>
                    <div class="validate_msg_short">50 长度的字母、数字、汉字和下划
线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">
                    <input   type="radio"   name="radFeeType"   id="monthly"
onclick="feeTypeChange(1);" />
                    <label for="monthly">包月</label>
                    <input  type="radio"  name="radFeeType"  checked="checked"
id="package" onclick="feeTypeChange(2);" />
                    <label for="package">套餐</label>
                    <input   type="radio"   name="radFeeType"   id="timeBased"
onclick="feeTypeChange(3);" />
                    <label for="timeBased">计时</label>
                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate msg long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的
数值</div>
```

```
                </div>
                <div class="text info clearfix"><span>单位费用：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的
数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info_high">
                    <textarea class="width300 height70"></textarea>
                    <div class="validate_msg_short error_msg">100 长度的字母、数
字、汉字和下划线的组合</div>
                </div>
                <div class="button info clearfix">
                    <input  type="button"  value=" 保 存 "  class="btn_save"
onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
    </body>
  </html>
```

2. 资费名唯一性校验-2

- **问题**

完成资费名唯一性校验。

- **方案**

在资费新增页面上，给资费名文本框增加光标移除事件，并绑定资费名唯一性校验函数。该函数中，使用 JQuery 发送异步请求访问资费名校验 action，并在回调函数中根据返回结果设置提示信息。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：引入 JQuery

在 WebRoot 下创建 js 文件夹，将 jquery-1.4.3.js 文件复制到此文件夹下，完成后 WebRoot 结构如下图：

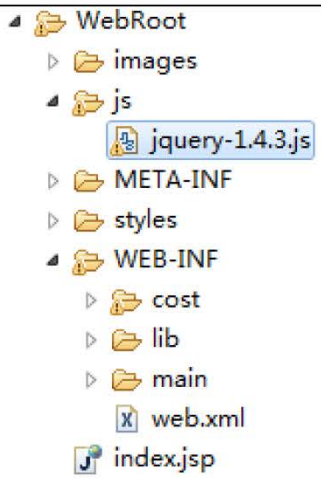


图-3

步骤二：新增页面上发起异步校验请求

在 add_cost.jsp 上，给资费名称文本框的 onblur 事件绑定资费名唯一性校验函数 check_name，然后在<head>通过 script 标记引入 JQuery，并实现 check_name 函数中的异步校验逻辑，代码如下：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all" href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all" href="../styles/global_color.css" />

        <script language="javascript" type="text/javascript" src="../js/jquery-1.4.3.js"></script>

        <script language="javascript" type="text/javascript">

          //校验资费名是否重复
          function check_name() {
              var cost_name = $("#cost_name").val();
              // 校验资费名是否为空
              if(cost_name == "") {
                  $("#name_msg").text("资费名不能为空.").addClass("error_msg");
                  return;
              }
              // 异步校验资费名是否重复
              $.post(
                  "checkCostName",
                  {"name":cost_name},
                  function(data) {
                      // 回调函数的参数就是返回的 info 属性
                      var info = data;
```

```
                        // 根据返回值设置提示信息
                        if(info.success) {
                            //  验证通过，设置提示信息并移除错误样式

    $("#name_msg").text(info.message).removeClass("error_msg");
                        } else {
                            // 验证失败，设置提示信息并添加错误样式

    $("#name_msg").text(info.message).addClass("error_msg");
                        }
                    }
                );
            }


            //保存结果的提示
            function showResult() {
                showResultDiv(true);
                window.setTimeout("showResultDiv(false);", 3000);
            }
            function showResultDiv(flag) {
                var divResult = document.getElementById("save_result_info");
                if (flag)
                    divResult.style.display = "block";
                else
                    divResult.style.display = "none";
            }

            //切换资费类型
            function feeTypeChange(type) {
                var                         inputArray                          =
document.getElementById("main").getElementsByTagName("input");
                if (type == 1) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = true;
                    inputArray[6].className += " readonly";
                    inputArray[6].value = "";
                }
                else if (type == 2) {
                    inputArray[4].readonly = false;
                    inputArray[4].className = "width100";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
                else if (type == 3) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
```

```
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                        href="../admin/admin list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                    href="../account/account list.html"
class="account_off"></a></li>
                <li><a                    href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill off"></a></li>
                <li><a                      href="../report/report list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information_off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save_result_info" class="save_fail">保存失败，资费名称重复！
</div>
            <form action="" method="" class="main_form">
                <div class="text_info clearfix"><span>资费名称：</span></div>
                <div class="input_info">

                    <input     type="text"     class="width300"     id="cost_name"
onblur="check_name();"/>

                    <span class="required">*</span>
                    <div class="validate_msg_short" id="name_msg">50 长度的字母、
数字、汉字和下划线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input info fee type">
                    <input   type="radio"   name="radFeeType"   id="monthly"
onclick="feeTypeChange(1);" />
                    <label for="monthly">包月</label>
                    <input  type="radio"  name="radFeeType"  checked="checked"
id="package" onclick="feeTypeChange(2);" />
                    <label for="package">套餐</label>
                    <input   type="radio"   name="radFeeType"   id="timeBased"
onclick="feeTypeChange(3);" />
                    <label for="timeBased">计时</label>
                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
```

```
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的
数值</div>
                </div>
                <div class="text_info clearfix"><span>单位费用：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的
数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info_high">
                    <textarea class="width300 height70"></textarea>
                    <div class="validate_msg_short error_msg">100 长度的字母、数
字、汉字和下划线的组合</div>
                </div>
                <div class="button_info clearfix">
                    <input  type="button"  value=" 保 存 "  class="btn_save"
onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
    </body>
</html>
```

步骤三：测试

重新部署项目并重启 tomcat，打开新增页面，输入一个已存在的资费名称，切换光标时效果如下图：

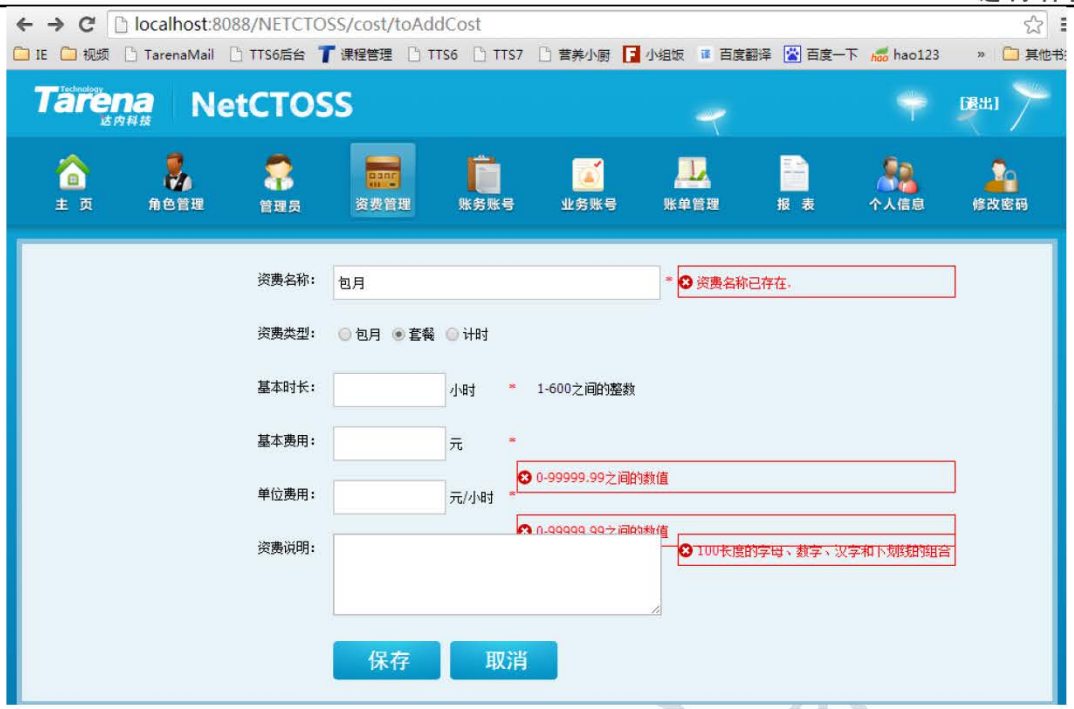


图-4

- **完整代码**

本案例的 add_cost.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内－NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript"
src="../js/jquery-1.4.3.js"></script>
        <script language="javascript" type="text/javascript">
          //校验资费名是否重复
          function check name() {
              var cost_name = $("#cost_name").val();
              // 校验资费名是否为空
              if(cost_name == "") {
                  $("#name_msg").text("资费名不能为
空.").addClass("error_msg");
                  return;
              }
              // 异步校验资费名是否重复
              $.post(
                  "checkCostName",
                  {"name":cost name},
                  function(data) {
                      // 回调函数的参数就是返回的 info 属性
                      var info = data;
                      // 根据返回值设置提示信息
```

```
                        if(info.success) {
                            //  验证通过，设置提示信息并移除错误样式

    $("#name_msg").text(info.message).removeClass("error_msg");
                        } else {
                            // 验证失败，设置提示信息并添加错误样式

    $("#name_msg").text(info.message).addClass("error_msg");
                        }
                    }
                );
            }

            //保存结果的提示
            function showResult() {
                showResultDiv(true);
                window.setTimeout("showResultDiv(false);", 3000);
            }
            function showResultDiv(flag) {
                var divResult = document.getElementById("save_result_info");
                if (flag)
                    divResult.style.display = "block";
                else
                    divResult.style.display = "none";
            }

            //切换资费类型
            function feeTypeChange(type) {
                var                         inputArray                      =
document.getElementById("main").getElementsByTagName("input");
                if (type == 1) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = true;
                    inputArray[6].className += " readonly";
                    inputArray[6].value = "";
                }
                else if (type == 2) {
                    inputArray[4].readonly = false;
                    inputArray[4].className = "width100";
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
                else if (type == 3) {
                    inputArray[4].readonly = true;
                    inputArray[4].value = "";
                    inputArray[4].className += " readonly";
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
```

```
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                        href="../admin/admin_list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                     href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                     href="../service/service_list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill off"></a></li>
                <li><a                       href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information off"></a></li>
                <li><a                       href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save_result_info" class="save_fail">保存失败，资费名称重复！
</div>
            <form action="" method="" class="main form">
                <div class="text info clearfix"><span>资费名称：</span></div>
                <div class="input info">
                    <input   type="text"   class="width300"   id="cost_name"
onblur="check_name();"/>
                    <span class="required">*</span>
                    <div class="validate_msg_short" id="name_msg">50 长度的字母、
数字、汉字和下划线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">
                    <input   type="radio"   name="radFeeType"   id="monthly"
onclick="feeTypeChange(1);" />
                    <label for="monthly">包月</label>
                    <input  type="radio"  name="radFeeType"  checked="checked"
id="package" onclick="feeTypeChange(2);" />
                    <label for="package">套餐</label>
                    <input   type="radio"   name="radFeeType"   id="timeBased"
onclick="feeTypeChange(3);" />
                    <label for="timeBased">计时</label>
                </div>
                <div class="text info clearfix"><span>基本时长：</span></div>
                <div class="input info">
                    <input type="text" value="" class="width100" />
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate msg long error msg">0-99999.99 之间的
数值</div>
                </div>
```

```
                <div class="text_info clearfix"><span>单位费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long error_msg">0-99999.99 之间的
数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info_high">
                    <textarea class="width300 height70"></textarea>
                    <div class="validate_msg_short error_msg">100 长度的字母、数
字、汉字和下划线的组合</div>
                </div>
                <div class="button_info clearfix">
                    <input  type="button"  value=" 保 存 "  class="btn_save"
onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
    </body>
  </html>
```

3. 资费修改

- **问题**

增加资费的修改功能，用于修改一条资费数据。本案例中只要求打开修改页面，并将要修改的内容默认显示在修改页面表单的文本框中即可。

- **方案**

打开修改页面逻辑与打开新增页面一致，但同时要查询出要修改的数据并发送给修改页面，在修改页面上将这些数据默认显示在表单上。

对于数据的默认显示，我们可以使用 Struts2 提供的 UI 标签来做，UI 标签的作用是可以生产框体同时设置其默认值。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：在 DAO 中追加根据 ID 查询资费的方法

在 ICostDao 中追加根据 ID 查询资费的方法，代码如下：

```
package com.netctoss.dao;
```

```
import java.util.List;
import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {
// 此处略去其他方法的声明...

 /**
  * 根据主键查询资费
  * @param id 主键
  * @return
  */
 Cost findById(int id);

}
```

在 CostDaoImpl 中实现该方法，代码如下：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;
import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {
 // 此处略去其他方法的实现...

 @Override
 public Cost findById(int id) {
     // 模拟根据 id 查询资费数据
     Cost c = new Cost();
     c.setId(97);
     c.setName("tarena");
     c.setBaseDuration(99);
     c.setBaseCost(9.9);
     c.setUnitCost(0.9);
     c.setDescr("tarena 套餐");
     c.setStatus("0");
     c.setCostType("2");
     return c;
 }

}
```

步骤二：创建打开修改页面的 Action

在 com.netctoss.action 包下，创建打开修改页面的类 ToUpdateCostAction，在该 Action 下定义输入属性为资费 id，输出属性为资费实体对象，并在业务方法中调用 DAO 根据 id 查询出资费的实体对象。代码如下：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;

public class ToUpdateCostAction {
 // input
 private int id;
 // output
 private Cost cost;

 public String execute() {
     ICostDao dao = DAOFactory.getCostDAO();
     try {
         cost = dao.findById(id);
     } catch (Exception e) {
         e.printStackTrace();
         return "error";
     }
     return "success";
 }

 public int getId() {
     return id;
 }

 public void setId(int id) {
     this.id = id;
 }

 public Cost getCost() {
     return cost;
 }

 public void setCost(Cost cost) {
     this.cost = cost;
 }

}
```

步骤三：在 struts.xml 中配置该 action

在 struts.xml 中，配置打开修改页面的 action，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>
 <!--
     资费模块配置信息：
     一般情况下，一个模块的配置单独封装在一个 package 下，
     并且以模块名来命名 package 的 name 和 namespace。
  -->
 <package name="cost" namespace="/cost" extends="json-default">
     <!—此处略去其他 Action 的配置... -->


     <!--  打开修改页面 -->
     <action name="toUpdateCost"
         class="com.netctoss.action.ToUpdateCostAction">
         <result name="success">
```

```
                /WEB-INF/cost/update_cost.jsp
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>

    </package>
    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!—此处略去其他 Action 的配置... -->
    </package>
   </struts>
```

步骤四：创建资费修改页面

在 WEB-INF/cost 下，创建资费修改页面 update_cost.jsp，并将静态页面代码复制到此 JSP 中。代码如下：

```
    <%@page pageEncoding="utf-8"%>
    <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>达内 - NetCTOSS</title>
            <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
            <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global_color.css" />
            <script language="javascript" type="text/javascript">
                //保存结果的提示
                function showResult() {
                    showResultDiv(true);
                    window.setTimeout("showResultDiv(false);", 3000);
                }
                function showResultDiv(flag) {
                    var divResult = document.getElementById("save result info");
                    if (flag)
                        divResult.style.display = "block";
                    else
                        divResult.style.display = "none";
                }

                //切换资费类型
                function feeTypeChange(type) {
                    var                        inputArray                        =
document.getElementById("main").getElementsByTagName("input");
                    if (type == 1) {
                        inputArray[5].readonly = true;
                        inputArray[5].value = "";
                        inputArray[5].className += " readonly";
                        inputArray[6].readonly = false;
                        inputArray[6].className = "width100";
                        inputArray[7].readonly = true;
                        inputArray[7].className += " readonly";
                        inputArray[7].value = "";
                    }
                    else if (type == 2) {
                        inputArray[5].readonly = false;
                        inputArray[5].className = "width100";
                        inputArray[6].readonly = false;
```

```
                    inputArray[6].className = "width100";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
                else if (type == 3) {
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = true;
                    inputArray[6].value = "";
                    inputArray[6].className += " readonly";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role off"></a></li>
                <li><a                        href="../admin/admin list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee list.html" class="fee on"></a></li>
                <li><a                     href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                     href="../service/service_list.html"
class="service off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill off"></a></li>
                <li><a                       href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user_info.html"
class="information off"></a></li>
                <li><a                       href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save_result_info" class="save_success">保存成功！</div>
            <form action="" method="" class="main_form">
                <div class="text_info clearfix"><span>资费 ID：</span></div>
                <div class="input_info"><input type="text" class="readonly"
readonly value="1" /></div>
                <div class="text_info clearfix"><span>资费名称：</span></div>
                <div class="input_info">
                    <input type="text" class="width300" value="包 20 小时"/>
                    <span class="required">*</span>
                    <div class="validate_msg_short">50 长度的字母、数字、汉字和下划
线的组合</div>
                </div>
```

```
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">
                    <input   type="radio"   name="radFeeType"   id="monthly"
onclick="feeTypeChange(1);" />
                    <label for="monthly">包月</label>
                    <input  type="radio"  name="radFeeType"  checked="checked"
id="package" onclick="feeTypeChange(2);" />
                    <label for="package">套餐</label>
                    <input   type="radio"   name="radFeeType"   id="timeBased"
onclick="feeTypeChange(3);" />
                    <label for="timeBased">计时</label>
                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>单位费用：</span></div>
                <div class="input_info">
                    <input type="text" value="" class="width100" />
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info_high">
                    <textarea class="width300 height70">没有启用的资费，可以修改除
ID 以外的所有信息
                    </textarea>
                    <div class="validate_msg_short">100 长度的字母、数字、汉字和下
划线的组合</div>
                </div>
                <div class="button info clearfix">
                    <input   type="button"   value=" 保 存 "   class="btn_save"
onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
```

```
    </body>
</html>
```

步骤五：设置修改按钮 URL

在 find_cost.jsp 上，设置修改按钮的请求路径为打开修改页面的 action，并传入参数 id。代码如下：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>达内 - NetCTOSS</title>
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
        <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
        <script language="javascript" type="text/javascript">
            //排序按钮的点击事件
            function sort(btnObj) {
                if (btnObj.className == "sort_desc")
                    btnObj.className = "sort_asc";
                else
                    btnObj.className = "sort_desc";
            }

            //启用
            function startFee() {
                var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
除。");
                document.getElementById("operate_result_info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费 action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
```

```
                <li><a                      href="../admin/admin list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                  href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                  href="../service/service list.html"
class="service off"></a></li>
                <li><a                      href="../bill/bill list.html"
class="bill_off"></a></li>
                <li><a                    href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                      href="../user/user info.html"
class="information off"></a></li>
                <li><a                    href="../user/user modi pwd.html"
class="password off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search_add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input type="button" value="基费" class="sort_asc"
onclick="sort(this);" />
                        <input type="button" value="时长" class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input type="button" value="增加" class="btn_add"
onclick="location.href='toAddCost';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate result info" class="operate success">
                    <img                        src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
                    删除成功！
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>
                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
```

```
                            <td><a          href="fee detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                               <input    type="button"    value="   启   用   "
class="btn_start" onclick="startFee();" />

                               <input type="button" value="修改" class="btn_modify"
onclick="location.href='toUpdateCost?id=<s:property value="id"/>';" />

                               <input    type="button"    value="   删   除   "
class="btn_delete" onclick="deleteFee(<s:property value="id"/>);" />
                            </td>
                        </tr>
                      </s:iterator>
                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                 </div>
                 <!--分页-->
                 <div id="pages">
                   <a href="#">上一页</a>
                     <a href="#" class="current_page">1</a>
                     <a href="#">2</a>
                     <a href="#">3</a>
                     <a href="#">4</a>
                     <a href="#">5</a>
                     <a href="#">下一页</a>
                 </div>
              </form>
           </div>
           <!--主要区域结束-->
           <div id="footer">
              <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
              <p>版权所有(C)加拿大达内 IT 培训集团公司 </p>
           </div>
        </body>
    </html>
```

步骤六：阶段测试

重新部署项目并重启 tomcat，打开资费列表页面，点击任意的修改按钮，效果如下图：

图-5

步骤七：在修改页面上使用 UI 标签回显修改数据

在 update_cost.jsp 上，使用 UI 标签生成框体并给框体设置默认值。代码如下：

```
    <%@page pageEncoding="utf-8"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>达内 - NetCTOSS</title>
            <link type="text/css" rel="stylesheet" media="all"
href="../styles/global.css" />
            <link type="text/css" rel="stylesheet" media="all"
href="../styles/global_color.css" />
            <script language="javascript" type="text/javascript">
                //保存结果的提示
                function showResult() {
                    showResultDiv(true);
                    window.setTimeout("showResultDiv(false);", 3000);
                }
                function showResultDiv(flag) {
                    var divResult = document.getElementById("save_result_info");
                    if (flag)
                        divResult.style.display = "block";
                    else
                        divResult.style.display = "none";
                }

                //切换资费类型
                function feeTypeChange(type) {
                    var inputArray =
document.getElementById("main").getElementsByTagName("input");
                    if (type == 1) {
```

```
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                    inputArray[7].readonly = true;
                    inputArray[7].className += " readonly";
                    inputArray[7].value = "";
                }
                else if (type == 2) {
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
                else if (type == 3) {
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = true;
                    inputArray[6].value = "";
                    inputArray[6].className += " readonly";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
            }
        </script>
    </head>

    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role list.html"
class="role off"></a></li>
                <li><a                        href="../admin/admin list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account off"></a></li>
                <li><a                    href="../service/service list.html"
class="service_off"></a></li>
                <li><a                          href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report list.html"
class="report off"></a></li>
                <li><a                          href="../user/user info.html"
class="information_off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
```

```
            <div id="save_result_info" class="save_success">保存成功！</div>
            <form action="" method="" class="main_form">
                <div class="text_info clearfix"><span>资费 ID：</span></div>
                <div class="input_info">

                    <s:textfield        name="cost.id"        cssClass="readonly"
readonly="true"/>

                </div>
                <div class="text_info clearfix"><span>资费名称：</span></div>
                <div class="input_info">

                    <s:textfield name="cost.name" cssClass="width300" />

                    <span class="required">*</span>
                    <div class="validate_msg_short">50 长度的字母、数字、汉字和下划
线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">

                    <s:radio  name="cost.costType"  list="#{'1':'包月','2':'套餐
','3':'计时' }" onclick="feeTypeChange(this.value)"/>

                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input_info">

                    <s:textfield name="cost.baseDuration" cssClass="width100"/>

                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input_info">

                    <s:textfield name="cost.baseCost" cssClass="width100"/>

                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>单位费用：</span></div>
                <div class="input_info">

                    <s:textfield name="cost.unitCost" cssClass="width100"/>

                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">0-99999.99 之间的数值</div>
                </div>
```

```
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input_info_high">

                  <s:textarea name="cost.descr" cssClass="width300 height70"/>

                   <div class="validate_msg_short">100 长度的字母、数字、汉字和下
划线的组合</div>
                </div>
                <div class="button_info clearfix">
                   <input  type="button"  value=" 保 存 "  class="btn_save"
onclick="showResult();" />
                   <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
        </div>
        <!--主要区域结束-->
        <div id="footer">
            <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
            <br />
            <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
        </div>
    </body>
  </html>
```

步骤八：测试

重新访问资费列表页面并点击修改按钮，效果如下图：

图-6

- **完整代码**

本案例的 ICostDao 完整代码：

```
package com.netctoss.dao;

import java.util.List;
import com.netctoss.entity.Cost;

/**
 * 资费 DAO 接口
 */
public interface ICostDao {

 /**
  * 查询全部资费数据
  */
 List<Cost> findAll();

 /**
  * 删除一条资费数据
  * @param id 主键
  */
 void delete(int id);

 /**
  * 根据名称查询资费
  * @param name 资费名
  * @return
  */
 Cost findByName(String name);

 /**
  * 根据主键查询资费
  * @param id 主键
  * @return
  */
 Cost findById(int id);

}
```

CostDaoImpl 完整代码：

```
package com.netctoss.dao;

import java.util.ArrayList;
import java.util.List;
import com.netctoss.entity.Cost;

/**
 *   当前阶段学习重点是 Struts2，对于 DAO 的实现就模拟实现了。
 *   同学们可以使用 JDBC/MyBatis 自行实现该 DAO。
 */
public class CostDaoImpl implements ICostDao {

 @Override
 public List<Cost> findAll() {
     // 模拟查询全部资费数据
     List<Cost> list = new ArrayList<Cost>();

     Cost c1 = new Cost();
     c1.setId(95);
```

```
        c1.setName("6元套餐");
        c1.setBaseDuration(66);
        c1.setBaseCost(6.6);
        c1.setUnitCost(0.6);
        c1.setDescr("6元套餐");
        c1.setStatus("0");
        c1.setCostType("2");
        list.add(c1);

        Cost c2 = new Cost();
        c2.setId(96);
        c2.setName("8元套餐");
        c2.setBaseDuration(88);
        c2.setBaseCost(8.8);
        c2.setUnitCost(0.8);
        c2.setDescr("8元套餐");
        c2.setStatus("0");
        c2.setCostType("2");
        list.add(c2);

        Cost c3 = new Cost();
        c3.setId(97);
        c3.setName("tarena");
        c3.setBaseDuration(99);
        c3.setBaseCost(9.9);
        c3.setUnitCost(0.9);
        c3.setDescr("tarena套餐");
        c3.setStatus("0");
        c3.setCostType("2");
        list.add(c3);

        return list;
    }

    @Override
    public void delete(int id) {
        // 模拟根据id删除资费数据
        System.out.println("删除ID为[" + id + "]的资费数据.");
    }

    @Override
    public Cost findByName(String name) {
        // 模拟根据名称查询资费数据，假设资费表中只有一条名为tarena的数据
        if("tarena".equals(name)) {
            Cost c = new Cost();
            c.setId(97);
            c.setName("tarena");
            c.setBaseDuration(99);
            c.setBaseCost(9.9);
            c.setUnitCost(0.9);
            c.setDescr("tarena套餐");
            c.setStatus("0");
            c.setCostType("2");
            return c;
        }
        return null;
    }

    @Override
    public Cost findById(int id) {
        // 模拟根据id查询资费数据
        Cost c = new Cost();
        c.setId(97);
        c.setName("tarena");
        c.setBaseDuration(99);
        c.setBaseCost(9.9);
```

```
        c.setUnitCost(0.9);
        c.setDescr("tarena 套餐");
        c.setStatus("0");
        c.setCostType("2");
        return c;
    }

}
```

ToUpdateCostAction 完整代码：

```
package com.netctoss.action;

import com.netctoss.dao.DAOFactory;
import com.netctoss.dao.ICostDao;
import com.netctoss.entity.Cost;

public class ToUpdateCostAction {

    // input
    private int id;

    // output
    private Cost cost;

    public String execute() {
        ICostDao dao = DAOFactory.getCostDAO();
        try {
            cost = dao.findById(id);
        } catch (Exception e) {
            e.printStackTrace();
            return "error";
        }
        return "success";
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Cost getCost() {
        return cost;
    }

    public void setCost(Cost cost) {
        this.cost = cost;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

    <!--
        资费模块配置信息:
```

```
        一般情况下，一个模块的配置单独封装在一个 package 下，
        并且以模块名来命名 package 的 name 和 namespace。
    -->
    <package name="cost" namespace="/cost" extends="json-default">
        <!-- 查询资费数据 -->
        <action name="findCost" class="com.netctoss.action.FindCostAction">
            <!--
                正常情况下跳转到资费列表页面。
                一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
             -->
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 删除资费 -->
        <action name="deleteCost"
            class="com.netctoss.action.DeleteCostAction">
            <!-- 删除完之后，重定向到查询 action -->
            <result name="success" type="redirectAction">
                findCost
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 打开资费新增页 -->
        <action name="toAddCost">
            <result name="success">
                /WEB-INF/cost/add_cost.jsp
            </result>
        </action>
        <!-- 资费名唯一性校验 -->
        <action name="checkCostName"
            class="com.netctoss.action.CheckCostNameAction">
            <!-- 使用 json 类型的 result 把结果输出给回调函数 -->
            <result name="success" type="json">
                <param name="root">info</param>
            </result>
        </action>
        <!--  打开修改页面 -->
        <action name="toUpdateCost"
            class="com.netctoss.action.ToUpdateCostAction">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
```

```
                该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
                会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 生成验证码 -->
        <action                                          name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!-- 使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!-- 指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>

    </struts>
```

find_cost.jsp 完整代码：

```
    <%@page pageEncoding="utf-8" isELIgnored="false"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>达内－NetCTOSS</title>
            <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
            <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global_color.css" />
            <script language="javascript" type="text/javascript">
                //排序按钮的点击事件
                function sort(btnObj) {
                    if (btnObj.className == "sort desc")
                        btnObj.className = "sort asc";
                    else
                        btnObj.className = "sort desc";
                }

                //启用
                function startFee() {
                    var r = window.confirm("确定要启用此资费吗？资费启用后将不能修改和删
```

```
除。");
                document.getElementById("operate result info").style.display
= "block";
            }
            //删除
            function deleteFee(id) {
                var r = window.confirm("确定要删除此资费吗？");
                if(r) {
                    // 如果用户确认，则调用删除资费action
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
        <!--Logo区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index_off"></a></li>
                <li><a                          href="../role/role_list.html"
class="role_off"></a></li>
                <li><a                        href="../admin/admin list.html"
class="admin off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account_off"></a></li>
                <li><a                    href="../service/service list.html"
class="service off"></a></li>
                <li><a                          href="../bill/bill list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report_list.html"
class="report_off"></a></li>
                <li><a                          href="../user/user info.html"
class="information off"></a></li>
                <li><a                      href="../user/user modi pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <form action="" method="">
                <!--排序-->
                <div class="search add">
                    <div>
                        <!--<input type="button" value="月租" class="sort_asc"
onclick="sort(this);" />-->
                        <input  type="button"  value="基费"  class="sort_asc"
onclick="sort(this);" />
                        <input  type="button"  value="时长"  class="sort_asc"
onclick="sort(this);" />
                    </div>
                    <input  type="button"  value="增加"  class="btn_add"
onclick="location.href='toAddCost';" />
                </div>
                <!--启用操作的操作提示-->
                <div id="operate_result_info" class="operate_success">
                    <img                          src="../images/close.png"
onclick="this.parentNode.style.display='none';" />
```

```
                    删除成功!
                </div>
                <!--数据区域：用表格展示数据-->
                <div id="data">
                    <table id="datalist">
                        <tr>
                            <th>资费 ID</th>
                            <th class="width100">资费名称</th>
                            <th>基本时长</th>
                            <th>基本费用</th>
                            <th>单位费用</th>
                            <th>创建时间</th>
                            <th>开通时间</th>
                            <th class="width50">状态</th>
                            <th class="width200"></th>
                        </tr>

                        <!-- 使用 Struts2 标签遍历集合，使用 OGNL 表达式输出内容。 -->
                        <s:iterator value="costs">
                         <tr>
                            <td><s:property value="id"/></td>
                            <td><a          href="fee detail.html"><s:property
value="name"/></a></td>
                            <td><s:property value="baseDuration"/></td>
                            <td><s:property value="baseCost"/></td>
                            <td><s:property value="unitCost"/></td>
                            <td><s:property value="createTime"/></td>
                            <td><s:property value="startTime"/></td>
                            <td>
                                <s:if test="status==0">开通</s:if>
                                <s:else>暂停</s:else>
                            </td>
                            <td>
                                <input    type="button"    value="   启   用   "
class="btn_start" onclick="startFee();" />
                                <input    type="button"    value="   修   改   "
class="btn modify"        onclick="location.href='toUpdateCost?id=<s:property
value="id"/>';" />
                                <input    type="button"    value="   删   除   "
class="btn delete" onclick="deleteFee(<s:property value="id"/>);" />
                            </td>
                         </tr>
                       </s:iterator>

                    </table>
                    <p>业务说明：<br />
                    1、创建资费时，状态为暂停，记载创建时间；<br />
                    2、暂停状态下，可修改，可删除；<br />
                    3、开通后，记载开通时间，且开通后不能修改、不能再停用、也不能删除；
<br />
                    4、业务账号修改资费时，在下月底统一触发，修改其关联的资费 ID（此触发
动作由程序处理）
                    </p>
                </div>
                <!--分页-->
                <div id="pages">
                   <!--
                       如果当前页是第一页，则不允许再点上一页。
                       否则点击上一页时访问 page-1 页。
                   -->
                   <s:if test="page==1">
                       <a href="#">上一页</a>
                   </s:if>
                   <s:else>
```

```
                        <a href="findCost?page=<s:property value="page-1"/>">
上一页</a>
                    </s:else>

                    <!-- 循环输出页码 -->
                    <s:iterator begin="1" end="totalPage" var="p">
                     <!--
                         1.如果循环页码等于当前页，则给与当前页样式current_page
                         2.链接findCost是相对路径，相对于当前浏览器地址栏的路径，
                             如果findCost的路径与地址栏路径的namespace相同，
                             则可以将namespace及之前的部分省略。
                      -->
                     <s:if test="#p==page">
                         <a href="findCost?page=<s:property value="#p"/>"
                                 class="current page"><s:property
value="#p"/></a>
                     </s:if>
                     <s:else>
                         <a                    href="findCost?page=<s:property
value="#p"/>"><s:property value="#p"/></a>
                     </s:else>
                      </s:iterator>

                      <!--
                         如果当前页是最后一页，则不允许再点下一页。
                         否则点击下一页时访问page+1页。
                     -->
                     <s:if test="page==totalPage">
                         <a href="#">下一页</a>
                     </s:if>
                     <s:else>
                         <a href="findCost?page=<s:property value="page+1"/>">
下一页</a>
                     </s:else>
                 </div>
             </form>
         </div>
         <!--主要区域结束-->
         <div id="footer">
             <p>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]</p>
             <p>版权所有(C)加拿大达内IT培训集团公司 </p>
         </div>
     </body>
 </html>
```

update_cost.jsp 完整代码：

```
 <%@page pageEncoding="utf-8"%>
 <%@taglib uri="/struts-tags" prefix="s"%>
 <!DOCTYPE   html   PUBLIC   "-//W3C//DTD   XHTML   1.0   Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml">
     <head>
         <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
         <title>达内－NetCTOSS</title>
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global.css" />
         <link       type="text/css"       rel="stylesheet"       media="all"
href="../styles/global color.css" />
         <script language="javascript" type="text/javascript">
             //保存结果的提示
             function showResult() {
                 showResultDiv(true);
                 window.setTimeout("showResultDiv(false);", 3000);
```

```
            }
            function showResultDiv(flag) {
                var divResult = document.getElementById("save_result_info");
                if (flag)
                    divResult.style.display = "block";
                else
                    divResult.style.display = "none";
            }

            //切换资费类型
            function feeTypeChange(type) {
                var                        inputArray                        =
document.getElementById("main").getElementsByTagName("input");
                if (type == 1) {
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                    inputArray[7].readonly = true;
                    inputArray[7].className += " readonly";
                    inputArray[7].value = "";
                }
                else if (type == 2) {
                    inputArray[5].readonly = false;
                    inputArray[5].className = "width100";
                    inputArray[6].readonly = false;
                    inputArray[6].className = "width100";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
                else if (type == 3) {
                    inputArray[5].readonly = true;
                    inputArray[5].value = "";
                    inputArray[5].className += " readonly";
                    inputArray[6].readonly = true;
                    inputArray[6].value = "";
                    inputArray[6].className += " readonly";
                    inputArray[7].readonly = false;
                    inputArray[7].className = "width100";
                }
            }
        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            <img src="../images/logo.png" alt="logo" class="left"/>
            <a href="#">[退出]</a>
        </div>
        <!--Logo 区域结束-->
        <!--导航区域开始-->
        <div id="navi">
            <ul id="menu">
                <li><a href="../index.html" class="index off"></a></li>
                <li><a                        href="../role/role list.html"
class="role_off"></a></li>
                <li><a                      href="../admin/admin list.html"
class="admin_off"></a></li>
                <li><a href="../fee/fee_list.html" class="fee_on"></a></li>
                <li><a                    href="../account/account_list.html"
class="account off"></a></li>
                <li><a                    href="../service/service list.html"
class="service_off"></a></li>
                <li><a                        href="../bill/bill_list.html"
class="bill_off"></a></li>
                <li><a                      href="../report/report_list.html"
```

```
class="report off"></a></li>
                <li><a                        href="../user/user info.html"
class="information_off"></a></li>
                <li><a                     href="../user/user_modi_pwd.html"
class="password_off"></a></li>
            </ul>
        </div>
        <!--导航区域结束-->
        <!--主要区域开始-->
        <div id="main">
            <div id="save result info" class="save success">保存成功！</div>
            <form action="" method="" class="main form">
                <div class="text info clearfix"><span>资费 ID：</span></div>
                <div class="input_info">
                  <s:textfield      name="cost.id"      cssClass="readonly"
readonly="true"/>
                </div>
                <div class="text info clearfix"><span>资费名称：</span></div>
                <div class="input info">
                    <s:textfield name="cost.name" cssClass="width300" />
                    <span class="required">*</span>
                    <div class="validate_msg_short">50 长度的字母、数字、汉字和下划
线的组合</div>
                </div>
                <div class="text_info clearfix"><span>资费类型：</span></div>
                <div class="input_info fee_type">
                    <s:radio name="cost.costType" list="#{'1':'包月','2':'套餐
','3':'计时' }" onclick="feeTypeChange(this.value)"/>
                </div>
                <div class="text_info clearfix"><span>基本时长：</span></div>
                <div class="input_info">
                    <s:textfield                    name="cost.baseDuration"
cssClass="width100"/>
                    <span class="info">小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">1-600 之间的整数</div>
                </div>
                <div class="text_info clearfix"><span>基本费用：</span></div>
                <div class="input info">
                    <s:textfield name="cost.baseCost" cssClass="width100"/>
                    <span class="info">元</span>
                    <span class="required">*</span>
                    <div class="validate msg long">0-99999.99 之间的数值</div>
                </div>
                <div class="text info clearfix"><span>单位费用：</span></div>
                <div class="input_info">
                    <s:textfield name="cost.unitCost" cssClass="width100"/>
                    <span class="info">元/小时</span>
                    <span class="required">*</span>
                    <div class="validate_msg_long">0-99999.99 之间的数值</div>
                </div>
                <div class="text_info clearfix"><span>资费说明：</span></div>
                <div class="input info high">
                  <s:textarea      name="cost.descr"      cssClass="width300
height70"/>
                    <div class="validate msg short">100 长度的字母、数字、汉字和下
划线的组合</div>
                </div>
                <div class="button_info clearfix">
                    <input  type="button"  value=" 保 存 "  class="btn_save"
onclick="showResult();" />
                    <input type="button" value="取消" class="btn_save" />
                </div>
            </form>
```

```
            </div>
            <!--主要区域结束-->
            <div id="footer">
                <span>[源自北美的技术，最优秀的师资，最真实的企业环境，最适用的实战项
目]</span>
                <br />
                <span>版权所有(C)加拿大达内 IT 培训集团公司 </span>
            </div>
        </body>
    </html>
```

4. 使用简单的 UI 标签

- **问题**

练习使用简单的 UI 标签生成框体，并给框体赋默认值，标签包含：

1) 表单标签
2) 提交按钮
3) 文本框
4) 密码框
5) 文本域
6) 布尔框

- **方案**

UI 标签最核心的功能是用于回显数据，因此往往被使用在修改功能中。为了贴合实际业务，我们模拟修改客户的场景，在修改客户的页面上使用上述标签来回显查询出的客户数据。

由于本案例侧重点在 UI 标签的使用，因此是模拟修改客户的场景，简化打开修改页面的代码即可，这里我们不需要设计数据库以及查询数据库，仅仅是模拟一份数据即可。

为了保证能够在模拟的场景中使用到每一个 UI 标签，因此客户实体类中要包含每一类的数据，这样的模拟并不符合真实的业务，而仅仅是为了练习标签做的假设，请同学们注意这一点。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建项目

先创建项目 StrutsDay04，然后导入 Struts2 核心包，接下来在 web.xml 中配置前端控制器，最后在 src 下创建 struts.xml，即 StrutsDay01 中的 Struts2 使用步骤。

步骤二：准备项目

模拟出客户修改的功能，先建包 entity，在包下创建客户实体类 Customer，代码如下：

```
package entity;

import java.util.List;

public class Customer {

 private String name; // 姓名
 private String password; // 密码
 private String desc; // 简介
 private boolean marry; // 是否已婚

 private String sex; // 性别
 private List<String> travelCities; // 去过的城市
 private String home; // 家乡

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

 public String getPassword() {
     return password;
 }

 public void setPassword(String password) {
     this.password = password;
 }

 public String getDesc() {
     return desc;
 }

 public void setDesc(String desc) {
     this.desc = desc;
 }

 public String getSex() {
     return sex;
 }

 public void setSex(String sex) {
     this.sex = sex;
 }

 public boolean isMarry() {
     return marry;
 }

 public void setMarry(boolean marry) {
     this.marry = marry;
 }

 public String getHome() {
     return home;
 }

 public List<String> getTravelCities() {
     return travelCities;
 }

 public void setTravelCities(List<String> travelCities) {
```

```
        this.travelCities = travelCities;
    }

    public void setHome(String home) {
        this.home = home;
    }

}
```

然后创建客户 DAO，并模拟根据 id 查询客户的方法，代码如下：

```
package dao;

import java.util.ArrayList;
import java.util.List;
import entity.Customer;

/**
 * 模拟客户 DAO
 */
public class CustomerDAO {

 /**
  * 模拟根据 ID 查询客户
  */
 public Customer findById() {
     // 实例化客户实体类
     Customer c = new Customer();
     // 设置一些默认值
     c.setName("Tarena");
     c.setPassword("123");
     c.setDesc("中华人民共和国公民");
     c.setSex("F");
     c.setMarry(true);

     List<String> list = new ArrayList<String>();
     list.add("beijing");
     list.add("guangzhou");
     c.setTravelCities(list);

     c.setHome("shanghai");
     return c;
 }

}
```

接下来，创建打开修改页面 Action 类 ToUpdateCustomerAction，在此 Action 中模拟处理打开修改页面，查询客户的请求，代码如下：

```
package action;

import dao.CustomerDAO;
import entity.Customer;

public class ToUpdateCustomerAction {

 // output
 private Customer customer;

 public String execute() {
     CustomerDAO dao = new CustomerDAO();
```

```
        // 模拟查询客户
        customer = dao.findById();
        return "success";
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

}
```

接下来，在 struts.xml 中配置打开修改页面 action，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

    <!-- 客户配置信息 -->
    <package name="customer"
        namespace="/customer" extends="struts-default">
        <!-- 打开修改页面 -->
        <action name="toUpdateCustomer"
            class="action.ToUpdateCustomerAction">
            <result name="success">
                /WEB-INF/customer/update_customer.jsp
            </result>
        </action>
    </package>

</struts>
```

最后，在 WEB-INF 下创建 customer 文件夹，并在此文件夹下创建修改客户的页面 update_customer.jsp，该页面简化处理，先输出一句话“模拟修改客户”即可，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
    <h1>模拟修改客户</h1>

</body>
</html>
```

完成上述模拟修改客户的代码后，项目结构如下图：

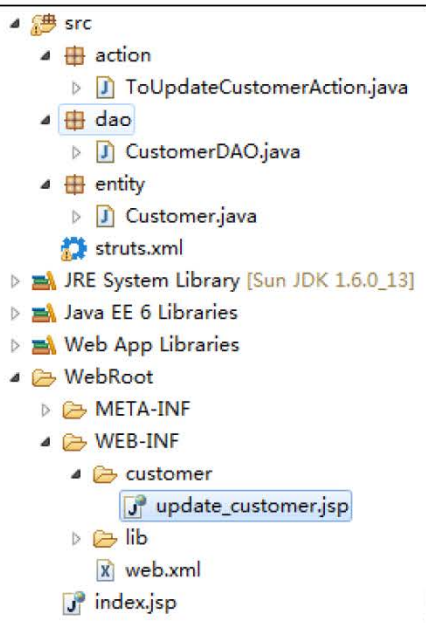


图-7

部署项目并启动 tomcat，在浏览器中访问修改客户，效果如下图：

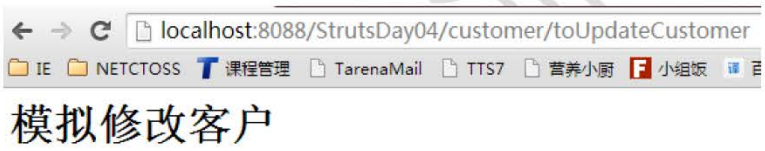


图-8

步骤三：表单标签

在 update_customer.jsp 上，使用表单标签生成 form 表单，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>


 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">

 </s:form>


</body>
</html>
```

刷新浏览器，修改页面效果如下：

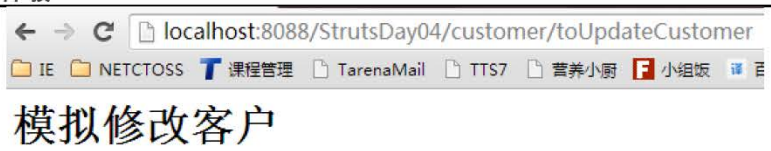


图-9

步骤四：文本框标签

在 update_customer.jsp 上，使用文本框标签生成文本框，并在框内显示客户名称，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">

     <!--
         2.文本框标签
          1）生成文本框
          2）根据 OGNL（customer.name）取值给文本框赋值
     -->
     <s:textfield name="customer.name" label="姓名"/>

 </s:form>

</body>
</html>
```

刷新浏览器，修改页面效果如下：

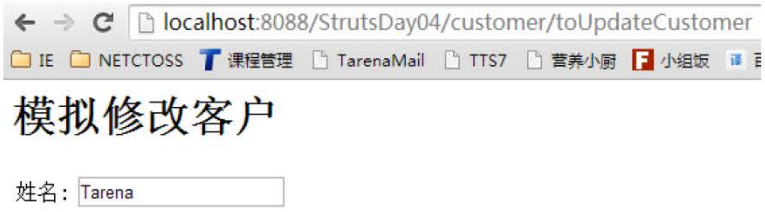


图-10

步骤五：密码框标签

在 update_customer.jsp 上，使用密码框标签生成密码框，并在框内显示客户密码，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
```

```
<head></head>
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">
     <!--此处略去其他标签...-->

     <!--
         3.密码框标签
          1）生成密码框
          2）根据 OGNL（customer.password）取值给密码框赋值
     -->
     <s:password name="customer.password" label="密码" showPassword="true"/>

 </s:form>

</body>
</html>
```

刷新浏览器，修改页面效果如下：

图-11

步骤六：文本域标签

在 update_customer.jsp 上，使用文本域标签生成文本域，并在框内显示客户简介，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">
     <!--此处略去其他标签...-->

     <!--
         4.文本域标签
          1）生成文本域
          2）根据 OGNL（customer.desc）取值给文本域赋值
     -->
```

```
        <s:textarea name="customer.desc" cols="30" rows="5" label="简介"/>

    </s:form>

   </body>
   </html>
```

刷新浏览器，修改页面效果如下：

图-12

步骤七：布尔框标签

在 update_customer.jsp 上，使用布尔框标签生成选择框，并根据客户是否已婚来决定是否勾选该选择框，代码如下：

```
   <%@page pageEncoding="utf-8"%>
   <%@taglib uri="/struts-tags" prefix="s"%>
   <html>
   <head></head>
   <body>
    <h1>模拟修改客户</h1>

    <%-- 1.表单标签 --%>
    <s:form action="#" method="post">
        <!--此处略去其他标签...-->

        <!--
            5.布尔标签
             1）生成一个 checkbox
             2）根据 OGNL（customer.marry）取值（布尔型）来确定是否勾选
        -->
        <s:checkbox name="customer.marry" label="是否已婚" labelposition="left"/>

    </s:form>

   </body>
   </html>
```

刷新浏览器，修改页面效果如下：

图-13

- **完整代码**

本案例的 Customer 完整代码：

```
package entity;

import java.util.List;

public class Customer {

 private String name; // 姓名
 private String password; // 密码
 private String desc; // 简介
 private boolean marry; // 是否已婚

 private String sex; // 性别
 private List<String> travelCities; // 去过的城市
 private String home; // 家乡

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

 public String getPassword() {
     return password;
 }

 public void setPassword(String password) {
     this.password = password;
 }

 public String getDesc() {
     return desc;
 }

 public void setDesc(String desc) {
     this.desc = desc;
 }

 public String getSex() {
     return sex;
 }
```

```
    public void setSex(String sex) {
        this.sex = sex;
    }

    public boolean isMarry() {
        return marry;
    }

    public void setMarry(boolean marry) {
        this.marry = marry;
    }

    public String getHome() {
        return home;
    }

    public List<String> getTravelCities() {
        return travelCities;
    }

    public void setTravelCities(List<String> travelCities) {
        this.travelCities = travelCities;
    }

    public void setHome(String home) {
        this.home = home;
    }

}
```

CustomerDAO 完整代码：

```
package dao;

import java.util.ArrayList;
import java.util.List;
import entity.Customer;

/**
 * 模拟客户 DAO
 */
public class CustomerDAO {

 /**
  * 模拟根据 ID 查询客户
  */
 public Customer findById() {
     // 实例化客户实体类
     Customer c = new Customer();
     // 设置一些默认值
     c.setName("Tarena");
     c.setPassword("123");
     c.setDesc("中华人民共和国公民");
     c.setSex("F");
     c.setMarry(true);

     List<String> list = new ArrayList<String>();
     list.add("beijing");
     list.add("guangzhou");
     c.setTravelCities(list);

     c.setHome("shanghai");
     return c;
 }
}
```

ToUpdateCustomerAction 完整代码：

```
package action;

import dao.CustomerDAO;
import entity.Customer;

public class ToUpdateCustomerAction {

 // output
 private Customer customer;

 public String execute() {
     CustomerDAO dao = new CustomerDAO();
     // 模拟查询客户
     customer = dao.findById();
     return "success";
 }

 public Customer getCustomer() {
     return customer;
 }

 public void setCustomer(Customer customer) {
     this.customer = customer;
 }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!-- 客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">
     <!-- 打开修改页面 -->
     <action name="toUpdateCustomer"
         class="action.ToUpdateCustomerAction">
         <result name="success">
             /WEB-INF/customer/update_customer.jsp
         </result>
     </action>
 </package>

</struts>
```

update_customer.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">
```

```
        <!--
            2.文本框标签
             1）生成文本框
             2）根据 OGNL（customer.name）取值给文本框赋值
        -->
        <s:textfield name="customer.name" label="姓名"/>
        <!--
            3.密码框标签
             1）生成密码框
             2）根据 OGNL（customer.password）取值给密码框赋值
        -->
        <s:password name="customer.password" label="密码" showPassword="true"/>
        <!--
            4.文本域标签
             1）生成文本域
             2）根据 OGNL（customer.desc）取值给文本域赋值
        -->
        <s:textarea name="customer.desc" cols="30" rows="5" label="简介"/>
        <!--
            5.布尔标签
             1）生成一个 checkbox
             2）根据 OGNL（customer.marry）取值（布尔型）来确定是否勾选
        -->
        <s:checkbox name="customer.marry" label="是否已婚" labelposition="left"/>
    </s:form>
   </body>
   </html>
```

5. 使用复杂的 UI 标签

- **问题**

练习使用简单的 UI 标签生成框体，并给框体赋默认值，标签包含：

7) 单选框标签
8) 多选框标签
9) 下拉选标签

- **方案**

在 StrutsDay04 案例的模拟修改客户功能基础上，模拟数据，练习使用这 3 类标签。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：单选框标签

单选框有 2 种用法，根据其初始化 radio 方式的不同，可以分为静态和动态 2 种方式。首先我们使用静态方式来初始化客户性别选项，并根据客户数据勾选客户性别，需要在 update_customer.jsp 上使用单选框标签来实现。代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">
     <!--此处略去其他标签...-->

     <!--
         6.1 单选框标签（静态）
          1）根据 OGNL（list 属性值）创建的 Map 生成一组 radio，
         其中 Map 的可以生成 radio 的 value 值，Map 的 value 生成 radio 的 label 值。
          2）根据 OGNL（customer.marry）取值来与生成 radio 的 value 比较，
             若与哪个 radio 的 value 值一致，则将其勾选。
     -->
     <s:radio name="customer.sex" list="#{'M':'男','F':'女' }" label="性别"/>

 </s:form>
</body>
</html>
```

刷新浏览器，修改页面效果如下：

图-14

静态方式初始化 radio 时是通过静态代码创建 Map 来实现的，很多时候这些选项不是固定的，是需要查询数据库得到的，即需要通过动态的方式来进行初始化选项，那么需要采用动态方式来实现。

要实现动态初始化选项，先要创建选项实体类，来封装选项信息，因此这里我们要创建性别的实体类 Sex，当然就这个场景而言并不合理，仅是出于练习的目的，大家了解这种用法即可。代码如下：

```
package entity;
public class Sex {
 private String code;
```

```
    private String name;

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

然后在 CustomerDAO 中追加查询全部性别的方法，这个方法是模拟的方法，并且不符合真实的业务场景，仅用于练习知识点。代码如下：

```
package dao;

import java.util.ArrayList;
import java.util.List;
import entity.Customer;
import entity.Sex;

/**
 * 模拟客户 DAO
 */
public class CustomerDAO {
//此处略去其他方法 ...


 /**
  * 模拟查询所有的性别，并不符合真实业务，仅仅是用于练习知识点
  */
 public List<Sex> findAllSex() {
     List<Sex> list = new ArrayList<Sex>();

     Sex s1 = new Sex();
     s1.setCode("M");
     s1.setName("男");
     list.add(s1);

     Sex s2 = new Sex();
     s2.setCode("F");
     s2.setName("女");
     list.add(s2);

     Sex s3 = new Sex();
     s3.setCode("O");
     s3.setName("其他");
     list.add(s3);

     return list;
 }


}
```

然后在 ToUpdateCustomerAction 中模拟查询所有的性别，代码如下：

```
package action;

import java.util.List;

import dao.CustomerDAO;
import entity.Customer;
import entity.Sex;

public class ToUpdateCustomerAction {

 // output
 private Customer customer; // 客户


 private List<Sex> sexes; // 性别


 public String execute() {
     CustomerDAO dao = new CustomerDAO();
     // 模拟查询客户
     customer = dao.findById();


     // 模拟查询全部的性别
     sexes = dao.findAllSex();


     return "success";
 }

 public Customer getCustomer() {
     return customer;
 }

 public void setCustomer(Customer customer) {
     this.customer = customer;
 }


 public List<Sex> getSexes() {
     return sexes;
 }

 public void setSexes(List<Sex> sexes) {
     this.sexes = sexes;
 }


}
```

最后，在 update_customer.jsp 上，使用动态的单选框标签，根据 Action 中返回的所有性别生成性别选项，并根据客户信息设置默认勾选，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>


 <%-- 1.表单标签 --%>
```

```
<s:form action="#" method="post">
    <!--此处略去其他标签...-->

    <!--
        6.2 单选框标签（动态）
         1）根据 OGNL（list 属性值）的取值初始化一组 radio，
            并根据 listKey 指定的实体类中的属性来生成 radio 的 value 值，
            根据 listValue 指定的实体类中的属性来生成 radio 的 label 值。
         2）根据 OGNL（customer.marry）取值来与生成 radio 的 value 比较，
            若与哪个 radio 的 value 值一致，则将其勾选。
    -->

    <s:radio name="customer.sex"
        list="sexes" listKey="code" listValue="name" label="性别"/>
</s:form>
</body>
</html>
```

重新部署项目并重启 tomcat，重新访问修改客户，效果如下图：

图-15

步骤二：多选框标签

多选框的用法与单选框用法类似，也分为静态初始化和动态初始化 2 种，区别在于对 checkbox 初始化方式的不同。

首先，我们通过静态初始化的方式生成一组城市的复选框，然后根据客户去过的城市来勾选这些城市。代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
<h1>模拟修改客户</h1>

<%-- 1.表单标签 --%>
<s:form action="#" method="post">
```

```
        <!--此处略去其他标签...-->
        <!--
            7.1 多选框标签（静态）
             1）根据 OGNL（list 属性值）创建的 Map 生成一组 checkbox，其中 Map 的 key
                生成 checkbox 的 value 值，Map 的 value 生成 checkbox 的 label 值。
             2）根据 OGNL（customer.travelCities）取值来与生成 checkbox 的
                value 比较，若与哪个 checkbox 的 value 值一致，则将其勾选。
        -->
        <s:checkboxlistname="customer.travelCities"
            list="#{'beijing':'北京','shanghai':'上海','guangzhou':'广州
','shenzhen':'深圳','chongqing':'重庆','diaoyudao':'钓鱼岛' }" label="旅游过的城市
"/>

    </s:form>

   </body>
   </html>
```

刷新浏览器，效果如下图：

图-16

接下来，我们在看看如何动态的初始化多选框的选项。先创建城市的实体类 City，用于封装所有的城市数据，代码如下：

```
package entity;

public class City {

 private String code;
 private String name;

 public City(String code, String name) {
     super();
     this.code = code;
     this.name = name;
 }

 public String getCode() {
     return code;
 }
```

```
    public void setCode(String code) {
        this.code = code;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

然后在 CustomerDAO 中模拟查询全部的城市，代码如下：

```
package dao;

import java.util.ArrayList;
import java.util.List;
import entity.City;
import entity.Customer;
import entity.Sex;

/**
 * 模拟客户 DAO
 */
public class CustomerDAO {

 // 此处略去其他方法...


 /**
  * 模拟查询所有的城市
  */
 public List<City> findAllCities() {
     List<City> cities = new ArrayList<City>();
     City c1 = new City("beijing", "北京");
     City c2 = new City("shanghai", "上海");
     City c3 = new City("guangzhou", "广州");
     City c4 = new City("shenzhen", "深圳");
     City c5 = new City("chongqing", "重庆");
     City c6 = new City("diaoyudao", "钓鱼岛");
     cities.add(c1);
     cities.add(c2);
     cities.add(c3);
     cities.add(c4);
     cities.add(c5);
     cities.add(c6);
     return cities;
 }

}
```

然后在 ToUpdateCustomerAction 中查询出全部的城市，代码如下：

```
package action;

import java.util.List;
```

```
import dao.CustomerDAO;
import entity.City;
import entity.Customer;
import entity.Sex;

public class ToUpdateCustomerAction {

 // output
 private Customer customer; // 客户
 private List<Sex> sexes; // 性别


 private List<City> cities; // 城市


 public String execute() {
     CustomerDAO dao = new CustomerDAO();
     // 模拟查询客户
     customer = dao.findById();
     // 模拟查询全部的性别
     sexes = dao.findAllSex();


     // 模拟查询全部的城市
     cities = dao.findAllCities();


     return "success";
 }

 public Customer getCustomer() {
     return customer;
 }

 public void setCustomer(Customer customer) {
     this.customer = customer;
 }

 public List<Sex> getSexes() {
     return sexes;
 }

 public void setSexes(List<Sex> sexes) {
     this.sexes = sexes;
 }


 public List<City> getCities() {
     return cities;
 }

 public void setCities(List<City> cities) {
     this.cities = cities;
 }


}
```

最后，在 update_customer.jsp 上，使用动态的多选框标签，根据 Action 中返回的所有城市生成城市选项，并根据客户信息设置默认勾选，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
```

```
    <head></head>
    <body>
     <h1>模拟修改客户</h1>

     <%-- 1.表单标签 --%>
     <s:form action="#" method="post">
         <!--此处略去其他标签...-->

         <!--
             7.2 多选框标签（动态）
              1）根据 OGNL（list 属性值）的取值初始化一组 checkbox，
                 并根据 listKey 指定的实体类中的属性来生成 checkbox 的 value 值，
                 根据 listValue 指定的实体类中的属性来生成 checkbox 的 label 值。
              2）根据 OGNL（customer.travelCities）取值来与生成 radio 的 value 比较，
                 若与哪个 radio 的 value 值一致，则将其勾选。
         -->
         <s:checkboxlist  name="customer.travelCities"  list="cities"  listKey="code"
listValue="name" label="旅游过的城市"/>

     </s:form>

    </body>
    </html>
```

重新部署项目并重启 tomcat，重新访问修改客户页面，效果如下图：

图-17

步骤三：下拉选标签

下拉选标签的用法与单选框标签、多选框标签也类似，分为静态初始化和动态初始化 2 种方式。首先我们来使用静态的方式来初始化下拉选选项，代码如下：

```
    <%@page pageEncoding="utf-8"%>
    <%@taglib uri="/struts-tags" prefix="s"%>
    <html>
    <head></head>
```

```
<body>
 <h1>模拟修改客户</h1>

 <%-- 1.表单标签 --%>
 <s:form action="#" method="post">
     <!--此处略去其他标签...-->

     <!--
         8.1 下拉选标签（静态）
          1）根据 OGNL（list 属性值）创建的 Map 生成一组 option，其中 Map 的 key
             生成 option 的 value 值，Map 的 value 生成 option 的显示值。
          2）根据 OGNL（customer.home）取值来与生成 option 的
             value 比较，若与哪个 option 的 value 值一致，则将其勾选。
     -->
     <s:select name="customer.home"
         list="#{'beijing':' 北 京 ','shanghai':' 上 海 ','guangzhou':' 广 州
','shenzhen':'深圳','chongqing':'重庆','diaoyudao':'钓鱼岛' }"
         label="家乡" headerKey="-1" headerValue="请选择"/>

 </s:form>

</body>
</html>
```

刷新浏览器，显示效果如下图：

图-18

对于动态的初始化的方式，我们还是使用全部的城市来初始化选项，代码如下：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
 <h1>模拟修改客户</h1>
```

```
<%-- 1.表单标签 --%>
<s:form action="#" method="post">
    <!--此处略去其他标签...-->

    <!--
        8.2 下拉选标签（动态）
         1）根据 OGNL（list 属性值）的取值初始化一组 option，
            并根据 listKey 指定的实体类中的属性来生成 option 的 value 值，
            根据 listValue 指定的实体类中的属性来生成 option 的 label 值。
         2）根据 OGNL（customer.home）取值来与生成 option 的 value 比较，
            若与哪个 option 的 value 值一致，则将其勾选。
    -->
    <s:select name="customer.home" list="cities" listKey="code" listValue="name"
        label="家乡" headerKey="" headerValue="请选择"/>


</s:form>

</body>
</html>
```

刷新浏览器，显示效果如下图：

图-19

- **完整代码**

本案例的 Customer 完整代码：

```
package entity;

import java.util.List;

public class Customer {
```

```
    private String name; // 姓名
    private String password; // 密码
    private String desc; // 简介
    private boolean marry; // 是否已婚

    private String sex; // 性别
    private List<String> travelCities; // 去过的城市
    private String home; // 家乡

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getDesc() {
        return desc;
    }

    public void setDesc(String desc) {
        this.desc = desc;
    }

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public boolean isMarry() {
        return marry;
    }

    public void setMarry(boolean marry) {
        this.marry = marry;
    }

    public String getHome() {
        return home;
    }

    public List<String> getTravelCities() {
        return travelCities;
    }

    public void setTravelCities(List<String> travelCities) {
        this.travelCities = travelCities;
    }

    public void setHome(String home) {
        this.home = home;
    }

}
```

Sex 完整代码：

```
package entity;

public class Sex {

 private String code;
 private String name;

 public String getCode() {
     return code;
 }

 public void setCode(String code) {
     this.code = code;
 }

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

}
```

City 完整代码：

```
package entity;

public class City {

 private String code;
 private String name;

 public City(String code, String name) {
     super();
     this.code = code;
     this.name = name;
 }

 public String getCode() {
     return code;
 }

 public void setCode(String code) {
     this.code = code;
 }

 public String getName() {
     return name;
 }

 public void setName(String name) {
     this.name = name;
 }

}
```

CustomerDAO 完整代码：

```
package dao;

import java.util.ArrayList;
import java.util.List;

import entity.City;
import entity.Customer;
import entity.Sex;

/**
 * 模拟客户 DAO
 */
public class CustomerDAO {

 /**
  * 模拟根据 ID 查询客户
  */
 public Customer findById() {
     // 实例化客户实体类
     Customer c = new Customer();
     // 设置一些默认值
     c.setName("Tarena");
     c.setPassword("123");
     c.setDesc("中华人民共和国公民");
     c.setSex("F");
     c.setMarry(true);

     List<String> list = new ArrayList<String>();
     list.add("beijing");
     list.add("guangzhou");
     c.setTravelCities(list);

     c.setHome("shanghai");
     return c;
 }

 /**
  * 模拟查询所有的性别，并不符合真实业务，仅仅是用于练习知识点
  */
 public List<Sex> findAllSex() {
     List<Sex> list = new ArrayList<Sex>();

     Sex s1 = new Sex();
     s1.setCode("M");
     s1.setName("男");
     list.add(s1);

     Sex s2 = new Sex();
     s2.setCode("F");
     s2.setName("女");
     list.add(s2);

     Sex s3 = new Sex();
     s3.setCode("O");
     s3.setName("其他");
     list.add(s3);

     return list;
 }

 /**
  * 模拟查询所有的城市
  */
 public List<City> findAllCities() {
     List<City> cities = new ArrayList<City>();
     City c1 = new City("beijing", "北京");
```

```
        City c2 = new City("shanghai", "上海");
        City c3 = new City("guangzhou", "广州");
        City c4 = new City("shenzhen", "深圳");
        City c5 = new City("chongqing", "重庆");
        City c6 = new City("diaoyudao", "钓鱼岛");
        cities.add(c1);
        cities.add(c2);
        cities.add(c3);
        cities.add(c4);
        cities.add(c5);
        cities.add(c6);
        return cities;
    }

}
```

ToUpdateCustomerAction 完整代码：

```
package action;

import java.util.List;

import dao.CustomerDAO;
import entity.City;
import entity.Customer;
import entity.Sex;

public class ToUpdateCustomerAction {

    // output
    private Customer customer; // 客户
    private List<Sex> sexes; // 性别
    private List<City> cities; // 城市

    public String execute() {
        CustomerDAO dao = new CustomerDAO();
        // 模拟查询客户
        customer = dao.findById();
        // 模拟查询全部的性别
        sexes = dao.findAllSex();
        // 模拟查询全部的城市
        cities = dao.findAllCities();
        return "success";
    }

    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public List<Sex> getSexes() {
        return sexes;
    }

    public void setSexes(List<Sex> sexes) {
        this.sexes = sexes;
    }

    public List<City> getCities() {
        return cities;
    }
```

```
    public void setCities(List<City> cities) {
        this.cities = cities;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

<!-- 客户配置信息 -->
<package name="customer"
    namespace="/customer" extends="struts-default">
    <!-- 打开修改页面 -->
    <action name="toUpdateCustomer"
        class="action.ToUpdateCustomerAction">
        <result name="success">
            /WEB-INF/customer/update customer.jsp
        </result>
    </action>
</package>

</struts>
```

update_customer.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head></head>
<body>
<h1>模拟修改客户</h1>

<%-- 1.表单标签 --%>
<s:form action="#" method="post">
    <!--
        2.文本框标签
         1）生成文本框
         2）根据 OGNL（customer.name）取值给文本框赋值
    -->
    <s:textfield name="customer.name" label="姓名"/>
    <!--
        3.密码框标签
         1）生成密码框
         2）根据 OGNL（customer.password）取值给密码框赋值
    -->
    <s:password name="customer.password" label="密码" showPassword="true"/>
    <!--
        4.文本域标签
         1）生成文本域
         2）根据 OGNL（customer.desc）取值给文本域赋值
    -->
    <s:textarea name="customer.desc" cols="30" rows="5" label="简介"/>
    <!--
        5.布尔框标签
         1）生成一个 checkbox
         2）根据 OGNL（customer.marry）取值（布尔型）来确定是否勾选
```

```
        -->
        <s:checkbox    name="customer.marry"    label="  是  否  已  婚  "
labelposition="left"/>
        <!--
            6.1 单选框标签（静态）
             1）根据 OGNL（list 属性值）创建的 Map 生成一组 radio，
                 其中 Map 的可以生成 radio 的 value 值，Map 的 value 生成 radio 的 label
值。
             2）根据 OGNL（customer.marry）取值来与生成 radio 的 value 比较，
                若与哪个 radio 的 value 值一致，则将其勾选。
        -->
        <s:radio name="customer.sex" list="#{'M':'男','F':'女' }" label="性别"/>
        <!--
            6.2 单选框标签（动态）
             1）根据 OGNL（list 属性值）的取值初始化一组 radio，
                并根据 listKey 指定的实体类中的属性来生成 radio 的 value 值，
                根据 listValue 指定的实体类中的属性来生成 radio 的 label 值。
             2）根据 OGNL（customer.marry）取值来与生成 radio 的 value 比较，
                若与哪个 radio 的 value 值一致，则将其勾选。
        -->
        <s:radio name="customer.sex"
            list="sexes" listKey="code" listValue="name" label="性别"/>
        <!--
            7.1 多选框标签（静态）
             1）根据 OGNL（list 属性值）创建的 Map 生成一组 checkbox，其中 Map 的 key
                生成 checkbox 的 value 值，Map 的 value 生成 checkbox 的 label 值。
             2）根据 OGNL（customer.travelCities）取值来与生成 checkbox 的
                value 比较，若与哪个 checkbox 的 value 值一致，则将其勾选。
        -->
        <s:checkboxlist name="customer.travelCities"
            list="#{'beijing':' 北 京 ','shanghai':' 上 海 ','guangzhou':' 广 州
','shenzhen':'深圳','chongqing':'重庆','diaoyudao':'钓鱼岛' }" label="旅游过的城市
"/>
        <!--
            7.2 多选框标签（动态）
             1）根据 OGNL（list 属性值）的取值初始化一组 checkbox，
                并根据 listKey 指定的实体类中的属性来生成 checkbox 的 value 值，
                根据 listValue 指定的实体类中的属性来生成 checkbox 的 label 值。
             2）根据 OGNL（customer.travelCities）取值来与生成 radio 的 value 比较，
                若与哪个 radio 的 value 值一致，则将其勾选。
        -->
        <s:checkboxlist      name="customer.travelCities"      list="cities"
listKey="code" listValue="name" label="旅游过的城市"/>
        <!--
            8.1 下拉选标签（静态）
             1）根据 OGNL（list 属性值）创建的 Map 生成一组 option，其中 Map 的 key
                生成 option 的 value 值，Map 的 value 生成 option 的显示值。
             2）根据 OGNL（customer.home）取值来与生成 option 的
                value 比较，若与哪个 option 的 value 值一致，则将其勾选。
        -->
        <s:select name="customer.home"
            list="#{'beijing':' 北 京 ','shanghai':' 上 海 ','guangzhou':' 广 州
','shenzhen':'深圳','chongqing':'重庆','diaoyudao':'钓鱼岛' }"
            label="家乡" headerKey="-1" headerValue="请选择"/>
        <!--
            8.2 下拉选标签（动态）
             1）根据 OGNL（list 属性值）的取值初始化一组 option，
                并根据 listKey 指定的实体类中的属性来生成 option 的 value 值，
                根据 listValue 指定的实体类中的属性来生成 option 的 label 值。
             2）根据 OGNL（customer.home）取值来与生成 option 的 value 比较，
                若与哪个 option 的 value 值一致，则将其勾选。
```

```
        -->
        <s:select    name="customer.home"    list="cities"    listKey="code"
listValue="name"
            label="家乡" headerKey="" headerValue="请选择"/>

    </s:form>

   </body>
   </html>
```

达内IT培训集团

课后作业

1. **json 类型的 Result 如何使用**

2. **如何在 Struts2 项目中实现异步请求的校验**

3. **Struts2 中有哪些 UI 标签，请简述其作用**

Struts 核心

Unit05

知识体系.....Page 268

拦截器	作用	拦截器的用途
	使用步骤	1、创建拦截器组件
		2、注册拦截器
		3、引用拦截器
	扩展	拦截器栈
		预置拦截器
		默认引用拦截器
		拦截器调用顺序
登录检查	需求描述	为什么要做登录检查
	开发思路	如何做登录检查
	开发步骤	登录检查的开发步骤
上传文件拦截器	FileUpload 介绍	FileUpload 拦截器介绍
		FileUpload 拦截器原理
	FileUpload 使用	FileUpload 拦截器使用步骤
		注意事项

经典案例.....Page 274

拦截器 HelloWorld	1、创建拦截器组件
	2、注册拦截器
	3、引用拦截器
扩展拦截器 HelloWorld	拦截器栈
	预置拦截器
	默认引用拦截器
	拦截器调用顺序
NetCTOSS 登录检查	登录检查的开发步骤
上传文件	FileUpload 拦截器使用步骤

课后作业.....Page 300

达内IT培训集团

1. 拦截器

1.1. 作用

1.1.1. 【作用】拦截器的用途

拦截器的用途

- 拦截器适合封装一些通用处理,便于重复利用。例如请求参数传递给Action属性,日志的记录,权限检查, 事务处理等。拦截器是通过配置方式调用,因此使用方法比较灵活,便于维护和扩展。

解说识知

1.2. 使用步骤

1.2.1. 【使用步骤】1、创建拦截器组件

1、创建拦截器组件

- 创建一个类，实现Interceptor接口，并实现intercept方法。

```
public String intercept(ActionInvocation invocation) {
    //拦截器--前部分处理
     invocation.invoke(); //执行action和result
    //拦截器--后续处理
}
```

解说识知

1.2.2. 【使用步骤】2、注册拦截器

2、注册拦截器

- 在struts.xml中，注册拦截器

```
<package>
    <interceptors>
            <interceptor name="别名" class="实现类"/>
            //...其他interceptor
    </interceptors>
</package>
```

解说识知

1.2.3. 【使用步骤】3、引用拦截器

3、引用拦截器

- 如果哪个Action希望被拦截器扩展，需要在此action配置下，引用拦截器

```
<action>
    <interceptor-ref name="拦截器"/>
    //...可以写多个
</action>
```

解说识知

1.3. 扩展

1.3.1. 【扩展】拦截器栈

拦截器栈

- 往往一个Action需要引用很多个拦截器，那么可以将这些拦截器打包成栈，直接引用这个栈即可，这样可以简化对拦截器的引用
- 拦截器栈仅仅是对拦截器的打包，方便引用，它在使用上完全等同于拦截器

```
<interceptor-stack name="myStack">
  <interceptor-ref name="firstInterceptor"/>
  <interceptor-ref name="secondInterceptor"/>
</interceptor-stack>
```

解说识知

1.3.2. 【扩展】预置拦截器

预置拦截器

- Struts2预置了大量的拦截器，见struts-default.xml第122行。
- 有一部分拦截器是Struts2框架自身要用到的，必须引用
  - 这些拦截器被打包在basicStack中
  - 我们项目中引用的拦截器，不能少于basicStack中注册的拦截器

```
<!-- Basic stack -->
<interceptor-stack name="basicStack">
    <interceptor-ref name="exception"/>
    <interceptor-ref name="servletConfig"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="checkbox"/>
    <interceptor-ref name="multiselect"/>
    <interceptor-ref name="actionMappingParams"/>
    <interceptor-ref name="params">
        <param name="excludeParams">dojo\..*,^struts\..*</param>
    </interceptor-ref>
    <interceptor-ref name="conversionError"/>
</interceptor-stack>
```

解说识知

1.3.3. 【扩展】默认引用拦截器

默认引用拦截器

解详识知

- 有一部分拦截器是最常用的，他们被打包在defaultStack中

```
<interceptor-stack name="defaultStack">
    <interceptor-ref name="exception"/>
    <interceptor-ref name="alias"/>
    <interceptor-ref name="servletConfig"/>
    <interceptor-ref name="i18n"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="chain"/>
    <interceptor-ref name="debugging"/>
    <interceptor-ref name="scopedModelDriven"/>
    <interceptor-ref name="modelDriven"/>
    <interceptor-ref name="fileUpload"/>
```

- 该拦截器栈是Struts2框架默认引用的，即我们不需要在Action中做任何处理，就相当于是引用了这个拦截器栈

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

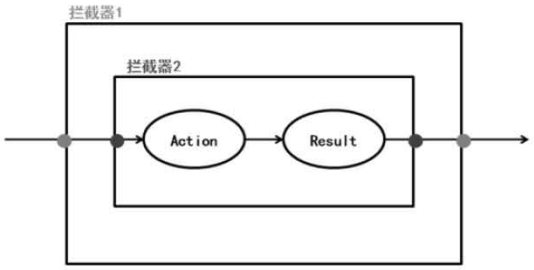
- 注意，一旦我们在action配置下引用了任何拦截器，那么上述默认引用的拦截器就失效了，因此在引用拦截器时，不要丢弃默认拦截器的引用

1.3.4. 【扩展】拦截器调用顺序

拦截器调用顺序

解详识知

- 拦截器在调用时，按照被引用的先后顺序，调用顺序如下图
- 调用顺序遵循先进后出原则



2. 登录检查

2.1. 需求描述

2.1.1. 【需求描述】为什么要做登录检查

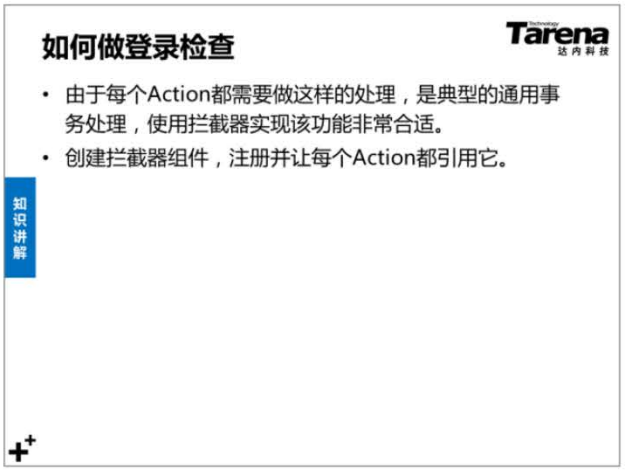
为什么要做登录检查

解详识知

- 目前NetCTOSS中，我们可以绕开登录，直接输入URL直接访问某Action，导致登录功能形同虚设。
- 要求访问每个Action时，验证用户是否已经登录，如果没有登录则跳转到登录页面，必须先行登录才能访问。

2.2. 开发思路

2.2.1. 【开发思路】如何做登录检查



2.3. 开发步骤

2.3.1. 【开发步骤】登录检查的开发步骤

3. 上传文件拦截器

3.1. FileUpload 介绍

3.1.1. 【FileUpload 介绍】FileUpload 拦截器介绍

FileUpload拦截器介绍

- FileUpload拦截器是Struts2预置的拦截器
  - 见struts-default.xml第134行
- FileUpload拦截器被默认拦截器栈defaultStack引用，即我们不需要做任何配置，即可使用该拦截器
  - 见struts-default.xml第275行
- FileUpload拦截器可以用于做文件上传

解说知识

3.1.2. 【FileUpload 介绍】FileUpload 拦截器原理

FileUpload拦截器原理

- FileUpload拦截器上传文件步骤
  - 首先，FileUpload拦截器将表单中提交的文件，以临时文件的形式保存到服务器临时路径下。
  - 之后，FileUpload拦截器将该临时文件对象注入给Action。Action自主处理该临时文件，如：将文件重新命名并复制到固定路径下。
  - FileUpload拦截器删除临时文件。

解说知识

3.2. FileUpload 使用

3.2.1. 【FileUpload 使用】FileUpload 拦截器使用步骤

FileUpload拦截器使用步骤

- 导包
  - 导入包commons-io.jar
- Action
  - 定义File类型属性（如some），接收拦截器注入的临时文件对象。
  - 如果想获取原始文件名，要定义String类型属性，属性名为“File类型属性+FileName”（如someFileName），拦截器会自动给该属性注入原始文件名。
- 表单设置
  - 必须满足method="post"
  - 必须满足enctype="multipart/form-data"

解说知识

3.2.2. 【FileUpload 使用】注意事项

达内IT培训集团

经典案例

1. 拦截器 HelloWorld

- **问题**

写一个拦截器的 HelloWorld 程序。

- **方案**

拦截器的使用步骤：

1) 创建拦截器组件，实现接口 Interceptor
2) 在 struts.xml 中注册拦截器
3) 在 action 的配置中引用拦截器

这里我们复用 StrutsDay04 项目的修改客户功能，并对打开修改页面的 action 引用自定义的拦截器。

- **步骤**

步骤一：创建拦截器

创建包 interceptor，在该包下创建拦截器组件 FirstInterceptor，并实现接口 Interceptor，在拦截方法中调用 action 业务方法，并且在调用 action 前后分别输出一些内容，以模拟对 action 请求的拦截。代码如下：

```
package interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 第一个拦截器
 */
public class FirstInterceptor implements Interceptor {

 @Override
 public void destroy() {
 }

 @Override
 public void init() {
 }

 @Override
 public String intercept(ActionInvocationai) throws Exception {
     System.out.println("FirstInterceptor 拦截前...");
     // 执行 action 业务方法
     ai.invoke();
     System.out.println("FirstInterceptor 拦截后...");
```

```
        /*
         * 返回值匹配对应的 result，但是一旦代码中调用了
         * ai.invoke 时，则此返回值无效，Struts2 会根据
         * action 的返回值匹配 result。如果当前代码中没有
         * 调用 ai.invoke，则此返回值生效。
         * */
        return "error";
    }

}
```

步骤二：注册拦截器

在 struts.xml 中，客户 package 下注册拦截器组件 FirstInterceptor，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">


     <interceptors>
         <!--注册拦截器 -->
         <interceptor name="first"
             class="interceptor.FirstInterceptor"/>
     </interceptors>



     <!--打开修改页面 -->
     <action name="toUpdateCustomer"
         class="action.ToUpdateCustomerAction">
         <result name="success">
             /WEB-INF/customer/update_customer.jsp
         </result>
     </action>
 </package>

</struts>
```

步骤三：引用拦截器

在修改客户 action 的配置下，引用已注册的拦截器，代码如下：

```
<?xmlversion="1.0"encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">
     <interceptors>
         <!--注册拦截器 -->
```

```
        <interceptor name="first"
            class="interceptor.FirstInterceptor"/>
    </interceptors>

    <!--打开修改页面 -->
    <action name="toUpdateCustomer"
        class="action.ToUpdateCustomerAction">


        <!--引用拦截器 -->
        <interceptor-ref name="first"/>


        <result name="success">
            /WEB-INF/customer/update customer.jsp
        </result>
    </action>
</package>

</struts>
```

步骤四：测试

为了便于观察拦截器与 Action 的执行顺序，在 Action 的构造方法及业务方法中，输出一些内容，代码如下：

```
package action;

importjava.util.List;

importdao.CustomerDAO;
importentity.City;
importentity.Customer;
importentity.Sex;

public class ToUpdateCustomerAction {
 // output
 private Customer customer; // 客户
 private List<Sex> sexes; // 性别
 private List<City> cities; // 城市


 publicToUpdateCustomerAction() {
     System.out.println("实例化 ToUpdateCustomerAction...");
 }


 public String execute() {


     System.out.println("调用 ToUpdateCustomerAction 业务方法...");


     CustomerDAOdao = new CustomerDAO();
     // 模拟查询客户
     customer = dao.findById();
     // 模拟查询全部的性别
     sexes = dao.findAllSex();
     // 模拟查询全部的城市
     cities = dao.findAllCities();
     return "success";
 }
```

```
    public Customer getCustomer() {
        return customer;
    }

    public void setCustomer(Customer customer) {
        this.customer = customer;
    }

    public List<Sex>getSexes() {
        return sexes;
    }

    public void setSexes(List<Sex> sexes) {
        this.sexes = sexes;
    }

    public List<City>getCities() {
        return cities;
    }

    public void setCities(List<City> cities) {
        this.cities = cities;
    }

}
```

重新部署并重启 tomcat，然后重新访问修改客户页面，控制台输出结果如下图：

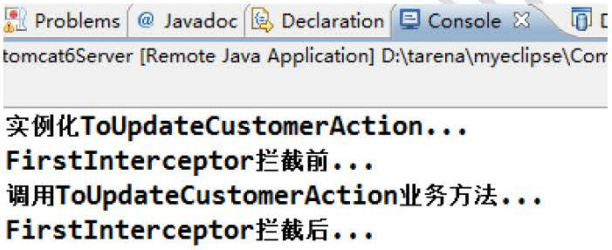


图-1

从输出结果中可以看出，拦截器是在实例化 Action 之后，调用业务方法之前开始调用的。

- **完整代码**

本案例的 FirstInterceptor 完整代码：

```
package interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 第一个拦截器
 */
public class FirstInterceptor implements Interceptor {

 @Override
 public void destroy() {
 }
```

```
    @Override
    public void init() {
    }

    @Override
    public String intercept(ActionInvocationai) throws Exception {
        System.out.println("FirstInterceptor 拦截前...");
        // 执行 action 业务方法
        ai.invoke();
        System.out.println("FirstInterceptor 拦截后...");
        /*
         * 返回值匹配对应的 result，但是一旦代码中调用了
         * ai.invoke 时，则此返回值无效，Struts2 会根据
         * action 的返回值匹配 result。如果当前代码中没有
         * 调用 ai.invoke，则此返回值生效。
         * */
        return "error";
    }

    }
```

ToUpdateCustomerAction 完整代码：

```
package action;

importjava.util.List;

importdao.CustomerDAO;
importentity.City;
importentity.Customer;
importentity.Sex;

public class ToUpdateCustomerAction {

 // output
 private Customer customer; // 客户
 private List<Sex> sexes; // 性别
 private List<City> cities; // 城市

 publicToUpdateCustomerAction() {
     System.out.println("实例化 ToUpdateCustomerAction...");
 }

 public String execute() {
     System.out.println("调用 ToUpdateCustomerAction 业务方法...");

     CustomerDAOdao = new CustomerDAO();
     // 模拟查询客户
     customer = dao.findById();
     // 模拟查询全部的性别
     sexes = dao.findAllSex();
     // 模拟查询全部的城市
     cities = dao.findAllCities();
     return "success";
 }

 public Customer getCustomer() {
     return customer;
 }
 public void setCustomer(Customer customer) {
     this.customer = customer;
 }
```

```
    public List<Sex>getSexes() {
        return sexes;
    }

    public void setSexes(List<Sex> sexes) {
        this.sexes = sexes;
    }

    public List<City>getCities() {
        return cities;
    }

    public void setCities(List<City> cities) {
        this.cities = cities;
    }

}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

<!-- 客户配置信息 -->
<package name="customer"
    namespace="/customer" extends="struts-default">
    <interceptors>
        <!-- 注册拦截器 -->
        <interceptor name="first"
            class="interceptor.FirstInterceptor"/>
    </interceptors>

    <!-- 打开修改页面 -->
    <action name="toUpdateCustomer"
        class="action.ToUpdateCustomerAction">
        <!-- 引用拦截器 -->
        <interceptor-ref name="first"/>
        <result name="success">
            /WEB-INF/customer/update customer.jsp
        </result>
    </action>
</package>

</struts>
```

2. 扩展拦截器 HelloWorld

- **问题**

扩展拦截器 HelloWorld 示例，练习使用拦截器栈，并观察多个拦截器的执行顺序。

- **方案**

创建一个新的拦截器，与第一个拦截器打包成栈，然后让修改客户的 action 引用这个拦截器栈，并观察控制台中这两个拦截器与 Action 的执行顺序。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建一个新的拦截器

创建一个新的拦截器组件 SecondInterceptor，代码如下：

```
package interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 第二个拦截器
 */
public class SecondInterceptor implements Interceptor {

 @Override
 public void destroy() {
 }

 @Override
 public void init() {
 }

 @Override
 public String intercept(ActionInvocationai) throws Exception {
     System.out.println("SecondInterceptor拦截前...");
     ai.invoke();
     System.out.println("SecondInterceptor拦截后...");
     return "error";
 }

}
```

步骤二：注册新拦截器，并打包成栈

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">
     <interceptors>
         <!--注册拦截器 -->
         <interceptor name="first"
             class="interceptor.FirstInterceptor"/>


         <interceptor name="second"
             class="interceptor.SecondInterceptor"/>
         <!--注册拦截器栈 -->
         <interceptor-stack name="myStack">
             <interceptor-ref name="first"/>
             <interceptor-ref name="second"/>
         </interceptor-stack>
```

```
        </interceptors>

        <!--打开修改页面 -->
        <action name="toUpdateCustomer"
            class="action.ToUpdateCustomerAction">
            <!--引用拦截器 -->
            <interceptor-ref name="first"/>
            <result name="success">
                /WEB-INF/customer/update_customer.jsp
            </result>
        </action>
    </package>

    </struts>
```

步骤三：引用拦截器栈

可以在 action 的配置下引用拦截器，也可以给一个包下所有的 action 统一设置默认使用的拦截器，代码如下：

```
    <?xml version="1.0" encoding="UTF-8" ?>
    <!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
        "http://struts.apache.org/dtds/struts-2.1.7.dtd">
    <struts>

    <!--客户配置信息 -->
    <package name="customer"
        namespace="/customer" extends="struts-default">
        <interceptors>
            <!--注册拦截器 -->
            <interceptor name="first"
                class="interceptor.FirstInterceptor"/>
            <interceptor name="second"
                class="interceptor.SecondInterceptor"/>
            <!--注册拦截器栈 -->
            <interceptor-stack name="myStack">
                <interceptor-ref name="first"/>
                <interceptor-ref name="second"/>
            </interceptor-stack>
        </interceptors>

        <!--设置当前包下所有 Action 默认引用的拦截器 -->
        <default-interceptor-ref name="myStack"/>

        <!--打开修改页面 -->
        <action name="toUpdateCustomer"
            class="action.ToUpdateCustomerAction">

            <!--引用拦截器 -->
            <!--<interceptor-ref name="first"/> -->

            <result name="success">
                /WEB-INF/customer/update_customer.jsp
            </result>
        </action>
    </package>
    </struts>
```

步骤四：测试

重新部署项目并重启 tomcat，重新访问修改客户页面时，控制台输出结果如下图：

图-2

从图中可以看出，多个拦截器拦截 Action 的顺序满足先进后出的原则，其顺序可以按照下图来理解和记忆：

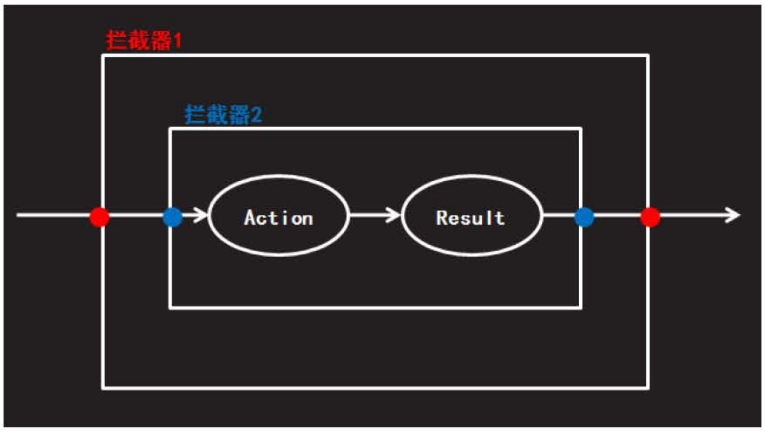


图-3

- **完整代码**

本案例的 SecondInterceptor 完整代码：

```
package interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 第二个拦截器
 */
public class SecondInterceptor implements Interceptor {

 @Override
 public void destroy() {
 }

 @Override
 public void init() {
 }
```

```
    @Override
    public String intercept(ActionInvocationai) throws Exception {
        System.out.println("SecondInterceptor拦截前...");
        ai.invoke();
        System.out.println("SecondInterceptor拦截后...");
        return "error";
    }
}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

    <!-- 客户配置信息 -->
    <package name="customer"
        namespace="/customer" extends="struts-default">
        <interceptors>
            <!-- 注册拦截器 -->
            <interceptor name="first"
                class="interceptor.FirstInterceptor"/>
            <interceptor name="second"
                class="interceptor.SecondInterceptor"/>
            <!-- 注册拦截器栈 -->
            <interceptor-stack name="myStack">
                <interceptor-ref name="first"/>
                <interceptor-ref name="second"/>
            </interceptor-stack>
        </interceptors>
        <!-- 设置当前包下所有Action默认引用的拦截器 -->
        <default-interceptor-ref name="myStack"/>

        <!-- 打开修改页面 -->
        <action name="toUpdateCustomer"
            class="action.ToUpdateCustomerAction">
            <!-- 引用拦截器 -->
            <!--<interceptor-ref name="first"/> -->
            <result name="success">
                /WEB-INF/customer/update_customer.jsp
            </result>
        </action>
    </package>

</struts>
```

3. NetCTOSS 登录检查

- **问题**

当前 NetCTOSS 项目中，我们可以直接在地址栏中输入地址访问资费模块的功能，这使得登录功能形同虚设。要求在没登录时不允许直接访问资费模块的任何 action，登录后才可以访问。

- **方案**

需要在访问资费 action 时进行校验，判断是否已进行了登录。这种判断每一个 action 都要处理，甚至以后有更多业务模块时也要做这样的事情，因此是 action 通用的业务逻辑，可以使用拦截器来处理。

我们可以创建拦截器组件，在拦截方法中调用 action 业务方法之前，判断是否进行过登录，从而确定是否可以继续访问该 action，然后给每一个业务模块注册该 action 即可实现此需求。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：创建登录检查拦截器

创建 com.netctoss.interceptor 包，在该包下创建登录检查拦截器 LoginInterceptor，代码如下：

```
packagecom.netctoss.interceptor;

importjava.util.Map;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 登录检查拦截器，用于检查用户是否登录。
 */
public class LoginInterceptor implements Interceptor {

 @Override
 public void destroy() {
 }

 @Override
 public void init() {
 }

 @Override
 public String intercept(ActionInvocationai) throws Exception {
     // 获取 session
     Map<String, Object> session = ai.getInvocationContext().getSession();
     // 从 session 中读取登录信息
     Object admin = session.get("admin");
     // 如果登录信息为空，则踢回登录页，而不调用业务 Action
     if (admin == null) {
         return "login";
     } else {
         // 如果登录信息不为空，则调用业务 Action
         returnai.invoke();
     }
 }

}
```

步骤二：使用登录检查拦截器

在 struts.xml 中注册该拦截器，然后将其与默认拦截器栈打包成新的拦截器栈 loginStack，再通过 default-interceptor-ref 标记设置 action 默认使用的拦截器为 loginStack。在检查到未登录时，需要找到名为 login 的 result，跳转回登录页面，这个result就需要是所有action公用的，可以在global-results标记下定义这个result，来实现被 action 复用的目的。代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>


<!--公共的包，封装了通用的拦截器、通用的 result -->
<package name="netctoss" extends="json-default">
    <interceptors>
        <!--登录检查拦截器 -->
        <interceptor name="loginInterceptor"
            class="com.netctoss.interceptor.LoginInterceptor"/>
        <!--登录检查拦截器栈 -->
        <interceptor-stack name="loginStack">
            <interceptor-ref name="loginInterceptor"/>
            <!--不要丢掉默认的拦截器栈，里面有很多 Struts2 依赖的拦截器 -->
            <interceptor-ref name="defaultStack"/>
        </interceptor-stack>
    </interceptors>
    <!--设置 action 默认引用的拦截器 -->
    <default-interceptor-ref name="loginStack"/>
    <!--全局的 result，包下所有的 action 都可以共用 -->
    <global-results>
        <!--跳转到登录页面的 result -->
        <result name="login" type="redirectAction">
            <param name="namespace">/login</param>
            <param name="actionName">toLogin</param>
        </result>
    </global-results>
</package>

<!--
    资费模块配置信息：
    一般情况下，一个模块的配置单独封装在一个 package 下，
    并且以模块名来命名 package 的 name 和 namespace。
 -->

<package name="cost" namespace="/cost" extends="netctoss">


    <!--查询资费数据 -->
    <action name="findCost" class="com.netctoss.action.FindCostAction">
        <!--
            正常情况下跳转到资费列表页面。
            一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
         -->
        <result name="success">
```

```
                /WEB-INF/cost/find cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!--删除资费 -->
        <action name="deleteCost"
            class="com.netctoss.action.DeleteCostAction">
            <!--删除完之后，重定向到查询 action -->
            <result name="success" type="redirectAction">
                findCost
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!--打开资费新增页 -->
        <action name="toAddCost">
            <result name="success">
                /WEB-INF/cost/add cost.jsp
            </result>
        </action>
        <!--资费名唯一性校验 -->
        <action name="checkCostName"
            class="com.netctoss.action.CheckCostNameAction">
            <!--使用 json 类型的 result 把结果输出给回调函数 -->
            <result name="success" type="json">
                <param name="root">info</param>
            </result>
        </action>
        <!--打开修改页面 -->
        <action name="toUpdateCost"
            class="com.netctoss.action.ToUpdateCostAction">
            <result name="success">
                /WEB-INF/cost/update cost.jsp
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!--登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
            会自动实例化默认的 Action 类 ActionSupport，
            该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
            会自动调用 execute 方法。
        -->
        <action name="toLogin">
            <result name="success">
```

```
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!--登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!--校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!--登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!--报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!--生成验证码 -->
        <action                                          name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!--使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!--指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>

    </struts>
```

注意：上述代码中将拦截器以及公共的 result 提出来放到了公共的包下，然后让资费的包继承此包，这样做的好处是可以复用这些公共的代码，因为将来还会有其他的业务模块，他们也需要使用这些代码。

步骤三：测试

部署项目并重启 tomcat，然后直接访问资费模块的某 action，会发现页面自动跳转到了登录页，这是因为拦截器的作用所致。在正常登录之后，再去输入地址访问资费模块某 action 时，就可以正常访问了。

- **完整代码**

本案例的 LoginInterceptor 完整代码：

```
    packagecom.netctoss.interceptor;

    importjava.util.Map;

    import com.opensymphony.xwork2.ActionInvocation;
    import com.opensymphony.xwork2.interceptor.Interceptor;

    /**
     * 登录检查拦截器，用于检查用户是否登录。
     */
    public class LoginInterceptor implements Interceptor {
```

```
    @Override
    public void destroy() {
    }

    @Override
    public void init() {
    }

    @Override
    public String intercept(ActionInvocationai) throws Exception {
        // 获取 session
        Map<String, Object> session = ai.getInvocationContext().getSession();
        // 从 session 中读取登录信息
        Object admin = session.get("admin");
        // 如果登录信息为空，则踢回登录页，而不调用业务 Action
        if (admin == null) {
            return "login";
        } else {
            // 如果登录信息不为空，则调用业务 Action
            returnai.invoke();
        }
    }

    }
```

struts.xml 完整代码：

```
    <?xml version="1.0" encoding="UTF-8" ?>
    <!DOCTYPE struts PUBLIC
       "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
       "http://struts.apache.org/dtds/struts-2.1.7.dtd">
    <struts>

    <!-- 公共的包，封装了通用的拦截器、通用的 result -->
    <package name="netctoss" extends="json-default">
        <interceptors>
            <!-- 登录检查拦截器 -->
            <interceptor name="loginInterceptor"
                class="com.netctoss.interceptor.LoginInterceptor"/>
            <!-- 登录检查拦截器栈 -->
            <interceptor-stack name="loginStack">
                <interceptor-ref name="loginInterceptor"/>
                <!-- 不要丢掉默认的拦截器栈，里面有很多 Struts2 依赖的拦截器 -->
                <interceptor-ref name="defaultStack"/>
            </interceptor-stack>
        </interceptors>
        <!-- 设置 action 默认引用的拦截器 -->
        <default-interceptor-ref name="loginStack"/>
        <!-- 全局的 result，包下所有的 action 都可以共用 -->
        <global-results>
            <!-- 跳转到登录页面的 result -->
            <result name="login" type="redirectAction">
                <param name="namespace">/login</param>
                <param name="actionName">toLogin</param>
            </result>
        </global-results>
    </package>

    <!--
        资费模块配置信息：
        一般情况下，一个模块的配置单独封装在一个 package 下，
        并且以模块名来命名 package 的 name 和 namespace。
     -->
    <package name="cost" namespace="/cost" extends="netctoss">
```

```
        <!-- 查询资费数据 -->
        <action name="findCost" class="com.netctoss.action.FindCostAction">
            <!--
                正常情况下跳转到资费列表页面。
                一般一个模块的页面要打包在一个文件夹下，并且文件夹以模块名命名。
             -->
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
            <!--
                错误情况下，跳转到错误页面。
                错误页面可以被所有模块复用，因此放在 main 下，
                该文件夹用于存放公用的页面。
             -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 删除资费 -->
        <action name="deleteCost"
            class="com.netctoss.action.DeleteCostAction">
            <!-- 删除完之后，重定向到查询 action -->
            <result name="success" type="redirectAction">
                findCost
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 打开资费新增页 -->
        <action name="toAddCost">
            <result name="success">
                /WEB-INF/cost/add_cost.jsp
            </result>
        </action>
        <!-- 资费名唯一性校验 -->
        <action name="checkCostName"
            class="com.netctoss.action.CheckCostNameAction">
            <!-- 使用 json 类型的 result 把结果输出给回调函数 -->
            <result name="success" type="json">
                <param name="root">info</param>
            </result>
        </action>
        <!--  打开修改页面 -->
        <action name="toUpdateCost"
            class="com.netctoss.action.ToUpdateCostAction">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
    </package>

    <!-- 登录模块 -->
    <package name="login" namespace="/login" extends="struts-default">
        <!--
            打开登录页面：
            1、action 的 class 属性可以省略，省略时 Struts2
               会自动实例化默认的 Action 类 ActionSupport，
               该类中有默认业务方法 execute，返回 success。
            2、action 的 method 属性可以省略，省略时 Struts2
               会自动调用 execute 方法。
        -->
```

```
        <action name="toLogin">
            <result name="success">
                /WEB-INF/main/login.jsp
            </result>
        </action>
        <!-- 登录校验 -->
        <action name="login" class="com.netctoss.action.LoginAction">
            <!-- 校验成功，跳转到系统首页 -->
            <result name="success">
                /WEB-INF/main/index.jsp
            </result>
            <!-- 登录失败，跳转回登录页面 -->
            <result name="fail">
                /WEB-INF/main/login.jsp
            </result>
            <!-- 报错，跳转到错误页面 -->
            <result name="error">
                /WEB-INF/main/error.jsp
            </result>
        </action>
        <!-- 生成验证码 -->
        <action                                                name="createImage"
class="com.netctoss.action.CreateImageAction">
            <!-- 使用 stream 类型的 result -->
            <result name="success" type="stream">
                <!-- 指定输出的内容 -->
                <param name="inputName">imageStream</param>
            </result>
        </action>
    </package>

   </struts>
```

4. 上传文件

- **问题**

在 Struts2 框架下实现文件上传。

- **方案**

Struts2 提供了拦截器可以自动实现文件上传，并且该拦截器存在于 defaultStack 中，是每个 action 默认使用的。

我们可以复用 StrutsDay04 的项目，在此基础上做文件上传示例。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：打开上传页面

在 struts.xml 中，配置打开上传页面的 action，代码如下：

```
   <?xml version="1.0" encoding="UTF-8" ?>
   <!DOCTYPE struts PUBLIC
      "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
      "http://struts.apache.org/dtds/struts-2.1.7.dtd">
```

```
<struts>

<!--客户配置信息 -->
<package name="customer"
    namespace="/customer" extends="struts-default">
    <interceptors>
        <!--注册拦截器 -->
        <interceptor name="first"
            class="interceptor.FirstInterceptor"/>
        <interceptor name="second"
            class="interceptor.SecondInterceptor"/>
        <!--注册拦截器栈 -->
        <interceptor-stack name="myStack">
            <interceptor-ref name="first"/>
            <interceptor-ref name="second"/>
        </interceptor-stack>
    </interceptors>
    <!--设置当前包下所有 Action 默认引用的拦截器 -->
    <default-interceptor-ref name="myStack"/>

    <!--打开修改页面 -->
    <action name="toUpdateCustomer"
        class="action.ToUpdateCustomerAction">
        <!--引用拦截器 -->
        <!--<interceptor-ref name="first"/> -->
        <result name="success">
            /WEB-INF/customer/update_customer.jsp
        </result>
    </action>
</package>


<!--文件上传示例 -->
<package name="demo" namespace="/demo" extends="struts-default">
    <!--打开上传文件页面 -->
    <action name="toUpload">
        <result name="success">/WEB-INF/jsp/upload.jsp</result>
    </action>
</package>

</struts>
```

在 WEB-INF 下创建文件夹 jsp，并在此文件夹下创建文件上传页面 upload.jsp，代码如下：

```
<%@page pageEncoding="utf-8"%>
<html>
<head></head>
<body>
 <form action="" method="post">
    <input type="file" name="some" />
    <input type="submit" value="提交" />
 </form>
</body>
</html>
```

重新部署并重启 tomcat，访问此页面，效果如下图：

图-4

步骤二：导包

上传文件需要依赖新的包 commons-io-1.3.2.jar，将其引入到项目中后，包结构如下图：

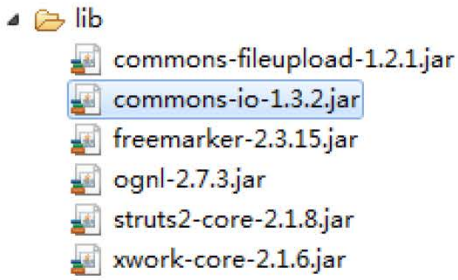


图-5

步骤三：Action 中处理上传请求

由于 Action 需要接收拦截器传入的临时文件，并对临时文件进行复制，因此需要提供一个文件操作的工具类 FileUtil，代码如下：

```
packageutil;

importjava.io.BufferedInputStream;
importjava.io.BufferedOutputStream;
importjava.io.File;
importjava.io.FileInputStream;
importjava.io.FileOutputStream;
importjava.io.IOException;

/**
 * 文件处理工具
 */
public class FileUtil {

 public static boolean copy(File src, File dest) {
     BufferedInputStreambis = null;
     BufferedOutputStreambos = null;
     try {
         bis = new BufferedInputStream(new FileInputStream(src));
         bos = new BufferedOutputStream(new FileOutputStream(dest));
         byte[] bts = new byte[1024];
         intlen = -1;
         while ((len = bis.read(bts)) != -1) {
             bos.write(bts, 0, len);
         }
         return true;
     } catch (Exception e) {
         e.printStackTrace();
         return false;
     } finally {
         if (bis != null) {
```

```
                try {
                    bis.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            if (bos != null) {
                try {
                    bos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

    }
}
```

然后创建上传文件 Action 类 UploadAction，并根据拦截器传入的临时文件，将其赋值到某路径下。代码如下：

```
package action;

importjava.io.File;
import org.apache.struts2.ServletActionContext;
importutil.FileUtil;

/**
 * 文件上传 Action
 */
public class UploadAction {

 /**
  * 接收拦截器传入的临时文件
  */
 private File some;
 /**
  * 接收拦截器注入的原始文件名
  */
 private String someFileName;

 publicString execute() {
     if (some == null)
         return "error";

     // 将文件放于项目部署路径下的 upload 文件夹下
     String path = "WEB-INF/upload/" + someFileName;
     // 根据相对部署路径计算完整路径
     path = ServletActionContext.getServletContext().getRealPath(path);
     // 将临时文件复制到上述路径下
     FileUtil.copy(some, new File(path));

     return "success";
 }

 public File getSome() {
     return some;
 }

 public void setSome(File some) {
     this.some = some;
 }
```

```
    public String getSomeFileName() {
        returnsomeFileName;
    }

    public void setSomeFileName(String someFileName) {
        this.someFileName = someFileName;
    }

    }
```

上述代码中，用意是将文件复制部署的项目路径下的 WEB-INF/upload 文件夹下，这样就可以不在项目中写死上传文件的路径了，而是随着项目部署位置的不同而自动变化，比较灵活。因此这里要求我们在项目中的 WEB-INF 下创建新的文件夹 upload，创建后该文件夹结构如下图：

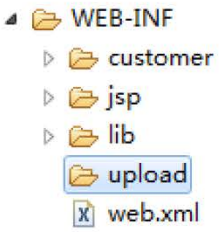


图-6

步骤四：struts.xml 中配置上传文件 action

在 struts.xml 中配置该 action，并且设置文件上传的最大值，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!--客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">
     <interceptors>
         <!--注册拦截器 -->
         <interceptor name="first"
             class="interceptor.FirstInterceptor"/>
         <interceptor name="second"
             class="interceptor.SecondInterceptor"/>
         <!--注册拦截器栈 -->
         <interceptor-stack name="myStack">
             <interceptor-ref name="first"/>
             <interceptor-ref name="second"/>
         </interceptor-stack>
     </interceptors>
     <!--设置当前包下所有 Action 默认引用的拦截器 -->
     <default-interceptor-ref name="myStack"/>

     <!--打开修改页面 -->
     <action name="toUpdateCustomer"
         class="action.ToUpdateCustomerAction">
         <!--引用拦截器 -->
         <!--<interceptor-ref name="first"/> -->
```

```
        <result name="success">
            /WEB-INF/customer/update customer.jsp
        </result>
    </action>
</package>

<!--设置上传文件最大值 -->
<constant name="struts.multipart.maxSize" value="5000000"/>

<!--文件上传示例 -->
<package name="demo" namespace="/demo" extends="struts-default">
    <!--打开上传文件页面 -->
    <action name="toUpload">
        <result name="success">/WEB-INF/jsp/upload.jsp</result>
    </action>

    <!--上传文件 -->
    <action name="upload" class="action.UploadAction">
        <result name="success">/WEB-INF/jsp/ok.jsp</result>
        <result name="error">/WEB-INF/jsp/error.jsp</result>
    </action>

</package>

</struts>
```

步骤五：JSP

上述配置中，如果上传成功则转至 ok.jsp，该页面代码如下：

```
<html>
<head></head>
<body>
 <h1>OK!</h1>
</body>
</html>
```

如果上传失败则转至 error.jsp，该页面代码如下：

```
<html>
<head></head>
<body>
 <h1 style="color:red">Error!</h1>
</body>
</html>
```

修改上传文件页面 upload.jsp 的表单属性值，代码如下：

```
<%@page pageEncoding="utf-8"%>
<html>
<head></head>
<body>

<!--
    上传文件对表单有 2 个要求：
     1、method="post"
```

```
        2、enctype="multipart/form-data"
    -->
    <form action="upload" method="post" enctype="multipart/form-data">

        <input type="file" name="some" />
        <input type="submit" value="提交" />
    </form>
    </body>
    </html>
```

步骤六：测试

重新部署项目并重启 tomcat，打开上传文件页面，选择一个文件后点提交按钮，文件被正确上传至项目部署路径下的 WEB-INF/upload 文件夹下，效果如下图：

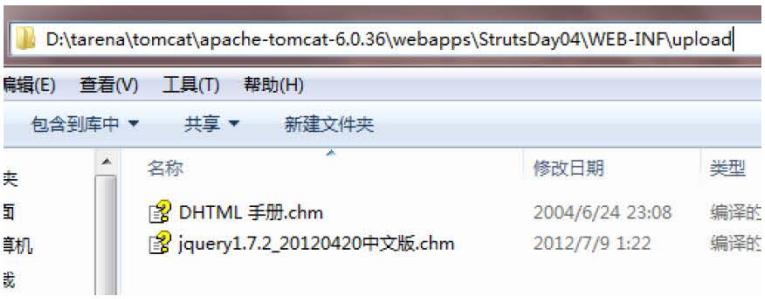


图-7

- **完整代码**

本案例的 FileUtil 完整代码：

```
packageutil;

importjava.io.BufferedInputStream;
importjava.io.BufferedOutputStream;
importjava.io.File;
importjava.io.FileInputStream;
importjava.io.FileOutputStream;
importjava.io.IOException;

/**
 * 文件处理工具
 */
public class FileUtil {

 public static boolean copy(File src, File dest) {
     BufferedInputStreambis = null;
     BufferedOutputStreambos = null;
     try {
         bis = new BufferedInputStream(new FileInputStream(src));
         bos = new BufferedOutputStream(new FileOutputStream(dest));
         byte[] bts = new byte[1024];
         intlen = -1;
         while ((len = bis.read(bts)) != -1) {
             bos.write(bts, 0, len);
         }
         return true;
     } catch (Exception e) {
         e.printStackTrace();
```

```
            return false;
        } finally {
            if (bis != null) {
                try {
                    bis.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            if (bos != null) {
                try {
                    bos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

    }
}
```

struts.xml 完整代码：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1.7//EN"
    "http://struts.apache.org/dtds/struts-2.1.7.dtd">
<struts>

 <!-- 客户配置信息 -->
 <package name="customer"
     namespace="/customer" extends="struts-default">
     <interceptors>
         <!-- 注册拦截器 -->
         <interceptor name="first"
             class="interceptor.FirstInterceptor"/>
         <interceptor name="second"
             class="interceptor.SecondInterceptor"/>
         <!-- 注册拦截器栈 -->
         <interceptor-stack name="myStack">
             <interceptor-ref name="first"/>
             <interceptor-ref name="second"/>
         </interceptor-stack>
     </interceptors>
     <!-- 设置当前包下所有 Action 默认引用的拦截器 -->
     <default-interceptor-ref name="myStack"/>

     <!-- 打开修改页面 -->
     <action name="toUpdateCustomer"
         class="action.ToUpdateCustomerAction">
         <!-- 引用拦截器 -->
         <!--<interceptor-ref name="first"/> -->
         <result name="success">
             /WEB-INF/customer/update customer.jsp
         </result>
     </action>
 </package>

 <!-- 设置上传文件最大值 -->
 <constant name="struts.multipart.maxSize" value="5000000"/>

 <!-- 文件上传示例 -->
 <package name="demo" namespace="/demo" extends="struts-default">
     <!-- 打开上传文件页面 -->
```

```
    <action name="toUpload">
        <result name="success">/WEB-INF/jsp/upload.jsp</result>
    </action>
    <!-- 上传文件 -->
    <action name="upload" class="action.UploadAction">
        <result name="success">/WEB-INF/jsp/ok.jsp</result>
        <result name="error">/WEB-INF/jsp/error.jsp</result>
    </action>
</package>

</struts>
```

UploadAction 完整代码：

```
package action;

importjava.io.File;
import org.apache.struts2.ServletActionContext;
importutil.FileUtil;

/**
 * 文件上传 Action
 */
public class UploadAction {

 /**
  * 接收拦截器传入的临时文件
  */
 private File some;
 /**
  * 接收拦截器注入的原始文件名
  */
 private String someFileName;

 publicString execute() {
     if (some == null)
         return "error";

     // 将文件放于项目部署路径下的 upload 文件夹下
     String path = "WEB-INF/upload/" + someFileName;
     // 根据相对部署路径计算完整路径
     path = ServletActionContext.getServletContext().getRealPath(path);
     // 将临时文件复制到上述路径下
     FileUtil.copy(some, new File(path));

     return "success";
 }

 public File getSome() {
     return some;
 }

 public void setSome(File some) {
     this.some = some;
 }

 public String getSomeFileName() {
     returnsomeFileName;
 }

 public void setSomeFileName(String someFileName) {
     this.someFileName = someFileName;
 }

}
```

upload.jsp 完整代码：

```
<%@page pageEncoding="utf-8"%>
<html>
<head></head>
<body>
 <!--
      上传文件对表单有 2 个要求：
      1、method="post"
      2、enctype="multipart/form-data"
  -->
 <form action="upload" method="post" enctype="multipart/form-data">
     <input type="file" name="some" />
     <input type="submit" value="提交" />
 </form>
</body>
</html>
```

ok.jsp 完整代码：

```
<html>
<head></head>
<body>
 <h1>OK!</h1>
</body>
</html>
```

error.jsp 完整代码：

```
<html>
<head></head>
<body>
 <h1 style="color:red">Error!</h1>
</body>
</html>
```

课后作业

1. Struts2 中的拦截器有什么用，与 Spring 中的 AOP 有什么区别和联系

2. Action 默认引用哪个拦截器，如果没有默认引用，会导致什么问题

3. Struts2 中如何实现文件上传