

arena teaching system

Spring+Struts+Hibernate开发实战



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. Spring 整合 Hibernate	3
1.1. 整合步骤	3
1.1.1. 导包	3
1.1.2. 配置 applicationContext.xml	3
1.1.3. 创建实体类和映射关系文件	3
1.1.4. 创建 DAO 接口及实现类	3
1.1.5. 声明 DAO 组件，注入 SessionFactory	4
2. Spring 整合 Struts2	4
2.1. 整合步骤	4
2.1.1. 导包	4
2.1.2. 配置 web.xml	5
2.1.3. 配置 applicationContext.xml	5
2.1.4. 创建并声明 Action	5
2.1.5. 配置 Action	6
2.1.6. 创建 JSP	6
2.2. 整合原理	6
2.2.1. 原理图	6
2.2.2. 整合的基本原理	7
3. SSH 整合	7
3.1. 整合步骤	7
3.1.1. 导包	7
3.1.2. 使用 Spring 整合 Hibernate	8
3.1.3. 使用 Spring 整合 Struts2	8
经典案例	9
1. 使用 Spring 整合 Hibernate	9
2. 使用 Spring 整合 Struts2	23
3. SSH 整合	29
Unit02	48
1. SSH 中的分页查询	49
1.1. SSH 分页查询的核心问题	49
1.1.1. 如何获取 Session	49
1.1.2. 代码执行顺序	49
2. SSH 中的延迟加载	50

2.1. SSH 中如何使用延迟加载方法	50
2.1.1. 使用延迟加载的问题	50
2.1.2. 如何使用 Open session in view	50
3. SSH 中的修改问题	51
3.1. 如何解决 SSH 中的修改问题	51
3.1.1. 修改时有什么问题	51
3.1.2. 如何解决这个问题	51
4. 补充案例	52
4.1. 资费新增	52
4.1.1. 资费新增功能开发思路	52
4.2. 资费删除	52
4.2.1. 资费删除功能开发思路	52
4.3. 异常处理	53
4.3.1. SSH 中如何处理异常	53
经典案例	54
1. SSH 中的分页查询	54
2. SSH 中使用延迟加载	67
3. 使用 Hibernate 进行修改	85
4. 资费新增	101
5. 资费删除	121
6. 异常处理	136

Spring + Struts + Hibernate 开发实战

Unit01

知识体系.....Page 3

Spring 整合 Hibernate	整合步骤	导包
		配置 applicationContext.xml
		创建实体类和映射关系文件
		创建 DAO 接口及实现类
		声明 DAO 组件，注入 SessionFactory
Spring 整合 Struts2	整合步骤	导包
		配置 web.xml
		配置 applicationContext.xml
		创建并声明 Action
		配置 Action
		创建 JSP
	整合原理	原理图
		整合的基本原理
SSH 整合	整合步骤	导包
		使用 Spring 整合 Hibernate
		使用 Spring 整合 Struts2

经典案例.....Page 9

使用 Spring 整合 Hibernate	导包
	配置 applicationContext.xml
	创建实体类和映射关系文件
	创建 DAO 接口及实现类
	声明 DAO 组件，注入 SessionFactory
使用 Spring 整合 Struts2	导包
	配置 web.xml

	配置 applicationContext.xml
	创建并声明 Action
	配置 Action
	创建 JSP
SSH 整合	导包
	使用 Spring 整合 Hibernate
	使用 Spring 整合 Struts2

1.1.4. 【整合步骤】创建 DAO 接口及实现类




创建DAO接口及实现类

- 创建DAO接口
- 创建DAO实现类，继承于HibernateDaoSupport，并实现DAO接口
- 在实现接口方法时，可以通过getHibernateTemplate()方法获取到HibernateTemplate工具类，用这个类来实现增删改查。Spring将Hibernate的API封装在这个类中，使用时更加简单




1.1.5. 【整合步骤】声明 DAO 组件，注入 SessionFactory



声明DAO组件，注入SessionFactory


- 声明DAO组件，将其纳入到Spring容器中
- 由于HibernateDaoSupport依赖于SessionFactory，因此需要在DAO实现类中注入SessionFactory



2. Spring 整合 Struts2


2.1. 整合步骤

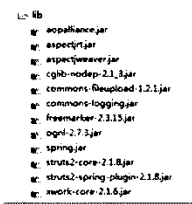
2.1.1. 【整合步骤】导包





导包

- 导入如下的包
 - 导入Struts2开发包
 - 导入Spring开发包
 - 导入Spring整合Struts2的开发包
- 完成之后，项目中的包结构如右图









2.1.2. 【整合步骤】配置 web.xml

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>配置web.xml</h4> <ul style="list-style-type: none"> 配置listener，用于在tomcat启动时自动加载Spring <pre> <listener> <listener-class> org.springframework.web.context.ContextLoaderListener </listener-class> </listener> <context-param> <param-name>contextConfigLocation</param-name> <param-value>classpath:applicationContext.xml</param-value> </context-param> </pre> <ul style="list-style-type: none"> 配置Struts2的前端控制器 <div style="text-align: right;">  </div>
---	---


2.1.3. 【整合步骤】配置 applicationContext.xml

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>配置applicationContext.xml</h4> <ul style="list-style-type: none"> 创建applicationContext.xml，开启注解扫描 <div style="text-align: right;">  </div>
---	---

2.1.4. 【整合步骤】创建并声明 Action


<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h4>创建并声明Action</h4> <ul style="list-style-type: none"> 创建Action 使用注解声明Action组件，将其纳入到Spring容器中 <div style="text-align: right;">  </div>
---	--

2.1.5. 【整合步骤】配置 Action



配置Action

- 在struts.xml中，配置Action
- 由于使用了Spring管理Action，在action元素中通过class属性指定Action类型时，需要指定Spring中的bean，即class= “组件ID”



2.1.6. 【整合步骤】创建 JSP




创建JSP

- 创建JSP，输出Action中的输出属性



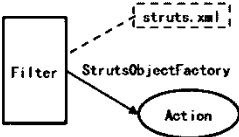
2.2. 整合原理

2.2.1. 【整合原理】原理图

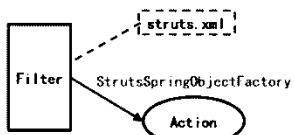



原理图

引入Spring前



引入Spring后





Tarena
达内科技

- ## 知识讲解



Tarena
达内教育

- ## 知识讲解

```
### if specified, the default object factory can be overridden here
### Note: short-hand notation is supported in some cases, such as "spring"
### Alternatively, you can provide a com.opensymphony.xwork2.ObjectFactory
$ struts.objectFactory = spring
```



3.1.1. 【整合步骤】导包

Tarena
結肉野郎

- 知识讲解

- `erle` 2.7.0.jar
- `osopallance.jar`
- `sun-atlas.jar`
- `sun.jar`
- `staxopt.jar`
- `staxopt-servlet.jar`
- `chp0-0.9.1.jar`
- `cglib-nodep-2.1.3.jar`
- `commons-collections-2.1.1.jar`
- `commons-logging-1.2.1.jar`
- `commons-logging-1.0.4.jar`
- `dom4j-1.6.1.jar`
- `freemarker-2.3.15.jar`
- `hibernate-tools.jar`
- `hibernate.jar`
- `jax.jar`
- `jax-2.4.jar`
- `log4j-1.2.11.jar`
- `ognl-2.7.3.jar`
- `ojdbc6.jar`
- `spring.jar`
- `struts-core-2.1.8.jar`
- `struts-plugin-2.1.8.jar`
- `struts-spring-plugin-2.1.8.jar`
- `struts-2.1.8.jar`



3.1.2. 【整合步骤】使用 Spring 整合 Hibernate

使用Spring整合Hibernate

达内科技

- 使用Spring整合Hibernate，完成DAO的接口及实现类，该操作请参考Spring整合Hibernate的步骤。

项目步骤

3.1.3. 【整合步骤】使用 Spring 整合 Struts2

使用Spring整合Struts2

达内科技

- 使用Spring整合Struts2，完成对请求的处理，该操作请参考Spring整合Struts2的步骤。

项目步骤

经典案例

1. 使用 Spring 整合 Hibernate

- 问题

使用 Spring 整合 Hibernate，并实现资费表的增、删、改、查。

- 方案

Spring 整合 Hibernate 的步骤：

- 1) 导包
- 2) 配置 applicationContext.xml
- 3) 创建实体类和映射关系文件
- 4) 创建 DAO 接口及实现类
- 5) 声明 DAO 组件，注入 SessionFactory

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：导包

创建 WEB 项目 SpringHibernate，并导入数据库驱动包、Hibernate 开发包以及 Spring 开发包，完成后项目中包结构如下图所示：



图-1

步骤二：配置 applicationContext.xml

引入 Spring 配置文件 applicationContext.xml，放在 src 根路径下。在该文件中配置数据源、SessionFactory、开启组件扫描、声明式事务，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xsi:schemaLocation="
           http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
           http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
           http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
           http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
           http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">

    <!-- 配置数据源 -->
    <bean id="ds"
          class="com.mchange.v2.c3p0.ComboPooledDataSource"
          destroy-method="close">

        <!-- 配置连接参数 -->
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe"/>
        <property name="driverClass" value="oracle.jdbc.OracleDriver"/>
        <property name="user" value="lhh"/>
        <property name="password" value="123456"/>

        <!-- 配置连接池 -->
        <property name="initialPoolSize" value="3"/>
        <property name="maxPoolSize" value="10"/>
        <property name="minPoolSize" value="1"/>
        <property name="acquireIncrement" value="3"/>
        <property name="maxIdleTime" value="60"/>
    </bean>

    <!-- 配置 SessionFactory -->
    <bean id="sessionFactory"
          class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

        <!-- 依赖数据源 -->
        <property name="dataSource" ref="ds"/>

        <!-- Hibernate 框架相关配置 -->
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">
                    org.hibernate.dialect.OracleDialect
                </prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.format_sql">true</prop>
            </props>
        </property>
    </bean>

    <!-- 开启注解扫描 -->
    <context:component-scan base-package="com.tarena"/>

```

```
<!-- 声明式事务管理,采用 AOP 形式切入 -->
<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="update*" propagation="REQUIRED" />
        <tx:method name="delete*" propagation="REQUIRED" />
        <tx:method name="add*" propagation="REQUIRED" />
        <tx:method name="load*" read-only="true" />
        <tx:method name="execute" propagation="REQUIRED" />
    </tx:attributes>
</tx:advice>
<aop:config proxy-target-class="true">
    <aop:advisor advice-ref="txAdvice"
        pointcut="within(com.tarena.action.*)" />
</aop:config>

</beans>
```

步骤三：创建实体类和映射关系文件

创建 com.tarena.entity 包，并在包下创建资费实体类和映射关系文件，这两个文件可以从 NETCTOSS 项目中复制过来。其中实体类 Cost 代码如下所示：

```
package com.tarena.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;
}
}
```

映射关系文件 Cost.hbm.xml 代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.tarena.entity.Cost" table="cost">
        <id name="id" type="integer" column="id">
            <!-- 用来指明主键的生成方式 -->
            <generator class="sequence">
                <param name="sequence">cost seq</param>
            </generator>
        </id>

        <property name="name"
            type="string" column="name" />
        <property name="baseDuration"
            type="integer" column="base_duration" />
        <property name="baseCost"
            type="double" column="base cost" />
        <property name="unitCost"
            type="double" column="unit cost" />
        <property name="status"
            type="string" column="status" />
        <property name="descr"
            type="string" column="descr" />
        <property name="createTime"
            type="date" column="createime" />
        <property name="startTime"
            type="date" column="starttime" />
        <property name="costType"
            type="string" column="cost_type" />
    </class>
</hibernate-mapping>
```

在 applicationContext.xml 中注册映射关系文件，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xsi:schemaLocation="
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">

    <!-- 配置数据源 -->
    <bean id="ds"
        class="com.mchange.v2.c3p0.ComboPooledDataSource"
        destroy-method="close">

        <!-- 配置连接参数 -->
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe"/>
```



```

<property name="driverClass" value="oracle.jdbc.OracleDriver"/>
<property name="user" value="lhh"/>
<property name="password" value="123456"/>

<!-- 配置连接池 -->
<property name="initialPoolSize" value="3"/>
<property name="maxPoolSize" value="10"/>
<property name="minPoolSize" value="1"/>
<property name="acquireIncrement" value="3"/>
<property name="maxIdleTime" value="60"/>
</bean>

<!-- 配置 SessionFactory -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <!-- 依赖数据源 -->
    <property name="dataSource" ref="ds"/>
    <!-- Hibernate 框架相关配置 -->
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.OracleDialect
            </prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.format_sql">true</prop>
        </props>
    </property>

    <property name="mappingResources">
        <list>
            <value>com/tarena/entity/Cost.hbm.xml</value>
        </list>
    </property>

</bean>

<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

<!-- 声明式事务管理, 采用 AOP 形式切入 -->
<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="update*" propagation="REQUIRED" />
        <tx:method name="delete*" propagation="REQUIRED" />
        <tx:method name="add*" propagation="REQUIRED" />
        <tx:method name="load*" read-only="true" />
        <tx:method name="execute" propagation="REQUIRED" />
    </tx:attributes>
</tx:advice>
<aop:config proxy-target-class="true">
    <aop:advisor advice-ref="txAdvice"
        pointcut="within(com.tarena.action.*)" />
</aop:config>

</beans>

```

步骤四：

创建包 `com.tarena.dao`，并在包下创建资费 DAO 接口，声明增、删、改、查的方法，代码如下所示：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    Cost findById(int id);

    void save(Cost cost);

    void update(Cost cost);

    void delete(int id);

}
```

创建 DAO 实现类 `CostDaoImpl` 继承于 `HibernateDaoSupport` 实现接口 `ICostDao`。代码如下所示：

```
package com.tarena.dao;

import java.util.List;
import javax.annotation.Resource;

import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;

import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public Cost findById(int id) {
        return (Cost) getHibernateTemplate().get(Cost.class, id);
    }

    @Override
    public void save(Cost cost) {
        getHibernateTemplate().save(cost);
    }

}
```

```
@Override
public void update(Cost cost) {
    getHibernateTemplate().update(cost);
}

@Override
public void delete(int id) {
    Cost c = new Cost();
    c.setId(id);
    getHibernateTemplate().delete(c);
}
}
```

步骤五：测试

在 `com.tarena.dao` 包下，创建 JUNIT 测试类 `TestDao`，分别写出资费的增、删、改、查测试方法，并执行这些方法进行测试。代码如下所示：

```
package com.tarena.dao;

import java.util.List;

import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.tarena.entity.Cost;

public class TestDao {

    private String conf = "applicationContext.xml";

    @Test
    public void test1() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        List<Cost> list = dao.findAll();
        for(Cost c : list) {
            System.out.println(
                c.getId() + " " + c.getName()
            );
        }
    }

    @Test
    public void test2() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = dao.findById(1);
        System.out.println(
            c.getId() + " " + c.getName()
        );
    }

    @Test
    public void test3() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = new Cost();
        c.setName("aaa");
        c.setBaseDuration(20);
    }
}
```

```

        c.setBaseCost(2.0);
        c.setUnitCost(0.2);
        c.setCostType("1");
        c.setStatus("0");
        dao.save(c);
    }

    @Test
    public void test4() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = dao.findById(351);
        c.setName("bbb");
        dao.update(c);
    }

    @Test
    public void test5() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        dao.delete(351);
    }
}

```

• 完整代码

本案例中 applicationContext.xml 的完整代码如下所示：

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xsi:schemaLocation="
        http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
        http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">

    <!-- 配置数据源 -->
    <bean id="ds"
        class="com.mchange.v2.c3p0.ComboPooledDataSource"
        destroy-method="close">
        <!-- 配置连接参数 -->
        <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe"/>
        <property name="driverClass" value="oracle.jdbc.OracleDriver"/>
        <property name="user" value="lhh"/>
        <property name="password" value="123456"/>
        <!-- 配置连接池 -->
        <property name="initialPoolSize" value="3"/>
        <property name="maxPoolSize" value="10"/>
        <property name="minPoolSize" value="1"/>
        <property name="acquireIncrement" value="3"/>
        <property name="maxIdleTime" value="60"/>
    </bean>

```

```

</bean>

<!-- 配置 SessionFactory -->
<bean id="sessionFactory"
      class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
  <!-- 依赖数据源 -->
  <property name="dataSource" ref="ds"/>
  <!-- Hibernate 框架相关配置 -->
  <property name="hibernateProperties">
    <props>
      <prop key="hibernate.dialect">
        org.hibernate.dialect.OracleDialect
      </prop>
      <prop key="hibernate.show_sql">true</prop>
      <prop key="hibernate.format_sql">true</prop>
    </props>
  </property>
  <property name="mappingResources">
    <list>
      <value>com/tarena/entity/Cost.hbm.xml</value>
    </list>
  </property>
</bean>

<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

<!-- 声明式事务管理, 采用 AOP 形式切入 -->
<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">
  <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
  <tx:attributes>
    <tx:method name="update*" propagation="REQUIRED" />
    <tx:method name="delete*" propagation="REQUIRED" />
    <tx:method name="add*" propagation="REQUIRED" />
    <tx:method name="load*" read-only="true" />
    <tx:method name="execute" propagation="REQUIRED" />
  </tx:attributes>
</tx:advice>
<aop:config proxy-target-class="true">
  <aop:advisor advice-ref="txAdvice"
    pointcut="within(com.tarena.action.*)" />
</aop:config>

</beans>

```

实体类 Cost 完整代码如下所示：

```

package com.tarena.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;
}
}
```

映射关系文件 Cost.hbm.xml 完整代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.tarena.entity.Cost" table="cost">
        <id name="id" type="integer" column="id">
            <!-- 用来指明主键的生成方式 -->
            <generator class="sequence">
                <param name="sequence">cost_seq</param>
            </generator>
        </id>

        <property name="name"
            type="string" column="name" />
        <property name="baseDuration"
            type="integer" column="base_duration" />
        <property name="baseCost"
            type="double" column="base_cost" />
        <property name="unitCost"
            type="double" column="unit cost" />
        <property name="status"
            type="string" column="status" />
        <property name="descr"
            type="string" column="descr" />
        <property name="createTime"
            type="date" column="createime" />
        <property name="startTime"
            type="date" column="starttime" />
        <property name="costType"
            type="string" column="cost type" />
    </class>
</hibernate-mapping>
```

DAO 接口 ICostDao 完整代码如下所示：

```
package com.tarena.dao;

import java.util.List;
import com.tarena.entity.Cost;
```

```
public interface ICostDao {  
  
    List<Cost> findAll();  
  
    Cost findById(int id);  
  
    void save(Cost cost);  
  
    void update(Cost cost);  
  
    void delete(int id);  
  
}
```

DAO 实现类 CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;  
  
import java.util.List;  
  
import javax.annotation.Resource;  
  
import org.hibernate.SessionFactory;  
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;  
import org.springframework.stereotype.Repository;  
  
import com.tarena.entity.Cost;  
  
@Repository  
public class CostDaoImpl  
    extends HibernateDaoSupport implements ICostDao {  
  
    @Resource  
    public void setSF(SessionFactory sf) {  
        super.setSessionFactory(sf);  
    }  
  
    @Override  
    public List<Cost> findAll() {  
        String hql = "from Cost";  
        return getHibernateTemplate().find(hql);  
    }  
  
    @Override  
    public Cost findById(int id) {  
        return (Cost) getHibernateTemplate().get(Cost.class, id);  
    }  
  
    @Override  
    public void save(Cost cost) {  
        getHibernateTemplate().save(cost);  
    }  
  
    @Override  
    public void update(Cost cost) {  
        getHibernateTemplate().update(cost);  
    }  
  
    @Override  
    public void delete(int id) {  
        Cost c = new Cost();  
        c.setId(id);  
        getHibernateTemplate().delete(c);  
    }  
  
}
```


测试类 TestDao 完整代码如下所示：

```
package com.tarena.dao;

import java.util.List;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.tarena.entity.Cost;

public class TestDao {

    private String conf = "applicationContext.xml";

    @Test
    public void test1() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        List<Cost> list = dao.findAll();
        for(Cost c : list) {
            System.out.println(
                c.getId() + " " + c.getName()
            );
        }
    }

    @Test
    public void test2() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = dao.findById(1);
        System.out.println(
            c.getId() + " " + c.getName()
        );
    }

    @Test
    public void test3() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = new Cost();
        c.setName("aaa");
        c.setBaseDuration(20);
        c.setBaseCost(2.0);
        c.setUnitCost(0.2);
        c.setCostType("1");
        c.setStatus("0");
        dao.save(c);
    }

    @Test
    public void test4() {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        Cost c = dao.findById(351);
        c.setName("bbb");
        dao.update(c);
    }

    @Test
    public void test5() {
        ApplicationContext ctx =
```

```

        new ClassPathXmlApplicationContext(conf);
        ICostDao dao = (ICostDao) ctx.getBean("costDaoImpl");
        dao.delete(351);
    }
}

```

2. 使用 Spring 整合 Struts2

• 问题

使用 Spring 整合 Struts2，并实现 Struts2 的 HelloWorld 案例。

• 方案

Spring 整合 Struts2 步骤：

- 1) 导包
- 2) 配置 web.xml
- 3) 配置 applicationContext.xml
- 4) 创建并声明 DAO
- 5) 创建并声明 Action
- 6) 配置 Action
- 7) 创建 JSP

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：导包

创建 WEB 项目 SpringStruts2，并导入 Struts2 开发包、Spring 开发包以及 Spring 整合 Struts2 开发包，完成后项目中包结构如下图所示：

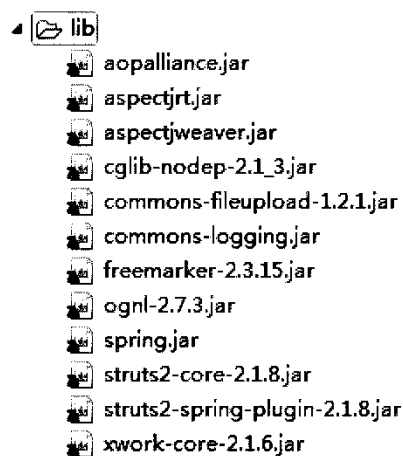


图-2

步骤二：配置 web.xml

在 web.xml 中配置一个 listener，用于 tomcat 启动时自动加载 Spring。再配置出 Struts2 的前端控制器，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  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_2_5.xsd">
  <display-name></display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <!-- 配置 listener，在容器启动时自动加载 Spring -->
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>

  <!-- 配置 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>

</web-app>
```

步骤三：配置 applicationContext.xml

引入 Spring 配置文件，放在 src 根路径下，在该文件下开启组件扫描，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:jee="http://www.springframework.org/schema/jee"
  xsi:schemaLocation="
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd
```

<http://www.springframework.org/schema/jee>
<http://www.springframework.org/schema/jee/spring-jee-2.5.xsd>>

```
<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

</beans>
```

步骤四：创建并声明 DAO

创建包 `com.tarena.dao`，并在包下创建一个类 `HelloDao`，在这个类中写一个 `say` 方法，其返回值为字符串，返回一句话即可。另外要使用 Spring 注解声明这个 DAO 组件，将其纳入到 Spring 容器中，代码如下所示：

```
package com.tarena.dao;

import org.springframework.stereotype.Repository;

@Repository
public class HelloDao {

    public String say() {
        return "Hello, Spring 整合 Struts2.";
    }

}
```

步骤五：创建并声明 Action

创建包 `com.tarena.action`，并在包下创建 `HelloAction`，在这个 Action 的业务方法中，调用 `HelloDao` 返回一句话并通过输出属性输出给页面。需要使用 Spring 注解声明这个 Action 组件，将其纳入到 Spring 容器中，另外也需要通过注解将 DAO 注入进来。代码如下所示：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao>HelloDao;

@Controller
public class HelloAction {

    @Resource
    private HelloDao dao;

    private String msg;

    public String execute() {
        // 通过 dao 获取输出消息
        msg = dao.say();
        return "success";
    }

    public String getMsg() {
        return msg;
    }

}
```

```
public void setMsg(String msg) {  
    this.msg = msg;  
}  
}
```

步骤六：配置 Action

在 `struts.xml` 中配置 Action。由于使用 Spring 管理 Action，在 `action` 元素中通过 `class` 属性指定 Action 类型时，需要指定的是组件的 ID。代码如下所示：

```
<?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 name="demo"  
        namespace="/demo" extends="struts-default">  
        <!--  
            由于使用 Spring 来管理 Action，因此需要通过 Spring  
            容器来创建 Action，因此 class 属性指定的是组件的 ID。  
        -->  
        <action name="hello" class="helloAction">  
            <result name="success">  
                /hello.jsp  
            </result>  
        </action>  
    </package>  
</struts>
```

步骤七：创建 JSP

在 `WebRoot` 下创建 `hello.jsp`，并在这个页面上输出 `HelloAction` 中的输出属性，代码如下所示：

```
<%@page pageEncoding="utf-8" isELIgnored="false"%>  
<%@taglib uri="/struts-tags" prefix="s"%>  
<html>  
    <head>  
    </head>  
    <body>  
  
        <h1>  
            <s:property value="msg"/>  
        </h1>  
  
    </body>  
</html>
```

步骤八：测试

部署项目并启动 tomcat，访问 `HelloAction`，页面输出效果如下图所示：

← ↻ 📄 localhost:8088/SpringStruts2/demo/hello
📁 IE 📁 NETCTOSS 📁 课程管理 📁 TarenaMail 📁 TTS7 📁 营养小厨 |

Hello, Spring整合Struts2.

图-3

• 完整代码

本案例中 web.xml 的完整代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
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_2_5.xsd">
    <display-name></display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <!-- 配置 listener, 在容器启动时自动加载 Spring -->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!-- 配置 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>
</web-app>
```

applicationContext.xml 完整代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:tx="http://www.springframework.org/schema/tx"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:jee="http://www.springframework.org/schema/jee"
xsi:schemaLocation="
    http://www.springframework.org/schema/tx
```

```
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">
```

```
<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

</beans>
```

HelloDao 完整代码如下所示：

```
package com.tarena.dao;

import org.springframework.stereotype.Repository;

@Repository
public class HelloDao {

    public String say() {
        return "Hello, Spring 整合 Struts2.";
    }

}
```

HelloAction 完整代码如下所示：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao.HelloDao;

@Controller
public class HelloAction {

    @Resource
    private HelloDao dao;

    private String msg;

    public String execute() {
        // 通过 dao 获取输出消息
        msg = dao.say();
        return "success";
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

}
```

struts.xml 完整代码如下所示：

```
<?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 name="demo"
    namespace="/demo" extends="struts-default">
    <!--
        由于使用 Spring 来管理 Action, 因此需要通过 Spring
        容器来创建 Action, 因此 class 属性指定的是组件的 ID。
    -->
    <action name="hello" class="helloAction">
        <result name="success">
            /hello.jsp
        </result>
    </action>
</package>
</struts>
```

hello.jsp 完整代码如下所示：

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

<h1>
    <s:property value="msg"/>
</h1>

</body>
</html>
```

3. SSH 整合

- 问题

使用 Spring 整合 Hibernate 及 Struts2 , 实现资费查询功能。

- 方案

SSH 整合步骤：

- 1) 导包
- 2) 使用 Spring 整合 Hibernate
- 3) 使用 Spring 整合 Struts2

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：导包

创建 WEB 项目 NETCTOSS-SSH，并导入数据库驱动包、Struts2 开发包、Hibernate 开发包以及 Spring 开发包，完成后项目中包结构如下图所示：



图-4

注意，Hibernate 开发包中的 freemarker.jar，与 Struts2 中的包重复，且版本较低，去掉。Spring 开发包中的 commons-logging.jar，与 Hibernate 中的包重复，且版本较低，去掉。

步骤二：使用 Spring 整合 Hibernate

在 src 根路径下引入并配置 applicationContext.xml，并在该文件中配置数据源、SessionFactory、开启组件扫描以及声明式事务，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:jee="http://www.springframework.org/schema/jee"
       xsi:schemaLocation="
           http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
           http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
           http://www.springframework.org/schema/beans
```

```
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">
```

```
<!-- 配置数据源 -->
<bean id="ds"
    class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
    <!-- 配置连接参数 -->
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe"/>
    <property name="driverClass" value="oracle.jdbc.OracleDriver"/>
    <property name="user" value="lhh"/>
    <property name="password" value="123456"/>
    <!-- 配置连接池 -->
    <property name="initialPoolSize" value="3"/>
    <property name="maxPoolSize" value="10"/>
    <property name="minPoolSize" value="1"/>
    <property name="acquireIncrement" value="3"/>
    <property name="maxIdleTime" value="60"/>
</bean>

<!-- 配置SessionFactory -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <!-- 依赖数据源 -->
    <property name="dataSource" ref="ds"/>
    <!-- Hibernate 框架相关配置 -->
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.OracleDialect
            </prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.format_sql">true</prop>
        </props>
    </property>
    <property name="mappingResources">
        <list>
            <value>com/tarena/entity/Cost.hbm.xml</value>
        </list>
    </property>
</bean>

<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

<!-- 声明式事务管理, 采用 AOP 形式切入 -->
<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="update*" propagation="REQUIRED" />
        <tx:method name="delete*" propagation="REQUIRED" />
        <tx:method name="add*" propagation="REQUIRED" />
        <tx:method name="load*" read-only="true" />
        <tx:method name="execute" propagation="REQUIRED" />
    </tx:attributes>
</tx:advice>
<aop:config proxy-target-class="true">
```

```
<aop:advisor advice-ref="txAdvice"
    pointcut="within(com.tarena.action.*)" />
</aop:config>

</beans>
```

在 `com.tarena.entity` 包下创建实体类 `Cost`，代码如下所示：

```
package com.tarena.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;
}
}

```

在 `com.tarena.entity` 包下创建映射关系文件 `Cost.hbm.xml`，代码如下所示：

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.tarena.entity.Cost" table="cost">
    <id name="id" type="integer" column="id">
        <!-- 用来指明主键的生成方式 -->
        <generator class="sequence">
            <param name="sequence">cost seq</param>
        </generator>
    </id>

```

```
<property name="name"
    type="string" column="name" />
<property name="baseDuration"
    type="integer" column="base_duration" />
<property name="baseCost"
    type="double" column="base cost" />
<property name="unitCost"
    type="double" column="unit cost" />
<property name="status"
    type="string" column="status" />
<property name="descr"
    type="string" column="descr" />
<property name="createTime"
    type="date" column="create_time" />
<property name="startTime"
    type="date" column="start_time" />
<property name="costType"
    type="string" column="cost_type" />
</class>
</hibernate-mapping>
```

在 `com.tarena.dao` 包下创建 DAO 接口 `ICostDao` ,并在接口中声明查询全部资费数据的方法，代码如下所示：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

}
```

在 `com.tarena.dao` 包下创建 DAO 实现类 `CostDaoImpl` ,继承于 `HibernateDaoSupport` ,并实现接口 `ICostDao` ,代码如下所示：

```
package com.tarena.dao;

import java.util.List;
import javax.annotation.Resource;

import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;

import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
```

```
        return getHibernateTemplate().find(hql);
    }
}
```

步骤三：使用 Spring 整合 Struts2

配置 web.xml，配置一个 listener，使 tomcat 启动时自动加载 Spring。另外再配置 Struts2 的前端控制器，代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name></display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <!-- 配置 listener，使 tomcat 启动时自动加载 Spring -->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!-- 配置前端控制器 -->
    <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>
```

在 com.tarena.action 包下创建 Action 类 FindCostAction，调用 DAO 实现对资费的查询，代码如下所示：

```
package com.tarena.action;

import java.util.List;
import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

@Controller
@Scope("prototype")
public class FindCostAction {

    @Resource
```

```
private ICostDao costDao;

private List<Cost> costs;

public String load() {
    costs = costDao.findAll();
    return "success";
}

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

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

在 src 根路径下引入 struts.xml 并配置 Action，代码如下所示：

```
<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
    </package>
</struts>
```

在 WEB-INF/cost 下创建页面 find_cost.jsp，实现资费数据的显示，代码如下所示：

```
<%@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() {
    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">
        
        <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>
        </form>
    </div>
</body>
</html>
```



```

<!--启用操作的操作提示-->
<div id="operate_result_info" class="operate_success">
    
    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();" />
                </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

部署项目并启动 tomcat，访问资费列表，效果如下图所示：

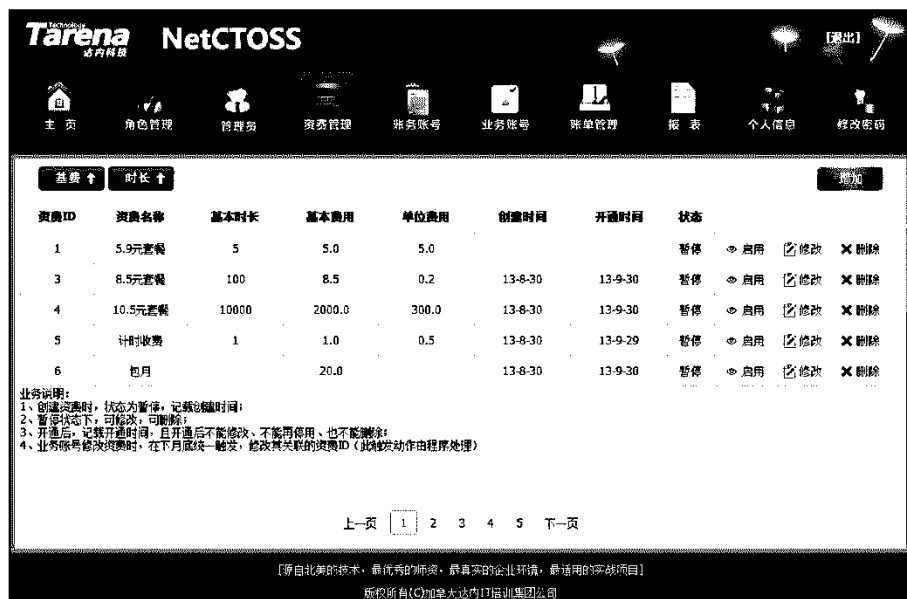


图-5

完整代码

本案例中 applicationContext.xml 的完整代码如下所示：

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:jee="http://www.springframework.org/schema/jee"

```

```
xsi:schemaLocation="
    http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.5.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://www.springframework.org/schema/jee
http://www.springframework.org/schema/jee/spring-jee-2.5.xsd">

<!-- 配置数据源 -->
<bean id="ds"
    class="com.mchange.v2.c3p0.ComboPooledDataSource"
    destroy-method="close">
    <!-- 配置连接参数 -->
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe"/>
    <property name="driverClass" value="oracle.jdbc.OracleDriver"/>
    <property name="user" value="lhh"/>
    <property name="password" value="123456"/>
    <!-- 配置连接池 -->
    <property name="initialPoolSize" value="3"/>
    <property name="maxPoolSize" value="10"/>
    <property name="minPoolSize" value="1"/>
    <property name="acquireIncrement" value="3"/>
    <property name="maxIdleTime" value="60"/>
</bean>

<!-- 配置SessionFactory -->
<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
    <!-- 依赖数据源 -->
    <property name="dataSource" ref="ds"/>
    <!-- Hibernate 框架相关配置 -->
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">
                org.hibernate.dialect.OracleDialect
            </prop>
            <prop key="hibernate.show_sql">true</prop>
            <prop key="hibernate.format_sql">true</prop>
        </props>
    </property>
    <property name="mappingResources">
        <list>
            <value>com/tarena/entity/Cost.hbm.xml</value>
        </list>
    </property>
</bean>

<!-- 开启注解扫描 -->
<context:component-scan base-package="com.tarena"/>

<!-- 声明式事务管理, 采用 AOP 形式切入 -->
<bean id="txManager"

class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
        <tx:method name="update*" propagation="REQUIRED" />
        <tx:method name="delete*" propagation="REQUIRED" />
        <tx:method name="add*" propagation="REQUIRED" />
        <tx:method name="load*" read-only="true" />
        <tx:method name="execute" propagation="REQUIRED" />
    </tx:attributes>
</tx:advice>
```

```

        </tx:attributes>
    </tx:advice>
    <aop:config proxy-target-class="true">
        <aop:advisor advice-ref="txAdvice"
            pointcut="within(com.tarena.action.*)" />
    </aop:config>

</beans>

```

Cost 完整代码如下所示：

```

package com.tarena.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;
    }
}
```

Cost.hbm.xml 完整代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.tarena.entity.Cost" table="cost">
        <id name="id" type="integer" column="id">
            <!-- 用来指明主键的生成方式 -->
            <generator class="sequence">
                <param name="sequence">cost seq</param>
            </generator>
        </id>

        <property name="name"
            type="string" column="name" />
    </class>
</hibernate-mapping>
```

```
<property name="baseDuration"
    type="integer" column="base duration" />
<property name="baseCost"
    type="double" column="base_cost" />
<property name="unitCost"
    type="double" column="unit cost" />
<property name="status"
    type="string" column="status" />
<property name="descr"
    type="string" column="descr" />
<property name="createTime"
    type="date" column="createTime" />
<property name="startTime"
    type="date" column="startTime" />
<property name="costType"
    type="string" column="cost_type" />
</class>
</hibernate-mapping>
```

ICostDao 完整代码如下所示：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

}
```

CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;

import java.util.List;

import javax.annotation.Resource;

import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;

import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

}
```

web.xml 完整代码如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name></display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <!-- 配置 listener, 使 tomcat 启动时自动加载 Spring -->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!-- 配置前端控制器 -->
    <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>
```

FindCostAction 完整代码如下所示：

```
package com.tarena.action;

import java.util.List;
import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

@Controller
@Scope("prototype")
public class FindCostAction {

    @Resource
    private ICostDao costDao;

    private List<Cost> costs;

    public String load() {
        costs = costDao.findAll();
        return "success";
    }

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

```

    }

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

}

```

struts.xml 完整代码如下所示：

```

<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
    </package>
</struts>

```

find_cost.jsp 完整代码如下所示：

```

<%@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() {
                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">
        
        <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">
                
                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>
                    </tr>
                </table>
            </div>
        </form>
    </div>

```

```

        <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();" />
            </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>

```

Spring + Struts + Hibernate 开发实战

Unit02

知识体系.....Page 49

SSH 中的分页查询	SSH 分页查询的核心问题	如何获取 Session
		代码执行顺序
SSH 中的延迟加载	SSH 中如何使用延迟加载方法	使用延迟加载的问题
		如何使用 Open session in view
SSH 中的修改问题	如何解决 SSH 中的修改问题	修改时有什么问题
		如何解决这个问题
补充案例	资费新增	资费新增功能开发思路
	资费删除	资费删除功能开发思路
	异常处理	SSH 中如何处理异常

经典案例.....Page 54

SSH 中的分页查询	如何获取 Session
	代码执行顺序
SSH 中使用延迟加载	如何使用 Open session in view
使用 Hibernate 进行修改	如何解决这个问题
资费新增	资费新增功能开发思路
资费删除	资费删除功能开发思路
异常处理	SSH 中如何处理异常

1. SSH 中的分页查询

1.1. SSH 分页查询的核心问题

1.1.1. 【SSH 分页查询的核心问题】如何获取 Session

Tarena
达内科技

如何获取Session

- Hibernate中的分页查询是通过Query对象设置参数来统一实现的，而使用Spring整合Hibernate后，session由Spring统一负责维护，并且没有直接暴露给开发者。如果想在这样的程序中获取到session，从而创建Query对象实现分页的话，需要使用HibernateTemplate中的executeFind方法实现查询，代码结构如下

Tarena
达内科技

Tarena
达内科技

如何获取Session (续1)

```

getHibernateTemplate().executeFind(new HibernateCallback() {
    @Override
    public Object doInHibernate(Session session)
        throws HibernateException, SQLException {
        // 在这里使用session
        return null;
    }
});
        
```

Tarena
达内科技

1.1.2. 【SSH 分页查询的核心问题】代码执行顺序

Tarena
达内科技

代码执行顺序

- executeFind方法中代码执行顺序如下图

```

graph TD
    Start(( )) --> CreateSession[创建session]
    CreateSession --> Decision1{存在事务?}
    Decision1 -- Y --> StartTransaction[开启事务]
    Decision1 -- N --> PassSession[为HibernateCallback对象传入session]
    StartTransaction --> CallDoInHibernate[调用 HibernateCallback.doInHibernate()]
    PassSession --> CallDoInHibernate
    CallDoInHibernate --> Decision2{存在事务?}
    Decision2 -- Y --> CloseTransaction[关闭事务]
    Decision2 -- N --> CloseSession[关闭Session]
    CloseTransaction --> End(( ))
    CloseSession --> End
        
```

Tarena
达内科技

2. SSH 中的延迟加载

2.1. SSH 中如何使用延迟加载方法

2.1.1. 【SSH 中如何使用延迟加载方法】使用延迟加载的问题


[illegible]

2.1.2. 【SSH 中如何使用延迟加载方法】如何使用 Open session in view


知识讲解

如何使用Open session in view

- 项目中可以使用Open session in view技术来解决延迟加载的问题，而该技术的实现手段有多种
 - Servlet下可以使用filter实现。
 - Struts2下可以使用Interceptor实现。
 - Spring下可以使用AOP实现。
- 在SSH框架下，一般采用filter方式实现，原因是Spring已经预置了一个实现Open session in view的filter，只需要配置这个filter即可，该filter的类名为org.springframework.orm.hibernate3.support.OpenSessionInViewFilter





Tarena
达内科技





3. SSH 中的修改问题



3.1. 如何解决 SSH 中的修改问题



3.1.1. 【如何解决 SSH 中的修改问题】修改时有什么问题

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h4 style="margin-top: 0;">修改时有什么问题</h4> <ul style="list-style-type: none"> 使用Hibernate修改时，它会根据映射关系文件，自动拼出一个update语句，然后执行修改，其中映射关系文件中往往配置表中全部的字段，而很多时候，页面上不需要修改表中全部的字段，只需要修改其中的一部分，因此页面上的字段少于表中字段，在提交保存时，缺少的字段就成了空值，那么再按照完整的update语句来执行更新，就会把这些不需要更新的字段更新为空。 <div style="text-align: right;">  </div>
---	---

3.1.2. 【如何解决 SSH 中的修改问题】如何解决这个问题

<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h4 style="margin-top: 0;">如何解决这个问题</h4> <p>方式一：提交完整字段</p> <ol style="list-style-type: none"> 提交时，保证所有字段都进行提交便可以解决问题。可以在表单中增加hidden标签，将那些不需要修改的字段回显在hidden中，这样既能使页面不显示这些字段，也能在提交时包含这些字段。 hidden标签语法如下，与文本框标签用法一致，可以将其看成是隐藏的文本框 <pre><s:hidden name= "OGNL" /></pre> 这种方式在不需要修改的字段很多时，需要写大量的hidden标签，并且执行效率也大大降低了。 <div style="text-align: right;">  </div>
---	--



<div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div> <div style="border-bottom: 1px solid black; height: 15px; margin-bottom: 5px;"></div>	<div style="text-align: right;">  </div> <h4 style="margin-top: 0;">如何解决这个问题（续1）</h4> <p>方式二：声明不需要更新的字段</p> <ol style="list-style-type: none"> 可以在映射关系文件中，property属性上设置update= "false" 来声明该字段在更新时不拼入update语句。 这种方式具有很大的局限性，因为有可能更新时不需要更新这些字段，但是在其他功能上却要更新这些字段，比如在修改时不需要更新状态，但是开通、暂停时却需要更新状态。而一旦设置了这个属性，那么就无法对其进行更新了，因此这种方式不推荐使用。 <div style="text-align: right;">  </div>
---	---

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>如何解决这个问题（续2）</h3> <p>方式三：动态更新</p> <ol style="list-style-type: none"> 1、可以在映射关系文件中，class属性上设置dynamic-update=“true”，来声明这个对象采用动态更新的方式，即更新时Hibernate会自动判断属性是否发生改变，若改变则将其拼入SQL，否则忽略。 2、这种方式要求传给Hibernate的对象必须是持久态的，而通过页面传入的对象是Struts2自动初始化的，是临时态的。我们可以通过ID查询出持久态对象，然后通过Spring中的BeanUtils工具类，将临时态对象的属性值复制给持久态对象，然后用这个持久态对象进行更新即可。 3、显然这种方式比较简单，推荐使用。 <div style="text-align: right;">  </div>
---	---

4. 补充案例



4.1. 资费新增

4.1.1. 【资费新增】资费新增功能开发思路

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>资费新增功能开发思路</h3> <ul style="list-style-type: none"> • 资费新增和资费修改十分相似，不同点在于新增页面不需要默认显示数据，因此该功能可以参考资费修改完成。 <div style="text-align: right;">  </div>
---	---

4.2. 资费删除

4.2.1. 【资费删除】资费删除功能开发思路

<hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/> <hr/>	<div style="text-align: right;">  </div> <h3>资费删除功能开发思路</h3> <ul style="list-style-type: none"> • 资费删除功能比较简单，在页面上选中一条资费，将ID传入Action，然后调用DAO使用Hibernate删除这条数据即可。 <div style="text-align: right;">  </div>
---	---

4.3. 异常处理

4.3.1. 【异常处理】SSH 中如何处理异常

知识讲解

SSH中如何处理异常

Tarena

达内科技

- 由于每个功能的业务代码中都有可能报错，因此这种处理异常的行为是所有业务代码都需要做的，是典型的通用业务逻辑，所以可以采用拦截器进行处理，发生异常时转向统一的错误页面即可。

++

经典案例

1. SSH 中的分页查询

- 问题

为资费列表页面增加分页功能。

- 方案

Hibernate 中的分页查询是通过 Query 对象设置参数来统一实现的，而使用 Spring 整合 Hibernate 后，session 由 Spring 统一负责维护，并且没有直接暴露给开发者。如果想在这样的程序中获取到 session，从而创建 Query 对象实现分页的话，需要使用 HibernateTemplate 中的 executeFind 方法实现查询，代码结构如下：

```
getHibernateTemplate().executeFind(new HibernateCallback() {  
  
    @Override  
    public Object doInHibernate(Session session)  
        throws HibernateException, SQLException {  
        // 在这里使用 session  
        return null;  
    }  
});
```

上面方法中，需要传入一个接口类型，我们往往传入一个匿名的对象，实现了该接口。这个接口 Spring 会自动调用并以它的返回值作为查询结果返回，在接口的实现中，我们可以使用 session 来自定义查询。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：在 DAO 中增加分页方法

在 ICostDao 接口中，增加分页方法，代码如下：

```
package com.tarena.dao;  
  
import java.util.List;  
  
import com.tarena.entity.Cost;  
  
public interface ICostDao {  
  
    List<Cost> findAll();  
  
    /**  
     * 查询某页资费数据  
     * @param page 页码
```

```

    * @param pageSize 页容量
    * @return
    */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);
}

```

在 CostDaoImpl 中，增加分页的实现方法，代码如下：

```

package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /**
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /**
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */

```

```

@Override
public Object doInHibernate(Session session)
    throws HibernateException, SQLException {
    String hql = "from Cost";
    Query query = session.createQuery(hql);
    /*
     * 设置分页参数，注意在内层函数中调用外层函数的参数，
     * 要求外层函数的参数必须是 final 的，因此需要将
     * page、pageSize 设置为 final。
     */
    query.setFirstResult((page-1)*pageSize);
    query.setMaxResults(pageSize);
    return query.list();
}

});
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}
}

```

步骤二：修改 Action，处理分页查询请求

修改 Action，增加分页输入条件 page、pageSize，增加输出条件 totalPage，并根据分页条件查询数据以及计算出总页数，代码如下：

```

package com.tarena.action;

import java.util.List;
import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

@Controller
@Scope("prototype")
public class FindCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private int page = 1;
    private int pageSize;

```

```
// output

private List<Cost> costs;

private int totalPage;

public String load() {

    costs = costDao.findByPage(page, pageSize);
    totalPage = costDao.findTotalPage(pageSize);

    return "success";
}

public int getPage() {
    return page;
}

public void setPage(int page) {
    this.page = page;
}

public int getPageSize() {
    return pageSize;
}

public void setPageSize(int pageSize) {
    this.pageSize = pageSize;
}

public int getTotalPage() {
    return totalPage;
}

public void setTotalPage(int totalPage) {
    this.totalPage = totalPage;
}

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

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

步骤三：注入页容量

修改 struts.xml，在查询 action 的配置中注入页容量 pageSize，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 name="cost"
    namespace="/cost" extends="json-default">
    <action name="findCost"
```

```
class="findCostAction" method="load">
```

```
<!-- 注入页容量 -->
```

```
<param name="pageSize">3</param>
```

```
<result name="success">
```

```
/WEB-INF/cost/find_cost.jsp
```

```
</result>
```

```
</action>
```

```
</package>
```

```
</struts>
```

步骤四：处理页面分页逻辑

修改 find_cost.jsp，增加分页的业务逻辑，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() {
        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">
      
      <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">
            
            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>
                </tr>
            </table>
        </div>
    </form>
</div>

```

```

        <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();" />
        </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；
    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">
    <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>

    <s:iterator begin="1" end="totalPage" var="p">
        <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>

```

```

        <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>
</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，访问资费列表功能，效果如下图所示：



图-1

完整代码

本案例中 ICostDao 的完整代码如下所示：

```

package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();
}

```



```
/**
 * 查询某页资费数据
 * @param page 页码
 * @param pageSize 页容量
 * @return
 */
List<Cost> findByPage(int page, int pageSize);

/**
 * 查询总页数
 * @param pageSize
 * @return
 */
int findTotalPage(int pageSize);
}
```

CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /**
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /**
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
```

```

        throws HibernateException, SQLException {
    String hql = "from Cost";
    Query query = session.createQuery(hql);
    /*
     * 设置分页参数，注意在内层函数中调用外层函数的参数，
     * 要求外层函数的参数必须是 final 的，因此需要将
     * page、pageSize 设置为 final。
     */
    query.setFirstResult((page-1)*pageSize);
    query.setMaxResults(pageSize);
    return query.list();
    }
    });
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}
}

```

FindCostAction 完整代码如下所示：

```

package com.tarena.action;

import java.util.List;
import javax.annotation.Resource;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

@Controller
@Scope("prototype")
public class FindCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private int page = 1;
    private int pageSize;

    // output
    private List<Cost> costs;
    private int totalPage;

    public String load() {
        costs = costDao.findByPage(page, pageSize);
        totalPage = costDao.findTotalPage(pageSize);
        return "success";
    }

    public int getPage() {
        return page;
    }
}

```

```

public void setPage(int page) {
    this.page = page;
}

public int getPageSize() {
    return pageSize;
}

public void setPageSize(int pageSize) {
    this.pageSize = pageSize;
}

public int getTotalPage() {
    return totalPage;
}

public void setTotalPage(int totalPage) {
    this.totalPage = totalPage;
}

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

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

```

struts.xml 完整代码如下所示：

```

<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find cost.jsp
            </result>
        </action>
    </package>
</struts>

```

find_cost.jsp 完整代码如下所示：

```

<%@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() {
        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">
        
        <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">
            
            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();" />
                        </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">
  <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:iterator begin="1" end="totalPage" var="p">
      <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>

    <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>

```

2. SSH 中使用延迟加载

- 问题

开发资费修改功能，要求可以打开修改页面，并显示出要修改的数据，其中对于修改数据的查询要求使用延迟加载的方法。

- 方案

由于 DAO 中使用延迟加载方法查询出修改数据，而该数据在修改页面中才被使用，即在使用数据时 session 已经关闭。因此，要想解决这个问题，需要使用 Open session in view 技术。在 SSH 中使用 Open session in view 技术很简单，只需要在 web.xml 中配置一个 filter 即可，并且这个 filter 已经由 Spring 预置好了，名为 org.springframework.orm.hibernate3.support.OpenSessionInViewFilter。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：在 DAO 中增加根据 ID 查询的方法

在 ICostDao 接口中，增加根据 ID 查询资费数据的方法，代码如下：

```
package com.tarena.dao;

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

public interface ICostDao {
    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据
     * @param id
     * @return
     */
    Cost findById(int id);
}
```

在 CostDaoImpl 中实现这个方法，代码如下：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
```

```

        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {
            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，
                 * 要求外层函数的参数必须是 final 的，因此需要将
                 * page、pageSize 设置为 final。
                 */
                query.setFirstResult((page-1)*pageSize);
                query.setMaxResults(pageSize);
                return query.list();
            }
        });
    }

    @Override
    public int findTotalPage(int pageSize) {
        String hql = "select count(*) from Cost";
        List<Object> list = getHibernateTemplate().find(hql);
        // 查询出总行数
        int rows = Integer.valueOf(list.get(0).toString());
        // 根据总行数计算总页数
        if (rows % pageSize == 0)
            return rows / pageSize;
        else
            return rows / pageSize + 1;
    }

    @Override
    public Cost findById(int id) {
        // 使用延迟加载的方法实现
        return (Cost) getHibernateTemplate().load(Cost.class, id);
    }
}

```


步骤二：创建打开修改页面的 Action

在 com.tarena.action 包下,创建打开修改页面的 Action ,实现根据 ID 查询资费数据的逻辑,代码如下:

```
package com.tarena.action;

import javax.annotation.Resource;

import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 加载修改数据
 */
@Controller
@Scope("prototype")
public class ToUpdateCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private int id;
    // output
    private Cost cost;

    public String load() {
        cost = costDao.findById(id);
        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：配置 Action

在 struts.xml 中配置这个 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>

    <package name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
```

```

        class="findCostAction" method="load">
        <!-- 注入页容量 -->
        <param name="pageSize">3</param>
        <result name="success">
            /WEB-INF/cost/find_cost.jsp
        </result>
    </action>

    <!-- 打开修改页面 -->
    <action name="toUpdateCost"
        class="toUpdateCostAction" method="load">
        <result name="success">
            /WEB-INF/cost/update_cost.jsp
        </result>
    </action>

</package>
</struts>

```

步骤四：创建修改页面

在 WEB-INF/cost 文件夹下，创建修改页面 update_cost.jsp，并通过 Struts2 的 UI 标签回显要修改的数据，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">
        
        <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
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
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>
```

步骤五：处理修改按钮

在 find_cost.jsp 中，将修改按钮的访问路径改为修改 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() {
        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">
      
      <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">
            
            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();" />
        </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">
    <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>

    <s:iterator begin="1" end="totalPage" var="p">
        <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>

    <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>

```

```

        </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>

```

步骤六：测试

重新部署项目并启动 tomcat，访问资费列表页面，并点击任意一行数据的修改按钮，显示出的修改页面效果如下图所示，可见除了 ID 字段外，其他字段都显示为空，这是由于采用了延迟加载查询数据，却没有保证 session 不提前关闭导致的。



图-2

步骤七：完善

在 web.xml 中配置一个 filter，用于保证 session 在视图层加载数据时是开启的，代码如下：

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <display-name></display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <!-- 配置listener，使tomcat启动时自动加载 Spring -->
  <listener>
    <listener-class>

```



```

        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
</context-param>

<!--
    配置过滤器，以保证在视图层 session 是开启的。
    该过滤器必须在 Struts2 前端控制器之前配置才有效。
-->
<filter>
    <filter-name>OpenSessionInView</filter-name>
    <filter-class>
        org.springframework.orm.hibernate3.support.OpenSessionInViewFilter
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>OpenSessionInView</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 配置前端控制器 -->
<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>

```

步骤八：测试

重新部署项目并启动 tomcat，再次访问资费修改页面，效果如下图所示：

图-3

• 完整代码

本案例中 ICostDao 完整代码如下所示：

```
package com.tarena.dao;

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

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据
     * @param id
     * @return
     */
    Cost findById(int id);
}
```

CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }
}
```

```
@Override
public List<Cost> findAll() {
    String hql = "from Cost";
    return getHibernateTemplate().find(hql);
}

@Override
public List<Cost> findByPage(final int page, final int pageSize) {
    /*
     * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
     * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
     * 将作为最终的结果返回。
     */
    return getHibernateTemplate().executeFind(new HibernateCallback() {

        /*
         * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
         * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
         * 创建和关闭。
         */
        @Override
        public Object doInHibernate(Session session)
            throws HibernateException, SQLException {
            String hql = "from Cost";
            Query query = session.createQuery(hql);
            /*
             * 设置分页参数，注意在内层函数中调用外层函数的参数，
             * 要求外层函数的参数必须是 final 的，因此需要将
             * page、pageSize 设置为 final。
             */
            query.setFirstResult((page-1)*pageSize);
            query.setMaxResults(pageSize);
            return query.list();
        }
    });
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}
}
```

ToUpdateCostAction 完整代码如下所示：

```
package com.tarena.action;

import javax.annotation.Resource;
```

```
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;

import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 加载修改数据
 */
@Controller
@Scope("prototype")
public class ToUpdateCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private int id;
    // output
    private Cost cost;

    public String load() {
        cost = costDao.findById(id);
        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 完整代码如下所示：

```
<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find cost.jsp
            </result>
        </action>
        <!-- 打开修改页面 -->
        <action name="toUpdateCost"
            class="toUpdateCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/update cost.jsp
            </result>
        </action>
    </package>
</struts>
```

```
</action>
</package>
</struts>
```

update_cost.jsp 完整代码如下所示：

```
<%@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>
    <div id="main">
      <div id="header">
        <div id="header_title">达内—NetCTOSS</div>
        <div id="header_nav">
          <a href="#">首页
          <a href="#">关于我们
          <a href="#">联系我们
          <a href="#">加入我们
          <a href="#">服务支持
          <a href="#">合作伙伴
          <a href="#">联系我们
        </div>
      </div>
      <div id="content">
        <div id="content_title">更新资费</div>
        <div id="content_form">
          <table border="1">
            <tr>
              <td>资费类型</td>
              <td>资费标准</td>
            </tr>
            <tr>
              <td>1. 资费类型</td>
              <td>1. 资费标准</td>
            </tr>
            <tr>
              <td>2. 资费类型</td>
              <td>2. 资费标准</td>
            </tr>
            <tr>
              <td>3. 资费类型</td>
              <td>3. 资费标准</td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

        </script>
    </head>
    <body>
        <!--Logo 区域开始-->
        <div id="header">
            
            <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>

```

web.xml 完整代码如下所示：

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
    <display-name></display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <!-- 配置 listener，使 tomcat 启动时自动加载 Spring -->
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!--
        配置过滤器，以保证在视图层 session 是开启的。
        该过滤器必须在 Struts2 前端控制器之前配置才有效。
    -->

```

```
<filter>
  <filter-name>OpenSessionInView</filter-name>
  <filter-class>
    org.springframework.orm.hibernate3.support.OpenSessionInViewFilter
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>OpenSessionInView</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<!-- 配置前端控制器 -->
<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>
```

3. 使用 Hibernate 进行修改

- 问题

完成资费修改保存功能，并在项目中解决使用 Hibernate 修改带来的问题。

- 方案

使用 Hibernate 修改时，它会根据映射关系文件，自动拼出一个 update 语句，然后执行修改，其中映射关系文件中往往配置表中全部的字段，而很多时候，页面上不需要修改表中全部的字段，只需要修改其中的一部分，因此页面上的字段少于表中字段，在提交保存时，缺少的字段就成了空值，那么再按照完整的 update 语句来执行更新，就会把这些不需要更新的字段更新为空。

这个问题如果使用 JDBC 或者 MyBatis 是不会存在的，因为 SQL 是自己写的。而 Hibernate 自动生成 SQL，就出现了这个问题。

本案例中，我们采用动态更新的方式来解决这个问题，即在映射关系文件中通过 dynamic-update="true" 来声明更新方式为动态更新，届时 Hibernate 在自动生成 update 语句时会判断属性值是否发生改变，若改变则将属性拼入 SQL，否则忽略掉这个属性。

这种方式要求传给 Hibernate 的对象必须是持久态的，而通过页面传入的对象是 Struts2 自动初始化的，是临时态的。我们可以通过 ID 查询出持久态对象，然后通过 Spring 中的 BeanUtils 工具类，将临时态对象的属性值复制给持久态对象，然后用这个持久态对象进行更新即可。

- 步骤

实现此案例需要按照如下步骤进行。

步骤一：在 DAO 中增加修改保存方法

在 ICostDao 中增加修改保存方法，代码如下：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据
     * @param id
     * @return
     */
    Cost findById(int id);

    /**
     * 修改资费数据
     * @param cost
     */
    void update(Cost cost);

}
```

在 CostDaoImpl 中实现这个方法，代码如下：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
```

```
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，
                 * 要求外层函数的参数必须是 final 的，因此需要将
                 * page、pageSize 设置为 final。
                 */
                query.setFirstResult((page-1)*pageSize);
                query.setMaxResults(pageSize);
                return query.list();
            }
        });
    }

    @Override
    public int findTotalPage(int pageSize) {
        String hql = "select count(*) from Cost";
        List<Object> list = getHibernateTemplate().find(hql);
        // 查询出总行数
        int rows = Integer.valueOf(list.get(0).toString());
        // 根据总行数计算总页数
        if (rows % pageSize == 0)
            return rows / pageSize;
        else
            return rows / pageSize + 1;
    }
}
```

```
@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

}
```

步骤二：创建修改保存 Action

在 com.tarena.action 包下，创建修改保存 Action 类，代码如下：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 修改保存
 */
@Controller
@Scope("prototype")
public class UpdateCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private Cost cost;

    public String update() {
        costDao.update(cost);
        return "success";
    }

    public Cost getCost() {
        return cost;
    }

    public void setCost(Cost cost) {
        this.cost = cost;
    }

}
```

步骤三：配置 Action

在 struts.xml 中配置这个 Action，代码如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
```

<!-- 修改保存 -->

修改 `update_cost.jsp`，配置好表单以及提交按钮，代码如下：

//保存结果的提示

```
//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">
        
        <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="updateCost" method="post" 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="submit" value="保存" class="btn_save" />

        <input type="button" value="取消" class="btn_save" />
    </div>
</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：测试

修改之前，资费表中数据如下图所示：

ID	NAME	BASE_DURATION	BASE_COST	UNIT_COST	STATUS	DESCR	CREATIME	STARTIME	COST_TYPE
1	5.9元套餐	5	5	5	0	5.9元20小时/月,超出部分0.4元/时	2014-07-17 12:07:34.0	2014-07-17 12:07:34.0	2
3	8.5元套餐	333	33	3	0	33元333小时/月,超出部分3元/时	2014-07-17 12:07:34.0	2014-07-17 12:07:34.0	2
4	10.5元套餐	10000	2000	300	1	10.5元200小时/月,超出部分0.1元/时	2013-08-30 22:36:40.0	2013-09-30 08:25:18.0	2
5	计时收费	1	1	0.5	1	0.5元/时,不使用不收费	2013-08-30 22:36:40.0	2013-09-29 22:00:00.0	2
6	包月	<NULL>	20	<NULL>	1	每月20元,不限使用时间	2013-08-30 22:36:40.0	2013-09-30 08:25:18.0	1

图-4

重新部署项目并启动 tomcat，再次访问修改功能，对 id=3 的数据进行一些修改，保存后，资费表中的数据如下图所示，可见此次更新将该条记录的 status、createtime、starttime 字段清空了。

ID	NAME	BASE_DURATION	BASE_COST	UNIT_COST	STATUS	DESCR	CREATIME	STARTIME	COST_TYPE
1	5.9元套餐	5	5	5	1	5.9元20小时/月,超出部分0.4元/时	<NULL>	<NULL>	2
3	8.5元套餐	333	33	3	<NULL>	33元333小时/月,超出部分3元/时	<NULL>	<NULL>	2
4	10.5元套餐	10000	2000	300	1	10.5元200小时/月,超出部分0.1元/时	2013-08-30 22:36:40.0	2013-09-30 08:25:18.0	2
5	计时收费	1	1	0.5	1	0.5元/时,不使用不收费	2013-08-30 22:36:40.0	2013-09-29 22:00:00.0	2
6	包月	<NULL>	20	<NULL>	1	每月20元,不限使用时间	2013-08-30 22:36:40.0	2013-09-30 08:25:18.0	1

图-5

步骤六：完善

在映射关系文件 Cost.hbm.xml 中，声明其更新方式为动态更新，代码如下：

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

    <!--
        通过 dynamic-update 将当前对象设置为动态更新，
        对象中没有发生变化的属性将不会拼入 update 语句。
    -->
    <class name="com.tarena.entity.Cost"
        table="cost" dynamic-update="true">

```

```
<id name="id" type="integer" column="id">
    <!-- 用来指明主键的生成方式 -->
    <generator class="sequence">
        <param name="sequence">cost_seq</param>
    </generator>
</id>

<property name="name"
    type="string" column="name" />
<property name="baseDuration"
    type="integer" column="base_duration" />
<property name="baseCost"
    type="double" column="base cost" />
<property name="unitCost"
    type="double" column="unit cost" />
<property name="status"
    type="string" column="status" />
<property name="descr"
    type="string" column="descr" />
<property name="createTime"
    type="date" column="createTime" />
<property name="startTime"
    type="date" column="startTime" />
<property name="costType"
    type="string" column="cost type" />
</class>
</hibernate-mapping>
```

在 UpdateCostAction 中，调用 DAO 更新方法之前，根据传入的对象，构造一个持久态的更新对象，然后利用这个构造出来的对象进行更新，代码如下：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.beans.BeanUtils;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 修改保存
 */
@Controller
@Scope("prototype")
public class UpdateCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private Cost cost;

    public String update() {

        // 查询出要修改的原始对象，该对象为持久态
        Cost c = costDao.findById(cost.getId());
        // 将传入对象的属性复制到原始对象上，忽视不需修改的列
        BeanUtils.copyProperties(cost, c,
            new String[] { "status", "createTime", "startTime" });
        /*
         * 对持久态对象更新，由于映射关系文件中设置了动态更新，

```



```

    * 因此持久对象中值为 null 的列将不被更新。
    */
    costDao.update(c);

    return "success";
}

public Cost getCost() {
    return cost;
}

public void setCost(Cost cost) {
    this.cost = cost;
}
}

```

步骤七：测试

重新部署项目并启动 tomcat，修改 id=4 的数据，保存后表中数据如下图所示。可见这次 status、createime、starttime 没有被更新为空值。

ID	NAME	BASE_DURATION	BASE_COST	UNIT_COST	STATUS	DESCR	CREATIME	STARTIME	COST_TYPE
1	5.9元套餐	5	5	5	0	5.9元20小时/月,超出部分0.4元/时	2014-07-17 12:14:18.0	2014-07-17 12:14:18.0	2
3	55元套餐	555	55	5	<NULL>	55元555小时/月,超出部分5元/时	<NULL>	<NULL>	2
4	88元套餐	888	88	8	1	88元888小时/月,超出部分8元/时	2013-08-30 00:00:00.0	2013-09-30 00:00:00.0	2
5	计时收费	1	1	0.5	1	0.5元/时,不使用不收费	2013-08-30 22:36:40.0	2013-09-29 22:00:00.0	2
6	包月	<NULL>	20	<NULL>	1	每月20元,不限制使用时间	2013-08-30 22:36:40.0	2013-09-30 08:25:18.0	1

图-6

完整代码

本案例中 ICostDao 完整代码如下所示：

```

package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据

```

```

    * @param id
    * @return
    */
    Cost findById(int id);

    /**
     * 修改资费数据
     * @param cost
     */
    void update(Cost cost);
}

```

CostDaoImpl 完整代码如下所示：

```

package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /**
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /**
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
            }
        });
    }
}

```

```
        * 设置分页参数，注意在内层函数中调用外层函数的参数，
        * 要求外层函数的参数必须是 final 的，因此需要将
        * page、pageSize 设置为 final。
        * */
        query.setFirstResult((page-1)*pageSize);
        query.setMaxResults(pageSize);
        return query.list();
    }
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}
}
```

UpdateCostAction 完整代码如下所示：

```
package com.tarena.action;

import javax.annotation.Resource;

import org.springframework.beans.BeanUtils;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 修改保存
 */
@Controller
@Scope("prototype")
public class UpdateCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private Cost cost;

    public String update() {
```

```
// 查询出要修改的原始对象，该对象为持久态
Cost c = costDao.findById(cost.getId());
// 将传入对象的属性复制到原始对象上，忽视不需修改的列
BeanUtils.copyProperties(cost, c,
    new String[] { "status", "createTime", "startTime" });
/*
 * 对持久态对象更新，由于映射关系文件中设置了动态更新，
 * 因此持久对象中值为 null 的列将不被更新。
 */
costDao.update(c);
return "success";
}

public Cost getCost() {
    return cost;
}

public void setCost(Cost cost) {
    this.cost = cost;
}
}
```

struts.xml 完整代码如下所示：

```
<?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 name="cost"
    namespace="/cost" extends="json-default">
    <action name="findCost"
        class="findCostAction" method="load">
        <!-- 注入页容量 -->
        <param name="pageSize">3</param>
        <result name="success">
            /WEB-INF/cost/find_cost.jsp
        </result>
    </action>
    <!-- 打开修改页面 -->
    <action name="toUpdateCost"
        class="toUpdateCostAction" method="load">
        <result name="success">
            /WEB-INF/cost/update_cost.jsp
        </result>
    </action>
    <!-- 修改保存 -->
    <action name="updateCost"
        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
</package>
</struts>
```

update_cost.jsp 完整代码如下所示：

```
<%@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">
      
      <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="updateCost" method="post" 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>
        </form>
    </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="submit" value="保存" class="btn_save" />
            <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>

```

Cost.hbm.xml 完整代码如下所示：

```

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <!--
        通过 dynamic-update 将当前对象设置为动态更新，
        对象中没有发生变化的属性将不会拼入 update 语句。
    -->
    <class name="com.tarena.entity.Cost"
        table="cost" dynamic-update="true">
        <id name="id" type="integer" column="id">
            <!-- 用来指明主键的生成方式 -->
            <generator class="sequence">
                <param name="sequence">cost_seq</param>
            </generator>
        </id>

        <property name="name"
            type="string" column="name" />
        <property name="baseDuration"
            type="integer" column="base_duration" />
        <property name="baseCost"
            type="double" column="base_cost" />
        <property name="unitCost"
            type="double" column="unit_cost" />
        <property name="status"
            type="string" column="status" />
        <property name="descr"
            type="string" column="descr" />
        <property name="createTime"
            type="date" column="create_time" />
        <property name="startTime"
            type="date" column="start_time" />
        <property name="costType"
            type="string" column="cost_type" />
    </class>
</hibernate-mapping>

```

4. 资费新增

• 问题

完成资费新增功能。

• 方案

资费新增和资费修改十分相似，不同点在于新增页面不需要默认显示数据，因此该功能可以参考资费修改完成。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置资费新增 Action

由于资费新增 Action 没有任何业务逻辑，因此不需要创建 Action 类，只需要直接在 struts.xml 配置即可。运行时 Struts2 会自动实例化 ActionSupport 类并调用默认的 execute 业务方法，该业务方法中返回了字符串“success”，根据该返回值配置好对应的新增页面即可，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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
        <!-- 打开修改页面 -->
        <action name="toUpdateCost"
            class="toUpdateCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
        </action>
        <!-- 修改保存 -->
        <action name="updateCost"
            class="updateCostAction" method="update">
            <result name="success" type="redirectAction">
                findCost
            </result>
        </action>

        <!-- 打开新增页面 -->
```



```
<action name="toAddCost">
    <result name="success">
        /WEB-INF/cost/add_cost.jsp
    </result>
</action>
```

```
</package>
</struts>
```

步骤二：创建新增页面

在 WEB-INF/cost 文件夹下，创建新增资费页面 add_cost.jsp，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 = "";
                }
            }
        </script>
    </head>
    <body>
        <div id="main">
            <div id="header">
                <div id="header_title">
                    达内 - NetCTOSS
                </div>
            </div>
            <div id="content">
                <div id="content_title">
                    新增资费
                </div>
                <div id="content_form">
                    <table border="1">
                        <tr>
                            <td>资费类型</td>
                            <td>
                                <input type="radio" value="1" /> 新增
                                <input type="radio" value="2" /> 修改
                                <input type="radio" value="3" /> 删除
                            </td>
                        </tr>
                        <tr>
                            <td>资费名称</td>
                            <td>
                                <input type="text" value="" />
                            </td>
                        </tr>
                        <tr>
                            <td>资费金额</td>
                            <td>
                                <input type="text" value="" />
                            </td>
                        </tr>
                        <tr>
                            <td>资费说明</td>
                            <td>
                                <input type="text" value="" />
                            </td>
                        </tr>
                    </table>
                </div>
                <div id="content_btn">
                    <input type="button" value="保存" />
                    <input type="button" value="取消" />
                </div>
            </div>
        </div>
    </body>
</html>
```

```

        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">
        
        <a href="#">[退出]</a>
    </div>
    <!--Logo 区域结束-->
    <!--导航区域开始-->

    <!--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：处理新增按钮

在资费列表页面 find_cost.jsp 上 修改新增按钮 将其 URL 指定为资费新增 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";
            }
        </script>
    </head>
    <body>
        <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>
    </body>
</html>

```

```

    }

    //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">
        
        <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">
        
        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();" />
                    </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">
    <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:iterator begin="1" end="totalPage" var="p">
        <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
value="#p"/>"><s:property value="#p"/></a>
          href="findCost?page=<s:property
value="#p"/>"><s:property value="#p"/></a>
        </s:else>
      </s:iterator>

      <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>

```

步骤四：阶段性测试

重新部署项目并启动 tomcat，访问资费列表页面，点击新增按钮，跳转后的页面如下图所示：

Tarena NetCTOSS [退出]

资源名称: * 50长度的字母、数字、汉字和下划线组合

资源类型: ☐ 包月 ☒ 套餐 ☐ 计时

基本时长: 小时 * 1-600之间的整数

基本费用: 元 * 0.999999-99.99之间的数值

单位费用: 元/小时 * 0.999999-99.99之间的数值

费用说明: * 100长度的字母、数字、汉字和下划线组合

保存 **取消**

[源自化美的技术，最优秀的师资，最真实的企业环境，最适用的实战项目]
版权所有(C)加拿大达内IT培训集团

图-7

步骤五：DAO 中增加新增保存方法

在 ICostDao 中增加新增保存的方法，代码如下：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据
     * @param id
     * @return
     */
    Cost findById(int id);

    /**
     * 修改资费数据
     * @param cost
     */
    void update(Cost cost);
}
```

```
/**
 * 新增资费数据
 * @param cost
 */
void save(Cost cost);

}
```

在 CostDaoImpl 中实现这个方法，代码如下：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，

```



```

        * 要求外层函数的参数必须是 final 的，因此需要将
        * page、pageSize 设置为 final。
        */
        query.setFirstResult((page-1)*pageSize);
        query.setMaxResults(pageSize);
        return query.list();
    }
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if(cost == null)
        return;
    getHibernateTemplate().save(cost);
}
}

```

步骤六：创建新增保存 Action

在 com.tarena.action 包下，创建新增保存 Action，代码如下：

```

package com.tarena.action;

import java.sql.Date;
import javax.annotation.Resource;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 新增保存资费数据
 */

```

```
@Controller
@Scope("prototype")
public class AddCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private Cost cost;

    public String add() {
        // 初始化一些默认值
        initDefaultValue(cost);
        costDao.save(cost);
        return "success";
    }

    /**
     * 初始化默认值
     */
    private void initDefaultValue(Cost cost) {
        // 状态默认为暂停态
        cost.setStatus("1");
        // 创建时间默认为当前系统时间
        cost.setCreateTime(
            new Date(System.currentTimeMillis()));
    }

    public Cost getCost() {
        return cost;
    }

    public void setCost(Cost cost) {
        this.cost = cost;
    }
}
```

步骤七：配置新增保存 Action

在 struts.xml 中配置这个 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>

    <package name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
        <!-- 打开修改页面 -->
        <action name="toUpdateCost"
            class="toUpdateCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
        </action>
    </package>
</struts>
```

```

        </result>
    </action>
    <!-- 修改保存 -->
    <action name="updateCost"
        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 打开新增页面 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add_cost.jsp
        </result>
    </action>

    <!-- 新增保存 -->
    <action name="addCost"
        class="addCostAction" method="add">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>

</package>
</struts>

```

步骤八：修改新增页面的表单

修改 add_cost.jsp，配置好表单、框体名称以及保存按钮，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">
        
        <a href="#">[退出]</a>
    </div>
    <!--Logo 区域结束-->
    <!--导航区域开始-->

    <!--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="addCost" method="post" class="main_form">

        <div class="text_info clearfix"><span>资费名称:</span></div>
        <div class="input_info">

            <input type="text" class="width300" name="cost.name"/>

            <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="cost.costType" value="1" id="monthly"
onclick="feeTypeChange(1);" />
        <label for="monthly">包月</label>
        <input type="radio" name="cost.costType" value="2"
checked="checked" id="package" onclick="feeTypeChange(2);" />
        <label for="package">套餐</label>
        <input type="radio" name="cost.costType" value="3"
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" name="cost.baseDuration" 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" name="cost.baseCost" 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" name="cost.unitCost" 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
name="cost.descr"></textarea>                                height70"

        <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="submit" value="保存" class="btn_save" />

```

```

        <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>

```

步骤九：测试

重新部署项目并启动 tomcat，访问新增页面，输入一些内容点击保存，成功后重定向到列表页面，可以看到列表页面上多了一条新增的数据（id=360），效果如下图所示：

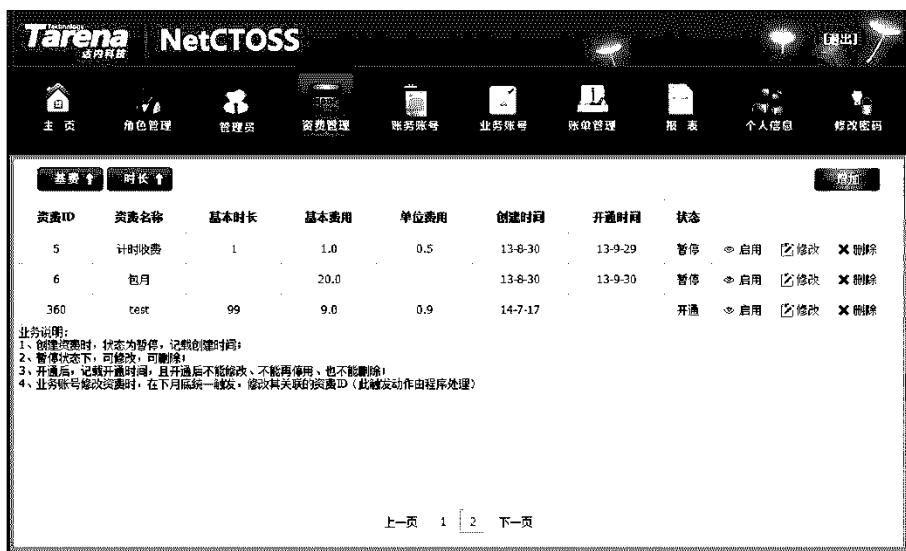


图-8

完整代码

本案例中 ICostDao 的完整代码如下所示：

```

package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     */
}

```

```
* @return
*/
List<Cost> findByPage(int page, int pageSize);

/**
 * 查询总页数
 * @param pageSize
 * @return
 */
int findTotalPage(int pageSize);

/**
 * 根据 ID 查询资费数据
 * @param id
 * @return
 */
Cost findById(int id);

/**
 * 修改资费数据
 * @param cost
 */
void update(Cost cost);

/**
 * 新增资费数据
 * @param cost
 */
void save(Cost cost);
}
```

CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
```

```

public List<Cost> findByPage(final int page, final int pageSize) {
    /*
     * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
     * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
     * 将作为最终的结果返回。
     */
    return getHibernateTemplate().executeFind(new HibernateCallback() {

        /*
         * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
         * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
         * 创建和关闭。
         */
        @Override
        public Object doInHibernate(Session session)
            throws HibernateException, SQLException {
            String hql = "from Cost";
            Query query = session.createQuery(hql);
            /*
             * 设置分页参数，注意在内层函数中调用外层函数的参数，
             * 要求外层函数的参数必须是 final 的，因此需要将
             * page、pageSize 设置为 final。
             */
            query.setFirstResult((page-1)*pageSize);
            query.setMaxResults(pageSize);
            return query.list();
        }
    });
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if(cost == null)
        return;
    getHibernateTemplate().save(cost);
}
}

```


AddCostAction 完整代码如下所示：

```
package com.tarena.action;

import java.sql.Date;
import javax.annotation.Resource;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;
import com.tarena.entity.Cost;

/**
 * 新增保存资费数据
 */
@Controller
@Scope("prototype")
public class AddCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private Cost cost;

    public String add() {
        // 初始化一些默认值
        initDefaultValue(cost);
        costDao.save(cost);
        return "success";
    }

    /**
     * 初始化默认值
     */
    private void initDefaultValue(Cost cost) {
        // 状态默认为暂停态
        cost.setStatus("1");
        // 创建时间默认为当前系统时间
        cost.setCreateTime(
            new Date(System.currentTimeMillis()));
    }

    public Cost getCost() {
        return cost;
    }

    public void setCost(Cost cost) {
        this.cost = cost;
    }

}
```

struts.xml 完整代码如下所示：

```
<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
```

```

        <!-- 注入页容量 -->
        <param name="pageSize">3</param>
        <result name="success">
            /WEB-INF/cost/find_cost.jsp
        </result>
    </action>
    <!-- 打开修改页面 -->
    <action name="toUpdateCost"
        class="toUpdateCostAction" method="load">
        <result name="success">
            /WEB-INF/cost/update_cost.jsp
        </result>
    </action>
    <!-- 修改保存 -->
    <action name="updateCost"
        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 打开新增页面 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add_cost.jsp
        </result>
    </action>
    <!-- 新增保存 -->
    <action name="addCost"
        class="addCostAction" method="add">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
</package>
</struts>

```

add_cost.jsp 完整代码如下所示：

```

<%@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">
        
        <a href="#">[退出]</a>
    </div>
    <!--Logo 区域结束-->
    <!--导航区域开始-->

    <!--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="addCost" method="post" class="main_form">
            <div class="text_info clearfix"><span>资费名称: </span></div>
            <div class="input_info">
                <input type="text" class="width300" name="cost.name"/>
                <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="cost.costType" value="1"
id="monthly" onclick="feeTypeChange(1);" />
                <label for="monthly">包月</label>
                <input type="radio" name="cost.costType" value="2"
checked="checked" id="package" onclick="feeTypeChange(2);" />
                <label for="package">套餐</label>
                <input type="radio" name="cost.costType" value="3"
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" name="cost.baseDuration"
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" name="cost.baseCost" 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" name="cost.unitCost" 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"
name="cost.descr"></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="submit" value="保存" class="btn_save" />
        <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>

```

5. 资费删除

• 问题

完成资费删除功能。

• 方案

资费删除功能比较简单, 在页面上选中一条资费, 将 ID 传入 Action, 然后调用 DAO 使用 Hibernate 删除这条数据即可。

- **步骤**

实现此案例需要按照如下步骤进行。

步骤一：DAO 中增加删除方法

在 ICostDao 中增加资费删除方法，代码如下：

```
package com.tarena.dao;

import java.util.List;

import com.tarena.entity.Cost;

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
    int findTotalPage(int pageSize);

    /**
     * 根据 ID 查询资费数据
     * @param id
     * @return
     */
    Cost findById(int id);

    /**
     * 修改资费数据
     * @param cost
     */
    void update(Cost cost);

    /**
     * 新增资费数据
     * @param cost
     */
    void save(Cost cost);

    /**
     * 删除资费数据
     * @param id
     */
    void delete(int id);

}
```

在 CostDaoImpl 中实现这个方法，代码如下：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，
                 * 要求外层函数的参数必须是 final 的，因此需要将
                 * page、pageSize 设置为 final。
                 */
                query.setFirstResult((page-1)*pageSize);
                query.setMaxResults(pageSize);
                return query.list();
            }
        });
    }
}
```

```
@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().save(cost);
}

@Override
public void delete(int id) {
    Cost cost = new Cost();
    cost.setId(id);
    getHibernateTemplate().delete(cost);
}
}
```

步骤二：创建资费删除 Action

在 com.tarena.action 包下，创建资费删除 Action，代码如下：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;

/**
 * 删除资费数据
 */
@Controller
@Scope("prototype")
public class DeleteCostAction {

    @Resource
    private ICostDao costDao;
```

```
// input
private int id;

public String delete() {
    costDao.delete(id);
    return "success";
}

public int getId() {
    return id;
}

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

步骤三：配置资费删除 Action

在 struts.xml 中，配置资费删除 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>

    <package name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
        <!-- 打开修改页面 -->
        <action name="toUpdateCost"
            class="toUpdateCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
        </action>
        <!-- 修改保存 -->
        <action name="updateCost"
            class="updateCostAction" method="update">
            <result name="success" type="redirectAction">
                findCost
            </result>
        </action>
        <!-- 打开新增页面 -->
        <action name="toAddCost">
            <result name="success">
                /WEB-INF/cost/add_cost.jsp
            </result>
        </action>
        <!-- 新增保存 -->
        <action name="addCost"
            class="addCostAction" method="add">
            <result name="success" type="redirectAction">
                findCost
            </result>
        </action>
    </package>
</struts>
```



```

        </result>
    </action>

    <!-- 删除 -->
    <action name="deleteCost"
        class="deleteCostAction" method="delete">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>

</package>
</struts>

```

步骤四：处理删除按钮

在 find_cost.jsp 中，将删除按钮的 URL 指向资费删除 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) {
                    window.location.href = "deleteCost?id="+id;
                }
            }

        </script>
    </head>
    <body>

```

```

<!--Logo 区域开始-->
<div id="header">
    
    <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">
            
            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">
    <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>

```

```

        <s:iterator begin="1" end="totalPage" var="p">
            <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>

        <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>

```

步骤五：测试

重新部署项目并启动 tomcat，访问资费列表页面，选择一条数据并删除，然后看列表页面上是否删除了这条数据。

• 完整代码

本案例中 ICostDao 完整代码如下所示：

```

package com.tarena.dao;

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

public interface ICostDao {

    List<Cost> findAll();

    /**
     * 查询某页资费数据
     * @param page 页码
     * @param pageSize 页容量
     * @return
     */
    List<Cost> findByPage(int page, int pageSize);

    /**
     * 查询总页数
     * @param pageSize
     * @return
     */
}

```

```
int findTotalPage(int pageSize);

/**
 * 根据 ID 查询资费数据
 * @param id
 * @return
 */
Cost findById(int id);

/**
 * 修改资费数据
 * @param cost
 */
void update(Cost cost);

/**
 * 新增资费数据
 * @param cost
 */
void save(Cost cost);

/**
 * 删除资费数据
 * @param id
 */
void delete(int id);
}
```

CostDaoImpl 完整代码如下所示：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法

```

```

    * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
    * 将作为最终的结果返回。
    * */
return getHibernateTemplate().executeFind(new HibernateCallback() {
    /*
    * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
    * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
    * 创建和关闭。
    */
    @Override
    public Object doInHibernate(Session session)
        throws HibernateException, SQLException {
        String hql = "from Cost";
        Query query = session.createQuery(hql);
        /*
        * 设置分页参数，注意在内层函数中调用外层函数的参数，
        * 要求外层函数的参数必须是 final 的，因此需要将
        * page、pageSize 设置为 final。
        */
        query.setFirstResult((page-1)*pageSize);
        query.setMaxResults(pageSize);
        return query.list();
    }
});
}

@Override
public int findTotalPage(int pageSize) {
    String hql = "select count(*) from Cost";
    List<Object> list = getHibernateTemplate().find(hql);
    // 查询出总行数
    int rows = Integer.valueOf(list.get(0).toString());
    // 根据总行数计算总页数
    if (rows % pageSize == 0)
        return rows / pageSize;
    else
        return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if(cost == null)
        return;
    getHibernateTemplate().save(cost);
}

@Override
public void delete(int id) {
    Cost cost = new Cost();
    cost.setId(id);
    getHibernateTemplate().delete(cost);
}
}

```

DeleteCostAction 完整代码如下所示：

```
package com.tarena.action;

import javax.annotation.Resource;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Controller;
import com.tarena.dao.ICostDao;

/**
 * 删除资费数据
 */
@Controller
@Scope("prototype")
public class DeleteCostAction {

    @Resource
    private ICostDao costDao;

    // input
    private int id;

    public String delete() {
        costDao.delete(id);
        return "success";
    }

    public int getId() {
        return id;
    }

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

}
```

struts.xml 完整代码如下所示：

```
<?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 name="cost"
        namespace="/cost" extends="json-default">
        <action name="findCost"
            class="findCostAction" method="load">
            <!-- 注入页容量 -->
            <param name="pageSize">3</param>
            <result name="success">
                /WEB-INF/cost/find_cost.jsp
            </result>
        </action>
        <!-- 打开修改页面 -->
        <action name="toUpdateCost"
            class="toUpdateCostAction" method="load">
            <result name="success">
                /WEB-INF/cost/update_cost.jsp
            </result>
        </action>
        <!-- 修改保存 -->
        <action name="updateCost"
```

```

        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 打开新增页面 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add cost.jsp
        </result>
    </action>
    <!-- 新增保存 -->
    <action name="addCost"
        class="addCostAction" method="add">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 删除 -->
    <action name="deleteCost"
        class="deleteCostAction" method="delete">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
</package>
</struts>

```

find cost.jsp 完整代码如下所示：

```
<%@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) {
                    window.location.href = "deleteCost?id="+id;
                }
            }
        </script>
    </head>
    <body>
```



```

    }
    </script>
</head>
<body>
    <!--Logo 区域开始-->
    <div id="header">
        
        <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">
                
                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>
                    </tr>
                </table>
            </div>
        </form>
    </div>

```

```

        <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">
    <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>

    <s:iterator begin="1" end="totalPage" var="p">
        <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>

    <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>

```

6. 异常处理

• 问题

当前系统中，如果业务代码报错，会在页面上直接输出错误信息，如下图，客户体验度很差。要求处理这些业务代码中的异常，当任何业务代码发生异常时，请跳转至一个统一的错误页面。



图-9

• 方案

由于每个功能的业务代码中都有可能报错，因此这种处理异常的行为是所有业务代码都需要做的，是典型的通用业务逻辑，所以可以采用拦截器进行处理，发生异常时转向统一的错误页面即可。

• 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建异常处理拦截器

创建 com.tarena.interceptor 包，并在包下创建异常处理拦截器 ExceptionInterceptor，代码如下：

```
package com.tarena.interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 异常处理拦截器
 */
public class ExceptionInterceptor implements Interceptor{

    @Override
    public void destroy() {

    }

    @Override
    public void init() {

    }

    @Override
    public String intercept(ActionInvocation ai) throws Exception {
        try {
            return ai.invoke();
        } catch (Exception e) {
            e.printStackTrace();
            // 处理异常, 报错时转向统一的错误页面
            return "error";
        }
    }
}
```

步骤二：注册及引用拦截器

在 struts.xml 中，注册该拦截器，并设置所有 Action 都引用这个拦截器，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>

    <!-- 公共的包，封装了通用的拦截器、通用的 result -->
    <package name="netctoss" extends="json-default">
        <interceptors>
            <!-- 异常处理拦截器 -->
            <interceptor name="exceptionInterceptor"
                class="com.tarena.interceptor.ExceptionInterceptor"/>
            <!-- 拦截器栈 -->
            <interceptor-stack name="netctossStack">
                <interceptor-ref name="exceptionInterceptor"/>
                <!-- 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>
    </package>
</struts>
```

```

</interceptors>
<!-- 设置 action 默认引用的拦截器 -->
<default-interceptor-ref name="netctossStack"/>
<!-- 全局的 result，包下所有的 action 都可以共用 -->
<global-results>
    <!-- 跳转到错误页面的 result -->
    <result name="error">
        /WEB-INF/main/error.jsp
    </result>
</global-results>
</package>

<package name="cost"
    namespace="/cost" extends="netctoss">

    <action name="findCost"
        class="findCostAction" method="load">
        <!-- 注入页容量 -->
        <param name="pageSize">3</param>
        <result name="success">
            /WEB-INF/cost/find_cost.jsp
        </result>
    </action>

    <!-- 打开修改页面 -->
    <action name="toUpdateCost"
        class="toUpdateCostAction" method="load">
        <result name="success">
            /WEB-INF/cost/update_cost.jsp
        </result>
    </action>

    <!-- 修改保存 -->
    <action name="updateCost"
        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>

    <!-- 打开新增页面 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add_cost.jsp
        </result>
    </action>

    <!-- 新增保存 -->
    <action name="addCost"
        class="addCostAction" method="add">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>

    <!-- 删除 -->
    <action name="deleteCost"
        class="deleteCostAction" method="delete">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>

</package>

</struts>

```

步骤三：创建统一的错误页面

在 WEB-INF/main 下，创建统一的错误页面 error.jsp，代码如下所示：

```
<%@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(showSeconds,1000);
      }

      var i = 5;
      function showSeconds() {
        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，&nbsp;<span id="secondes">5</span>&nbsp;<span id="secondes">秒后将自动跳转，立即跳
      转请点击&nbsp;<a href="javascript:resetTimer();">返回</a>
    </h1>
  </body>
</html>
```

步骤四：模拟一个业务异常

在 CostDaoImpl 中分页查询方法中模拟一个异常，代码如下所示：

```
package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;
```

```
import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;
```

```
@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {
```

```
    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }
```

```
    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }
```

```
    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
```

```
        Integer.valueOf("abc");
```

```
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {
```

```
            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
```

```
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，
                 * 要求外层函数的参数必须是 final 的，因此需要将
                 * page、pageSize 设置为 final。
                 */
                query.setFirstResult((page-1)*pageSize);
                query.setMaxResults(pageSize);
                return query.list();
            }
        });
```

```
    }

    @Override
    public int findTotalPage(int pageSize) {
        String hql = "select count(*) from Cost";
        List<Object> list = getHibernateTemplate().find(hql);
```

```
// 查询出总行数
int rows = Integer.valueOf(list.get(0).toString());
// 根据总行数计算总页数
if (rows % pageSize == 0)
    return rows / pageSize;
else
    return rows / pageSize + 1;
}

@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().save(cost);
}

@Override
public void delete(int id) {
    Cost cost = new Cost();
    cost.setId(id);
    getHibernateTemplate().delete(cost);
}
}
```

步骤五：测试

重新部署项目并启动 tomcat，访问资费列表，效果如下图所示：



图-10

- **完整代码**

本案例中 ExceptionInterceptor 的完整代码如下所示：

```
package com.tarena.interceptor;

import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

/**
 * 异常处理拦截器
 */
public class ExceptionInterceptor implements Interceptor{

    @Override
    public void destroy() {

    }

    @Override
    public void init() {

    }

    @Override
    public String intercept(ActionInvocation ai) throws Exception {
        try {
            return ai.invoke();
        } catch (Exception e) {
            e.printStackTrace();
            // 处理异常,报错时转向统一的错误页面
            return "error";
        }
    }
}
```

struts.xml 完整代码如下所示：

```
<?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>
            <!-- 异常处理拦截器 -->
            <interceptor name="exceptionInterceptor"
                class="com.tarena.interceptor.ExceptionInterceptor"/>
            <!-- 拦截器栈 -->
            <interceptor-stack name="netctossStack">
                <interceptor-ref name="exceptionInterceptor"/>
                <!-- 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="netctossStack"/>
        <!-- 全局的 result,包下所有的 action 都可以共用 -->
        <global-results>
```

```

        <!-- 跳转到错误页面的 result -->
        <result name="error">
            /WEB-INF/main/error.jsp
        </result>
    </global-results>
</package>

<package name="cost"
    namespace="/cost" extends="netctoss">
    <action name="findCost"
        class="findCostAction" method="load">
        <!-- 注入页容量 -->
        <param name="pageSize">3</param>
        <result name="success">
            /WEB-INF/cost/find cost.jsp
        </result>
    </action>
    <!-- 打开修改页面 -->
    <action name="toUpdateCost"
        class="toUpdateCostAction" method="load">
        <result name="success">
            /WEB-INF/cost/update cost.jsp
        </result>
    </action>
    <!-- 修改保存 -->
    <action name="updateCost"
        class="updateCostAction" method="update">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 打开新增页面 -->
    <action name="toAddCost">
        <result name="success">
            /WEB-INF/cost/add_cost.jsp
        </result>
    </action>
    <!-- 新增保存 -->
    <action name="addCost"
        class="addCostAction" method="add">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
    <!-- 删除 -->
    <action name="deleteCost"
        class="deleteCostAction" method="delete">
        <result name="success" type="redirectAction">
            findCost
        </result>
    </action>
</package>
</struts>

```

CostDaoImpl 完整代码如下所示：

```

package com.tarena.dao;

import java.sql.SQLException;
import java.util.List;
import javax.annotation.Resource;

import org.hibernate.HibernateException;
import org.hibernate.Query;
import org.hibernate.Session;

```

```
import org.hibernate.SessionFactory;
import org.springframework.orm.hibernate3.HibernateCallback;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Repository;
import com.tarena.entity.Cost;

@Repository
public class CostDaoImpl
    extends HibernateDaoSupport implements ICostDao {

    @Resource
    public void setSF(SessionFactory sf) {
        super.setSessionFactory(sf);
    }

    @Override
    public List<Cost> findAll() {
        String hql = "from Cost";
        return getHibernateTemplate().find(hql);
    }

    @Override
    public List<Cost> findByPage(final int page, final int pageSize) {
        Integer.valueOf("abc");
        /*
         * executeFind 方法需要传入接口对象，该接口对象的 doInHibernate 方法
         * 会在执行查询时自动被 Spring 调用，并且 doInHibernate 方法的返回值
         * 将作为最终的结果返回。
         */
        return getHibernateTemplate().executeFind(new HibernateCallback() {

            /*
             * doInHibernate 方法类似于回调函数，是在 executeFind 方法的内部被调用的。
             * 该方法中可以直接使用 session，这个 session 由 Spring 负责管理，不需要我们
             * 创建和关闭。
             */
            @Override
            public Object doInHibernate(Session session)
                throws HibernateException, SQLException {
                String hql = "from Cost";
                Query query = session.createQuery(hql);
                /*
                 * 设置分页参数，注意在内层函数中调用外层函数的参数，
                 * 要求外层函数的参数必须是 final 的，因此需要将
                 * page、pageSize 设置为 final。
                 */
                query.setFirstResult((page-1)*pageSize);
                query.setMaxResults(pageSize);
                return query.list();
            }
        });
    }

    @Override
    public int findTotalPage(int pageSize) {
        String hql = "select count(*) from Cost";
        List<Object> list = getHibernateTemplate().find(hql);
        // 查询出总行数
        int rows = Integer.valueOf(list.get(0).toString());
        // 根据总行数计算总页数
        if (rows % pageSize == 0)
            return rows / pageSize;
        else
            return rows / pageSize + 1;
    }
}
```

```
@Override
public Cost findById(int id) {
    // 使用延迟加载的方法实现
    return (Cost) getHibernateTemplate().load(Cost.class, id);
}

@Override
public void update(Cost cost) {
    if (cost == null)
        return;
    getHibernateTemplate().update(cost);
}

@Override
public void save(Cost cost) {
    if(cost == null)
        return;
    getHibernateTemplate().save(cost);
}

@Override
public void delete(int id) {
    Cost cost = new Cost();
    cost.setId(id);
    getHibernateTemplate().delete(cost);
}
}
```