一：项目第一天（系统框架）

# 1：项目介绍

\*参考【文档】中的【项目介绍】

# 2:项目框架搭建

创建数据库itcastelec0710,创建表

**#测试表**

**CREATE TABLE Elec\_Text(**

**textID varchar(50) not null primary key,**

**textName varchar(50),**

**textDate datetime,**

**textRemark varchar(500)**

**)**

## （1）持久层

1:在itest.elec.domain中创建一个javabean和对应的映射文件

ElecText.java

**package** itest.elec.domain;

**import** java.util.Date;

@SuppressWarnings("serial")//此处是注解？

**public** **class** ElecText **implements** java.io.Serializable{

**private** String textID; //测试id

**private** String textName; //测试name

**private** Date textDate; //测试时间

**private** String textRemark; //测试备注

**public** String getTextID() {

**return** textID;

}

**public** **void** setTextID(String textID) {

**this**.textID = textID;

}

**public** String getTextName() {

**return** textName;

}

**public** **void** setTextName(String textName) {

**this**.textName = textName;

}

**public** Date getTextDate() {

**return** textDate;

}

**public** **void** setTextDate(Date textDate) {

**this**.textDate = textDate;

}

**public** String getTextRemark() {

**return** textRemark;

}

**public** **void** setTextRemark(String textRemark) {

**this**.textRemark = textRemark;

}

}

2:在同级目录下，创建ElecText.hbm.xml

<?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 model-->

<hibernate-mapping>

<class name=*"itest.elec.domain.ElecText"* table=*"Elec\_Text"*>

<id name=*"textID"* type=*"string"* column=*"textID"*>

<generator class=*"uuid"*></generator>

</id>

<property name=*"textName"* type=*"string"* column=*"textName"*></property>

<property name=*"textDate"* type=*"date"* column=*"textDate"*></property>

<property name=*"textRemark"* type=*"string"* column=*"textRemark"*></property>

</class>

</hibernate-mapping>

3:在类路径下，创建hibernate.cfg.xml

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!--数据库连接信息 -->

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/itcastelec0710?useUnicode=true&amp;characterEncoding=utf8</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>ghq123</property>

<!-- hibernate自动提交事务 -->

<property name=*"hibernate.connection.autocommit"*>true</property>

<!--其他配置 -->

<property name=*"hibernate.dialect"*> org.hibernate.dialect.MySQL5Dialect</property>

<property name=*"hibernate.show\_sql"*>true</property>

<property name=*"hibernate.hbm2ddl.auto"*>update</property>

<!--加载映射文件 -->

<mapping resource=*"itest/elec/domain/ElecText.hbm.xml"*/>

</session-factory>

</hibernate-configuration>

4:在junit包下，使用TesstHiberate进行测试，同时导入一个log4日志的文件

**import** java.util.Date;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** org.junit.Test;

**import** itest.elec.domain.ElecText;

**public** **class** TestHibernate {

@Test

/\*\*

\*保存

\*/

**public** **void** save(){

//默认加载路径下的Hibernate配置文件(hibernate.cfg.xml)和映射文件

Configuration configration = **new** Configuration();

configration.configure();

//模板代码

SessionFactory sf = configration.buildSessionFactory();

Session s = sf.openSession();

Transaction tr = s.beginTransaction();

ElecText elecText = **new** ElecText();

elecText.setTextName("测试名称");

elecText.setTextDate(**new** Date());

elecText.setTextRemark("测试Hibernate备注");

s.save(elecText);

tr.commit();

s.close();

}

}

## （2）Dao层（Spring）

1：在itest.elec.dao中创建2个接口

IcommonDao接口

**package** itest.elec.dao;

**public** **interface** ICommonDao<T> {

**public** **void** save(T entry);

}

IElecTextDao接口

**package** itest.elec.dao;

**import** itest.elec.domain.ElecText;

**public** **interface** IElecTextDao **extends** ICommonDao<ElecText>{

// public void findElecText();

**public** **static** **final** String ***SERVICE\_NAME***="itest.elec.dao.impl.ElecTextImpl";

}

2：在itest.elec.dao.impl中创建2个接口的实现类

CommomDaoImpl

**package** itest.elec.dao.impl;

**import** javax.annotation.Resource;

**import** org.hibernate.SessionFactory;

**import** org.springframework.orm.hibernate3.support.HibernateDaoSupport;

**import** itest.elec.dao.ICommonDao;

**public** **class** CommondImpl<T> **extends** HibernateDaoSupport **implements** ICommonDao<T> {

/\*HibernateDaoSupport的作用：需要使用模板来写顶层方法save()等。。。

\* 要使用模板，就要有容器，此时Spring就出现了

\* bean.xml去写容器。使用注解的方式。

\*

\*/

/\* spring容器中定义

\* <bean id="commondDao" class="itest.elec.dao.impl.CommondImpl">

<property name="sessionFactory" ref="sessionFactory"></property>

</bean>

\*

\* \*/

@Resource(name="sessionFactory")

public final void setSessionFactoryDi(SessionFactory sessionFactory){

this.setSessionFactory(sessionFactory);

}

@Override

/\*保存\*/

**public** **void** save(T entry) {

**this**.getHibernateTemplate().save(entry);

}

}

ElecTextDaoImpl类

/\*\*相当于在Spring中定义了：

\* <bean id = "itest.elec.dao.impl.ElecTextImpl" class = "itest.elec.dao.impl.ElecTextImpl"/>

\* <bean>\*/

@Repository(IElecTextDao.***SERVICE\_NAME***)

**public** **class** ElecTextImpl **extends** CommondImpl<ElecText> **implements** IElecTextDao{

}

3：在src下创建spring的容器（beans.xml）

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

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-3.0.xsd"*>

<!-- 1:配置注解的自动扫描的范围 在类和方法上添加对注解的支持-->

<context:component-scan base-package=*"itest.elec"*></context:component-scan>

<!-- 2， -->

<!-- 3 创建SessionFactory对象，这是spring整合hibernate的核心(LocalSessionFactoryBean)-->

<bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate3.LocalSessionFactoryBean"*>

<property name=*"configLocation"*>

<value>classpath:hibernate.cfg.xml</value>

</property>

</bean>

<!-- 4:创建事务管理器 -->

<bean id=*"txManage"* class=*"org.springframework.orm.hibernate3.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"*></property>

</bean>

<!-- 5:以注解的形式管理事务 -->

<tx:annotation-driven transaction-manager=*"txManage"*/>

<!--不要 <bean id="hibernateTemplate" class="org.springframework.orm.hibernate3.hibernateTemplate">

<property name="sessionFactory" ref="sessionFactory"></property>

</bean>

<bean id="commondDao" class="itest.elec.dao.impl.CommondImpl">

<property name="hibernateTemplate" ref="hibernateTemplate"></property>

</bean> -->

<!--剪切去，采用纯注解的方式 <bean id="commondDao" class="itest.elec.dao.impl.CommondImpl">

<property name="sessionFactory" ref="sessionFactory"></property>

</bean> -->

<!--5: XML配置文件的写法,管理事务

<tx:advice id="aa" transaction-manager="txManage">

<tx:attributes>

<tx:method name="save\*" isolation="DEFAULT" propagation="REQUIRED" read-only="false"/>

<tx:method name="update\*" isolation="DEFAULT" propagation="REQUIRED" read-only="false"/>

<tx:method name="delete\*" isolation="DEFAULT" propagation="REQUIRED" read-only="false"/>

<tx:method name="\*" read-only="true"/>

</tx:attributes>

</tx:advice>

<aop:config>

<aop:pointcut id="bb" expression="execution(\* itest.elec.service..\*.\*(..))" />

<aop:advisor advice-ref="aa" pointcut-ref="bb"/>

</aop:config>

-->

</beans>

4:在junit包下的TestDao进行测试

public class TestDao {

@Test

/\*\*

\*保存

\*

\*先要加载Spring容器，容器来加载Hibernate

\*/

public void save(){

// 1. 加载容器

ApplicationContext ac = new ClassPathXmlApplicationContext("beans.xml");

// 2.实例化、初始化一个javabean，即调用Dao，这里使用多态，

// 应认真思考以加重对多态的印象

IElecTextDao elecTextDao = (IElecTextDao)ac.getBean(IElecTextDao.*SERVICE\_NAME*);

// 3.使用一个javabean对象，加载数据，并调用save();

ElecText elecText = new ElecText();

elecText.setTextName("测试Dao名称");

elecText.setTextDate(new Date());

elecText.setTextRemark("测试Dao备注");

elecTextDao.save(elecText);

}

}

结论：在Dao层要对公用方法进行丰足昂，好处在于有些方法不需要在每个Dao中都体现，这样开发麻烦，代码重复性高。解决方案：将公用的方法封装到CommonDaoImlp中。

## （3）业务层

1；在itest.elec.service包中创建接口

**package** itest.elec.service;

**import** itest.elec.domain.ElecText;

**public** **interface** IElecTextService {

**public** **static** **final** String ***SERVICE\_NAME*** = "itest.elec.service.impl.ElecTextServiceImpl";

**public** **void** save(ElecText elecText);

}

2：在在itest.elec.service.impl包中创建接口的实现类

@Service(IElecTextService.***SERVICE\_NAME***)

/\*\*@Transactional的作用是Spring管理事务要加上的注解，

\* 全局有效

\* 默认是可写，开发时候类级别的事务可以定义为只读

\* 在save()里面就会报错，此时可以在save()加上注解

\* @Transactional(isolation=Isolation.DEFAULT,propagation=Propagation.REQUIRED,readOnly=false)

\* 即方法里面改变类的属性？

\* \*/

**@Transactional(readOnly=true)**

**public** **class** ElecTextServiceImpl **implements** IElecTextService {

@Resource(name=IElecTextDao.***SERVICE\_NAME***)

**private** IElecTextDao elecTextDao;

/\*\*保存\*/

**@Transactional(isolation=Isolation.*DEFAULT*,propagation=Propagation.*REQUIRED*,readOnly=false)**

**public** **void** save(ElecText elecText) {

elecTextDao.save(elecText);

}

}

3：使用junit测试

**public** **class** TestService {

@Test

/\*\*

\*保存

\*

\*先要加载Spring容器，容器来加载Hibernate

\*/

**public** **void** save(){

// 1. 加载容器

ApplicationContext ac = **new** ClassPathXmlApplicationContext("beans.xml");

// 2.调用Dao，这里使用多态，应认真思考以加重对多态的印象

IElecTextService elecTextService = (IElecTextService)ac.getBean(IElecTextService.***SERVICE\_NAME***);

// 3.实例化一个javabean对象，加载数据，并调用save();

ElecText elecText = **new** ElecText();

elecText.setTextName("测试Service名称");

elecText.setTextDate(**new** Date());

elecText.setTextRemark("测试Service备注");

elecTextService.save(elecText);

}

}

结论：使用声明式事务处理（aop编程的实现），控制业务层的代码，相当于代码访问业务层的方法之前产生一个代理对象，在代理对象中完成事物的操作。

**解决的问题：**

**1:eclpsec的配置，jsp不能被识别。在tomcat中的一个jar包引入即可。**

Struts2和struts1都可以实现MVC的设计模式。

M：model，模型，存放页面获得的值和传递给页面的值。

V：veiw，视图，表现层，例如jsp

C：controller，控制层，处理Action业务的转发，使用Action类连接视图

VO对象：是一个javabean，表示值得对象，用来传递和操作页面表单的值

PO对象:：是一个javabean，表示的持久对象，用来连接数据库中的表和字段的值

Struts2的又是在于拦截器

Struts1开发的时候，从页面上获取VO 对象，vo对象只能获取String类型的或者是String类型的数组，此时需要vo对象转化成po对象，才能执行保存

Struts2开发的时候，由于struts2使用类型转换过滤器，实现了对各种数据类型的转换，此时vo对象不成需要转换成po对象，方便程序开发。

问题：每个Action都要实现一个ModelDerive，而且要在每个Avtion中定义一个javabean：例如：**private** ElecText elecText = **new** ElecText();

**public** ElecText getModel() {

**return** elecText;

}

解决方案：

使用一个Action的父类BaceAction完成这个操作。

将Action中经常定义的request和response对象也放置到BaseAction中

## （4）控制层

1：在itest.elec.web.action中创建两个Action类

ElecTextAction类

@SuppressWarnings("serial")

@Controller("elecTextAction")

@Scope(value="prototype")

/\*\*

\* 相当于spring中定义了

\* <bean id= "elecTextAction"class="itest.elec.web.action.ElecTextAction">\*/

**public** **class** ElecTextAction **extends** **BaseAction**<ElecText> {

//封装模型驱动在BaseAction，接着需要泛型转换，再封装（将request和response和模

//型驱动的对象统一放在BaseAction中封装。）！！！

**private** ElecText elecText = **this**.getModel();

@Resource(name=IElecTextService.***SERVICE\_NAME***)

**private** IElecTextService elecTextService;

/\*\*保存 \*/

**public** String save(){

elecTextService.save(elecText);

System.***out***.println(request.getParameter("textDate")+" "+request.getParameter("textName"));

**return** "save";

}

}

父类BaseAction类

**public** **class** BaseAction<T> **extends** ActionSupport **implements** ModelDriven<T>,ServletRequestAware,ServletResponseAware {

**protected** HttpServletRequest request;

**protected** HttpServletResponse response;

**private** T entity;

**public** BaseAction(){

//泛型转换

Class entityClass = GenericTypeUtils.*getGenericSuperClass*(**this**.getClass());

**try** {

entity = (T)entityClass.newInstance();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** T getModel() {

**return** entity;

}

**public** **void** setServletRequest(HttpServletRequest req) {

**this**.request = req;

}

**public** **void** setServletResponse(HttpServletResponse res) {

**this**.response = res;

}

}

结论：

1.将request和response和模型驱动的对象统一放在BaseAction中封装,

.泛型转换

2.在itest.elec.util包下创建泛型转换类的公用方法Utils

**public** **static** Class getGenericSuperClass(Class entity) {

//泛类转换，目的是将对应的泛型，装换成真实的对象类型。

//此时type表示，BaseAction< itest.elec.domain.ElecText>

ParameterizedType type = (ParameterizedType)entity.getGenericSuperclass();

//entityClass:itest.elec.domain.ElecText

Class entityClass = (Class)type.getActualTypeArguments()[0];

**return** **null**;

}

3.在类路径下创建struts.xml文件

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

<!DOCTYPE struts PUBLIC

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

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

<struts>

<!-- 配置Struts的模式为开发模式，优势是自动加载配置文件struts。xml和资源文件，显示更多错误信息 -->

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

<!-- 设置UI主题，去掉Struts提供的样式，设置为simple，由美工添加样式 -->

<constant name=*"struts.ui.theme"* value=*"simple"*></constant>

<!-- 改变struts方法的后缀，友action该为do -->

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

<!-- <constant name="struts.objectFactory" value="spring" /> -->

<!-- 系统管理 -->

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

<!-- 全局转发 -->

<global-results>

<result name=*"close"*>/close.jsp</result>

</global-results>

<!-- 测试 -->

<action name=*"elecTextAction\_\*"* class=*"elecTextAction"* method=*"{1}"*>

<result name=*"save"*>/system/textAdd.jsp</result>

</action>

</package>

</struts>

4.在web.xml中定义监听器和过滤器

<!--当web容器启动的时候，自动加载spring容器 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:beans.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 添加struts2的过滤器，这是struts2执行的核心 -->

<filter>

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

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

</filter>

5.导入页面的原型，使用system下的testAdd.jsp进行测试。

# 3:底层方法的封装

新增、修改、查询（使用id）删除（使用id删除、使用集合删除集合中的所有对象）

查询（指定条件查询）

/\*\*更新\*/

**public** **void** update(T entry) {

**this**.getHibernateTemplate().update(entry);

}

/\*\*使用主键id查询对象\*/

**public** T findObjectByID(Serializable id) {

**return** (T) **this**.getHibernateTemplate().get(entityClass, id);

}

/\*\*删除（使用一个主键id和多个主键id的数组）\*/

**public** **void** deleteObjectByIDS(Serializable... ids) {

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

**for** (Serializable id : ids) {

Object entity = **this**.findObjectByID(id);

**this**.getHibernateTemplate().delete(entity);

}

}

}

/\*\*删除（将对象封装成集合，使用集合删除集合中存放的所有对象）\*/

**public** **void** deleteObjectByCollection(List<T> list) {

**this**.getHibernateTemplate().deleteAll(list);

}

/\*指定页面传递的条件，查询对应的结果集的信息，返回一个List<ElecText> 不分页\*/

/\*\*SELECT \* FROM elec\_text o WHERE 1=1 #Dao层

\* AND o.textName LIKE '%张%' #Servicec层

\* AND o.textRemark LIKE '%张%' #Servicec层

\* ORDER BY o.teextDate ASC ,o.textRemark DESC #Servicec层

\* \*/

**public** List<T> findCollectionByConditionNoPage(String condition, Object[] params, Map<String, String> orderby) {

String hql = " select o from " + entityClass.getSimpleName()+" o where 1=1 ";

//解析Map集合，获取排序的语句

String orderbyhql = **this**.orderby(orderby);

**final** String finalHql = hql + condition + orderbyhql;

//方式一

//List<T> list = this.getHibernateTemplate().find(finalHql, params);

//使用hibernate提供的回调函数，回调session.实质上，这段代码实现this.getHibernateTemplate().find(finalHql, params);

List<T> list = (List<T>) **this**.getHibernateTemplate().execute(**new** HibernateCallback() {

@Override

**public** Object doInHibernate(Session session) **throws** HibernateException, SQLException {

Query query = session.createQuery(finalHql);

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

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

query.setParameter(i, params[i]);

}

}

**return** query.list();

}

});

**return** list;

}

//解析Map集合，获取排序的语句 ORDER BY o.teextDate ASC ,o.textRemark DESC #Servicec层

**private** String orderby(Map<String, String> orderby) {

StringBuffer buffer = **new** StringBuffer("");

**if**(orderby!=**null** && orderby.size()>0){

buffer.append(" order by ");

**for**(Map.Entry<String, String> map:orderby.entrySet()){

buffer.append(map.getKey()+" "+map.getValue()+",");

}

//删除最后一个逗号

buffer.deleteCharAt(buffer.length()-1);

}

**return** buffer.toString();

}

## 4：项目用整合svn

Svn版本控制，管理项目中的源码，文档，管理每个人开发的代码

项目使用svn的部署

\* 参数【技术资料】中svn的文件夹中的《svn.doc》

## 5：项目开发可以在myeclipse中开发，也可以在eclipse开发

* 不同之处：

1：Myeclipse收费，Eclipse不收费

2：Myeclipse是Eclipse的一个插件，也就是说Eclipse没有任何插件，需要人工导入

3：使用Myeclipse发布项目，是发布到外部的tomcat下的webapp的目录中

Eclipse是将Tomcat服务器整合到Eclipse开发

4：myeclipse是将class文件放置到

D:\workspaceProject\itcast1115elec\WebRoot\WEB-INF\classes

Eclipse是将class文件放置到

D:\workspaceEclipse\itcast1115ElecEclipse\build\classes

5：myeclipse放置jsp文件的文件夹默认叫做：WebRoot

Eclipse放置jsp文件的文件夹默认叫做：WebContent

## 6：项目编码规范

项目开发前，统一开发习惯

见：【文档】，《国家电力检测中心项目和设备资源管理系统开发规范.doc》

作用：统一项目中每个成员的开发习惯，保证项目每个模块相当于是1个人开发是一样的

项目第一天问答

1：什么事MVC结构？

M：model，模型。表示javabean，用来传递和接收页面的表单数据

V:view视图。表现层

C:controller控制层，Action

2：在ssh项目中，项目的事务处理是如何控制的？

采用Spring的声明式事务管理，使用注解的方式。在Spring容器配置中设置组件的扫描，在Service层配置一个注解@Transcational

3：在ssh项目中，项目的Dao层是如何封装的？需要每个Dao都重新写同样的方法吗？

Dao层是封装的。将公用的方法提取出来，定义一个公用的接口和类，让每个Dao的类都继承公用类，每个Dao的接口继承公用的接口。在每个Dao的接口中和每个Dao的实现类中 都定义真实类型，然后再公用的累中定义一个泛型转换。

4：在struts2和hibernate中的项目。Vo对象和po对象的区别？

Vo：值对象，表示javabean。用来传递和接受页面的表单参数。

Po：持久对象，表示javabean。每个属性用来关联数据库表的字段。

2 项目第二天

1：系统登录+首页显示

1. index.jsp

<jsp:forward page=*"menu/index.jsp"*/>

1. 登录页面：menu包下的index.jsp

提供登录名和密码

1. 系统登录首页home.jsp

之系统框架划分：

Frame和iframe的使用

<FRAMESET border=*0* frameSpacing=*0* rows=*82,\** frameBorder=*0* id=*"mainparent"*>

<FRAME name=*topFrame* src=*"*${pageContext.request.contextPath }*/menu/title.jsp"* noResize scrolling=*no*>

<FRAMESET id=*"main"* border=*"0"* frameSpacing=*"0"* frameBorder=*"0"* cols=*"153,1%,\*"*>

<FRAME name=*"leftFrame"* src=*"*${pageContext.request.contextPath }*/menu/left.jsp"* noResize>

<frame name=*"changeButton"* src=*"*${pageContext.request.contextPath }*/menu/change.jsp"* frameBorder=*0* marginHeight=*0* marginWidth=*0* scrolling=*no* noresize>

<FRAME name=*"mainFrame"* src=*"*${pageContext.request.contextPath }*/menu/loading.jsp"* >

</FRAMESET>

</FRAMESET>

（4）菜单left.jsp

<TABLE border=*0* width=*"20"*>

<TR>

<TD width=*340px* align=*center* valign=*top*>

<div class=*"zTreeDemoBackground"*>

<ul id=*"*menuTree*"* class="tree" ></ul>

</div>

</TD>

</TR>

</TABLE>

**Tree.js**

**var** menu = {

setting: {

isSimpleData: **true**,

treeNodeKey: "mid",

treeNodeParentKey: "pid",

showLine: **true**,

root: {

isRoot: **true**,

nodes: []

}

},

loadMenuTree:**function**(){

$("#menuTree").zTree(menu.setting, privilegeDate);

}

};

$().ready(**function**(){

menu.loadMenuTree();

});

menuDate.js

privilegeDate = [{

mid: 'aa',

pid: '0',

isParent: **true**,

icon:'../images/MenuIcon/jishusheshiweihuguanli.gif',

//open:true,

name: '技术设施维护管理',

nodes:[

{

（5）Change.jsp

框架改变

<script language=*"javascript"*>

**function** shiftwindow()

{

**if**(parent.document.getElementById("main").cols=="153,1%,\*")

{

parent.document.getElementById("main").cols='0,1%,99%';

document.all.image.src="${pageContext.request.contextPath }/images/you.gif";

}

**else** **if**(parent.document.getElementById("main").cols=="0,1%,99%")

{

parent.document.getElementById("main").cols='153,1%,\*';

document.all.image.src="${pageContext.request.contextPath }/images/zuo.gif";

}

}

</script>

（6）loading.jsp

<table width=*"100%"* border=*"0"* height=*"88"* border=*"1"* background=${pageContext.request.contextPath }/images/back1.jpg>

<tr>

<td colspan=*3* class=*"ta\_01"* align=*"center"* background=*"*${pageContext.request.contextPath }*/images/b-info.gif"*><strong>系统首页</strong></td>

</tr>

<tr>

<td width=*"50%"* height=*"84"* align=*"left"* valign=*"top"* id=*"stationtd"*>

<fieldset id=*"stationset"* style=*"width: 500px; height: 430px; padding: 1 background:*${pageContext.request.contextPath }*/images/back1.JPG"*><legend>

<font color=*"#0000FF"*>

<img border=*"0"* src=*"*${pageContext.request.contextPath }*/images/zoom.gif"* width=*"14"* height=*"14"*><a href=*"#"* onclick="shiftiframe('1')">站点运行情况</a></font></legend>

<IFRAME src=*"alermStation.jsp"* name=*"station"* id=*"station"* frameBorder=*"0"* width=*"500"* scrolling=*"auto"* height=*"400"*></IFRAME>

</fieldset>

</td>

<td width=*"50%"* align=*"left"* valign=*"top"* id=*"devtd"*>

<fieldset id=*"devset"* style=*"width: 500px; height: 430px; padding: 1 background:*${pageContext.request.contextPath }*/images/back1.JPG"*><legend>

<font color=*"#0000FF"*>

<img border=*"0"* src=*"*${pageContext.request.contextPath }*/images/zoom.gif"* width=*"14"* height=*"14"*><a href=*"#"* onclick="shiftiframe('2')">设备运行情况</a></font></legend>

<IFRAME src=*"alermDevice.jsp"* name=*"dev"* id=*"dev"* frameBorder=*"0"* width=*"500"* scrolling=*"auto"* height=*"400"*></IFRAME>

</fieldset>

</td>

</tr>

<tr><td height=*2*></td></tr>

</table>

改变iframe的宽度

<script language=*"javascript"*>

**function** shiftiframe(value)

{

**if**(value==1){

**if**(document.all.station.width==500)

{

document.all.station.width=1100;

document.getElementById("devtd").style.display="none";

}

**else** **if**(document.all.station.width==1100)

{

document.all.station.width=500;

document.getElementById("devtd").style.display="";

}

}

**else**{

**if**(document.all.dev.width==500)

{

document.all.dev.width=1100;

document.getElementById("stationtd").style.display="none";

}

**else** **if**(document.all.dev.width==1100)

{

document.all.dev.width=500;

document.getElementById("stationtd").style.display="";

}

}

}

每隔十分钟刷新页面，保证实时性

/\*\*添加10分钟后自动刷新页面，站点和设备运行的实时性\*/

window.onload=**function**(){

setTimeout('refresh10()',1000\*60\*10) ;

}

**function** refresh10(){

window.location.reload();

}

</script>

2运行监控

（1）：首页td内容显示（JS）

当显示内容的表格td不能显示全部文本内容时候，可以使用一个div，显示所有的内容，当鼠标移动到td时候，显示div内容，当鼠标移出td的时候，div再次隐藏起来。



1：jsp页面

<link href="${pageContext.request.contextPath }/css/showText.css" type="text/css" rel="stylesheet">

<script language=*"javascript"* src="${pageContext.request.contextPath }/script/showText.js"></script>

2：在jsp中定义

<div id="showInfomation" style="visibility: hidden"></div>放置到任意位置可以。

3：在jsp页面中定义一个鼠标移动div，显示td中的需要显示的内容。

<div class=*"scrollStyle"* align=*"left"*

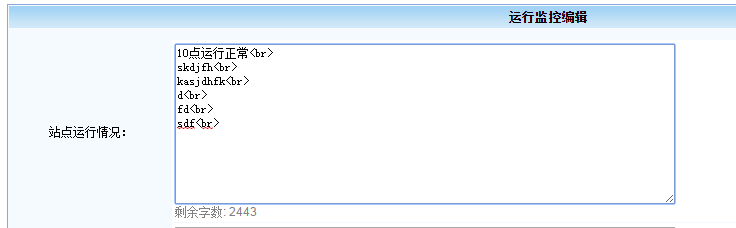
onmouseover="showInfoWithPanel(this)"

onmouseout="hiddenInfoPanel(this)"

style="table-layout:*fixed*;">

<s:property value=*"devRun"*/></div>

（2）：防止jsp页面输入的内容长度超出数据库的长度限制。



1：引入js

<script language=*"javascript"* src=*"*${pageContext.request.contextPath }*/script/limitedTextarea.js"*></script>

2:在页面中定义

**function** checkTextAreaLen(){

**var** stationRun = **new** Bs\_LimitedTextarea('stationRun', 2500);

stationRun.infolineCssStyle = "font-family:arial; font-size:11px; color:gray;";

stationRun.draw();

**var** devRun = **new** Bs\_LimitedTextarea('devRun', 2500);

devRun.infolineCssStyle = "font-family:arial; font-size:11px; color:gray;";

devRun.draw();

}

window.onload=**function**(){

checkTextAreaLen();

}

3：在页面中定义

<s:textarea name="stationRun" id="stationRun" cssStyle="width: 500px; height: 160px; padding: 1;FONT-FAMILY: 宋体; FONT-SIZE: 9pt" onkeydown="if(event.keyCode==13)addEnter('stationRun');" ></s:textarea>

怎么消除<br>?添加escape="false"

在

<tr>

<td align=*"left"* valign=*"middle"* style="word-break: *break-all*">

<span class=*"style1"*>

<s:property value=*"stationRun"* escape=*"false"*/>

</span></td>

</tr>

（3）进度条

需求 ：

1：大批量数据执行保存、更新、删除，可以添加进度条

2:文件的上传和下载，可以添加进度条

3：文件的导入和导出，可以添加进度条

4：在进行远程数据访问的时候，可以添加进度条

详情：请见文档中的【技术资料】【进度条】

（4）整合FCK文本编辑器

详情见：【技术资料】。FCK文本编辑器/最新版本/ckeditor配置资料/CKEditor\_CKFinder\_整合项目开发.doc

扩展（问题）

由于项目中使用文本编辑器，很有可能在开发的时候，进行保存数据的时候，超出数据库支持的字段的最大值。如何解决？

方案一：

之前数据库设计

主键ID 站点运行情况 设备运行情况 创建日期

现在数据库设计

运行监控表

主键ID 站点运行情况 设备运行情况 创建日期

1 1 2 2013-11-17 14:39:00

运行监控数据表

主键ID 站点运行/设备运行的标识 数据内容 数据显示排序

1 1 XXXXXXX 1

2 1 YYYYYYY 2

3 2 AAAAAAA 1

4 2 BBBBBBB 2

注意：数据内容在保存的时候，先将收内容进行分割，按照2500个字符进行分割，分割数组的形式进行保存

查询的时候，查询运行监控表，使用站点运行情况的1，查询运行监控数据表，将查询获取的数据内容组织成1个新的字符串，将新的字符串显示XXXXXXXYYYYYYYY

（5）导出设置保存设置:

1：弹出窗口（exportExcel.jsp -> close.jsp function.js中的js）

Js:window.open();

Js:windex.close();

a.jsp中可以使用window.open();，此时弹出b.jsp

在b.jsp中可以使用opener属性，调用a.jsp的内容

类似于：使用framest的时候，在子页面之中调用parent，用来调用父页面的数据

并在struts.xml中设置全局转发

<!-- 全局转发 -->

<global-results>

<result name=*"close"*>/close.jsp</result>

</global-results>

2：使用<select></select>的框架

3：由于使用<select></select>的框架的某些原因，使用5个隐藏域存放数据库保存的值

<s:hidden name=*"belongTo"* id=*"belongTo"*></s:hidden>

belongTo：ID

另外4个：在页面上组织好保存的参数（详见exportExcel.jsp）

<s:hidden name=*"expNameList"* id=*"expNameList"* ></s:hidden>

<s:hidden name=*"expFieldName"* id=*"expFieldName"*></s:hidden>

<s:hidden name=*"noExpNameList"* id=*"noExpNameList"*></s:hidden>

<s:hidden name=*"noExpFieldName"* id=*"noExpFieldName"*></s:hidden>

4:【导出】：将数据导出成Excel文件（动态导出Excel字段）在第十一天完成。

5：扩展：

导出设置开发的思想，不仅能满足Excel报表的字段动态导出，

同时也能满足，对应的jsp页面的字段显示，实现动态显示的效果。

开发数据字典的一些要求：

1数据字典由管理员（admin）进行配置，因为数据字典的每一项都影响系统数据的使用

2在开发过程中，进行录入数据字典的值，方便系统的开发和维护

3系统一旦上线，数据字典的值一般不会发生变化

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"* destroy-method=*"close"*>

<property name=*"driverClass"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"jdbcUrl"* value=*"jdbc:mysql://localhost:3306/itcastelec0710?useUnicode=true&amp;characterEncoding=utf-8"*/>

<property name=*"user"* value=*"root"* />

<property name=*"password"* value=*"ghq123"* />

<!--连接池中保留的最小连接数。-->

<property name=*"minPoolSize"* value=*"5"* />

<!--连接池中保留的最大连接数。Default: 15 -->

<property name=*"maxPoolSize"* value=*"30"* />

<!--初始化时获取的连接数，取值应在minPoolSize与maxPoolSize之间。Default: 3 -->

<property name=*"initialPoolSize"* value=*"10"*/>

<!--最大空闲时间,60秒内未使用则连接被丢弃。若为0则永不丢弃。Default: 0 -->

<property name=*"maxIdleTime"* value=*"60"*/>

<!--当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3 -->

<property name=*"acquireIncrement"* value=*"5"* />

<!--JDBC的标准参数，用以控制数据源内加载的PreparedStatements数量。但由于预缓存的statements

属于单个connection而不是整个连接池。所以设置这个参数需要考虑到多方面的因素。

如果maxStatements与maxStatementsPerConnection均为0，则缓存被关闭。Default: 0-->

<property name=*"maxStatements"* value=*"0"* />

<!--每60秒检查所有连接池中的空闲连接。Default: 0 -->

<property name=*"idleConnectionTestPeriod"* value=*"60"* />

<!--定义在从数据库获取新连接失败后重复尝试的次数。Default: 30 -->

<property name=*"acquireRetryAttempts"* value=*"30"* />

<!--获取连接失败将会引起所有等待连接池来获取连接的线程抛出异常。但是数据源仍有效

保留，并在下次调用getConnection()的时候继续尝试获取连接。如果设为true，那么在尝试

获取连接失败后该数据源将申明已断开并永久关闭。Default: false-->

<property name=*"breakAfterAcquireFailure"* value=*"true"* />

</bean>

百分比进度条

多线程ajax技术

Get和post区别

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。  
4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。  
5. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。   
  
建议：  
1、get方式的安全性较Post方式要差些，包含机密信息的话，建议用Post数据提交方式；  
2、在做数据查询时，建议用Get方式；而在做数据添加、修改或删除时，建议用Post方式；

扩展：

Ajax技术用来加载数据非常方便（比如：extJS）

可以使用多线程的方式加载数据（对应ext来说，一次加载所有的ajax。故第一次加载的时候会比较慢）

普通的Javaweb只能是单线程（只有访问了页面，才能加载）

解决什么需求：

大批量的数据进行保存（10000）

如果使用单线程【保存】（10000）

在使用一个ajax，在开启一个线程，则保存5000，设置一个标识，设置为1。

另一个保存5000,接收到标识不是1的话

是1,则完成整个操作。

项目第四天

1.ajax实现二级联动

分析：

//将数据字典的值转换成json数据

//1.使用jquery的ajax获取 到所属单位的名称

//2。以选择的数据类型作为条件，查询对应数据类型的集合List<ElecSystemDDL>

//3.将List放置到栈顶，将list使用struts2的方式转换成json数据（需要导入struts2支持的json的struts2-json-plugin）

（1）导入struts2支持json的jar包

Struts2-json-plugin-2.3.3.jar

（2）struts.xml文件定义

<package name=*"system"* namespace=*"/system"* extends=*"*json-default*"*>

<result name=*"findJctUnit"* type=*"json"*>

<param name=*"includeProperties"*>\[d+\]\.ddlCode,\[d+\]\.ddlName</param>

</result>

</package>

(3)在Action类中定义

/\*\*

\* @Name: findJctUnit（方法的名称）

\* @Description:选择页面中选择所属单位，使用所属单位查询对应的所属单位的名称（Ajax的二级联动）（方法的描述）

\* @Author: ghq（作者）

\* @Version: V1.00 （版本号）

\* @Create Date: 2016-10-01（创建日期）

\* @Parameters: 无

\* @Return: 跳转到system/userAdd.jsp

\*/

public String findJctUnit(){

//将数据字典的值转换成json数据

//1.使用jquery的ajax获取 到所属单位的名称

String keyword = elecUser.getJctID();

//2。以选择的数据类型作为条件，查询对应数据类型的集合List<ElecSystemDDL>

List<ElecSystemDDL> list = elecSystemDDLService.findSystemDDLByKeyword(keyword);

//3.将List放置到栈顶，将list使用struts2的方式转换成json数据（需要导入struts2支持的json的struts2-json-plugin）

ValueStackUtils.*setValueStack*(list);

return "findJctUnit";

}

（4）页面中显示二级联动

的js

//ajax的二级联动，使用选择的所属单位，查询该所属单位下对应的单位名称列表

function findJctUnit(o){

//货物所属单位的文本内容

var jct = $(o).find("option:selected").text();

/\*\*

jquery对dom对象进行了封装

$post:表示post请求

\*参数一：发送请求的地址

\*参数二：发送数据，json数据格式

\*参数三：function(data,textStatus) 回调函数， data表示从服务器获取的值

\*/

$.post("elecUserAction\_findJctUnit.do",{"jctID":jct},function(data,textStatus){

//先删除单位名称的下拉菜单，但是请选择要留下

$("#jctUnitID option").remove();

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

for(var i=0;i<data.length;i++){

var ddlCode = data[i].ddlCode;

var ddlName = data[i].ddlName;

//添加到单位名称的下拉菜单中

var $option = $("<option></option>");

$option.attr("value",ddlCode);

$option.text(ddlName);

$("#jctUnitID").append($option);

}

}

});

}

（5）同时在页面中要存在两个表单2个select

<s:select list="#request.jctList" name="jctID" id="jctID"

listKey="ddlCode" listValue="ddlName"

headerKey="" headerValue=""

cssStyle="width:155px"

onchange="findJctUnit(this)">

</s:select>

<select id="jctUnitID" name="jctUnitID" style="width:155px"></select>

2.ajax完成登录名校验

分析：

//获取页面传递的登录名

//1.以登录名作为条件查询用户表，返回List<ElecUser>

//2.如果登录名为空，message=1，如果list为空，返回message = 3，如果list不为空，说明数据库中有存在一个该值，返回message = 2.

String message ="";

\* message = 1:表示登录名不能为空

\* message = 2：表示登录名在数据库中已经存在，此时不能保存

\* message = 3：表示登录名不存在，此时可以保存

（1）在jsp页面中定义

/\*\*校验登录名是否出现重复\*/

**function** checkUser(o){

//alert(o.value);//dom的写法

//alert($(o).val());//jquery的写法

**var** logonName = $(o).val();

//以登录名作为查询条件，查询该登录名是否在数据库表中存在记录

$.post("elecUserAction\_checkUser.do",{"logonName":logonName},**function**(data){

**if**(data==1){

$("#check").html("<font color='red'>登录名不能为空</font>");

$(o)[0].focus();

$("#BT\_Submit").attr("disabled","none");

}

**else** **if**(data==2){

$("#check").html("<font color='red'>登录名已经存在</font>");

/\* $(o)[0].focus(); \*/

o.focus();

$("#BT\_Submit").attr("disabled","none");

}

**else**{

$("#check").html("<font color='green'>登录名可以使用</font>");

$("#BT\_Submit").attr("disabled","");

}

});

}

同时下面的jsp中添加校验的文本框

<s:textfield name=*"logonName"* id=*"logonName"* maxLength=*"25"* size=*"20"* onblur=*"checkUser(this)"*></s:textfield>

<div id=*"check"*></div>

（2）在struts.xml的配置

<action name="elecUserAction\_\*" class="elecUserAction" method="{1}">

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

<!--模型驱动中存在一个叫message的属性，将查询结果放置到字符串类型的message中，就可以传递给页面 -->

<param name="root">message</param>

</result>

</action>

（3）字符串形式传递值得时候，在模型驱动的对象中，添加message属性（在javabean中定义message）

ElecUser.java

Private String message;

Set和get方法

（4）在Action中定义checkUser();

/\*\*

\* @Name: checkUser（方法的名称）

\* @Description:使用登陆名作为条件，查询登录名在数据库中是否出现重复（ajax的二级联动）（方法的描述）

\* String message ="";

\* message = 1:表示登录名不能为空

\* message = 2：表示登录名在数据库中已经存在，此时不能保存

\* message = 3：表示登录名不存在，此时可以保存

\*

\* @Author: ghq（作者）

\* @Version: V1.00 （版本号）

\* @Create Date: 2016-10-01（创建日期）

\* @Parameters: 无

\* @Return:ajax。使用struts2的ajax转发

\*/

public String checkUser(){

//获取页面传递的登录名

//1.获取登录名以登录名作为条件查询用户表，返回List<ElecUser>

String logonName = elecUser.getLogonName();

String message = elecUserService.checkUser(logonName);

//将message放置到栈顶

elecUser.setMessage(message);

return "checkUser";

}

3：在jsp中校验输入的值是否正确（正则表达式）

学会正则表达式的语法，真正用的时候借助于搜索。

**function** verifyDateTime(obj){

**var** v = obj.value;

**var** pattern =/^([1990-2500]{4})-((((0)[0-9])|((1)[0-2])))-(((0|1|2)\d)|((3)[0-1])) (((0|1)\d)|((2)[0-4]))\:(([0-5]\d)|(60))$/;

//

flag = pattern.test(v);

**if**(!flag){

alert("您的日期（yyyy-mm-dd hh:mm）验证有误，请重新输入！");

obj.select();

obj.focus();

**return** **false**;

}**else**{

**return** **true**;

}

}

4：学会使用md5密码加密

场景：针对用户的新增和修改保存

（1）引入md5keybean，放置到utils包下

（2）使用md5keybean

**public** **static** **void** main(String args[]) {

MD5keyBean m = **new** MD5keyBean();

String md5 = m.getkeyBeanofStr("123456");

System.***out***.println(md5);

}

5：数据字典查询时候（性能优化），使用hibernate的二级缓存

（1）导入对应的jar包

需要引入三个jar包

在hibernate下能找到

hibernate-distribution-3.5.6-Final\lib\optional\ehcache\ehcache-1.5.0.jar

在srping下能找到

..\lib\concurrent\backport-util-concurrent.jar

..\lib\jakarta-commons\commons-logging.jar

（2）在hibernate.cfg.xml中配置

<!-- 开启二级缓存 -->

<property name=*"hibernate.cache.use\_second\_level\_cache"*>true</property>

<!--提供EhCache的供应商 -->

<property name=*"hibernate.cache.provider\_class"*>org.hibernate.cache.EhCacheProvider</property>

<!-- 开启查询缓存（因为电力系统中使用了查询缓存（hql语句）） -->

<property name=*"hibernate.cache.use\_query\_cache"*>true</property>

<!-- 指定使用二级缓存的类 放在maping下面 -->

<!-- 配置类级别的二级缓存 -->

<class-cache class=*"itest.elec.domain.ElecSystemDDL"* usage=*"read-write"*/>

（3）：项目中使用二级缓存的目的

在使用数据字典的时候  
 \*使用数据类型查询数据字典，获取对应的数据项的值，列表List<ElecSystemDDL>，查询数据库

\*使用数据项的编号和数据类型，获取对应的数据项的值，查询数据库

\*使用数据项的值和数据类型，获取对应的数据项的编号，查询数据库

每次查询都要查询数据库，性能得不到保证

能不能使用一条sql语句查询获取结构，就不要用多条语句查询

\*使用联合查询

（4）项目中使用查询缓存，在类路径下，添加ehcache.xml文件

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">

<diskStore path="java.io.tmpdir"/>

<defaultCache

maxElementsInMemory="10000"

eternal="true"

overflowToDisk="true"

maxElementsOnDisk="10000000"

diskPersistent="false"

diskExpiryThreadIntervalSeconds="120"

memoryStoreEvictionPolicy="LRU"

/>

</ehcache>

（5）：使用查询缓存，在所有Query接口中定义hql语句执行的过程中，最后要使用query .setCacheable(true);开启二级缓存

（6）：注意事项

类级别的缓存区域

<class>

\*存放对象的散装数据，而且是对象中的所有的属性都会放置到类级别的缓存区域

集合级别的缓存区域

<Set>电力系统没有涉及到

\*配置二级缓存，因为查询缓存依赖于二级缓存，依赖于类级别的缓存

\*查询缓存中存放的对象，oid，而真正oid对应的实体的值，任然放置到类级别的缓存区域

查询级别的缓存区域

\*配置二级缓存，因为查询缓存依赖于二级缓存，依赖于类级别的缓存

\*查询缓存中存放的对象，oid，而真正oid对应的实体的值，任然放置到类级别的缓存区域

Sql语句联合查询优化

能不能呢使用一条sql语句查询结果就不要使用多条语句查询

\*使用联合查询即可

left outer join on (left join on); left关键字左侧表的数据全部显示出来，如果2个匹配结果没有关联的字段用null表示

select a.userID,a.logonName,a.userName,a.contacTel,b.ddlName From elec\_user a

left outer join elec\_systemddl b on a.sexID = b.ddlCode

right outer join on (right join on); right关键字右侧表的数据全部显示出来，如果2个匹配结果没有关联的字段用null表示

inner join on (join on);2个表必须完全匹配，才能查询到对应的结果

1:jquery的ajax常用的有几种方法？语法是什么？

$.get(url,{},function(data){

});

$.post(url),{},function(data){

});

$.ajax({

Type:”post”,

url:””,

data:{},

success:function(data){}

})

2使用struts2开发，如何使用ajax调用json数据的

（1）struts支持json的插件包引进来

（2）struts.xml文件中的<package extends=”json-default”>

(3)在<action>中的类型上添加type=”json”

3项目中使用了二级缓存了吗？为什么要使用二级缓存，使用二级缓存的那个缓存，这个缓存还需要注意些什么

（1）使用了；

（2）提升效率，每次操作不用查询数据录，直接差缓存；

在数据字典中，每次检索数据项，也包括数据项进行转换的时候，频繁地查询数据库，由于数据字典在上线后，不会发生变化，将查询或取得数据字典的结果放置到缓存中，下次再查询的时候，直接查询缓存获取；

（3）查询缓存

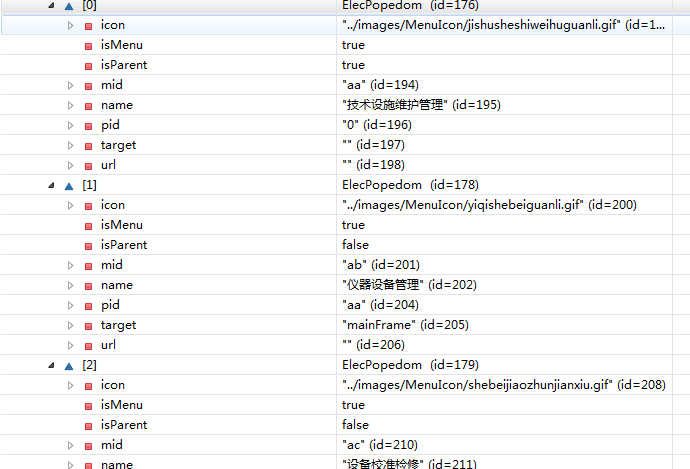
（4）要求：频繁读取，不会发生变化的数据

查询缓存依赖于危机缓存（类级别的二级缓存）

项目第五天

1：便历树形结构的表格

(1)服务器组织查询所有值List<>



页面显示



使用Struts2的标签实现

<table cellspacing=*"0"* align=*"center"* width=*"100%"* cellpadding=*"1"* rules=*"all"* bordercolor=*"gray"* border=*"1"*

style="BORDER-RIGHT:*gray 1px solid*; BORDER-TOP:*gray 1px solid*; BORDER-LEFT:*gray 1px solid*; WORD-BREAK:*break-all*; BORDER-BOTTOM:*gray 1px solid*; BORDER-COLLAPSE:*collapse*; BACKGROUND-COLOR:*#f5fafe*; WORD-WRAP:*break-word*">

<s:if test=*"#request.popedomList!=null && #request.popedomList.size()>0"*>

<s:set value="%{'1'}" var="flag" scope="request"></s:set>

<s:iterator value=*"#request.popedomList"* var=*"popedom"*>

<!--父节点 -->

<s:if test=*"#popedom.isParent"*>

<s:if test=*"#request.flag=='2'"*>

</td>

</tr>

<s:set value=*"%{'1'}"* var=*"flag"* scope=*"request"*></s:set>

</s:if>

<tr onmouseover="this.style.backgroundColor = 'white'" onmouseout="this.style.backgroundColor = '#F5FAFE';">

<td class=*"ta\_01"* align=*"left"* width=*"18%"* height=*"22"* background=*"../images/tablehead.jpg"* >

<input type=*"checkbox"* name=*"selectoper"* id=*"*<s:property value=*"#popedom.mid"*/>*\_*<s:property value=*"#popedom.mid"*/>*"* value=*""* onClick='goSelect(this.id)' >

<s:property value=*"#popedom.name"*/>:

</s:if>

<!--子节点 -->

<s:else>

<s:if test=*"#request.flag=='1'"*>

<td class=*"ta\_01"* align=*"left"* width=*"82%"* height=*"22"*>

<s:set value=*"%{'2'}"* var=*"flag"* scope=*"request"*></s:set>

</s:if>

<div>

<input type=*"checkbox"* name=*"selectoper"* id=*"*<s:property value=*"#popedom.pid"*/>*\_*<s:property value=*"#popedom.mid"*/>*"* value=*""* onClick='goSelect(this.id)' >

<s:property value=*"#popedom.name"*/>

</div>

</s:else>

</s:iterator>

</s:if>

</table>