nirvana1988

浏览: 2810 次

性别:

来自: 郑州

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

绿窗明月 long588

xiaokang1582 hanfeng450

博客分类

- 全部博客 (10)
- 框架整合 (2)
- web前端 (5)
- j2se基础 (2)
- JVM (1)

我的留言簿 >>更多留言

其他分类

- 我的收藏 (19)
- 我的代码 (0)
- 我的书籍 (1)
- 我的论坛主题帖 (0)
- 我的所有论坛帖 (35)
- 我的精华良好帖 (0)

精确获取JS变量类型

2011-03-14

mybatis3.0 多对多映射CRUD操作实例

博客分类: 框架整合

iBATIS XML junit Apache JDBC

mybatis3.0添加了association和collection标签专门用于对多个相关实体类数据进行级联查询，但仍不支持多个相关实体类数据的级联保存和级联删除操作。因此在进行实体类多对多映射表设计时，需要专门建立一个关联对象类对相关实体类的关联关系进行描述。下文将以“User”和“Group"两个实体类之间的多对多关联映射为例进行CRUD操作。

建立user表，对应实体类"User"，建表语句如下：

<!--StartFragment-->

Sql代码

```sql
CREATE TABLE `user` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) collate utf8_unicode_ci default NULL,
  `password` varchar(20) collate utf8_unicode_ci default NULL,
  `createtime` timestamp NULL default CURRENT_TIMESTAMP,
  PRIMARY KEY  (`id`)
)
```

建立group_info表，对应实体类"Group"，建表语句如下：

Sql代码

```sql
CREATE TABLE `group_info` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) collate utf8_unicode_ci default NULL,
  `createdate` timestamp NULL default CURRENT_TIMESTAMP,
  `state` int(1) default '0' COMMENT '0:可见;1:不可见',
  PRIMARY KEY  (`id`)
)
```

建立user_group表，对应实体类"UserGroupLink"(该类为User和Group两个实体类之间的关系描述)，建表语句如下：

Sql代码

```sql
CREATE TABLE `user_group` (
  `user_id` int(11) default NULL,
  `group_id` int(11) default NULL,
  `createdate` timestamp NULL default CURRENT_TIMESTAMP,
  KEY `FK_user_group_user_id` (`user_id`),
  KEY `FK_user_group_group_id` (`group_id`),
  CONSTRAINT `FK_user_group_group_id` FOREIGN KEY (`group_id`) REFERENCES `group_info` (`id`),
  CONSTRAINT `FK_user_group_user_id` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`)
)
```

建立实体类User，代码如下：

最近加入群组

- Vim编辑器
- Jquery
- 每日一题JAVA
- 济南JavaEyer群
- 设计模式

存档

- 2011-08 (1)
- 2011-07 (1)
- 2011-06 (3)
- 更多存档...

评论排行榜

- spring与mybatis三种整合方法
- AJAX传值中文乱码解决方法
- mybatis3.0 多对多映射CRUD操作实例

RSS
Add to Google

Java代码

```java
package com.xxt.ibatis.dbcp.domain;

import java.util.Date;
import java.util.List;

/**
 * @describe: User实体类
 * @author:    Nirvana
 * @version:   V1.0  2011-3-11下午06:06:49 create
 */

public class User {

        private long id;

        private String name;

        private String password;

        private Date createTime;

        private List<Group> group;

        public Date getCreateTime() {
                return createTime;
        }

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

        public long getId() {
                return id;
        }

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

        public String getName() {
                return name;
        }

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

        public String getPassword() {
                return password;
        }

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

        public List<Group> getGroup() {
                return group;
        }

        public void setGroup(List<Group> group) {
                this.group = group;
        }
}
```

建立实体类Group,代码如下:

Java代码

```java
package com.xxt.ibatis.dbcp.domain;

import java.util.Date;
import java.util.List;

/**
 * @describe: Group实体类
 * @author: Nirvana
 * @version: V1.0 2011-3-7下午08:10:29 create
 */
public class Group {

        private long id;

        private String name; // 组名

        private Date createTime;

        private int state; // 0可见状态 1不可见状态

        private List<User> user;

        public Date getCreateTime() {
                return createTime;
        }

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

        public long getId() {
                return id;
        }

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

        public String getName() {
                return name;
        }

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

        public int getState() {
                return state;
        }

        public void setState(int state) {
                this.state = state;
        }

        public List<User> getUser() {
                return user;
        }

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

}
```

建立实体类UserGroupLink，用于描述User和Group之间的对应关系，代码如下：

Java代码

```java
package com.xxt.ibatis.dbcp.domain;

import java.util.Date;

/**
 * @describe: 描述User和Group之间的映射关系
 * @author: Nirvana
 * @version: V1.0 2011-3-11下午02:57:52 create
 */
public class UserGroupLink {

        private User user;

        private Group group;

        private Date createTime;

        public Date getCreateTime() {
                return createTime;
        }

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

        public Group getGroup() {
                return group;
        }

        public void setGroup(Group group) {
                this.group = group;
        }

        public User getUser() {
                return user;
        }

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

建立user实体类的映射文件user.map.xml，代码如下：

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxt.ibatis.dbcp.domain.User">
        <resultMap type="User" id="userMap">
                <id property="id" column="id" />
                <result property="name" column="name" />
                <result property="password" column="password" />
                <result property="createTime" column="createtime" />
        </resultMap>

        <resultMap type="User" id="userGroupMap" extends="userMap">
                <collection property="groups" ofType="Group">
                        <id property="id" column="goupId" />
                        <result property="name" column="groupName" />
                        <result property="state" column="state" />
                        <result property="createTime" column="groupCreateTime" />
                </collection>
        </resultMap>


        <!-- 根据user表中的id查询用户信息 -->
        <select id="selectUser" parameterType="long" resultMap="userMap">
                select * from user where id = #{id}
        </select>

        <!-- 根据user表中的id查询用户和组信息 -->
        <select id="selectUserGroup" parameterType="long"
                resultMap="userGroupMap">
                select u.id,u.name,u.password,u.createtime, gi.id as
                goupId,gi.name as groupName, gi.createdate as groupCreateTime,
                gi.state from user u left join user_group ug on u.id=ug.user_id
                left join group_info gi on ug.group_id=gi.id where u.id = #{id}
        </select>

        <!-- 插入用户信息 -->
        <insert id="saveUser" parameterType="User" keyProperty="id"
                useGeneratedKeys="true">
                insert into user(name,password) values(#{name},#{password})
        </insert>

        <!-- 保存用户和组之间的关系信息 -->
        <insert id="saveRelativity" parameterType="UserGroupLink">
                insert into user_group(user_id,group_id)
                values(#{user.id},#{group.id})
        </insert>

        <select id="selectAllUser" resultMap="userMap">
                select * from user
        </select>

</mapper>
```

建立group实体类的映射文件group.map.xml，代码如下:

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.xxt.ibatis.dbcp.domain.Group">
        <resultMap type="Group" id="groupMap">
                <id property="id" column="id" />
                <result property="name" column="name" />
                <result property="createTime" column="createdate" />
        </resultMap>

        <resultMap type="Group" id="groupUserMap" extends="groupMap">
                <collection property="users" ofType="User">
                        <id property="id" column="userId" />
                        <result property="name" column="userName" />
                        <result property="password" column="password" />
                <result property="createTime" column="userCreateTime" />
            </collection>
        </resultMap>


        <!-- 根据Group表中的id或name查询组信息和组内用户信息 -->
        <select id="selectGroupUser" parameterType="Group"
                resultMap="groupUserMap">
                select u.id as userId,u.name as userName,
                u.password,u.createtime as userCreateTime,
                gi.id,gi.name,gi.createdate,gi.state from group_info gi left
                join user_group ug on gi.id=ug.group_id left join user u on
                uug.user_id=u.id
                <where>
                        <!--当id为初始值0，不再使用id作为查询条件 -->
                        <if test="id != 0  ">gi.id=#{id}</if>
                        <!-- 当name为空或为空串时，不再使用name作为查询条件 -->
                        <if test="name != null and name != ''">
                                or gi.name = #{name}
                        </if>
                </where>
        </select>

        <!-- 根据id查询group组信息 -->
        <select id="selectGroup" parameterType="Date"
                resultMap="groupMap">
                select * from group_info where id=#{group_id}
        </select>

        <!--根据name查询group组信息  -->
        <select id="getGroupByName" parameterType="String"
                resultMap="groupMap">
                select * from group_info where name=#{name}
        </select>

        <!-- 插入组信息 -->
        <insert id="saveGroup" parameterType="Group" keyProperty="id"
                useGeneratedKeys="true">
                insert into group_info(name) values(#{name})
        </insert>

        <!-- 删除组与组内成员之间的对应关系 -->
        <delete id="deleteGroupUser" parameterType="UserGroupLink">
                delete from user_group
                <where>
                        <if test="user.id != 0">user_id = #{user.id}</if>
                        <if test="group.id != 0">and group_id = #{group.id}</if>
                </where>
        </delete>

        <!--根据组id或者组name删除组信息 -->
        <delete id="deleteGroup" parameterType="Group">
                delete from group_info
                <where>
                        <if test="id!=0">id=#{id}</if>
                        <if test="name!=null || name!=''">and name=#{name}</if>
                </where>
        </delete>

        <!-- 更新根据组id或者组name更新组状态 -->
        <update id="updateGroupState" parameterType="Group">
                update group_info set state=#{state}
                <where>
                        <if test="id!=0">id=#{id}</if>
                        <if test="name!=null || name!=''">and name=#{name}</if>
                </where>
        </update>
</mapper>
```

建立ibatis总配置文件salMapConfig.xml,代码如下:

Xml代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
    PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
        <properties resource="jdbc.properties" />

        <typeAliases>
                <typeAlias type="com.xxt.ibatis.dbcp.domain.User" alias="User" />
                <typeAlias type="com.xxt.ibatis.dbcp.domain.Group"
                        alias="Group" />
                <typeAlias type="com.xxt.ibatis.dbcp.domain.UserGroupLink"
                        alias="UserGroupLink" />
        </typeAliases>

        <environments default="development">
                <environment id="development">
                        <transactionManager type="JDBC" />
                        <dataSource type="UNPOOLED">
                                <property name="driver" value="${driver}" />
                                <property name="url" value="${url}" />
                                <property name="username" value="${username}" />
                                <property name="password" value="${password}" />
                        </dataSource>
                </environment>
        </environments>

        <mappers>
                <mapper resource="com/xxt/ibatis/dbcp/domain/user.map.xml" />
                <mapper resource="com/xxt/ibatis/dbcp/domain/group.map.xml" />
        </mappers>
</configuration>
```

测试用例，代码如下:

Java代码

```java
package com.xxt.ibatis.dbcp.test;

import java.io.Reader;
import java.text.SimpleDateFormat;
import java.util.List;

import org.junit.AfterClass;
import org.junit.BeforeClass;
import org.junit.Test;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import com.xxt.ibatis.dbcp.domain.Group;
import com.xxt.ibatis.dbcp.domain.User;
import com.xxt.ibatis.dbcp.domain.UserGroupLink;

```

```
/**
 * @describe: 测试用例
 * @author:    Nirvana
 * @version:   V1.0  2011-3-4下午03:21:17 create
 */
public class IbatisUserTest {

    private final static String IBATIS_CONFIG_XML  = "sqlMapConfig.xml";
    private static SqlSession session;
    private SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd k:mm:ss");

        @BeforeClass
        public static void setUpBeforeClass() throws Exception {
                 Reader reader = Resources.getResourceAsReader(IBATIS_CONFIG_XML);  //读取ibatis配置文件
                 SqlSessionFactory sqlMapper = new SqlSessionFactoryBuilder().build(reader);
                  session = sqlMapper.openSession(true);
                  session.commit(false); //将默认提交事务属性设置为否
        }


        //保存用户信息
        @Test
        public void saveUserTest(){
                User user = new User();
                user.setName("张三");
                user.setPassword("123456");
                session.insert("com.xxt.ibatis.dbcp.domain.User.saveUser", user);
        }

        //获取用户信息
        @Test
        public void getUserTest(){
                User user = (User) session.selectOne("com.xxt.ibatis.dbcp.domain.User.selectUser", 1L);
                System.out.println("用户名:  "+user.getName());
                System.out.println("用户密码:  "+user.getPassword());
                System.out.println("用户创建日期: "+sdf.format(user.getCreateTime()));
        }

        //获取用户和用户所在组信息
        @Test
        public void getUserAndGroupTest(){
                 User  user =  (User)session.selectOne("com.xxt.ibatis.dbcp.domain.User.selectUserGroup", 4L);

                 System.out.println(user.getName() +"所属组信息:");
                for(Group group : user.getGroups() ){
                        System.out.println("    组名:"+group.getName()+", 组创建时间:"+sdf.format(group.getCreateTime()));
                }


        }

        //保存用户和用户所在组信息
    //当用户所在组不存在时，创建该组，并生成映射关系
        @Test
        public void saveUserAndGroupTest(){
        User user = new User();
            user.setName("无常鬼");
                user.setPassword("wuchang");
                session.insert("com.xxt.ibatis.dbcp.domain.User.saveUser", user);

        Group groupImpl = (Group)session.selectOne("com.xxt.ibatis.dbcp.domain.Group.getGroupByName","用户组4");//获取组实例
        UserGroupLink ugl = new UserGroupLink();//声明User和Group实体间映射关系实例

        //查询到的组实例为空时的逻辑处理
        if(groupImpl == null){
             Group group = new Group();
             group.setName("用户组4");
            session.insert("com.xxt.ibatis.dbcp.domain.Group.saveGroup", group);//持久化创建好的组实例

            //设置映射关系实例相关的属性
            ugl.setUser(user);
            ugl.setGroup(group);
      session.insert("com.xxt.ibatis.dbcp.domain.User.saveRelativity",ugl);//持久化映射关系实力

        }else{
          ugl.setGroup(groupImpl);
            ugl.setUser(user);
            session.insert("com.xxt.ibatis.dbcp.domain.User.saveRelativity",ugl);
        }
    }


    //删除组信息的同时，取消组内所有的成员与该组的关联关系
    @Test
    public void deleteGroupTest(){
        Group group = new Group();
        //group.setId(1L);  //以组id作为查询条件
        group.setName("用户组1"); //以组name作为查询条件
        Group groupImpl = (Group)session.selectOne("com.xxt.ibatis.dbcp.domain.Group.selectGroupUser",group);//获取组实例

        //组实例存在时
      if(groupImpl != null){

        List<User> users = groupImpl.getUsers();

     //查看用户组1中是否存在用户
        if(users != null && users.size() > 0){

            //存在用户时，先删除组与用户的对应关系
            UserGroupLink ugl = new UserGroupLink();
            for(User user : users){
               ugl.setUser(user);
               ugl.setGroup(groupImpl);
               session.delete("com.xxt.ibatis.dbcp.domain.Group.deleteGroupUser",ugl );
            }
        }
        //删除组信息
        session.delete("com.xxt.ibatis.dbcp.domain.Group.deleteGroup", groupImpl);

      }else{
        throw new RuntimeException("查询的组不存在!!");
      }


    }


    //更新组状态，当组状态由可见状态变成不可见时，取消该组下的用户与该组的映射关系
    @Test
```

```java
        public void updateGroupStateTest(){
                Group group = new Group();
                group.setName("用户组2");
                Group groupImpl = (Group) session.selectOne("com.xxt.ibatis.dbcp.domain.Group.selectGroupUser",group);

                if(groupImpl != null){
                        int state = groupImpl.getState() == 1 ? 0 : 1;


                        //组状态由0变成1时, 即由可见变为不可见
                        if(state == 1){
                                List<User> users = groupImpl.getUsers();
           //查看用户组2中是否存在用户
                        if(users != null && users.size() > 0){

                                //存在用户时, 删除组与用户的对应关系
                                UserGroupLink ugl = new UserGroupLink();
                                for(User user : users){
                                   ugl.setUser(user);
                                   ugl.setGroup(groupImpl);
                                   session.delete("com.xxt.ibatis.dbcp.domain.Group.deleteGroupUser",ugl );
                                }
                        }
                        }

                        //更新组状态
                        groupImpl.setState(state);
                }
                else{
                        throw new RuntimeException("查询的组不存在!!");
                }

        }

        @AfterClass
        public static void tearDownAfterClass() throws Exception {
          if(session != null){
                     session.commit(); //提交事务
                     session.close(); //关闭连接
          }
        }
}
```

分享到:

精确获取JS变量类型

19:52 | 评论 / 浏览 (1 / 364) | 分类:企业架构 | 相关推荐 MORE

评论

1 楼 LanderRooter 2011-08-22

楼主, 您好,

public class UserGroupLink {

  private User user;

  private Group group;

```
    private Date createTime;
    .....
}
```

Java代码

```
<resultMap type="User" id="userMap">
    <id property="id" column="id" />
    <result property="name" column="name" />
    <result property="password" column="password" />
    <result property="createTime" column="createtime" />
</resultMap>

<resultMap type="User" id="userGroupMap" extends="userMap">
    <collection property="groups" ofType="Group">
        <id property="id" column="goupId" />
        <result property="name" column="groupName" />
        <result property="state" column="state" />
        <result property="createTime" column="[b]groupCreateTime[/b]" />
    </collection>
</resultMap>
```

文章中如上的映射，并没有将UserGroupLink的createDate映射到UserGroupLink，而是把groupCreateTime映射到Group的createTime属性

Java代码

```
<select id="selectUserGroup" parameterType="long"
    resultMap="userGroupMap">
    select u.id,u.name,u.password,u.createtime, gi.id as
    goupId,gi.name as groupName, gi.createdate as [b]groupCreateTime[/b],
    gi.state from user u left join user_group ug on u.id=ug.user_id
    left join group_info gi on ug.group_id=gi.id where u.id = #{id}
</select>
```

关联对象类UserGroupLink中的createTime属性该如何映射呢，在此例中并没有把UserGroupLink的createTime 映射出来，如果此关联对象类还有boolean enable等属性，可不可以直接将结果映射成UserGroupLink呢？，如何可以的话，该如何映射呢，谢谢。小弟最近在项目中要Mybatis,也遇到类似的case，不是很清楚如何解决，谢谢。

发表评论

您还没有登录,请您登录后再发表评论