Grouper Client JDBC API

If you are syncing from Grouper to a JDBC database, you might want to use the Grouper client. The Atlassian connector does this. You can use the Grouper Client JDBC API if you like (or whatever you are comfortable with, Spring, Hibernate, whatever). If you want to use the Grouper JDBC API, you can map pojos to tables with annotations and use GcDbAccess to manage the persistence. Here is an example for an Atlassian user:

```
/**
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouperAtlassianConnector.db;

import java.sql.Timestamp;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

import edu.internet2.middleware.grouperClient.jdbc.GcDbAccess;
import edu.internet2.middleware.grouperClient.jdbc.GcPersist;
import edu.internet2.middleware.grouperClient.jdbc.GcPersistableClass;
import edu.internet2.middleware.grouperClient.jdbc.GcPersistableField;
import edu.internet2.middleware.grouperClient.jdbc.GcSqlAssignPrimaryKey;


/**
 *
 */
@GcPersistableClass(tableName="CWD_USER", defaultFieldPersist=GcPersist.persistIfPersistableField)
public class AtlassianCwdUser implements GcSqlAssignPrimaryKey {

  /**
   * id col
   */
  @GcPersistableField(primaryKey=true, primaryKeyManuallyAssigned=true)
  private Long id;

  /**
   * id col
   * @return the id
   */
  public Long getId() {
    return this.id;
  }


  /**
   * id col
   * @param id1 the id to set
   */
  public void setId(Long id1) {
    this.id = id1;
  }

  /**
   * user_name col
   */
  @GcPersistableField
  private String userName;

  /**
   * user_name col
   * @return the userName
   */
  public String getUserName() {
    return this.userName;
  }
```

```
/**
 * user_name col
 * @param userName1 the userName to set
 */
public void setUserName(String userName1) {
  this.userName = userName1;
}

/**
 * active col
 */
@GcPersistableField
private Long active;

/**
 * active col
 * @return the active
 */
public Long getActive() {
  return this.active;
}


/**
 * active col
 * @param active1 the active to set
 */
public void setActive(Long active1) {
  this.active = active1;
}

/**
 * updated_date col
 */
@GcPersistableField
private Timestamp updatedDate;

/**
 * updated_date col
 * @return the updatedDate
 */
public Timestamp getUpdatedDate() {
  return this.updatedDate;
}


/**
 * updated_date col
 * @param updatedDate1 the updatedDate to set
 */
public void setUpdatedDate(Timestamp updatedDate1) {
  this.updatedDate = updatedDate1;
}

/**
 * directory_id col
 */
@GcPersistableField
private Long directoryId;

/**
 * directory_id col
 * @return the directoryId
 */
public Long getDirectoryId() {
  return this.directoryId;
}

/**
 * directory_id col
 * @param directoryId1 the directoryId to set
```

```
 */
public void setDirectoryId(Long directoryId1) {
  this.directoryId = directoryId1;
}

/**
 * lower_user_name col
 */
@GcPersistableField
private String lowerUserName;



/**
 * lower_user_name col
 * @return the lowerUserName
 */
public String getLowerUserName() {
  return this.lowerUserName;
}


/**
 * lower_user_name col
 * @param lowerUserName1 the lowerUserName to set
 */
public void setLowerUserName(String lowerUserName1) {
  this.lowerUserName = lowerUserName1;
}

/**
 * created date col
 */
@GcPersistableField
private Timestamp createdDate;



/**
 * created date col
 * @return the createdDate
 */
public Timestamp getCreatedDate() {
  return this.createdDate;
}


/**
 * created date col
 * @param createdDate1 the createdDate to set
 */
public void setCreatedDate(Timestamp createdDate1) {
  this.createdDate = createdDate1;
}

/**
 * lower_display_name col
 */
@GcPersistableField
private String lowerDisplayName;


/**
 * lower_display_name col
 * @return the lowerDisplayName
 */
public String getLowerDisplayName() {
  return this.lowerDisplayName;
}
```

```
/**
 * lower_display_name col
 * @param lowerDisplayName1 the lowerDisplayName to set
 */
public void setLowerDisplayName(String lowerDisplayName1) {
  this.lowerDisplayName = lowerDisplayName1;
}

/**
 * lower_email_address col
 */
@GcPersistableField
private String lowerEmailAddress;


/**
 * lower_email_address col
 * @return the lowerEmailAddress
 */
public String getLowerEmailAddress() {
  return this.lowerEmailAddress;
}


/**
 * lower_email_address col
 * @param lowerEmailAddress1 the lowerEmailAddress to set
 */
public void setLowerEmailAddress(String lowerEmailAddress1) {
  this.lowerEmailAddress = lowerEmailAddress1;
}

/**
 * display_name col
 */
@GcPersistableField
private String displayName;


/**
 * display_name col
 * @return the displayName
 */
public String getDisplayName() {
  return this.displayName;
}


/**
 * display_name col
 * @param displayName1 the displayName to set
 */
public void setDisplayName(String displayName1) {
  this.displayName = displayName1;
}


/**
 * email_address col
 */
@GcPersistableField
private String emailAddress;


/**
 * email_address col
 * @return the emailAddress
 */
public String getEmailAddress() {
  return this.emailAddress;
}
```

```
  /**
   * email_address col
   * @param emailAddress1 the emailAddress to set
   */
  public void setEmailAddress(String emailAddress1) {
    this.emailAddress = emailAddress1;
  }


  /**
   * external_id col
   */
  @GcPersistableField
  private String externalId;


  /**
   * external_id col
   * @return the externalId
   */
  public String getExternalId() {
    return this.externalId;
  }


  /**
   * external_id col
   * @param externalId1 the externalId to set
   */
  public void setExternalId(String externalId1) {
    this.externalId = externalId1;
  }


  /**
   * @see java.lang.Object#toString()
   */
  @Override
  public String toString() {
    return "AtlassianCwdUser [id=" + this.id + ", userName=" + this.userName + ", active=" + this.active
        + ", updatedDate=" + this.updatedDate + ", directoryId=" + this.directoryId + ", lowerUserName="
        + this.lowerUserName + ", createdDate=" + this.createdDate + ", lowerDisplayName=" + this.
lowerDisplayName
        + ", lowerEmailAddress=" + this.lowerEmailAddress + ", displayName=" + this.displayName
        + ", emailAddress=" + this.emailAddress + ", externalId=" + this.externalId + "]";
  }

  /**
   * note, you need to commit this (or at least flush) so the id is incremented
   */
  public void initNewObject() {

    this.setExternalId(UUID.randomUUID().toString());
//    Long maxId = new GcDbAccess().sql("select max(id) from cwd_user").select(Long.class);
//    this.setId(maxId + 1);
    this.setActive(1L);
    Timestamp now = new Timestamp(System.currentTimeMillis());
    this.setCreatedDate(now);
    this.setUpdatedDate(now);
    this.setDirectoryId(1L);

  }

  /**
   * get all users
   * @return the users or null by map of username to user
   */
  public static Map<String, AtlassianCwdUser> retrieveUsers() {
```

```
    List<AtlassianCwdUser> resultList = new GcDbAccess().selectList(AtlassianCwdUser.class);
    Map<String, AtlassianCwdUser> resultMap = new LinkedHashMap<String, AtlassianCwdUser>();
    for (AtlassianCwdUser atlassianCwdUser : resultList) {
      resultMap.put(atlassianCwdUser.getUserName(), atlassianCwdUser);
    }
    return resultMap;
  }

  /**
   * store this record insert or update
   */
  public void store() {
    new GcDbAccess().storeToDatabase(this);
  }

  /**
   * delete this record
   */
  public void delete() {
    new GcDbAccess().deleteFromDatabase(this);
  }


  /**
   * @see edu.internet2.middleware.grouperClient.jdbc.GcSqlAssignPrimaryKey#gcSqlAssignNewPrimaryKeyForInsert()
   */
  public void gcSqlAssignNewPrimaryKeyForInsert() {
    if (this.id != null) {
      throw new RuntimeException("Why setting primary key if already exists! " + this.id);
    }
    Long maxId = new GcDbAccess().sql("select max(id) from cwd_user").select(Long.class);
    this.setId(maxId + 1);
  }

}
```

The DB connect info is in the grouper.client.properties. Currently there is no connection pooling since this is for simple feeds, but you should open a connection and use that connection for all DB access, then close it.

```
    new GcDbAccess().callbackConnection(new GcConnectionCallback() {

     @Override
     public Object callback(Connection connection) {


       //do work (connection is in threadlocal, all DB access will use it)
     }
    });
```

Config

```
 # default database connection name
grouperClient.jdbc.defaultName = default

# the part between jdbc. and the last . is the name of the connection, in this case "default"
# e.g. mysql:           com.mysql.jdbc.Driver
# e.g. p6spy (log sql): com.p6spy.engine.spy.P6SpyDriver
#   for p6spy, put the underlying driver in spy.properties
# e.g. oracle:          oracle.jdbc.driver.OracleDriver
# e.g. hsqldb:          org.hsqldb.jdbcDriver
# e.g. postgres:        org.postgresql.Driver
# e.g. mssql:           com.microsoft.sqlserver.jdbc.SQLServerDriver
grouperClient.jdbc.default.driver = oracle.jdbc.driver.OracleDriver

# e.g. mysql:           jdbc:mysql://localhost:3306/grouper
# e.g. p6spy (log sql): [use the URL that your DB requires]
# e.g. oracle:          jdbc:oracle:thin:@server.school.edu:1521:sid
# e.g. hsqldb (a):      jdbc:hsqldb:dist/run/grouper;create=true
# e.g. hsqldb (b):      jdbc:hsqldb:hsql://localhost:9001/grouper
# e.g. postgres:        jdbc:postgresql://localhost:5432/database
# e.g. mssql:           jdbc:sqlserver://localhost:3280
grouperClient.jdbc.default.url = jdbc:oracle:thin:@server.school.edu:1521:sid

grouperClient.jdbc.default.user = user
grouperClient.jdbc.default.pass = pass
```