Managing Objects

This chapter contains the following sections:

Object Store

The Object Store provides simple APIs for database persistence. A module that needs to persist objects into the database typically uses the Object Store APIs to perform all the CRUD (Create, Read, Update, and Delete) operations.

Cisco UCS Director uses MySQL as its database. The platform runtime makes use of the Java Data Object (JDO) library provided by DataNucleus to abstract all the SQL operations through an Object Query representation. This simplifies and speeds up the development with respect to data persistence. The Object Store documentation include sections that show how CRUD operations are realized using JDO.

Note This documentation uses the acronym POJO (Plain Old Java Object) to refer to a java class that does not extend any other class or implement any interfaces.

Marking a Class for Persistence

A POJO class that needs to be persisted in the database has to be defined and marked with suitable JDO annotations. The class shown below is marked for JDO persistence.

In this class, note that

- `foo_netapp_filer` is attached on top of the class declaration.
- The `table` attribute specifies the name of the table to be used.
- `foo` is the name of the module.
- `@Persistent` is attached to the field that needs persistence.

```
package com.cloupia.lib.cIaaS.netapp.model;
```

```
@PersistenceCapable(detachable = "true", table = "foo_netapp_filer")
public class  NetAppFiler
{
    @Persistent
    private String filerName;

    @Persistent
    private String accountName;

    @Persistent
    private String dcName;
}
```

The above class has two annotations: `@PersistenceCapable` and `@Persistent`. These are defined in the JDO, and the Cisco UCS Director Platform runtime expects all persistent classes to be marked with these two annotations. Cisco UCS Director uses a flat schema, so creating a nested schema, though possible and allowed in JDO, is not recommended in a Module.

What to do next

The persistence class is now ready for CRUD operations against the database.

Publishing the Persistence Class

A class that is marked with suitable JDO annotation has to be published so that the Platform Runtime can pick up the class.

SUMMARY STEPS

1. Create a file with the name `jdo.files` in the same directory (package) as that of the persistence class.
2. Add the name of the class to the file as follows:

DETAILED STEPS

Step 1 Create a file with the name `jdo.files` in the same directory (package) as that of the persistence class.

Step 2 Add the name of the class to the file as follows:

Example:

```
Linux# cat jdo.files

// Copyright (C) 2010 Cisco Inc. All rights reserved.
//
// Note: all blank lines and lines that start with // are ignored
//
// Each package that has Persistable Objects shall have a file called jdo.files
// Each line here indicates one class that represents a persistable object.
//
// Any line that starts with a + means package name is relative to current package
// If a line starts without +, then it must be complete fully qualified java class name
// (for example: com.cloupia.lib.xyz.MyClass)

+NetAppFiler

Linux#
```

Performing CRUD Operations on the Persistence Class

When a persistence class is ready for CRUD operations against the database, you can perform the different operations available, as shown in the following examples.

Create a New Instance of the Object

```
NetAppFiler filer = new NetAppFiler();
filer.setAccountName("netapp-account");
filer.setDcName("Default Datacenter");
filer.setfilerName("filer0");
filer.setIpAddress("192.168.0.1");

ObjStore store = ObjStoreHelper.getStore(NetAppFiler.class);
    store.insert(filer);
```

Modify a Single Instance of the Object

```
ObjStore store = ObjStoreHelper.getStore(NetAppFiler.class);
String query = "filerName == 'filer0'";
//Use Java field names as parameter,

    // can use && , || operators in the query.
    store.modifySingleObject(query, filer);
```

Querying All the Instances from the Database

```
ObjStore store = ObjStoreHelper.getStore(NetAppFiler.class);

List filerList = store.queryAll();
```

Querying the Instances with a Filer Query

```
ObjStore store = ObjStoreHelper.getStore(NetAppFiler.class);

String query = "dcName == 'Default Datacenter'";
List filerList = store.query(query);
```