Java iControl Objects - LTM Pool

Joe Pruitt, 2010-22-09

The first comment I usually get when developers look at the iControl API has something to do with it's "flat" nature. Developers are used to working with "objects". They create an instance of an object and then get and set attributes on that object. In iControl-speak, they would create a pool object and then get and set the load balancing method on that pool they were holding on to.

iControl was meant to be a very low level API to give you access to every type of object and it's attributes. But iControl was designed as a Web Services API, and thus, it was exposed as a set of methods, not as objects.

But, that doesn't mean we can't make iControl look like a set of objects, does it!

Rounding The Flatness

Fortunately, iControl is grouped into a set of "interfaces" that, in most cases, can be treated as objects. Pools, PoolMembers, VLANs, and WideIPs, can all be thought of as objects.

This is the first article in a series centered around creating a object library around the various types of iControl objects. In this article I will start with where most of the initial iControl applications begin with: Pools. I would have started with Pool Members, but a Pool Member needs to belong to a Pool and I couldn't create the PoolMember object without first defining the Pool object. So, be on the lookout for that one coming next time around.

Prerequisites

The code in this article, relies on the iControl library for Java that can be downloaded in it's DevCentral labs project. If you haven't watched it already, check out my previous article on Configuring Eclipse for iControl development with Java.

The Class

I'll be using the package "iControl.Objects" for this code library I'm building. The classes will be in sub packages for the product (LTM, GTM, ASM, etc) and the class names will be the same as the existing iControl interfaces. For the Pool class, we need the underlying iControl.Interfaces object from the iControl Library for Java as well as a name for the pool.

```
1: package iControl.Objects.LTM;
2:
3: public class Pool {
4:     private iControl.Interfaces _interfaces = null;
5:     private String _name = null;
6:
7:
8:     //--------------------------------------------------------------------
9:     // Member Accessors
10:    //--------------------------------------------------------------------
11:    public iControl.Interfaces getInterfaces() { return _interfaces; }
12:    public void setInterfaces(iControl.Interfaces interfaces) { _interfaces = interfaces; }
13:
14:    public String getName() { return _name;    }
15:    public void setName(String name) { _name = name; }
16:
17:    //--------------------------------------------------------------------
18:    // Constructors
19:    //--------------------------------------------------------------------
20:    public Pool(iControl.Interfaces interfaces, String name)
21:    {
22:        _interfaces = interfaces;
23:        _name = name;
24:    }
25:    ...
```

Above, I've defined the member variables, the accessor methods for those variables, and the constructor that takes as input the defining values of the Pool.

Object Attribute Accessors

If you look at the iControl API methods in the Pool interface, you'll see lots of "get_*" and "set_*" methods. We defined the API in this way to make it very easy to know which attributes you can set and which you can retrieve. Some are read-only and, in that case, will only have a get_* method. The next section of the class will include all the get_ and set_ methods. Here's the code for the get_lb_method() and set_lb_method() methods.

```
1: // lb_method
2: public void setLBMethod(iControl.LocalLBLBMethod lbMethod) throws Exception
3: {
4:     validateMembers();
5:
6:     String [] pool_names = { _name };
7:     iControl.LocalLBLBMethod [] lb_methods = { lbMethod};
8:
9:     _interfaces.getLocalLBPool().set_lb_method(pool_names, lb_methods);
10: }
11: public iControl.LocalLBLBMethod getLBMethod() throws Exception
12: {
13:     validateMembers();
14:
15:     String [] pool_names = { _name };
16:     iControl.LocalLBLBMethod [] lb_methods =
17:         _interfaces.getLocalLBPool().get_lb_method(pool_names);
18:
19:     return lb_methods[0];
20: }
```

You'll notice that I've chosen to abstract away all the arrays in the method calls. This removes the burden of packaging up single and two-dimensional arrays before making method calls for a single object.

I've repeated this type of methodology for all the other get_* and set_* methods in the LocalLB.Pool interface.

Action Methods

Each iControl interface has a set of actions you can perform. Whether it's creating a new pool, removing a pool, or adding members to a pool, these methods need to be exposed to the user. Below is how I chose to expose the ability to add new members to the Pool. It essentially wraps the iControl LocalLB.Pool.add_member() method.

```
1: public void addMembers(iControl.CommonIPPortDefinition [] members) throws Exception
2: {
3:     validateMembers();
4:
5:     String [] pool_names = { _name };
6:     iControl.CommonIPPortDefinition [][] membersAofA = { members };
7:
8:     _interfaces.getLocalLBPool().add_member(pool_names, membersAofA);
9: }
```

I've also included methods that allow you to create the pool, delete the pool, delete persistence records, querying the pool members, removing pool members, and querying statistics.

Static Methods

Static methods are meant to be able to be used without an instance of the object. Querying the list of available pools on the system seems to fit into this category. You need to know the name of a pool to be able to create the instance, but it is also part of the Pool class. This is illustrated below.

```
1: public static String [] getList(iControl.Interfaces interfaces) throws Exception
2: {
3:     String [] pool_list = null;
4:
5:     if ( null != interfaces )
6:     {
7:         pool_list = interfaces.getLocalLBPool().get_list();
8:     }
9:
10:     return pool_list;
11: }
```

An Example Usage

Using the above class is very straightforward. First you must create the core iControl.Interfaces object defined in the iControl Library for Java. This must be initialized with the connection information (hostname, username, and password) to the BIG-IP. I've then included some code to create a new pool, query some of it's attributes, and then delete it.

```
1: // Initialize the iControl Interfaces Object
2: iControl.Interfaces interfaces = new iControl.Interfaces();
3: interfaces.initialize("bigip addr", "username", "password");
```

```
4:
5: // Call the static getList method to return the list of existing pools.
6: String [] poolList = iControl.Objects.LTM.Pool.getList(interfaces);
7:
8: // Create a new Pool object with the name dummyPool.
9: iControl.Objects.LTM.Pool pool = new iControl.Objects.LTM.Pool(interfaces, "dummyPool");
10:
11: // Create pool
12: iControl.CommonIPPortDefinition[] membersToAdd = new iControl.CommonIPPortDefinition[1];
13: membersToAdd[0] = new iControl.CommonIPPortDefinition("10.10.10.10", 80);
14: pool.create(iControl.LocalLBLBMethod.LB_METHOD_ROUND_ROBIN, membersToAdd);
15:
16: System.out.println("Pool " + poolName + " created...");
17:
18: // Query LB Method
19: iControl.LocalLBLBMethod lbMethod = pool.getLBMethod();
20: System.out.println("  + LB Method : " + lbMethod.toString());
21:
22:
23: // Query pool members
24: iControl.CommonIPPortDefinition [] membersInPool = pool.getMembers();
25: System.out.println("  + Members");
26: for(int i=0; i<membersInPool.length; i++)
27: {
28:   System.out.println("    [" + i + "] : " +
29:     membersInPool[i].getAddress() + ":" + membersInPool[i].getPort());
30: }
31:
32: // Query statistics
33: iControl.CommonStatistic [] poolStatistics = pool.getStatistics(null);
34: System.out.println("  + Statistics");
35: for(int i=0; i<poolStatistics.length; i++)
36: {
37:   iControl.CommonStatisticType type = poolStatistics[i].getType();
38:   iControl.CommonULong64 ul64 = poolStatistics[i].getValue();
39:   StatisticValue sv = new StatisticValue(ul64);
40:
41:   System.out.println("    [" + type + "] : " + sv.doubleValue());
42: }
43:
44: // Delete pool
45: pool.remove();
46: System.out.println("Pool " + poolName + " deleted...");
```

View The Source

The source code can be found in the iControl CodeShare under JavaObjectLtmPool.

Related Articles on DevCentral

- DevCentral Wiki: iControl Wiki Home
- Audio White Paper - F5 iControl
- DevCentral Wiki: Java Pool Member Control
- Custom BIG-IP Object MetaData With Data Groups > DevCentral > F5 ...
- Getting Started With iControl And Java – Setting Up Eclipse ...
- SSL Trust Provider for Java
- iControl 101 #22 - GTM Data Centers > DevCentral > F5 DevCentral ...
- How to instrument your Java EE applications for a virtualized ...

Technorati Tags: Java, iControl, Object, Joe Pruitt

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com