Java iControl Objects - LTM Pool Member

Joe Pruitt, 2010-29-09

This is the second article in the "Java iControl Objects" series in which I define a set of software objects that implement the iControl methods in various iControl interfaces. In my last article, I defined the LocalLB Pool class where you could define your application pools of servers and control attributes on any given pool. In that article I mentioned that I couldn't do the PoolMember class without having a Pool defined first. So, since the Pool is there, this article will focus on the PoolMember.

The PoolMember represents a server in your application pool. If you have a server farm of dozens of servers, those servers will be represented in the LTM configuration as a PoolMember within a given pool. Most iControl-based automation tasks surrounding application server management revolve around the PoolMember interface. Typical uses involve enabling and gracefully disabling servers from provisioning and querying statistics. With the LTM PoolMember Java Object, you will be able to create an instance of that PoolMember and do all that and everything else the iControl PoolMember interface provides.

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

The package I will be using is "iControl.Objects" with the various classes being in sub-packages for the related products (LTM, GTM, etc) and the class names will be the same as their corresponding iControl interfaces. For the PoolMember class, we will need the underlying iControl.Interfaces object defined in the iControl Library for Java. We will also need the containing Pool class as well as the address and port defining that pool member.

```
 1: package iControl.Objects.LTM;
 2:
 3: public class PoolMember
 4: {
 5:   private iControl.Interfaces _interfaces = null;
 6:
 7:   private Pool _pool = null;
 8:   private String _address = null;
 9:   private long _port = 0;
10:
11:   //--------------------------------------------------------------------
12:   // Member Accessors
13:   //--------------------------------------------------------------------
14:   public iControl.Interfaces getInterfaces() { return _interfaces; }
15:   public void setInterfaces(iControl.Interfaces interfaces) { _interfaces = interfaces; }
16:
17:   public String getAddress() { return _address;  }
18:   public void setAddress(String address) { _address = address; }
19:
20:   public Pool getPool() { return _pool;  }
21:   public void setPool(Pool pool) { _pool = pool; }
22:
23:   public long getPort() { return _port;  }
24:   public void setPort(long port) { _port = port; }
25:
26:   //--------------------------------------------------------------------
27:   // Constructors
28:   //--------------------------------------------------------------------
29:   public PoolMember(iControl.Interfaces interfaces, Pool pool, String address, long port)
30:   {
31:     _interfaces = interfaces;
32:     _pool = pool;
33:     _address = address;
34:     _port = port;
35:   }
```

Object Attribute Accessors

For each of the attribute methods in the iControl PoolMember interface, we will define associated methods in the class definition. The iControl API was meant for bulk method calls – meaning you can change attributes on as many like-objects at a time with a single method call. Since this doesn't make sense in an object model, I've masked all the arrays with scalar values to make it easier to code from the client side. There are still some arrays floating around, but they are only used when necessary within the single object instance.

The following section of code shows the wrappers around the iControl LocalLB.PoolMember.set_connection_limit() and get_connection_limit() methods.

```
// connection_limit
public void setConnectionLimit(long value) throws Exception
{
  validateMembers();

  String [] pool_names = { _pool.getName() };

  iControl.LocalLBPoolMemberMemberConnectionLimit memberConnLimit =
    new iControl.LocalLBPoolMemberMemberConnectionLimit();
  memberConnLimit.setMember(new iControl.CommonIPPortDefinition());
  memberConnLimit.getMember().setAddress(_address);
  memberConnLimit.getMember().setPort(_port);
  memberConnLimit.setConnection_limit(value);

  iControl.LocalLBPoolMemberMemberConnectionLimit [] memberConnLimitA =
    { memberConnLimit };
  iControl.LocalLBPoolMemberMemberConnectionLimit [][] memberConnLimitAofA =
    { memberConnLimitA };

  _interfaces.getLocalLBPoolMember().set_connection_limit(
    pool_names, memberConnLimitAofA);
}
public long getConnectionLimit() throws Exception
{
  validateMembers();
  long value = 0;

  String [] pool_names = { _pool.getName() };

  iControl.LocalLBPoolMemberMemberConnectionLimit [][] memberConnLimitAofA =
    _interfaces.getLocalLBPoolMember().get_connection_limit(pool_names);

  iControl.LocalLBPoolMemberMemberConnectionLimit [] memberConnLimitA =
    memberConnLimitAofA[0];

  for(int i=0; i<memberConnLimitA.length; i++)
  {
    iControl.CommonIPPortDefinition member = memberConnLimitA[i].getMember();
    if ( member.getAddress().equals(_address) && (member.getPort() == _port) )
    {
      value = memberConnLimitA[i].getConnection_limit();
      break;
    }
  }
  return value;
}
```

Action Methods

In addition to attribute accessor methods, there are a set of actions you can take on the objects. Enabling and disabling are probably the most popular methods as they aid in the automation of server provisioning. Questions about gracefully disabling servers come up all the time in the DevCentral forums, so I chose to include that logic in the disable method. This method will disable any new connections to the PoolMember, query the statistics and wait for the current connections to drop to zero, and then force down the PoolMember. This logic is illustrated below.

```
public void disable() throws Exception
{
  validateMembers();

  String [] pool_list = { _pool.getName() };

  //System.out.println("Disabling Session Enabled State...");

  iControl.LocalLBPoolMemberMemberSessionState memberSessionState =
    new iControl.LocalLBPoolMemberMemberSessionState();
  memberSessionState.setMember(new iControl.CommonIPPortDefinition(_address, _port));
  memberSessionState.setSession_state(iControl.CommonEnabledState.STATE_DISABLED);

  iControl.LocalLBPoolMemberMemberSessionState [] sessionStateA = { memberSessionState };
  iControl.LocalLBPoolMemberMemberSessionState [][] sessionStateAofA = { sessionStateA };

  _interfaces.getLocalLBPoolMember().set_session_enabled_state(pool_list, sessionStateAofA);

  //System.out.println("Waiting for current connections to drop to zero...");

  iControl.CommonIPPortDefinition memberDef =
    new iControl.CommonIPPortDefinition(_address, _port);

  iControl.CommonIPPortDefinition [] memberDefA = { memberDef };
  iControl.CommonIPPortDefinition [][] memberDefAofA = { memberDefA };

  long cur_connections = 1;

  while ( cur_connections > 0 )
  {
    iControl.LocalLBPoolMemberMemberStatistics [] memberStatsA =
```

```
      _interfaces.getLocalLBPoolMember().get_statistics(pool_list, memberDefAofA);

    iControl.LocalLBPoolMemberMemberStatistics memberStats = memberStatsA[0];

    iControl.LocalLBPoolMemberMemberStatisticEntry [] statsEntryA =
      memberStats.getStatistics();
    iControl.LocalLBPoolMemberMemberStatisticEntry  statsEntry = statsEntryA[0];

    iControl.CommonStatistic [] statsA = statsEntry.getStatistics();

    for(int i=0; i<statsA.length; i++)
    {
      iControl.CommonStatisticType type = statsA[i].getType();
      iControl.CommonULong64 value64 = statsA[i].getValue();

      if ( type == iControl.CommonStatisticType.STATISTIC_SERVER_SIDE_CURRENT_CONNECTIONS )
      {
        cur_connections = value64.getLow();
        //System.out.println("Current Connections: " + cur_connections);
      }
    }
    Thread.currentThread();
    Thread.sleep(1000);
  }

  //System.out.println("Disabling Monitor State...");

  iControl.LocalLBPoolMemberMemberMonitorState memberMonitorState =
    new iControl.LocalLBPoolMemberMemberMonitorState();
  memberMonitorState.setMember(new iControl.CommonIPPortDefinition(_address, _port));
  memberMonitorState.setMonitor_state(iControl.CommonEnabledState.STATE_DISABLED);

  iControl.LocalLBPoolMemberMemberMonitorState [] monStateA = { memberMonitorState };
  iControl.LocalLBPoolMemberMemberMonitorState [][] monStateAofA = { monStateA };

  _interfaces.getLocalLBPoolMember().set_monitor_state(pool_list, monStateAofA);
}
```

Static Methods

Static methods are ones that you can call on a class without having to create an instance of that class. For the PoolMember, I created an “exists” method that allows you to query whether a specified pool member exists in the running configuration. This is done by creating a temporary Pool and PoolMember class and then calling the “exists” action method in the PoolMember class.

```
public static boolean exists(iControl.Interfaces interfaces, String poolname, String address, long port) throws Exception
{
  Pool pool = new Pool(interfaces, poolname);
  PoolMember poolMember = new PoolMember(interfaces, pool, address, port);
  return poolMember.exists();
}
```

An Example Usage

Below is a sample routine that illustrates how to use the PoolMember object. An iControl.Interfaces object is created and inialized with the supplied input arguments. For this example, I hard coded in a few temporary variables for the pool name and member definition.

The code then instantiates a Pool and a PoolMember object. An empty pool is created, the member is added, the members are queried, and then the dynamic_ration and connection_limit accessor methods are tested. When all is done, the Pool is then removed from the configuration.

```
public void testPoolMember(String [] args)
{
  if ( args.length >= 2)
  {
    try
    {
      iControl.Interfaces interfaces = new iControl.Interfaces();
      interfaces.initialize(args[0], args[1], args[2]);

      String poolName = "dummyPool";
      String memberAddr = "99.99.99.99";
      long memberPort = 80;

      iControl.Objects.LTM.Pool pool = new iControl.Objects.LTM.Pool(interfaces, poolName);
      iControl.Objects.LTM.PoolMember poolMember = new iControl.Objects.LTM.PoolMember(interfaces, pool, memberAddr, memberPort);

      System.out.println("> Creating pool " + pool.getName());
      iControl.CommonIPPortDefinition[] members = null;
      pool.create(LocalLBLBMethod.LB_METHOD_ROUND_ROBIN, members);

      System.out.println("> Creating pool member " + pool.getName() + "/" + poolMember.getAddress() + ":" + poolMember.getPort() + "...");
      poolMember.create();

      members = pool.getMembers();
      System.out.println("Pool '" + pool.getName() + "' members:");
      for(int i=0; i<members.length; i++)
```

```
27:       {
28:         System.out.println("  " + members[i].getAddress() + ":" + members[i].getPort());
29:       }
30:
31:       long dynRatio = poolMember.getDynamicRatio();
32:       System.out.println("Dynamic Ratio -> " + dynRatio);
33:
34:       long connLimit = poolMember.getConnectionLimit();
35:       System.out.println("Connection Limit -> " + connLimit);
36:
37:       System.out.println("> Setting connection limit to 1000...");
38:       poolMember.setConnectionLimit(1000);
39:
40:       connLimit = poolMember.getConnectionLimit();
41:       System.out.println("Connection Limit -> " + connLimit);
42:
43:       iControl.LocalLBObjectStatus objStatus = poolMember.getObjectStatus();
44:       System.out.println("Object Status :");
45:       System.out.println("  Availability -> " + objStatus.getAvailability_status());
46:       System.out.println("  Enabled      -> " + objStatus.getEnabled_status());
47:       System.out.println("  Description  -> " + objStatus.getStatus_description());
48:
49:       iControl.CommonEnabledState sessionState = poolMember.getSessionEnabledState();
50:       System.out.println("Session Enabled State -> " + sessionState);
51:
52:       System.out.println("> Gracefully Disabling pool member...");
53:       poolMember.disable();
54:
55:       objStatus = poolMember.getObjectStatus();
56:       System.out.println("Object Status :");
57:       System.out.println("  Availability -> " + objStatus.getAvailability_status());
58:       System.out.println("  Enabled      -> " + objStatus.getEnabled_status());
59:       System.out.println("  Description  -> " + objStatus.getStatus_description());
60:
61:       System.out.println("> Enabling pool member...");
62:       poolMember.enable();
63:
64:       objStatus = poolMember.getObjectStatus();
65:       System.out.println("Object Status :");
66:       System.out.println("  Availability -> " + objStatus.getAvailability_status());
67:       System.out.println("  Enabled      -> " + objStatus.getEnabled_status());
68:       System.out.println("  Description  -> " + objStatus.getStatus_description());
69:
70:       System.out.println("Deleting pool member...");
71:       poolMember.remove();
72:
73:       System.out.println("Deleting pool...");
74:       pool.remove();
75:
76:     }
77:     catch(Exception ex)
78:     {
79:       ex.printStackTrace(System.out);
80:     }
81:   }
82: }
```

View The Source

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

Related Articles on DevCentral

- Getting Started With iControl And Java – Setting Up Eclipse ...
- DevCentral Wiki: Java System Info
- Java iControl Objects - LTM Pool > DevCentral > F5 DevCentral ...
- DevCentral Wiki: Java Pool Member Control
- DevCentral Wiki: Java Object Ltm Pool
- iRules 101 #05 - Selecting Pools, Pool Members, and Nodes ...
- F5 DevCentral > Community > Group Details - iControl Java Wrapper ...
- How to instrument your Java EE applications for a virtualized ...
- DevCentral Wiki: Power Shell Pool Member Availability
- Java Assembly - Marking down a Node - DevCentral - F5 DevCentral ...
- add pool member in a disabled state - DevCentral - F5 DevCentral ...

Technorati Tags: Java, PoolMember, 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