**M**[**anaging Camel routes with JMX APIs**](http://www.consulting-notes.com/2010/08/managing-camel-routes-with-jmx-apis.html)

here is a quick example of how to programmatically access Camel MBeans to monitor and manipulate routes...  
  
**first, get a connection to a JMX server (assumes localhost, port 1099, no auth)**  
*note, always cache the connection for subsequent requests (can cause memory utilization issues otherwise)*  
  
JMXServiceURL url = new JMXServiceURL("service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi");  
JMXConnector jmxc = JMXConnectorFactory.connect(url);  
MBeanServerConnection server = jmxc.getMBeanServerConnection();  
  
**use the following to iterate over all routes and retrieve statistics (state, exchanges, etc)...**  
  
ObjectName objName = new ObjectName("org.apache.camel:type=routes,\*");  
List<ObjectName> cacheList = new LinkedList(server.queryNames(objName, null));  
for (Iterator<ObjectName> iter = cacheList.iterator(); iter.hasNext();)  
{  
    objName = iter.next();  
    String keyProps = objName.getCanonicalKeyPropertyListString();  
    ObjectName objectInfoName = new ObjectName("org.apache.camel:" + keyProps);  
    String routeId = (String) server.getAttribute(objectInfoName, "RouteId");  
    String description = (String) server.getAttribute(objectInfoName, "Description");  
    String state = (String) server.getAttribute(objectInfoName, "State");  
    ...  
}  
  
**use the following to execute operations against a Camel route (stop,start, etc)**  
  
ObjectName objName = new ObjectName("org.apache.camel:type=routes,\*");  
List<ObjectName> cacheList = new LinkedList(server.queryNames(objName, null));  
for (Iterator<ObjectName> iter = cacheList.iterator(); iter.hasNext();)  
{  
    objName = iter.next();  
    String keyProps = objName.getCanonicalKeyPropertyListString();  
    if(keyProps.contains(routeID))  
    {  
        ObjectName objectRouteName = new ObjectName("org.apache.camel:" + keyProps);  
        Object[] params = {};  
        String[] sig = {};  
        server.invoke(objectRouteName, operationName, params, sig);  
        return;  
    }  
}  
  
**summary**  
  
These APIs can easily be used to build a web or command line based tool to support remote Camel management features. All of these features are available via the JMX console and Camel does provide a web console to support some management/monitoring tasks.