Monitoring Servoy with JMX

Java Management Extensions (JMX) provides a way to manage and monitor resources dynamically at runtime. These resources are represented by objects called Managed Beans (MBeans). The MBeans are registered to the MBean Server from where they can be accessed via any JMX client, like JConsole for example.

Servoy supports server monitoring (Clients,Solutions,Transactions and Locks) and datasource monitoring using JMX, that is monitoring the database servers /connections.

Server monitoring is exposed by the Application server as an MBean in the JMX domain name of com.servoy.server and then under 5 types: **(Server, Clients,Solutions,Transactions,Locks**). So the full name for the Clients bean is

```
com.servoy.server:type=Clients
```

But these are exposed also per web applicaton context :

```
com.servoy.server:type=Clients,context=ROOT
```

Where ROOT is the value of the context name the application/war is deployed at, so for a servoy applcation server its always ROOT, but for a WAR deployment it is the context name (very often the same name as the war file name)

Server bean gives information of the running server like Servoy and Repository version, it has the option to put the server in maintenance mode and get or set any property (the properties that are displayed on the admin page)

Clients bean gives you the NumberOfConnectedClients and the clients list itself, you can send message to all or to a specific client or shutdown a client.

Solutions bean gives you a list of solutions on the server including the type and information about releases, and the abiltiy to set a specific releas the a active or compact/flush/delete a solution.

Locks bean gives you a list of current locks and the abiltiy to release them (This should normally not be done, use with care!)

Transactions bean gives you a list of current transactions and the ability to commit or rollback them (This should normally not be done, use with care!)

Each enabled/active data server is exposed by the Application Server as an MBean in the JMX domain name `com.servoy.datasource`. For each data server the MBean is exposed with the configuration details of the server.

To enable connecting to the JMX bean of the ApplicationServer, a java option needs to be added to the startup commands:

For pure localhost access

```
-Dcom.sun.management.jmxremote
```

For remote access

```
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.port=8484
-Dcom.sun.management.jmxremote.rmi.port=8484
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=<IP_OR_HOSTNAME>
```

For remote access you need to specify 2 ports (that can be the same value) then that port needs to be opened in the firewall.

The java.rmi.server.hostname is also set in servoy (admin page/network settings) if rmi is used for the smart client. So if there 127.0.0.1 or localhost is defined,then the above system property is overwritten again with that value that servoy configures when the web application starts up. Make sure that both settings have the same value. (If smart client is used with a public dns, try to use that public dns also for jmx internally). The value can not be localhost/127. 0.0.1 in both settings because then the remote client can't reach the server.

There is also a GaugeMonitor per server, named `NumActiveMonitor_{serverName}`, to monitor active connections and get notifications when limits are passed. The limits can be modified via the attributes. Notifications are pushed to the JMX client.

The JMX interface can be used with any JMX client. Java comes with a simple JMX client, called jconsole, which is a simple GUI to configure the attributes and show the monitoring data. The jconsole is found in the `bin` folder of the installed Java JDK.

The tomcat manager application has a special JMX Proxy Servlet build in:

https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#JMX_Get_command

If you configured that one right (in tomcat-users.xml) then you can query through http the servoy stuff (and the system stuff java already provides)

you can get for the total number of clients:

manager/jmxproxy/?get=com.servoy.server:type=Clients,context=ROOT&att=NumberOfConnectedClients

or do a query for the full bean:

manager/jmxproxy/?qry=com.servoy.server:type=Clients,context=ROOT (or 'manager/jmxproxy/?qry=com.servoy.server:type=Clients,context=' for a root deployed war (so kind of empty string)

to explain the parts:

com.servoy.server == domain where all the server related stuff of servoy is under

type=Clients == the type of the bean that you want to query or get from, servoy supports currently Clients,Solutions,Transactions and Locks

context=ROOT == the context name, so if you deploy a war under the context "myapp" then it should be context=myapp (there can be multiply servoy "installations" in a same app server)

att=NumberOfConnectedClients == 'att' is a mandatory value for the "get" query above so that you specify to get a specific property of that bean.

if you want to see all the beans there are:

manager/jmxproxy/?qry

for example the basic thing that java itself already has can also be asked for:

manager/jmxproxy/?get=java.lang:type=Memory&att=HeapMemoryUsage

you can filter this to just get the used or max for example:

manager/jmxproxy/?get=java.lang:type=Memory&att=HeapMemoryUsage&key=used
or
manager/jmxproxy/?get=java.lang:type=Memory&att=HeapMemoryUsage&key=max