**Guidelines to implement JMX in project**

In our project JMX is exposed by two ways:

* Implemented JMX in Service by exposing MBeanExporter in GTNWebContext.xml file
* Implemented JMX in UI by creating MBean interface in GtnFramework package.
* Both service and UI JMX code is generic for all modules in Global Files.

1. **Steps to implement JMX in service:**

* Create a class to expose JMX (ex: GtnWebServiceJmx.java)
* Expose that class in GTNWebContext.xml and create an id for that class.
* Need to expose the MBeanExporter bean for that id which is created (ex: gtnWebServerJmx) as shown in Figure 1.

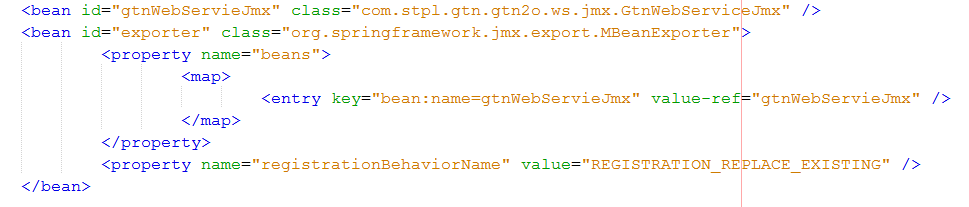
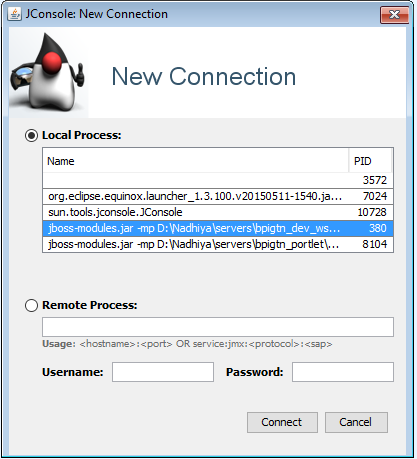
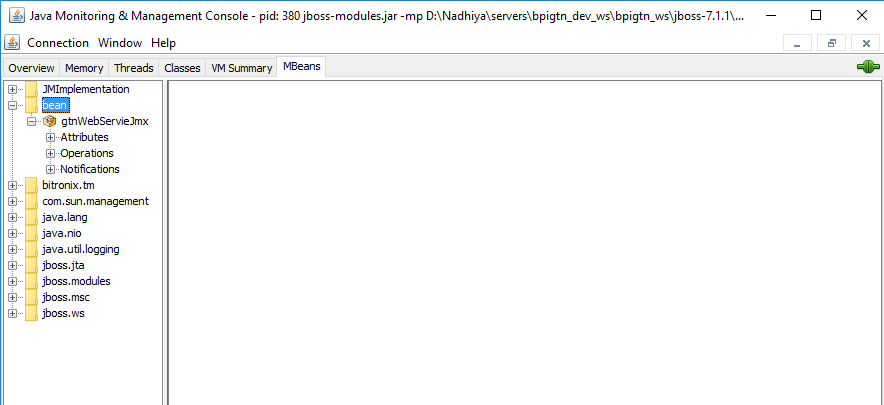


Figure 1

* Open the Jconsole by entering as “Jconsole “in command prompt
* Select Local Process as jboss (bpigtn\_dev\_ws) and click connect



* Click on “Insecure connection” and then select the MBeans tab in Jconsole
* In MBeans tab, user can find the bean which is exposed in GTNWebContext.xml



* Under bean attributes and operations will display the methods and variables which is implemented in the GtnWebServiceJmx.java class.

**For example: Get average time taken for each service**

* Create a filter class called GtnWebServiceJmxFilter to get the start and end time for each service as shown in the Figure 2.

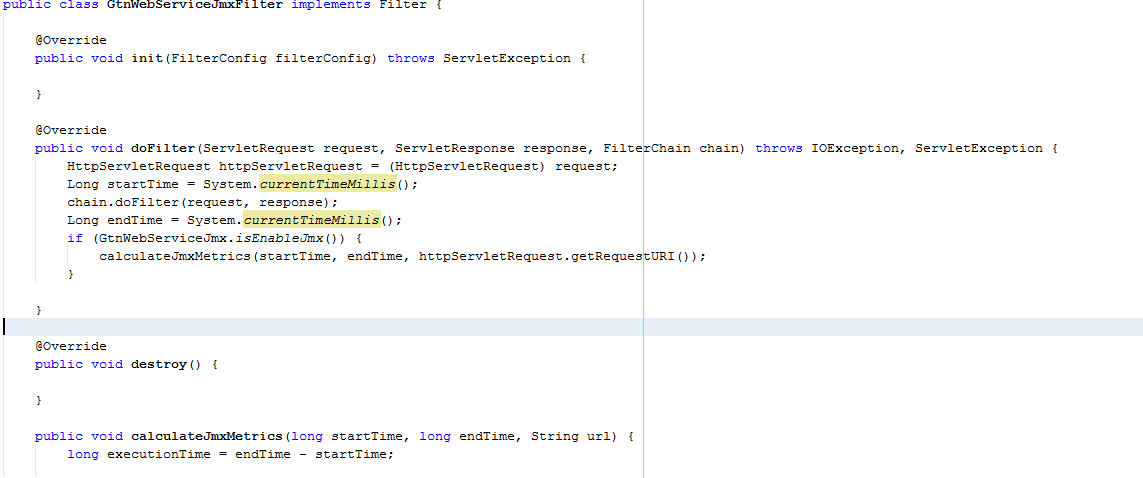


Figure 2

* Then implement method in GtnWebServiceJmx.java according to the requirement.

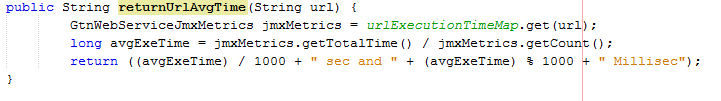
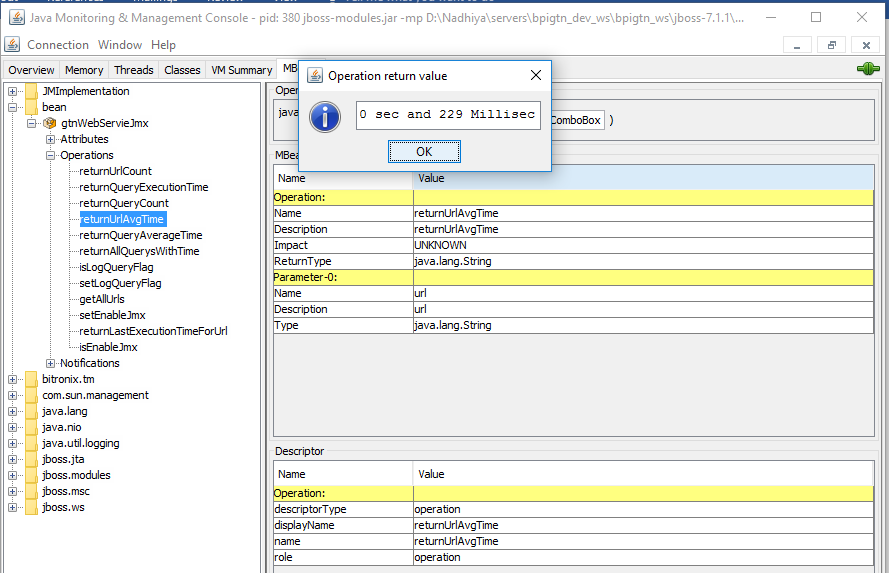
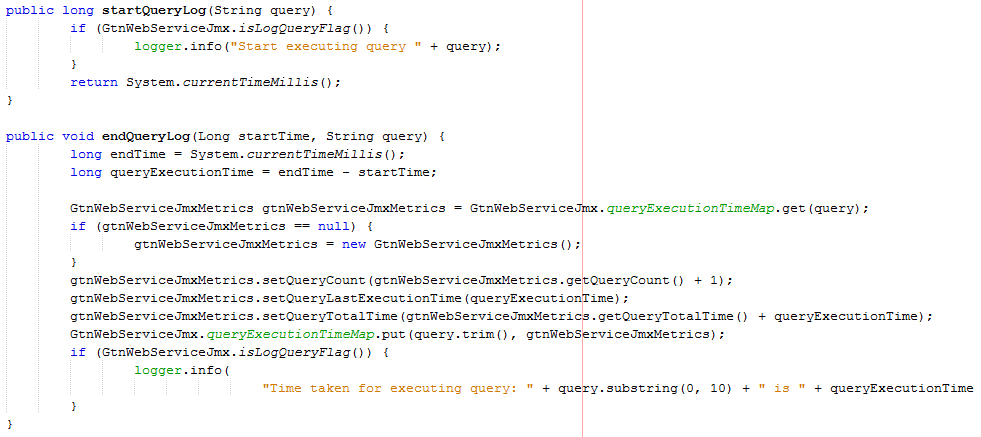


Figure 3

* In this above Figure 3, getting the total time and total count how many times that particular url has been called.
* Dividing total time/count to get the average time taken for each url
* Build and deploy the service war and then open the Jconsole to check the results
* Enter the url as input in Jconsole and get the average time taken for tat url as shown below



* Geting query execution time for all modules from GtnWSLogger.java. Find the code snippet below



**Implemented methods for service and its purpose:**

* returnUrlcount() : it will return the count for each service, how many times that particular service has been called.
* returnQueryExecutionTime(): It will return the query execution time for each query.
* returnQueryCount(); It will return the count for each query, how many times that query has been executed.
* returnUrlAvgTime(): It will return the average time for each url
* returnQueryAverageTime(): it will return the average time for each query
* setQueryLogFlag() : Set the flag, whether the query should be printed in backend or not.
* setEnableJmx(): Set the flag, whether the JMX should be enable or not
* returnLastExecutionTimeForUrl(): It will return the last execution time for that particular url

1. **Steps to implement JMX in UI:**

* Create a MBean interface (ex: GtnUIFrameworkJmxMBean.java)
* Create a class to implement MBean interface for JMX. Class name should be same as interface name (ex: GtnUIFrameworkJmx.java)
* Need to register Mbean GtnUIFrameworkEngine.java class(Because this s generic implementation for all modules) as given in the Figure 4.



Figure 4

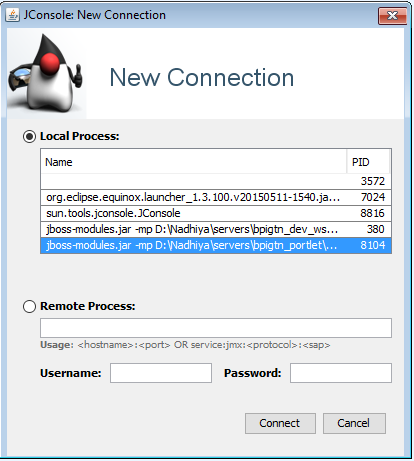
* In UI, we can get the vaadin components count that has been created while the screen is loading and count for each component (ex: TextField count,Combobox Count, Etc)

**Implemented methods in GtnUIFrameworkJmx.java for UI and its purpose:**

* getVaadinComponentsCount(): It will return the vaadin components count
* returnTextFieldCount(): It will return the Textfield components count
* returnComboBoxCount(): It will return the comboBox components count
* returnPopUpTextFieldCount(): It will return the popup TextField components count
* returnPopDateFieldCount(): It will return the popupDateField components count
* returnExtPagedTableCount(): It will return the ExtPagedTable components count
* returnButtonCount(): It will return the buttons count
* returnCustomMenuBarCount() : It will return the custom menu bar count
* returnExtFilterTableCount(): It will return the ExtFielter table compenents count
* returnTextAreaCount(): It will return the TextArea components count
* returnOptionGroupCount(): It will return the OptionGroup Count
* returnFreezePagedTreeTableCount(): It will return the FreezePagedTree table count
* objectListCount(): It will return the objects count that has been created while the screen s loading
* clearMemory(): GC will be performed
* clearMap(): It will clear all maps
* isEnableJMX(): It will enable/disable jmx for UI
* returnFieldFactoryTextFieldCount(): It will return the field factory textField count
* returnFieldFactoryComboBoxCount(): It will return the field facory comboBox count
* returnFieldFactoryCustomTextFieldCount(): It will return the field factory custom TextField count
* returnFieldFactoryPopupDateFieldCount(): It will return the field factory popup date field count
* returnFieldFactoryButtonCount(): It will return the field factory buttons count
* returnFieldFactoryTextAreaCount(): It will return the field factory text area count

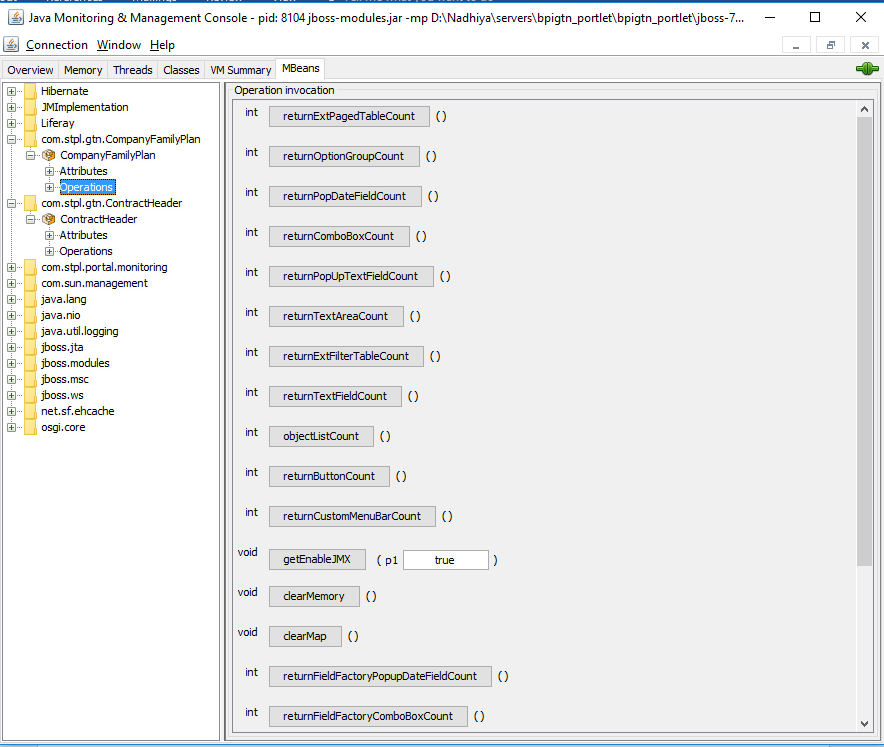
Open the jconsole by entering Jconsole in run command

Select local process as jboss(bpigtn\_portlet) and click connect



Click on insecure connection and select MBean tab, For each and every module one beans will be created as shown in the below snippet.

Click on operations in any module and check the methods are available which is created in MBeans interface.

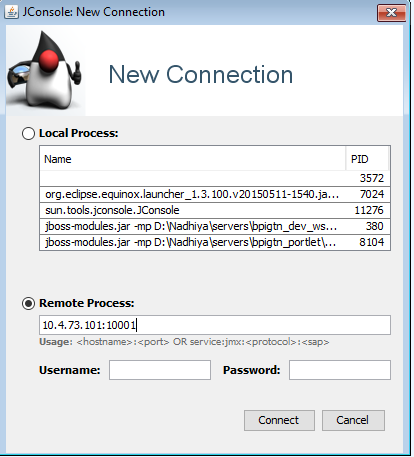


1. **Remote JMX configuration for windows/Linux:**

* Enter as Jconsole in command prompt and press enter, Jconsole will be opened.
* Add the following VM arguments for windows in standalone.config.bat file which is present in this path "jboss-7.1.1/bin"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote=true"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote.authenticate=false"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote.ssl=false"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote.port=<PORT>"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote.host=<IP\_ADDRESS>"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Dcom.sun.management.jmxremote.local.only=false"
  + set "JAVA\_OPTS=%JAVA\_OPTS% - Djava.util.logging.manager=org.jboss.logmanager.LogManager -Xbootclasspath/p:../modules/org/jboss/logmanager/main/jboss-logmanager-1.2.2.GA.jar -Xbootclasspath/p:../modules/org/jboss/logmanager/log4j/main/jboss-logmanager-log4j-1.0.0.GA.jar -Xbootclasspath/p:../modules/org/apache/log4j/main/log4j-1.2.16.jar"
  + set "JAVA\_OPTS=%JAVA\_OPTS% -Djboss.modules.system.pkgs=org.jboss.logmanager"
* Need to add the below VM arguments for Linux
* Add the following VM arguments in standalone.config file which is present in this path "jboss-7.1.1/bin
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote=true"
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote.authenticate=false"
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote.ssl=false"
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote.port=<PORT>"
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote.host=<IP\_ADDRESS>"
  + JAVA\_OPTS="$JAVA\_OPTS -Dcom.sun.management.jmxremote.local.only=false"
  + JAVA\_OPTS="$JAVA\_OPTS -Djava.util.logging.manager=org.jboss.logmanager.LogManager -Xbootclasspath/p:$JBOSS\_HOME/modules/org/jboss/logmanager/main/jboss-logmanager-1.2.2.GA.jar - Xbootclasspath/p:$JBOSS\_HOME/modules/org/jboss/logmanager/log4j/main/jboss-logmanager-log4j-1.0.0.GA.jar -Xbootclasspath/p:$JBOSS\_HOME/modules/org/apache/log4j/main/log4j-1.2.16.jar"
  + JAVA\_OPTS="$JAVA\_OPTS -Djboss.modules.system.pkgs=org.jboss.logmanager"
* Place this "jboss-logmanager-1.2.2.GA.jar" in the below path
* /jboss-7.1.1/modules/org/jboss/logmanager/main/jboss-logmanager-.2.2.GA.jar
* Place this "jboss-logmanager-log4j-1.0.0.GA.jar" in the below path
* /jboss-7.1.1/modules/org/jboss/logmanager/log4j/main/jboss-logmanager-log4j-1.0.0.GA.jar
* Place this "log4j-1.2.16.jar in below path"
* /jboss-7.1.1/modules/org/apache/log4j/main/log4j-1.2.16.jar
* Restart the server
* Enter as Jconsole in command prompt and press enter, Jconsole will be opened
* Select remote process option in Jconsole
* Enter <Hostename>:<port> (enter the hostname and port which is given in the standalone.config.bat file).

**For Example:**

* In Remote process, entering system IP: Host which is mentioned in server.
* Port will be different for each server so user can connect to service or portlet server with port (both the servers should run by bind mode).
* Then click on connect.



**Note:**

* Replace $JBOSS\_HOME with jboss directory
* JMX <PORT> for portlet server is 21556
* JMX <PORT> for service server is 21555
* <IP\_ADDRESS> should be the server ip address
* Jars for JMX are committed in GTN framework project.
* Remote Jconsole is working fine for windows but it’s not working for Linux (still in process)