# JMX

## Jetty JMX Integration

The jetty JMX integration is based on the ObjectMBean implementation of DynamicMBean. This implementation allows an arbitrary POJO to be wrapped in an MBean and for meta data to be provided by properties files

The creation of MBeans is coordinated by the MBeanContainer implementation of the Container.Listener interface. The Jetty Server and it's components use a Container to maintain a containment tree of components and to support notification of changes to that tree. The MBeanContainer class listens for Container events and creates and destroys MBeans as required to wrap all Jetty components.

### Using Jetty MBeans

The simplest way to use Jetty MBeans is to use the JVM supplied jconsole utility. If you use mx4j, then an internal HTTP agent can also be used (see jetty-jmx.xml config file comments).

See Running jetty with jconsole for instructions on how to configure jmx for use with Sun's jconsole.

### Jetty Standalone

The MBeanContainer instance can be configured for a Jetty server by the jetty-jmx.xml configuration file. This can be run with the standard configuration file as follows:

```
java -jar start.jar etc/jetty-jmx.xml etc/jetty.xml
```

> Jetty 7 requires `-DOPTIONS=jmx` to be used when starting up Jetty. That is, instead of the above, use this:
> `java -DOPTIONS=jmx -jar start.jar etc/jetty-jmx.xml etc/jetty.xml`

### Jetty Maven Plugin

If you are using the jetty maven plugin you should copy the etc/jetty-jmx.xml file into your webapp project somewhere, such as src/etc, then add a <jettyConfig> element to the plugin <configuration>:

```
<plugin>
        <groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>

<version>${project.version}</version>
        <configuration>

<scanIntervalSeconds>10</scanIntervalSeconds
>

<jettyConfig>src/etc/jetty-jmx.xml</jettyCon
fig>
          <userRealms>
            <userRealm
implementation="org.mortbay.jetty.security.H
ashUserRealm">
              <name>Test Realm</name>

<config>../../etc/realm.properties</config>
            </userRealm>
          </userRealms>
        </configuration>
</plugin>
```

If you are using mx4j with the jetty-plugin then you should add the mx4j-tools dependecy to the jetty plug-in declaration. It goes something like:

```
<plugin>
  <groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>
  <configuration>...</configuration>
  <dependencies>
    <dependency>
      <groupId>mx4j</groupId>
      <artifactId>mx4j-tools</artifactId>
      <version>3.0.1</version>
    </dependency>
  </dependencies>
</plugin>
```

### Running on Java 1.4 and below (with MX4J)

To run on Java 1.4 and below, mx4j library is used to provide the MBean server. The commented out section of jetty-jmx.xml needs to be added to create the mbean server and to register the JMX RMI connector:

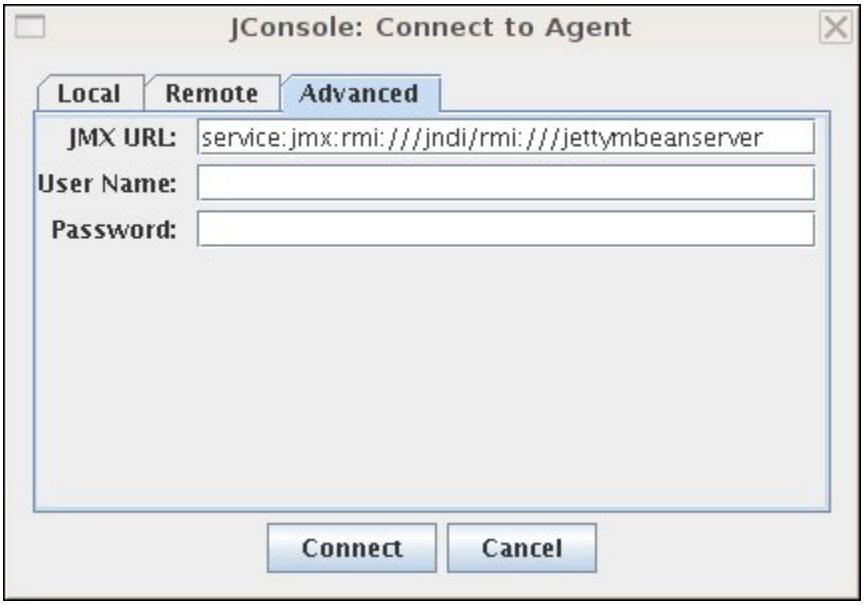
```
<Call id="MBeanServer"
class="javax.management.MBeanServerFactory"
name="createMBeanServer"/>

<Call id="jmxConnector"
class="javax.management.remote.JMXConnectorS
erverFactory" name="newJMXConnectorServer">
  <Arg>
    <New
class="javax.management.remote.JMXServiceURL
">

<Arg>service:jmx:rmi:///jndi/rmi:///jettymbe
anserver</Arg>
    </New>
  </Arg>
  <Arg/>
  <Arg><Ref id="MBeanServer"/></Arg>
  <Call name="start"/>
</Call>
```

Then, you will need to ensure that you have the `rmiregistry` running. The start sequence becomes:

```
rmiregistry &
java -Dcom.sun.management.jmxremote -jar
start.jar etc/jetty-jmx.xml etc/jetty.xml
[config files]
jconsole &
```

You then need to configure jconsole to find this remote connector using the `Advanced` panel and enter the

connector url like so:




Click the `Connect` button and you will be able to manipulate the Jetty mbeans.