Der JmxApdator

Überblick über das JmxApdator-Plugin

by Peter Roßbach

NOTICE:

Der JmxApdator ermöglicht den MBean-Zugriff mit einem Browser. Im Standard ist vorgesehen die Ausgabe in HTML und XML zu ermöglichen.

1. Idee des JmxApdator

Die gesamte Administration der Centaurus-Platform soll extern erfolgen können. Der zugrundeliegende Tomcat 5 basiert auf dem JMX API und seinen MBeanServer kann auch Remote verfügbar gemacht werden. Die Tomcat Admin-Anwendung beruht auf einem entsprechenden MBean API.

Die Admin-Anwendung hat leider einige Einschänkungen in der Pflege und Manipulation der MBeans. Leider arbeitet die Anwendung nicht generisch und ist auch nicht in der Lage eigene Erweiterungen zu integrieren. Der Standard Http-Adaptor der sich in der Datei *jk2.properties* aktivieren läßt, ist vollständig ungesichert und damit problematisch im Interneteinsatz.

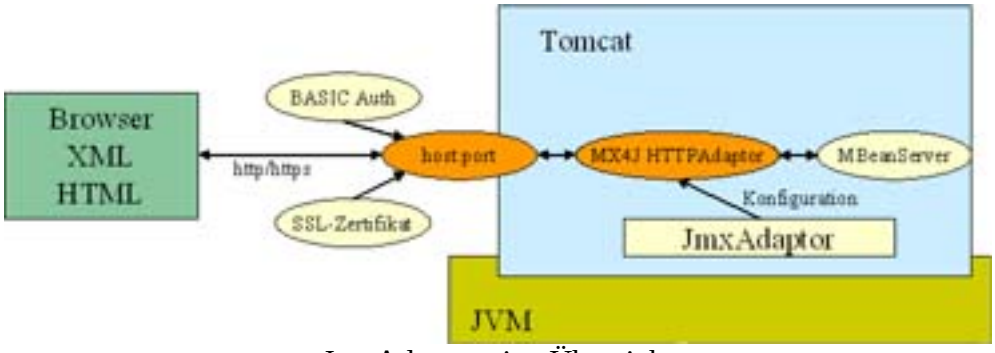
2. Die Lösung ist ein konfiguierbarer MX4J Http Apdator

Das von der Centaurus-Platform genutze MX4J-Projekt bringt einen Http-Adaptor und einen Remote-Connector auf der Basis des JSR 160 mit. Wir haben uns erstmal zur Unterstützung des Http-Adators entschlossen. Mit diesem Adaptor kommt man aktuell durch die meisten Firewalls und kann somit eine erweitertet Remote-Administration der Centaurus-Platform ermöglichen. Folgende Eigenschaften sind mit dem JmxAdaptor realisiert:

HTTP-Zugang	Alle Informationen sind durch einem HTTP-Port zugänglich.

Verschlüssug mit SSL	Der Adaptor kann auch nur mit Https erreichbar gemacht werden. Eine solche sichere Verbindung ist sehr ratsam, wenn Sie auf den Server über ein ungesichertes Netz zugreifen. Immerhin können Sie mit JMX auf alle Komponenten des laufenden Server aktiv Einfluß nehmen.
Zugangsbeschränkung	Der Http-Zugang kann auf bestimmte Nutzer beschränkt werden.
Flexible Ausgabesteuerung	Die Ausgabe kann direkt in XML, oder HTML erfolgenden. Die HTML-Ausgabe kann durch eigene XSL-Stylesheets gesteuert werden.

Table 1:



JmxAdaptor eine Übersicht

Klicken Sie auf das Bild für die Vollansicht

Note:

Warnung: Geben Sie nur wirklich vertrauenswollen Personen Zugang zu diesem Adpator. Mit dem JmxApdator können Sie vertrauliche Informationen des gesamten Servers einsehen und aktiv ändern.

Note:

Einschränkung: Die Centaurus-Platform nutzt das Release MX4J 2.0.1. Mit dieser MX4J-Version stehen die RMI- und IIOP-Adaptoren nicht mehr direkt zur Verfügung. Im Source sind die Adaptoren noch enthalten. Wir werden versuchen in einem der nächsten Release ein JSR 160 konformen JMX-Adaptor bereitzustellen. Dieser Adaptor bietet sehr umfangreiche Möglichkeiten zur Benutzerkontrolle und Zugangsbeschränkung. Wir hoffen, das wir dann mit dieser Remote Client-Lösung eine Integration in bestehende JMX-Consolen ermöglichen (MC4J).

Note:

Nutzung MX4J 1.1.1: Die Nutzung des MX4J 1.1.1 ist theoretisch auch möglich. Hierfür müssen Sie dann das *mx4j.jar* und *mx4j-tools.jar* in den System-ClassPath in der Datei *conf/wrapper.xml* eingetragen werden. Nicht vergessen die MX4J 2.0.1 *jmx.jar*, *mx4j-tools.jar* und *mx4j-remote.jar* aus dem Pfad zu entfernen und im Listener die entsprechenden MX4J 1.1.1-Adaptor, -Processor und SSL Factory-Klassen umzusetzen.

Note:

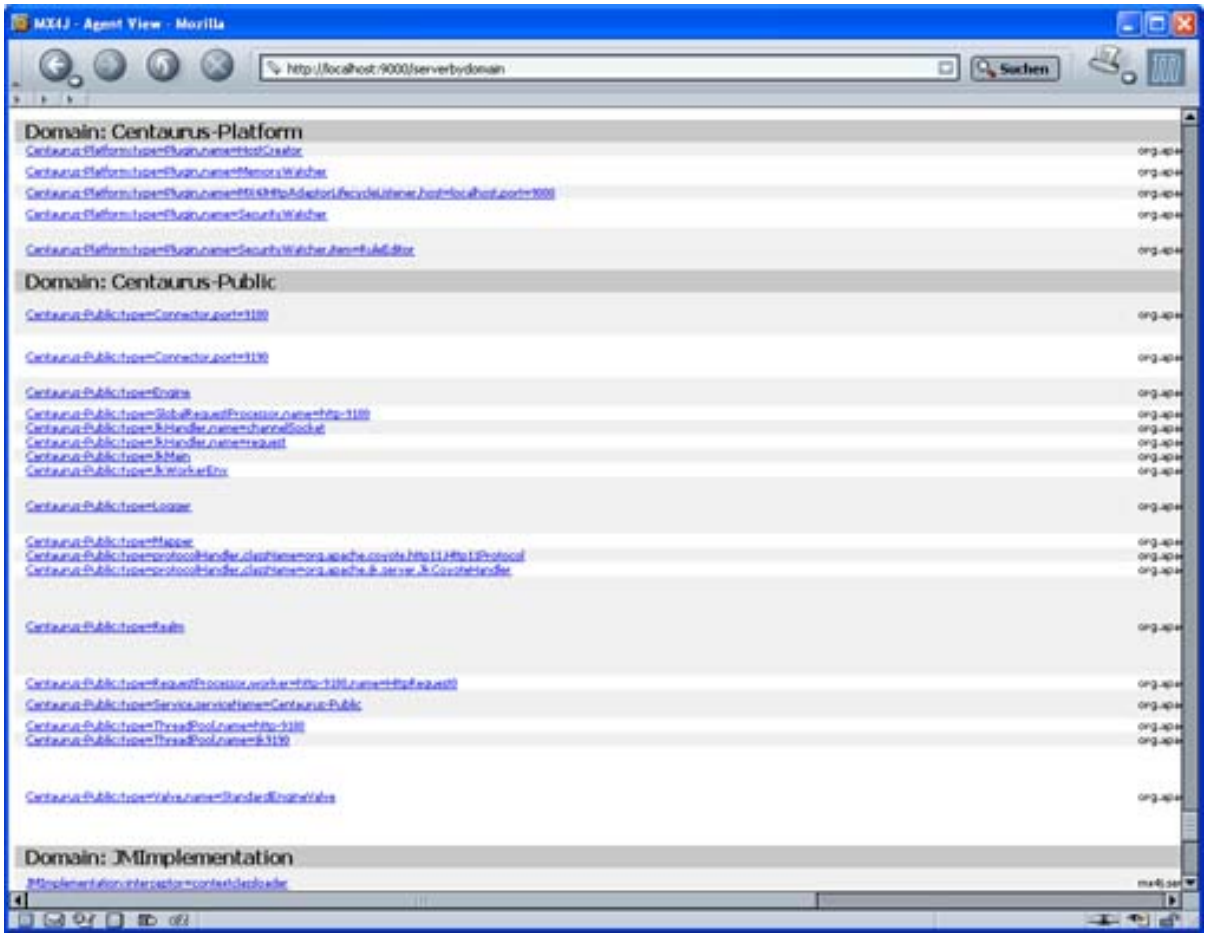
Eine Zugangsbeschränkung auf bestimmte Operationen oder bestimmte MBeans wird leider nicht von MX4J Http-Adaptor angeboten. Theoretisch könnten Sie die XML-Ausgabe durch entsprechende XSL-Strylesheets natürlich beschränken. Für die Beschränkung der Operation müssen Sie auf einen entsprechend konfiguierten Proxy zurückgreifen und auf den wirklichen JmxAdaptor umleiten.

3. Funktionsweise

Wir nutzen die JMX-Bibliotheken des MX4J-Projekts. Der JmxAdaptor hat sehr viele Parameter, die Sie für Ihre Bedürfnisse anpassen können. Es ist sogar möglich, gleichzeitig verschiedene JmxAdaptoren in einem Server mit unterschiedlichen Konfigurationen zu betreiben. Die Integration in den Tomcat erfolgt über die Datei `${centaurus.base}/conf/server.xml`.
Suchen Sie dort den entsprechenden Listener-Eintrag:

```
<Listener
className="de.centaurus.platform.plugins.jmxadaptor.MX4JHttpAdaptorLifecy
... />
```

Nach der Standardinstallation steht der JmxAdaptor unter dem Port *9000* bereit. Als Autorisierung gilt Ihr Admin-Account und die HTML-Ausgabe ist aktiv.



JmxAdaptor mit Html GUI

Klicken Sie auf das Bild für die Vollansicht

Für die Konfiguration einer SSL-Verbindung generieren Sie einen eigenen Java-Keystore oder importieren bitte ein offizielles Zertifikat. Ein Testzertifikat können Sie mit dem Skript *$centaurus.base/scripts/jmxadaptor/genkey.xml* erstellen. Die Parameterisierung kann direkt im Skript oder besser in einer eigenen Datei *build.properties* erfolgen.

scripts/jmxadaptor/build.properties

```
mx4j.storepass=changeit
mx4j.keypass=changeit
mx4j.keystore=mx4j.keystore
mx4j.host=localhost
mx4j.unit=Software Deveplopment
mx4j.org=Centaurus-Platform
```

```
mx4j.city=Bochum
mx4j.code=DE
```

Leider muss sich das Zertifikat für den MX4J Http-Adaptor im System-ClassPath befinden. Mit dem Aufruf des folgenden Ant-Skript wird Ihr Testzertifikat generiert und in ein Java Archiv (JAR) gepackt: `ant -f genkey.xml serverkey.jar` . Die Konfiguration des System-ClassPath erfolgt, nach der Kopie des Testzertifikat-Jars nach *$centaurus.base/bin*, in der Datei *$centaurus.base/conf/wrapper.xml*.

```
<property name="wrapper.java.classpath.7"
    value="${centaurus.base}/bin/mx4j.keystore.jar"></property>
```

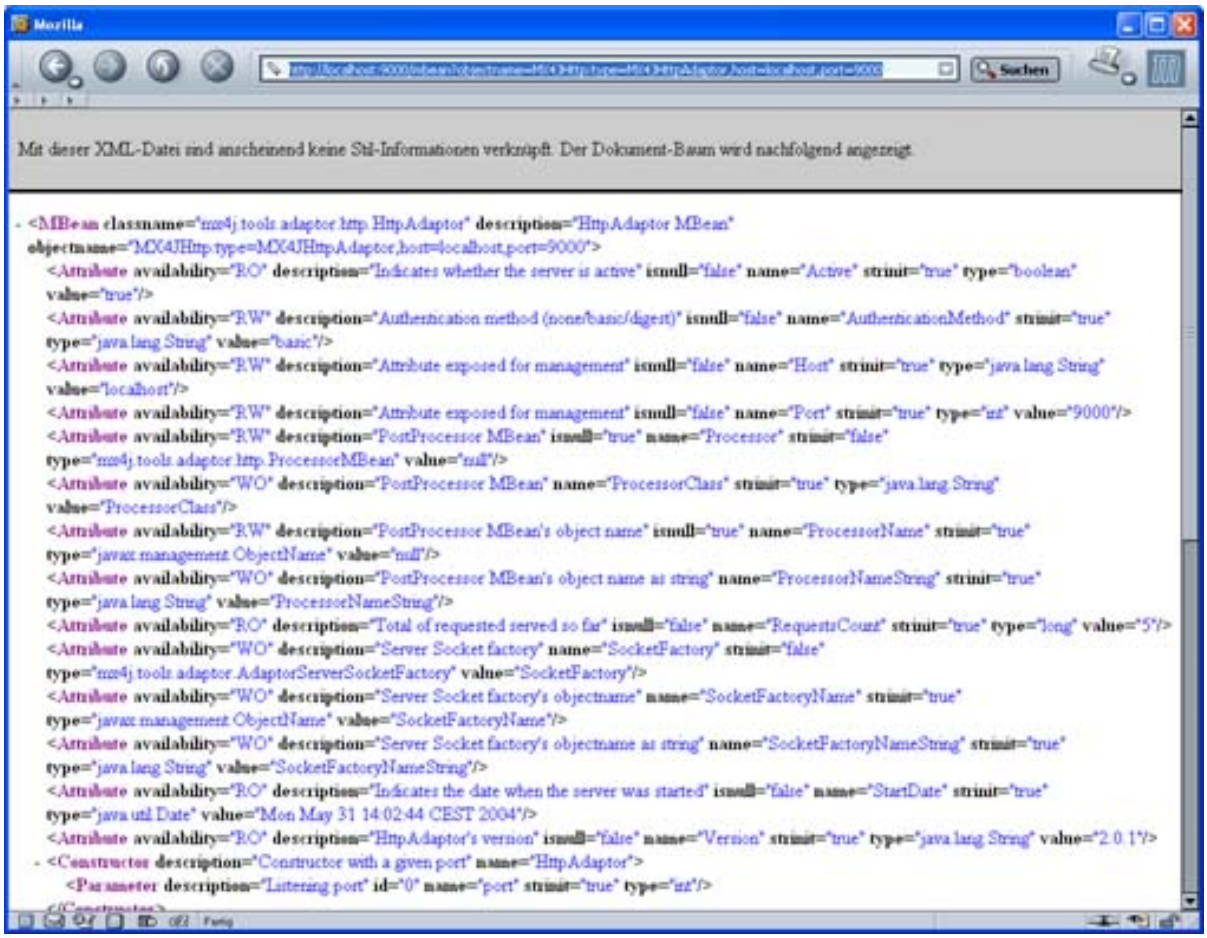
Als letzten Schritt erfolgt nun die Konfiguration der SSL-Parameter im Listener:

```
<Listener
    className="de.centaurus.platform.plugins.jmxadaptor.MX4JHttpAdaptorLifecycleListene
    enabled="true"
    mx4jDomain="MX4JHttp"
    httpPort="9000"
    httpHost="localhost"
    xslProcessor="true"
    xslPathInJar="mx4j/tools/adaptor/http/xsl"
    adaptorClassName="de.centaurus.platform.plugins.jmxadaptor.CentaurusHttpAdaptor"
    processorClassName="mx4j.tools.adaptor.http.XSLTProcessor"
    user="manager"
    password="tomcat"
    realmMode="true"
    realmMBeanName="Catalina:type=Realm,host=localhost"
    realmAllowedRole="admin"
    ssl="true"
    keyStoreType="JKS"
    keyStoreName="mx4j.keystore"
    keyStorePassword="changeit"
    keyManagerAlgorithm="SunX509"
    keyManagerPassword="changeit"
    sslProtocol="TLS"
    factoryClassName="mx4j.tools.adaptor.ssl.SSLAdaptorServerSocketFactory"
   >
 </Listener>
```

Die XML-Ausgabe wird folgendermaßen aktiviert:

```
<Listener
    className="de.centaurus.platform.plugins.jmxadaptor.MX4JHttpAdaptorLifecycleListen
    ...
    xslProcessor="false"
    processorClassName="mx4j.tools.adaptor.http.DefaultProcessor"
    ...
   >
```

```
</Listener>
```



JmxAdaptor mit XML GUI

Klicken Sie auf das Bild für die Vollansicht

> **Note:**
>
> **Änderung** am JmxAdaptor sollten Sie nur durchführen, wenn der Server gestoppet ist. Ein kleiner Tippfehler in den Dateien *server.xml* oder *wrapper.xml* können zu ziemlich merkwürdigen Situationen führen, aus der Sie sich nur noch durch *kill -9* oder dem Restart des gesamten Systems befreien können.

4. Parameter des JMXAdaptor-Listeners

Folgende Parameter können Sie konfigurieren:

Allgemeine Parameter

enabled	Aktivierung des Apdators (**Standard: true**)
mx4jDomain	JMX-Domain für diesen Listener (**Standard: MX4JHttp**)
httpPort	Port unter der Adaptor erreichbar ist (**Standard: 9000**)
httpHost	Der Adaptor kann auch auf bestimme Interfaces beschränkt werden.(**Standard: localhost**). Mit einer Angabe von *0.0.0.0* lauscht der JmxAdaptor auf allen Interfaces des jeweiligen Rechner und ist von überall erreichbar!
adaptorClassName	Implementierungsklasse die für den HTTP-Zugang verwendet werden soll. **mx4j.tools.adaptor.http.HttpAdaptor** In MX4j 1.1.x ist der ClassName *mx4j.adaptor.http.HttpAdaptor*. Eigene Erweiterungen mit speziellen Befehlshandlern oder eigenen Methoden zur Zugangsbeschränkungen sind möglich.

Table 1:

Parameter für Filterung der Ausgabe

xslProcessor	Ist Xsl Processor gewählt und einer der beiden Parameter *xslPathInJar* oder *xslPath* gesetzt, ist die eigenständige Ausgabeaufbereitung des XML-Stroms möglich. (Default **true**).
processorClassName	Implementierungsklasse für die XSLT-Filterung **mx4j.tools.adaptor.http.XSLTProcessor**. In MX4j 1.1.x ist der ClassName *mx4j.adaptor.http.XSLTProcessor* Für eine reine XML-Ausgabe muss der Processor auf *mx4j.adaptor.http.DefaultProcessor* gesetzt werden.
xslPathInJar	Pfad der XSL-Skripte in einem JAR, das im System-ClassPath (*wrapper.xml*) vor dem *mx4j-tools.jar* eingetragen sein sollte. Die Voreinstellung ist **mx4j/tools/adaptor/http/xsl**.
xslPath	Pfad im Dateisystem zu den XSL-Skripten, ist entweder relative zu *$centaurus.base* oder besser eine absolute Angabe, zu wählen (Beispiel: **mx4j/tools/adaptor/http/xsl**).

Table 2:

Autorisierungsinformationen

user	Nutzer für den Http Adaptor-Zugang (Beispiel **manager**).
password	Geheimwort für den Http Adaptor-Zugang (Beispiel **tomcat**)
authResource	Ist authResource gesetzt, wird eine Filterung der Admin-UserDatabase *$centaurus.base/conf/users/centaurus-users.xml* durchgeführt. Die Filterung erfolgt auf der Basis des Filters in *conf/xsl/jmxadaptor.xsl*. Es werden alle Accounts die zur Rolle *admin* gehören für die Nutzung des MX4J Http-Adaptors herangezogen. Die Datei wird unter *$centaurus.base/temp/$authResource* gespeichert.
realmMode	Hiermit wird ein Tomcat-Realm als Autorisierungsdatenbank genutzt. Aktuell ist dies mit der Wahl der Adaptorklasse *adaptorClassName=de.centaurus.platform.plugins.jmxadaptor.Cer* möglich.
realmMBeanName	Realm MBean Name (Beispiel **Catalina:type=Realm,host=localhost**)
realmAllowedRole	Rolle der Nutzer im Realm den der Zugang via JMX gestattet wird. **admin**)

SSL-Parameter

ssl	Https-Zugang nutzen (Default **false**)
keyStoreType	Store-Type für den Http Adaptor-Zugang (Default **JKS**).
keyStoreName	Die Schlüsseldatei muss sich als Ressource im System-ClassPath befinden (Default **mx4j.keystore**).
keyStorePassword	Geheimwort für den Schlüssel im Key-Store
keyManagerAlgorithm	Zertifikatsalogrithmus (Default **SunX509**).
keyManagerPassword	Geheimwort für die Nutzung des Stores
sslProtocol	SSL-Protokoll (Default **TLS**).

factoryClassName	SSL Socket-Factory (Default **mx4j.tools.adaptor.ssl.SSLAdaptorServerSocketFactory**).

5. MX4J-Befehle des Http-Adaptors

Die exakte Dokumentation aller MX4J-Befehle mit weiteren Beispielen finden Sie aktuell in der MX4-HttpAdaptor- Dokumentation.

5.1. JMX-Informationen erfragen

5.1.1. Ausgabe sortiert nach der JMX-Domain

Syntax des Befehls
http://host:port/serverbydomain

- Ausgabe aller MBeans eines Types:
  http://localhost:9000/serverbydomain?instanceof=org.apache.catalina.mbeans.UserMBean
- Ausgabe aller MBeans aller Domains:
  http://localhost:9000/serverbydomain?querynames=*:*
- Ausgabe aller MBeans der Centaurus-Public Domain:
  http://localhost:9000/serverbydomain?querynames=Centaurus-Public:*
- Ausgabe aller MBeans der Plugins:
  http://localhost:9000/serverbydomain?querynames=Centaurus-Platform:*

> **Note:**
> Die meisten MBeans basieren auf dem generischen Typ *org.apache.commons.modeler.BaseModelMBean*.

5.1.2. Ausgabe aller MBeans eines Types

Syntax des Befehls
http://host:port/server

- Ausgabe des Http-Adaptors:
  http://localhost:9000/server?instanceof=mx4j.tools.adaptor.http.HttpAdaptor

5.1.3. Ausgabe eines MBeans mit allen Informationen

Syntax des Befehls
http://host:port/mbean?
objectname=XXX

- Ausgabe des UserDatabase-MBeans:
  http://localhost:9000/mbean?

objectname=Catalina:type=Resource,resourcetype=Global,
class=org.apache.catalina.UserDatabase,name=UserDatabase

- Ausgabe des MX4J HttpAdaptor-MBeans:
  http://localhost:9000/mbean?
  objectname=MX4JHttp:type=MX4JHttpAdaptor,host=localhost,port=9000

5.1.4. Ausgabe eines bestimmte MBean-Attributes

Syntax des Befehls
http://host:port/getattribute?objectname=XXX&attribute=XXX&format=ZZZ

- Ausgabe des UserDatabase MBean-Attribute *auth*:
  http://localhost:9000/getattribute?
  objectname=Catalina:type=Resource,resourcetype=Global,
  class=org.apache.catalina.UserDatabase,name=UserDatabase&attribute=auth

Ausgabe von Collections können mit dem *format*-Parameter gesteuert werden.
`format=array|collection|map`

5.1.5. Ping für den Server

Syntax des Befehls
http://host:port/empty

Senden einer Alive-Nachricht

5.2. JMX-Informationen manipulieren

5.2.1. Setzen gleich mehrer Attribute eines MBeans

Syntax des Befehls
http://host:port/setattributes?
objectname=XXX&value_YYY=XXX&value_YYY2=XXX2&[set_XXX=Set | setall]

5.2.2. Aufrufen einer JMX-Operation

Syntax des Befehls
http://host:port/invoke?
objectname=XXX&operation=XXX&type0=XXX&value0=XXX...

Mit diesem Befehl können Sie einen gesamten Host samt aller Anwendung stoppen oder den Connector kurzzeitig außer Betrieb nehmen.

5.2.3. Löschen eines MBeans

Syntax des Befehls
http://host:port/delete?objectname=XXX

Warnung Das Löschen eines MBeans sollten Sie sich sehr gut überlegen und nur durchführen, wenn vorher das Objekt gestoppt wurde.

5.2.4. Ausgabe des Construtors Metainformationen eines MBeans

Syntax des Befehls
http://host:port/constructors?
classname=mx4j.tools.adaptor.http.HttpAdaptor

Information über die Verfügbarkeit und Parameter der Konstruktoren eines MBeans.

5.2.5. Erzeugen eines neue MBeans

Syntax des Befehls
http://host:port/create?
class=XXX&objectname=XXX&type0=XXX&value0=XXX...

Selbstverständlich können Sie auch neue MBeans erzeugen. Sehr interessant ist in diesem Zusammenhang die MBeanFactory (Objectname *Catalina:type=MBeanFactory*). Dieses MBean können Sie zur kontrollierten Erzeugung von Tomcat MBeans heranziehen. In der Regel gibt es aber leider Probleme solche MBeans durch entsprechenden Operationen an die jeweiligen Laufzeitinstanzen des Tomcats zu binden.