Die Magie von MBeans und JMX

Andreas Chatziantoniou
Foxglove-IT BV
Utrecht, Niederlande

Einleitung

Beim Betrieb von WebLogic und Fusion Middleware Umgebungen sind die Konsole und die FMW Control geeignete Tools aber um sehr detaillierte Informationen zu erhalten ist es notwendig um die MBeans abzufragen. In dieser Session wird gezeigt wie MBeans organisiert sind, wie sie ausgelesen werden können und welche Informationen sie liefern. Weiterhin wird der Zusammenhang von MBeans mit JMX erläutert. Dies resultiert in der Entwicklung eigener Anwendungen welche genau die aktuellen Daten anzeigen die Sie brauchen um zu analysieren wie ihre Anwendung sich verhält.

Was sind MBeans?

Eine MBean ist in Java die Repräsentation einer Resource. MBeans haben an der Außenseite ein Management Interface mit dem das Auslesen von MBeans-internen Werten aber auch die Veränderung dieser Werte ermöglicht wird. Hierdurch kann eine MBean Informationen über den Verlauf der Anwendung geben.

In der Implementierung von MBeans werden die Variablen definiert die Informationen enthalten.

WLS und MBeans

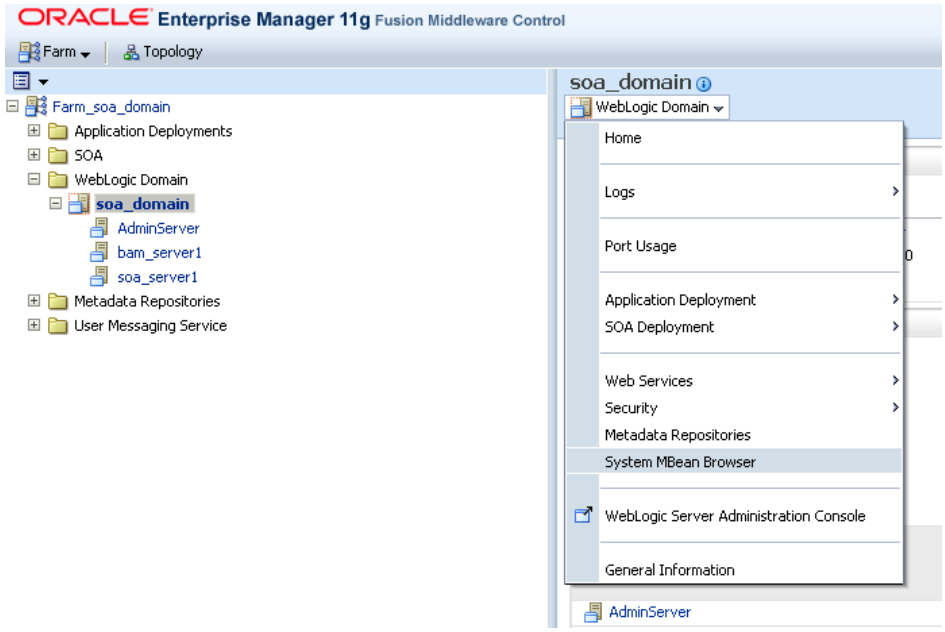
Alle WebLogic Server MBeans unterschieden werden in MBeans die das Verhalten des WLS überwachen oder den WLS konfigurieren.

Runtime MBeans liefern Daten über den aktuellen Status des WLS. Diese Daten sind flüchtig, da sie sich ständig ändern. Es findet keine Speicherung dieser Daten statt.

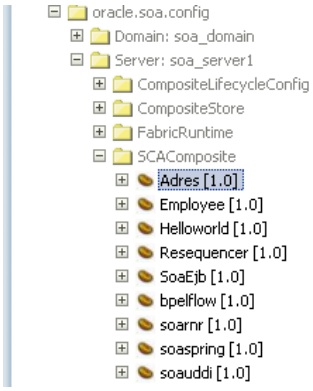
Configuration MBeans beschreiben die aktuelle Konfiguration. Da diese Daten den Neustart des WLS überstehen müssen werden sie in den XML Konfigurationsdateien gespeichert.

Im WLS existieren Werkzeuge um diese MBeans zu lesen bzw. zu verändern:

So ist der System MBean Browser auch auf dem Level einer Domain vorhanden:



Innerhalb der Domain werden Anwendungen deployed. Diese haben oft ihre eigenen MBeans die eine Konfiguration beschreiben.



Weiterhin bietet die WLS Konsole auch ein Interface um die verfügbaren MBeans und auch die vorhandenen Operationen zu sehen.

Application Defined MBeans: SCAComposite:Adres [1.0]

Hide MBean Information

MBean Name: oracle.soa.config:Location=soa_server1,label=5a43edd9-0199-4717-815f-1eeb5d1215cc, Application=soa-infra,name="Adres",revision=1.0,j2eeType=SCACor wsconfigtype=WebServicesConfig

Description: SCA Composite

Attributes | Operations | Notifications

	Name	Description	Access	Value
1	ApplicationName	Composite Application Name	R	default
2	Components	Component MBeans associated with this composite	R	oracle.soa.config:SCAComposite="Adres",Location=soa_server1,label=5
3	CompositeXML	XML representation of the composite model	R	<?xml version = '1.0' encoding = 'UTF-8'?> <composite name="Adres" ap
4	ConfigMBean	If true, it indicates that this MBean is a Config MBean.	R	false
5	DN	Composite Distinguished Name	R	default/Adres!1.0*5a43edd9-0199-4717-815f-1eeb5d1215cc
6	EJB21ContextRoot	EJB 2.1 Context Root	R	
7	eventProvider	If true, it indicates that this MBean is an event provider as defined by JSR-77.	R	true
8	eventTypes	All the event's types emitted by this MBean.	R	jmx.attribute.change
9	Imports	Imports	R	[Ljavax.management.openmbean.CompositeData;@11360a3
10	Mode	Composite Mode	R	active
11	Name	Composite Name	R	Adres [1.0]
12	objectName	The MBean's unique JMX name	R	oracle.soa.config:label=5a43edd9-0199-4717-815f-1eeb5d1215cc,Applic
13	Properties	access to the properties	RW	[Ljavax.management.openmbean.CompositeData;@12d7824
14	ProviderConfigs	The provider-port element Config MBean names	R	
15	ReadOnly	If true, it indicates that this MBean is a read only MBean.	R	false
16	References	Reference MBeans associated with this composite	R	oracle.soa.config:Location=soa_server1,j2eeType=SCAComposite.SCAR
17	RestartNeeded	Indicates whether a restart is needed.	R	false
18	Revision	Composite Revision	R	1.0
19	Services	Service MBeans associated with this composite	R	oracle.soa.config:Location=soa_server1,j2eeType=SCAComposite.SCAS
20	State	Composite State	R	on
21	stateManageable	If true, it indicates that this MBean provides State Management capabilities as defined by JSR-77.	R	false
22	statisticsProvider	If true, it indicates that this MBean is a statistic provider as defined by JSR-77.	R	false
23	SystemMBean	If true, it indicates that this MBean is a System MBean.	R	false
24	WebServiceConfigs	The webservice-description element Config MBean names	R	oracle.soa.config:Location=soa_server1,j2eeType=SCAComposite.SCAS
25	Wires	Get the information about the wires in this composite	R	[Ljavax.management.openmbean.CompositeData;@2a99a1

Application Defined MBeans: SCAComposite:Adres [1.0]

Hide MBean Information

MBean Name: oracle.soa.config:Location=soa_server1,label=5a43edd9-0199-4717-815f-1eeb5d1215cc, Application=soa-infra,name="Adres",revision=1.0,j2eeType=SCAComposite, wsconfigtype=WebServicesConfig

Description: SCA Composite

Attributes | **Operations** | Notifications

	Name	Description	Parameters	Return Type
1	attachProperty	add a new property	1	void
2	getPolicySubjects	Returns an array of ObjectName instances for those policy subjects associated with this object that satisfy the filter criteria	3	Array of javax.management.ObjectName
3	hasProviderConfigs	The provider-port element Config MBean names	0	boolean
4	keepEMDiscoveryAlive	Set if the EM Discovery MBean should remain registered after this MBean is unregistered	1	void
5	removeProperty	remove the specified property	1	void
6	removeProperty	remove the specified property	1	void
7	save	Persist composite edit event.	0	void
8	setProperty	add a new property	2	void

Wie setze ich JMX ein?

Die Java Platform bietet in ihrer Modularität die Möglichkeit um Anwendungen so zu monitoren, dass CPU, Memory und die Benutzung von Resources sowie andere Metriken.

Im Gegensatz zu Systemtools kann hiermit ein Detailniveau der Anwendung untersucht werden, womit die Möglichkeit besteht um genau zu sehen welcher Teil der Anwendung diese Resources benutzt. Anwendungen sollten daher auch eigene MBeans mitbringen damit kein eigenes Monitoring Interface und die dazugehörige Infrastruktur aufgebaut werden muss.

Beim Einsatz von JMX ist es möglich um direkt in den JVM einzusteigen. Dies kann sowohl lokal als auch remote stattfinden.

In der Basis hat JMX einen Agenten und einen Viewer. Der Agent bietet die Informationen innerhalb der Anwendung an. Der Viewer (z.B. Java Mission Control) bietet normalerweise Informationen über der Prozess aber auch über die Resources die benutzt werden. Außerdem kann man auch mit einfachen Mitteln seinen eigenen JMX Viewer entwickeln.

Um seine Anwendungen vor der "Bespitzelung" durch fremde JMX Viewers zu beschützen ist es wichtig um den Zugang zum JMX Agenten zu sichern. Dies kann z.B. durch ein Passwort in einer Datei geschehen ($JRE/lib/management/ jmxremote.password.template)

Custom metrics in JMX für Entwickler

Gerade für Entwickler ist es wichtig um mehr Informationen über den Verbrauch von Resources zu erhalten. Hiermit kann eine Reihe von anwendungsbasierten Daten erhalten werden. Ein Beispiel wäre das Verarbeiten von Batches. Statt den Fortgang in immer weiteren *print-statements* auf die Konsole zu schreiben kann eine eigene MBean diese Daten an einen JMX Viewer weiter senden.

Solche Werte können natürlich auch für den Betrieb der Anwendung benutzt werden.

Kontaktadressen:

Andreas Chatziantoniou
Foxglove-IT BV
Texel 18
NL-3524 AP Utrecht

Telefon: +31623259167
E-Mail andreas@foxglove-it.nl