Configuration Support for GlassFish V3

Kedar Mhaswade, Lloyd Chambers, Jerome Dochez, Kohsuke Kawaguchi

Introduction

This document focuses on ideas about integrating configuration support for GlassFish V3. codebase and others will be part of the V3 Config support. Few goals that we tried to satis

- Extensibility
- Support for JMX/AMX
- Dynamic reconfiguration
- Batching configuration changes and allowing for role based administration
- Provision of Validation

Reader is assumed to be familiar with basic principles of V3, Java 5 and JMX.

Overall Scheme

At some level, a V3 distribution is a set of HK-2 modules. This document discusses the ne that create these modules. The configuration of these *objects* can optionally be stored in dc modules come into being (according to a set of contracts) and a habitat of configuration o the domain.xml is injected into the configuration objects according to certain annotations a configuration objects themselves may well be the MBeans that are exposed for managemer

Let's take a snap-shot of such a runtime. Typically, this is how it looks like:

- You have a few modules started (those that satisfy the @ModuleStartup contract) WebContainer. These modules are configured by certain configuration objects, w

Attributes in domain.xml. Needless to say, each configuration object is an instanc
Interface).

- A set of configuration objects and corresponding MBeans identified by domain.xml
read from domain.xml. Let's work with following snippet placed at a convenient loc

```
<http-listener id="ls1" port="8080" address="0.0.0
    <property name="GrizzlyBufferSizeInBytes" valu
<http-listener id="ls2" port="8181" address="0.0.0
    <ssl alias="verisign-1"/>
<http-listener>
<virtual-server id="vs1" hosts="www.abc.com">
```

- Thus, at this point in time, there are five MBeans in the runtime. They have defined
note that the MBeans might have a number of management operations not evident fr
creation and deletion of a child element. Another subtle point is that these MBeans a
predefined with certain conventions. This is required to look them up according to a
property names in the property list of ObjectName. For example:
    - type specifies the class name of the configuration object, like org.glassfish.w
    - parent specifies the type of the logical parent of a given configuration object

With this snap-shot, this section outlines how administrative tools (via configuration infrast
basic administrative operation is any of:

1. Modification of an existing element in terms of attribute values.
2. Addition or Deletion of an existing element by name.
3. Listing elements of the same class (e.g. all http-listener elements).
4. Accessing an element attribute, a set of element attributes, an element or a set of elen
specification.
5. Invoking a long-running operation that accesses (or mutates) a set of configuration o
thought process for this is yet to be complete).

Here is how this is proposed to be done:

1. Administrative interfaces find out about the MBean of interest (somehow). For exam
that its ObjectName is on1 . Interfaces also establish an MBeanServerConnection (n
2. The meta-data of this MBean is made available through mbsc.getMBeanInfo(on
3. Five operations are of essence:
    1. mbsc.setAttribute(on1, attribute1), where attribute1 is: "port" -> 9080 (instead
    2. mbsc.setAttributes(on1, list1), where list1 is: "port"->9080, "address"->"mym
    3. mbsc.invoke(on1, "createchild" ...)

    4. mbsc.getAttribute(on1, "port")
    5. mbsc.getAttributes(on1, list1)
4. At the MBeanServer level, when it knows which MBean(s) are getting modified or a Transaction. When an MBean is placed in a Transaction this way, it is immediatel MBean are made on the clone. Thus, during the Transaction, a copy of MBean corre While a particular MBean is in a Transaction and the operation might mutate the ME may or may not involve that MBean fail (See the unresolved question #8 to decide w
5. During an in-flight Transaction, an MBean or a set of MBeans is modified. The syst MBeans along with the state of rest of the configuration. The Validation phase the same Transaction. Till now, only the copies of MBeans are modified. With every mo AttributeChangeNotification is emitted to which a Transaction-specific JMX N responsible for maintaining the list of changes (Delta) during the Transaction.
6. When the Validation fails, the MBeans/configuration objects in the Transaction are i
7. When the Validation succeeds, it means few things (and this is something where lot
    1. The domain.xml is kosher and can be flushed.
    2. Co-located View: When the (copy of) configuration objects are *shared* betv Grizzly here), a successful change in state implies that the runtime actually res not only valid, but also currently applied. This is a significant deviation from h the Co-located View.
    3. Distributed View: When the (copy of) configuration objects are *not shared* (this is the classic enterprise case, where admin server sees/makes the changes implies that only validation is successful. Thus, in this view, since domain.xm not adapted to the changes). Call this item as the Distributed View.

       It is important that we understand and agree upon the differences in Co-locate need to modify the design.

Unresolved Questions

1. Should MBeanServer be injected into the configuration objects? Most likely, MBear associated with remote JMX Connectors. This is better, for it gives us a control over MBeanServer Interceptors.
2. Do we use VetoablePropertyChangeEvent or JMX AttributeChangeNotif
3. Where should the configuration of an arbitrary class (a module that is new) be specif the referential structure of the domain.xml?

4. Since there is no schema, what should be done with default values of attributes? Hov
5. How to do the long-running operations?
6. Should the Validation be delegated to the module classes?
7. Should the read-only operations be done outside a Transaction?
8. Should the entire set of MBeans be locked while a Transaction is in flight or we can
   semantics have to be carefully done.