VIRTUAL HOSTING FEATURES IN GLASSFISH

Jan Luehe

Sun Microsystems, Inc.

Agenda

- Virtual hosting concepts
- Root context configuration choices
- Security
- Access logging
- Customization features
- Dynamic reconfiguration
- Isolation
- Q&A

Virtual Hosting: How does it work?

- Refers to the ability to run several web sites (*domains*) from a single physical server machine with a single IP address
- Also known as S*hared IP Hosting*
- Each web site is identified by its domain (*host*) name
- HTTP requests are routed to the appropriate domain, based on the host name in their URL
- DNS must be configured in such a way that each hosted domain name resolves to the server's IP address
- Enables ISP/ASP business models

Example

http://mydomain.com:1024/
http://yourdomain.com:1024/

- mydomain.com and yourdomain.com are virtual server (host) names that resolve to the same IP address
- The HTTP listener on port 1024 dispatches incoming requests to the appropriate virtual server, based on the host name of the request URL
- mydomain.com and yourdomain.com may share the same HTTP port(s) (as in this example), or use different ports

Virtual Servers and HTTP Listeners

- A <virtual-server> receives requests from only those HTTP listeners listed in its “http-listeners” attribute
- An <http-listener> knows about the virtual servers that it is supposed to dispatch to
- If an <http-listener> receives a request with an unknown host name, it dispatches the request to the virtual server referenced by its “default-virtual-server” attribute

Example: Virtual Servers and HTTP Listeners

Protocol	Domain Name	Port Number	URL Prefix	Target Virtual Server
http	mydomain.com	1111	http://mydomain.com:1111	vs-1
https	mydomain.com	2222	https://mydomain.com:2222	vs-1
http	yourdomain.com	3333	http://yourdomain.com:3333	vs-2
https	yourdomain.com	4444	https://yourdomain.com:4444	vs-2

Example (cont.): Virtual Servers and HTTP Listeners

```
<http-listener id="http-listener-1" port="1111"/>
<http-listener id="http-listener-2" port="2222" security-
   enabled="true"/>
<http-listener id="http-listener-3" port="3333"/>
<http-listener id="http-listener-4" port="4444" security-
   enabled="true"/>

<virtual-server id="vs-1" hosts="mydomain.com"
     http-listeners="http-listener-1,http-listener-2"/>
<virtual-server id="vs-2" hosts="yourdomain.com"
     http-listeners="http-listener-3,http-listener-4"/>
```

Virtual Server States

- A virtual server may be in one of 3 states:
  - *on* (default)
  - *off*
    - Any requests mapped to the virtual server will result in 404 responses
  - *disabled*
    - Any requests mapped to the virtual server will result in 403 responses
- A virtual server's state is configured via the “state” attribute of <virtual-server> in domain.xml

Root Context Configuration Choices

- A virtual server's *root context* is defined as the context with the empty path (/)
- Any requests that cannot be mapped to any of the web modules deployed on a virtual server will be mapped to the virtual server's root context
- GlassFish supports the following root context configuration choices:
  - Docroot
  - Default web module
  - Alternate docroots

docroot

- Specifies the physical location of a virtual server's root context
- Specified via the virtual server's "docroot" attribute and property
- Every virtual server may be configured with its own individual docroot

Example: docroot

```
<virtual-server
    hosts="mydomain.com"
    docroot="${com.sun.aas.instanceRoot}/docroot"/>
```

http://mydomain.com/

Contents of
${com.sun.aas.instanceRoot}/docroot/index.html

Default Web Module

- Makes it possible to map a virtual server's root context to one of its deployed web modules
- Convenient way for having a single web module occupy both its designated context root as well as the virtual server's root context
- "Shadows" the virtual server's docroot (that is, the docroot is ignored for as long as the default web module is in place)
- When the default web module is no longer in place, the virtual server's docroot is reinstantiated as the virtual server's root context
- Specified via the "default-web-module" attribute of <virtual-server>

Example: Default Web Module

```
<virtual-server default-web-module="abc" hosts="mydomain.com" />
<web-module
   context-root="abc"
   location="${com.sun.aas.instanceRoot}/applications/j2ee-
    modules/abc"
   name="abc"/>
```

http://mydomain.com/

http://mydomain.com/abc/

Contents of

${com.sun.aas.instanceRoot}/applications/j2ee-modules/abc/index.html

Alternate Docroots

- In GlassFish, a virtual server may specify alternate docroots in addition to its main docroot
- Each alternate docroot is associated with one or more URL patterns
- An alternate docroot is selected over the main docroot if the request matches one of its URI patterns
- Both path (prefix) and extension matches are supported
- If a request matches multiple alternate docroot URL patterns, the following precedence order is used:
  - Exact match
  - Longest path match
  - Extension match
- Alternate docroots allow virtual servers to share a subset of their

Example: Alternate Docroots

```
<virtual-server id="server" [...]>
  <property name="alternatedocroot_1"
            value="from=/images/* dir=/usr/gifs"/>
  <property name="alternatedocroot_2"
            value="from=*.jpg dir=/usr/gifs"/>
  <property name="alternatedocroot_3"
            value="from=*.jsp dir=/usr/jsps"/>
  <property name="docroot"
            value="${com.sun.aas.instanceRoot}/docroot"/>
</virtual-server>
```

Security

- Single Sign On (SSO)
- Authentication realms

Security: Single Sign on (SSO)

- Enables web applications to share client authentication state
- Configured via virtual server's "sso-enabled" property
- When enabled, any user who successfully authenticates to any one web application becomes implicitly logged in to any other web application deployed on the same virtual server that shares the same authentication realm

Security: Authentication Realm

- Webapps that require authentication specify the name of an authentication realm
- Authentication realm may be specified at virtual server level (via "authRealm" property), with the expectation that it apply to all web modules deployed on the virtual server
- Referenced authentication realm must exist in domain.xml
- Standalone web modules inherit the authentication realm of their virtual server, unless they specify their own

Access Logging

- Every virtual server has its own individual access log file
- Access logging may be configured via a combination of the following elements in domain.xml:
  - ${appserver.instance.name}.http-service.property.accessLoggingEnabled: Enables or disables access logging
  - ${appserver.instance.name}.http-service.access-log: Configures access log format (pattern), as well as rotation policy, interval, and suffix
  - ${appserver.instance.name}.http-service.virtual-server.*[vs]*.http-access-log: Configures access log file location, and whether to log the client's IP address or DNS name
  - ${appserver.instance.name}.http-service.virtual-server.*[vs]*.property.accesslog: Configures access log file location

Access Log Patterns

- Any combination of the following tokens may be used to specify an access log pattern in domain.xml:

  %auth-user-name% %client.dns% %client.name%

  %cookie.value% %datetime% %header.accept% %header.%

  %header.auth% %header.date" %header.if-mod-since%

  %header.user-agent% %header.referer% %http-method%

  %http-uri% %http-version% %query-str% %referer%

  %request% %response.length% %status% %user.agent%

  %vs.id%

- GlassFish also supports “shortcuts” for some well-known access log patterns

Support for Apache Access Log Formats

- *common*
  - http://httpd.apache.org/docs/2.2/logs.html#common
  - domain.xml:

```
<http-service>
  <access-log format="common"/>
</http-service>
```

- *combined*
  - http://httpd.apache.org/docs/2.2/logs.html#combined
  - domain.xml:

```
<http-service>
  <access-log format="combined"/>
</http-service>
```

Custom Valves and Listeners

- GlassFish allows custom implementations of the following interfaces to be added to any <virtual-server> in domain.xml:
  - org.apache.catalina.Valve: Allows to intercept the request and response objects passed through the virtual server
  - org.apache.catalina.ContainerListener: Gets notified of any web context and valve registrations with, and unregistrations from the virtual server
  - org.apache.catalina.LifecycleListener: Receives notifications for any lifecycle changes of the virtual server
- Implementations are specified as <virtual-server> properties, using their fully qualified class names

Example: Custom Valves and Listeners

```
<virtual-server id=”server” [...]>
  <property name="valve_1"
                value="com.sun.enterprise.MyValve"/>
  <property name="listener_1”

   value="com.sun.enterprise.MyContainerListener"/>
  <property name="listener_2"
                value="com.sun.enterprise.MyLifecycleListener"/>
  </virtual-server>
```

Dynamic Reconfiguration

- All aspects of virtual hosting, including
  - the creation and deletion of virtual servers
  - the manipulation of virtual server attributes and properties

  are dynamically reconfigurable, i.e., do not require any domain restart in order to take effect

Profile Dependent Defaults

- GlassFish V2 introduces 3 *profiles* to better address the specific configuration requirements of various runtime environments:
  - *developer*
  - *cluster*
  - *enterprise*
- *Profiles* supersede earlier product *editions*
- Access logging and SSO are disabled by default in the *developer* and *cluster* profiles

Isolation

- If co-located on a single instance, virtual servers and the webapps deployed on them may interfere with each other:
  - A buggy (or malicious) webapp deployed on one virtual server may affect all other virtual servers
  - No "jailed manager"
- If isolation of virtual servers is a concern:
  - Create one domain for each virtual server (GlassFish V1 and V2)
  - Create one standalone instance, or cluster of instances, for each virtual server in a single domain (GlassFish V2)

Comparison with other Products

- GlassFish and Tomcat comparison table at:

  http://www.glassfishwiki.org/gfwiki/Wiki.jsp?page=GlassFishVsTomcat

- GlassFish community effort: Originally started by Roger Keays, then turned into the above Wiki by Jason Lee
- Tomcat's “Configure same webapp differently" feature is being added to GlassFish as we speak

Q&A

VIRTUAL HOSTING FEATURES IN GLASSFISH

Jan Luehe
jan.luehe@sun.com