DZone Refcardz

Brought to you by Sun microsystems

CONTENTS INCLUDE:

Getting Started with GlassFish Application Server v3

By Masoud Kalali

ABOUT GLASSFISH

GlassFish is a Java EE application server hosted at http://glassfish.dev.java.net and is sponsored mainly by Sun Microsystems. Based on the Java EE reference implementation, GlassFish is the first application server to support new specifications and standards. It is distributed under two open source licenses - GPL v2 with the Classpath exception and CDDL.

What Makes GlassFish Different

Several unique features differentiate GlassFish v3 from other application servers: it's the first to support Java EE 6 and Web Profile; modularity based on OSGi; session retention on application redeploy; support for the latest Web services specification and proven interoperability with Microsoft WCF; easy-to-use administration console and command line; low-overhead, fine grained monitoring, in addition to already available monitoring facilities in its administration channels; and runs dynamic languages and frameworks like RoR using native containers. For now, those requiring high availability clustering and centralized administration with Java EE 5 applications should consider GlassFish v2. These features are in the GlassFish v3 roadmap.

Support for GlassFish is available in a community driven mode (free) and commercial support provided by Sun Microsystems. It's worth noting that to move from community driven support to commercial support there is no need to re-install. Install the Sun Branding and license from the Update Center, enter a commercial relationship with Sun Microsystems, and have best of the breed application server along with support from a leading Java EE vendor.

GlassFish v3

GlassFish v3 is the first application server based on OSGi with full support of Java EE 6 umbrella specification. OSGi and a highly modular architecture turn GlassFish v3 into the most flexible and extensible application server available on the market. It offers many GlassFish v2 features, and many new features and enhancements introduced in different layers - from application server security and stability to developer productivity improvements. Some additional new features include:

- Embeddable; dynamically extensible
- IPS as main packaging, distribution and updating system
- Uses OpenInstaller for installation; optional zip installer
- RESTful monitoring interface

Web and Enterprise Profiles

GlassFish v3 fully supports Java EE 6 and also provides both Web Profile and Java EE distributions. These distributions are different in terms of provided services and footprint. The beauty of modularity is the option to install the Java EE or Web Profile distributions and simply use the update center to download/install/uninstall the desired modules. The Web Profile distribution does not provide enterprise integration services such as JCA and JMS.

DOWNLOADING AND INSTALLING

To download GlassFish visit https://glassfish.dev.java.net/public/downloadsindex.html and selecting the appropriate operating system. JDK 6 (version 1.6.0_13) or a newer version is required for installation. Supported operating systems on different architectures include: Solaris, OpenSolaris, Linux, AIX, and Windows. Installing GlassFish is as easy as following the automatic installer's steps. Alternatively, GlassFish v3 can be installed using a zip installer; just download, unzip, and go. For the remainder of this Refcard, assume GlassFish is installed in gf_home.

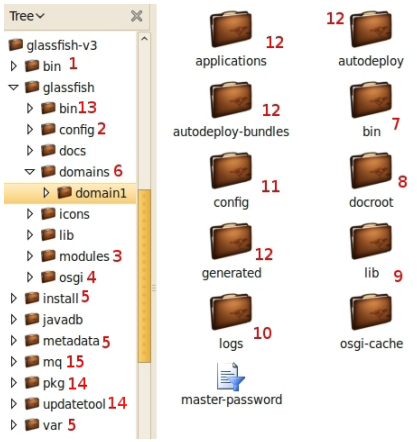

Figure 1: GlassFish installation directory structure

GlassFish Domains

GlassFish, like other application servers, uses the concept of domains: entities which define the scope of administration.

Each GlassFish v3 installation has its own domain, although support for multiple domains – an existing GlassFish v2 feature - is in the GlassFish v3 roadmap.

GlassFish Directory Structure

The directory structure of GlassFish v3 has changed to cope with new architecture and features. Figure 1 (page1) shows the new directory layout and Table 1 provides descriptions of important directories.

All directories in Figure 1 are numbered to ease locating them in the table and in Figure 2 (page3).

Table 1: Directory structure description

Dir	Description
1	Contains links to asadmin and update center scripts. Directory no. 13 contains Java EE related scripts in addition to what is available in this directory.
2	GlassFish configuration files.
3	This is where new OSGi modules are placed to allow the application server platform to pick them up.
4	OSGi platform used by GlassFish.
5	OpenInstaller and un-installation related scripts.
6	The default directory which GlassFish domains are created in. Domain related CLI commands look for target domain inside this directory
7	Domain start and stop scripts.
8	Default document root for serving http://IP:PORT/ Another set of files may be included here if required.
9	This is where shared libraries such as JDBC drivers, Apache Commons, etc. are placed.
10	Stores all logs, including server logs, access logs, and transaction logs.
11	Includes domain related configuration files like domain.xml, default-web.xml, key stores, etc.
12	Deployed applications, ready to be deployed applications and container generated files for deployed applications are stored in these directories.
13	Update center scripts and PKG (5) image packaging system are located inside these directories.
14	Open MQ which is the default MQ implementation of GlassFish is located here.

GLASSFISH ADMINISTRATION

GlassFish provides multiple administration channels for administration flexibility.

Web Based Administration Console

To access the Web Console, point a browser to http://IP_ADDRESS:PORT/. By default the port number is 4848 and the listener listens on all available network interfaces of the server machine. The default administration credentials are admin with no password, although one can be provided during installation. GlassFish uses a separate Virtual Server for Web Console for independent configuration.

Command Line Administration Console

This is the preferred approach for experienced administrators who choose to use CLI for administration purposes. Some tasks like creating/removing/backing up and restoring a domain are only possible using the CLI. The CLI also enables automation of routine tasks through shell scripts, as well as integration with provisioning tools. The administration CLI is accessible through the asadmin utility located at gf_home/bin directory. It is either a batch file named asadmin.bat for Windows or a shell script named asadmin for Linux and UNIX.

The asadmin script has two modes. The first, invoking schema, is more suitable for creating custom scripts or for executing only one command:

```
./asadmin [ program options] command_name *[[--param] values]
```

The asadmin options are shown in Table 2:

Table 2: The asadmin program options

`--host`	The application server administration listener IP address or host name.
`--port`	The administration listener port number.
`--user`	A username in admin realm. admin by default.
`--passwordfile`	The password file containing administrator's password.
`--terse`	Add new type of container to host; for example another dynamic language applications.
`--interactive`	If specified, CLI utility will ask for required parameter in an interactive way.
`--secure`	If specified, CLI utility will use HTTPs to communicate with administration application to prevent possible security breach.

Not all of these parameters are required for executing a command and only important items are detailed in this Refcard.

The second mode of using asadmin script is entering the shell and running commands inside the shell. This is a bit faster than the first method which needs to run the asadmin utility each time a command is invoked.

To enter the shell and invoke commands, execute the asadmin script without any trailing command or parameter. The following example shows the asadmin shell and how to execute a command inside it.

```
asadmin>help create-domain
```

The invoking schema is the same for both methods but there are some usage differences. The CLI can execute most commands against a remote server by providing the address and port number of the administration listener. If not, it will execute the command against the default local server which is 127.0.0.1:4848.

Hot Tip You can get a list of all commands along with a brief explanation about them by invoking help command. To save the output for later review use the following command: `./asadmin help > path_to_file.txt`

JMX/AMX Administration Channel

This channel is widely preferred by developers who need to interact with GlassFish administration and management layer using the Java language. Administration tasks can also be executed by writing Java code using JMX. The GlassFish JMX listener listens on port 8686 by default. The username and password are the same as those for the CLI and Web Console. Any JMX console can be used, like the JDK's JConsole or MC4J, to examine the GlassFish JMX MBeans and see what functions are availble.

RESTful Administration Channel

This interface exposes complete GlassFish management and monitoring functionalities through a RESTful interface, where output can be formatted using HTML, JSON or XML. The root URL for RESTful administration is http://ADMIN_ADDRESS:ADMIN_PORT/management/domain/, which can accept requests from a browser, tools like curl or wget, or even Java applications using JAX-RS client libraries. Use the POST method to update the configuration, and use the GET method to view the configuration. For example to view the monitoring levels, send the following GET request from any web browser:

```
http://localhost:4848/management/domain/configs/config/server-config/
monitoring-service/module-monitoring-levels
```

To create a new HTTP listener we can POST the following command to the server.

```
curl -X POST -d "web-container=ON" -H "Accept: application/json"
http://localhost:4848/management/domain/configs/config/server-config/
monitoring-service/module-monitoring-levels
```

DAILY ADMINISTRATION TASKS

Administration is performed on a day-to-day basis. Some tasks may be easier using the CLI, others with the Web Console.

Common Administration Tasks in CLI

Table 3 shows the list of common CLI commands. Some are exclusive to CLI, and others are easier to perform this way. Local commands are marked using an asterisk *.

Table 3: Common CLI Commands

Command	Description
list-commands	List local and remote commands. Remote commands are listed if the remote server is up and running.
create-domain*	Create a new domain.
start-domain*	Start the given domain.
stop-domain*	Stop the given domain.
deploy	Deploy an application.
undeploy	Undeploy an application.
backup-domain*	Create a backup of domain configurations. It does not include deployed applications.
restore-domain*	Restore the given backup.
list-domains*	List all domains in the default or the given domains directory.
verify-domain-xml*	Command to verify the the domain.xml file. Useful when edited manually.

More CLI commands are covered in the Security section of this Refcard. There are tens of other commands for administrating Java EE managed resources and application server functionalities. The complete list of commands can be viewed by invoking the help command. An example of how to execute the create-domain command to create a domain named domain2 is as follows:

```
create-domain --adminuser admin --adminport 4848 --instanceport 8080
--domainproperties domain.jmxPort=8686:http.ssl.port=8181 domain2
```

Administration Tasks in Web Console

The Web Console is suitable for tasks where there are several parameters and attributes involved, or where the CLI is unavailable. It is very easy to create JDBC connection pools, JMS destinations, configure listeners and thread pools, some security related configuration, containers configuration, cluster-wide configuration and management, and so on. The Web Console has a very organized structure to make it easy to find and locate tasks. Figure 2 shows a part of the tree based menu structure of Web administration console.

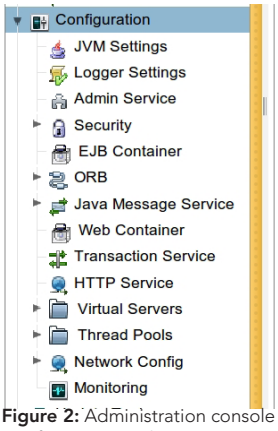

Figure 2: Administration console Configuration node.

Administration Tasks and JMX Console

GlassFish v2 can be monitored and managed using JMX- and AMX-enabled tools such as jconsole, or by developing simple custom applications to take care of repetitive daily tasks or automatic response to some events. Figure 3 shows the GlassFish v3 MBeans tree in JConsole.

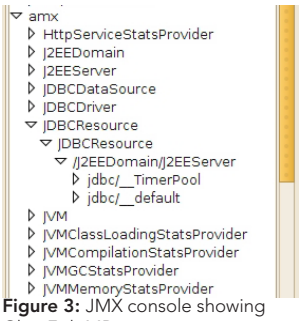

Figure 3: JMX console showing GlassFish MBeans tree

The AMX Beans form a dynamic tree which represents the entire application server and any manageable object inside it. The AMX is a dynamic proxy layer on top of JMX MBeans to ease interaction with the GlassFish management core.

SECURITY

It is wise to spend time securing the application server instead of gazing at the disaster a security leak caused.

Passwords Security

The administration and master passwords should be changed frequently. Change them using provided CLI commands. Table 4 lists the required tasks for overall password security along with a description and command used to complete that security task.

Table 4: Security tasks and related commands

Task	Description and Command
Change administrator password	Change the administrator password from time to time using change-admin-password command.
Change master password which protects the key store files	Change the default master password from "changeit" to something else using change-master-password command.
Alias of the passwords used for connection pools, JMS hosts and so on	Passwords required for connection pools are stored in plain text; to avoid this, use password aliasing. For example: `create-password-alias Alias_Name` and use the alias name in the following format instead of the password: `${ALIAS=sample-alias}` A complete set of commands for creating, updating, listing and deleting aliases is provided.
Do not enter passwords	The CLI is typically used to administer multiple servers; to prevent entering clear-text the passwords on the command line, use a password file containing AS_ADMIN_PASSWORD=${ALIAS=Alias_name} AS_ADMIN_MASTERPASSWORD=${ALIAS=Al_name} and pass it to commands invoked using --passwordfile parameter. And yes, use password aliases here too.
Do not enter passwords	The login command is helpful in interactive mode. After successfully logging into a server, asadmin will store the password and re-use it when a command is invoked on that server. The format is: asadmin login --host HOST_ADDRESS --port ADMIN_PORT The command will enter the interactive mode and ask for passwords.

Listeners Security (Network Security)

Listeners are the interaction channels of the application server with the outside world. Both the administration listener and the ORB listener should be protected from un-authorized access. Sometimes securing listeners means binding them to a specific network interface of the server machine, and sometimes it means mutual authentication protection using digital certificates. Table 5 lists the listeners, security measures and how to access them through Web Console.

Table 5: Listeners and Listeners Security Measures

Listener	How to access	Security Measures
HTTP Listeners	`Tree>Configuration>Network Config> Network Listeners`	Listeners should listen on specific interfaces, not on 0.0.0.0 (all available interfaces).

HTTP Listeners	`Tree>Configuration>Network` Config> Protocols	Check off the Security check box if HTTPS is required for this listener.
ORB Listeners	`Tree>Configuration>ORB>IIOP Listeners`	Make sure to leave required listeners active. Be sure to change the network address to the appropriate one. Use secure listeners if possible.
JMX Listeners	`Tree>Configuration>Admin service`	Enable security (SSL) if required. Use only required network addresses. Change authentication realm if required.

Hot Tip New users with administration permission can be added to admin-realm. To do this, navigate to Tree>Configuration>Security>Realms>admin-realm and then click on the Manage Users button. Add as many users as needed.

Security Audition

Auditing is useful for checking logged events and conducting analysis to locate potential problems. Security auditing makes it possible to review events like failed logins. The default auditing module provided with GlassFish logs all security events in the server.log file. To enable auditing hit Tree>Configuration>Security>Audit Modules>default and then enable the module by changing the auditOn property's value to true. Implementing new auditing modules is as simple as extending the com.sun.appserv.security.AuditModule class, putting the implementation in the domain classpath and adding it as a new audit module in Tree>Configuration>Security>Audit Modules>. After adding the auditing module, it can be enabled by changing the appropriate attribute defined, if any.

HTTP Access Log

Storing HTTP access logs can help with reviewing HTTP access events. Access logs can be enabled in two ways: HTTP Service enables it for all HTTP listeners (Tree>Configuration>HTTP Service) and Virtual Server level enables HTTP Access Log for a particular Virtual Server (Tree>Configuration>Virtual Server). Access logs are stored in a directory named access inside the logs directory of each domain.

Hot Tip A virtual server, sometimes called a virtual host, is an object that allows the same physical server to host multiple Internet domain names. Virtual Servers can use dedicated HTTP listeners and configurations for authentication realms and default web applications. You can create Virtual Server by navigating to Tree>Configuration>Virtual Server.

MONITORING

Monitoring is an essential part of enterprise applications, either in application server services layers or in enterprise applications themselves. GlassFish provides advanced and extensible monitoring capabilities for both applications deployed in GlassFish and for GlassFish services themselves. The monitoring information is accessible though different channels, including all three administration channels in addition to the newly included RESTful interface.

Web Console and Monitoring

To monitor a GlassFish services like JDBC connection pools, HTTP services and so on, enable monitoring for the services of interest. To enable monitoring for a service navigate to Tree>Configuration>Monitoring and change the monitoring level of the services and containers of interest. Three levels of monitoring are available:

1. OFF: No monitoring information is gathered.
2. LOW: Cumulative statistics are gathered.
3. HIGH: Detailed statistics are gathered.

When enabled, monitoring affects the overall performance of the application server, specifically the observed services. It is better not to use HIGH in a production environment.
After changing the monitoring level, GlassFish starts gathering monitoring information which can be viewed in the Web Console or other channels. To view the monitoring information, navigate to Tree>Application Server and select the Monitor tab. This page shows all gathered statistics for any service with statistics collection enabled.

For example, the JDBC Connection Pools performance metrics, with counts the maximum number of connections, maximum available connections, and more, and various timing factors like maximum wait time, and so on.

CLI and Monitoring

For CLI advocates, the CLI provides thorough monitoring capabilities. The CLI offers fine-grained control on which performance factors to monitor. Table 6 lists all CLI commands related to monitoring.

Table 6: CLI monitoring commands

Command	Description
`get`	Getting the value for any configuration attribute in the system. View the monitoring level of a service or multiple services using this command.
`set`	Setting the value for any configuration attribute in the system. Set the monitoring level of a service or multiple services using this command.
`monitor`	Monitoring one or multiple attributes of GlassFish services like transaction, JMS, JDBC connection pools and so on. The command is very flexible about information representation.

Configuration attributes can be accessed using a concept called Dotted Names. Dotted names, as the name may imply, is a hierarchy of attributes representing all servers, services, and managed objects of the GlassFish Application Server in a tree which starts with father nodes and comes down to attribute leafs. Each level is separated from the adjacent level by a dot. The following dotted names can further clarify the concept.

```
resources.jdbc-connection-pool.DerbyPool.allow-non-component-callers
servers.server.server.resource-ref.jdbc/__TimerPool.enabled
configs.config.server-config.ejb-container.max-pool-size
```

Now, let's use these three commands to do some monitoring.

```
asadmin get server.monitoring-service.module-monitoring-levels.*
```

Checks the monitoring levels of all services using the wild-card character.

```
asadmin set server.monitoring-service.module-monitoring-levels.jdbc-
connection-pool=HIGH
```

This last command set the JDBC connection pools monitoring level to HIGH.

```
asadmin monitor --type jdbcpool --filter sample-mysql-pool  --filename
/opt/out2.csv --interval 5 server
```

This monitor command will gather sampling data each 5 seconds for a JDBC connection pool named sample-mysql-pool which is managed by a server instance named server. The command will store the sampling data to a CSV file located at /opt/out2.csv.

JMX and Monitoring

JMX enables application server to be monitored directly from Java source code, and can respond to sampling data in an appropriate way. An example would be a swing-based monitoring console or integration of GlassFish monitoring into a bigger monitoring console to monitor the entire infrastructure from a single console. Another common example is monitoring and changing application server configuration in a headless way.

RESTful Monitoring Interface

One of the unique features of GlassFish v3 is the RESTful monitoring interface that allows virtually any programming language to access GlassFish monitoring information using a RESTful interface.

The RESTful interface represents a hierarchy of GlassFish services, server environment, JVM, deployed applications and managed resources like JDBC connection pools.

To access monitoring information using the RESTful interface, send a GET request to http://ADMIN_ADDRESS:ADMIN_PORT/monitoring/domain/ which will return a list of all child objects for this node (monitoring must be enabled). Going down the hierarchy to get monitoring information along with a list of children for any node in the hierarchy. For example sending a get request to http://localhost:4848/monitoring/domain/server/jvm/memory (with JVM monitoring enabled) will result in an output similar to Figure 4.

Figure 4: JVM memory monitoring information

The RESTful interface can format the output to JSON or XML when the appropriate trailing parameters are included. For JSON format include application/json in the URL and for XML include application/xml in the URL trail. For example:

```
curl http://localhost:4848/monitoring/domain/server/jvm/
memory   -H "application/xml"
```

The RESTful monitoring interface enables developing monitoring solutions using scripting languages like Perl, Python and so on.

For GlassFish Enterprise Server v3 commercial customers, Sun Microsystems provides a monitoring scripting client, which enable ad-hoc querying using fine-grained probes, similar to DTrace on Solaris.

PERFORMANCE TUNING

Enterprise application performance is a big concern, and performance tuning is a hurdle which all enterprise application developers are involved with. While Java EE performance tuning is out of the scope of this refcard, some GlassFish performance tuning information is provided. Additional tuning information is a available through performance whitepapers available through Sun (http://www.sun.com/software/products/glassfish_portfolio/resources.jsp). In addition, GlassFish Enterprise Server subscriptions include Enterprise Manager, which offers tools to "auto-tune" the configuration based on a simple set of questions about the deployment environment.

Although the JVM uses sophisticated garbage collection algorithms, performance and memory can be tuned by changing memory management and garbage collection parameters, depending on the deployment environment.

Tools like VisualVM give a good sense of what is going on the application server. VisualVM is now included in JDK and is accessible using the java_home/bin/visualvm. Installing visual GC plug-in for Visual VM can help with memory tuning. More information about Visual VM is available at http://visualvm.dev.java.net or JDK documentation. To change the application server's JVM option navigate to Tree>Configuration>JVM Settings and select the JVM Options tab.

Web Container Performance Tuning

Tuning the Web container will lead to the application server processing Web requests in a more efficient way and will result in higher throughput. Some determining factors for Web container performance are:

- HTTP listener thread pool attributes.
- File cache support in HTTP Protocol of each HTTP listener.
- Disabling the access log.
- Disabling monitoring.

Thread pool attributes customization highly depends on the deployed applications' characteristics in relation to whether they are memory consumers, process consumers, IO bound and so on.

Tuning the Web container is easier by analyzing the statistics driven from monitoring the Web container and deployed applications. Web Container configuration is available at Tree>Configuration>Web Container node. Listener and protocols configuration are available at Tree>Configuration>Network Config>Protocols.

EJB Container Performance Tuning

The EJB container can affect performance more than the Web container if deployed applications use EJBs extensively for business logic processing. Fine-tuning the EJB container involves tuning the EJB container thread pool and its cache size. Although the timer service effects are usually not intensive, for applications with a large number of timers it is recommended to reduce the timer service load by reducing the service precision.

Fine-tuning Message Driven Beans pool affects the overall performance of the system if it is highly dependent on JMS and message processing.

Decreasing the pool size results in fewer MDBs processing the JMS messages and less overhead on the system. However it will also result in longer wait time for asynchronous tasks and requests.

Monitoring the EJB container using GlassFish built-in capabilities helps to tune the EJB container. Enable monitoring, let the system work for some time, and then decide on new values for container attributes. EJB container configuration is accessible through Tree>Configuration>EJB Container.

JDBC Connection Pool Performance Tuning

JDBC connection pools often cause bottlenecks in Web applications which use JDBC to access a database. Monitoring JDBC connection pools using GlassFish monitoring can give a fair amount of information about whether application performance is related to JDBC connection pool configuration. Usually, maximum wait time, average wait time, high watermark and minimum available free connections are parameters that help us decide whether to further limit the size of the connection pool, or expand it (to prevent requests processing waits longer than necessary for a connection to become free).

Detecting connection leaks is another very helpful feature of GlassFish JDBC monitoring. Simply scan the monitoring results to determine if there aree JDBC connection leaks.

JMS Server Interaction

GlassFish supports three different methods for integration with a JMS broker. A local or embedded broker means the JMS broker load is on the same machine running the application server. If the JMS service integration type is set to REMOTE, processing load is transferred from the application server machine to another machine which only hosts the JMS broker or set of brokers. However, this change only affects JMS-bound applications that rely heavily on messaging. JMS Service configuration is available at Tree>Configuration>Java Message Service.

EMBEDDED GLASSFISH

Embedded GlassFish is available at https://embedded-glassfish.dev.java.net/. The embedded distribution hosts applications embedded into a larger, single JVM process. For example, use embedded distribution for unit testing. The following sample code shows how to create a server instance, start it, and deploy a Web application into the embedded instance.

```
Server server = new Server.Builder("emServer").build();
server.createPort(8080);
server.addContainer(new   org.glassfish.web.embed.impl.
EmbeddedWebContainer());
server.start();
File sampleWar = new File("sample.war");
server.getDeployer().deploy(sampleWar);

// Insert customer logic here

server.stop();
```

Using the embedded mode is as simple as the few lines above. The embedded mode is useful for unit testing and cases when Web or enterprise applications run in-process instead of being hosted in a stand-alone server.

GLASSFISH UPDATE TOOL

GlassFish v3 is a modular application server, and update and distribution mechanisms can benefit from the modular architecture. GlassFish v3 uses IPS/PKG(5) for distribution and updating. The IPS/PKG(5) is an operating system-independent, network-based distribution mechanism for applications, from a simple phone book to systems as large as operating systems like OpenSolaris.

Bootstrapping the Update Tool

During the installation process installer provides us with the option of installing and configuring the Update Tool for us. If we do select this option we can use the updatetool script located in the `gf_home/bin` to bootstrap it at any time after the installation. To start the bootstrapping we just need to run `./updatetool` or `updatetool.bat`; it will automatically detect whether the update center is installed or not and will continue accordingly.

Using Update Tool

The PKG 5 IPS provides a command line script for using IPS to install, update, upgrade and remove an application. The script is named pkg and is located inside the `gf_home/bin`. In addition to the CLI command, there is a very polished GUI named Update Tool which we can run using the `updatetool` script available in the same directory. Running the update tool will open a GUI similar to Figure 5. This shows what new updates are available for our installed modules, or what new modules are available which we many need to install. We can use the update tool and IPS to distribute our Web or Enterprise application updates, to create our own update centers, to distribute our GlassFish modules, and even to distribute our custom applications using IPS.

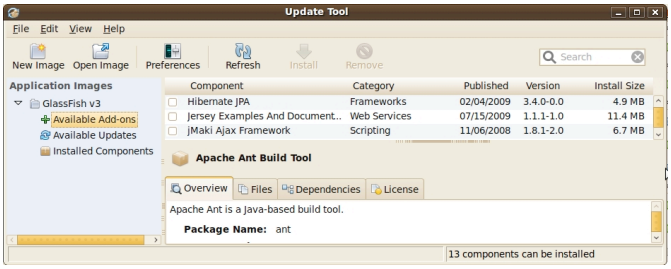

Figure 5: GlassFish Update Tool

Another good feature of the Update Tool is that it can manage multiple installation images of different applications and different versions. We just need to register new images using File menu.

Like many other applications, the GlassFish Update Tool will notify of available updates through a system tray icon. To register its system try icon we need to open a command window, navigate to `gf_home/updatetool` and then execute the following command.

```
./updatetool --register
```

It will simply sit in the system tray and let us know when an update for our installed modules is available. Imagine that we can update the Hibernate or Spring or any other framework's libraries automatically and through a notification system.

ARCHITECTURE

GlassFish v3 benefits from a modular architecture based on OSGi modularity system for platform and bundle management, and HK2 for service layer modularity. The HK2, available at http://hk2.dev.java.net, forms the GlassFish micro-kernel and is roughly based on Java Platform Modularity specification. The whole modularity in the service layer is based on the concept of contracts and contracts providers which are simply Java interfaces and interface implementations along with some annotations.

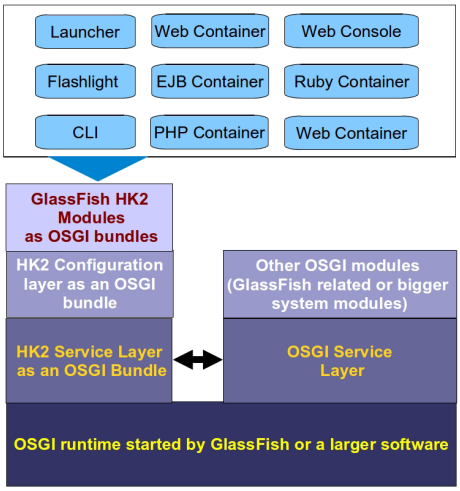

Figure 6: GlassFish architecture from Mars

GlassFish can be extended by developing new providers for contracts, and the good news is that a registry of extension information is not required; instead, all of the extension discovery and life cycle management is completely automatic. Extensions are "installed" by simply placing them in the modules directory.

Table 7: GlassFish extensibility points

Extension point	Sample use case
CLI Extensibility	Add new CLI command; for example a command to start or stop a new type of container.
Web Administration Console Extensibility	Add a whole new set of pages and nodes to administration console to address new administration requirements like administrating a new container.
Monitoring Extensibility	Export monitoring information to a specific format or from a specific container.
Container Extensibility	Add new type of container to host; for example applications developed by another dynamic language.

ABOUT THE AUTHOR

Masoud Kalali holds a software engineering degree and has been working on software development projects since 1998. He has experience with a variety of technologies (.Net, J2EE, CORBA, and COM+) on diverse platforms (Solaris, Linux, and Windows). His experience is in software architecture, design and server side development. Masoud has several articles in Java.net. He is one of founder members of NetBeans Dream Team. Masoud's main area of research and interest includes Web Services and Service Oriented Architecture along with large scale and high throughput systems' development and deployment.

Blog: http://weblogs.java.net/blog/kalali/
Contact: Kalali@gmail.com

RECOMMENDED BOOK

The complete guide to installing and configuring the GlassFish Application Server and developing Java EE 5 applications to be deployed to this server.

BUY NOW
books.dzone.com/books/glassfish

DZone, Inc.
1251 NW Maynard
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-934238-62-2
ISBN-10: 1-934238-62-7
50795
9 781934 238622
$7.95