GlassFish v3
ein Vorgeschmack auf die Application Server der nächsten Generation

Peter Doschkinow
ORACLE Deutschland B.V. & Co. KG
München

Schlüsselworte:

GlassFish v3, Java EE 6, Application Server

Einleitung

GlassFish ist der beliebte und weit verbreitete Open Source Application Server, der seit 2006 als Referenzimplementierung der Java EE Plattform von der GlassFish Community und damals Sun, heute Oracle, ausgeliefert wird. Seine Popularität ist vor allem darin begründet, dass er gleichermaßen die Bedürfnisse von Entwicklern, Administratoren und Betreibern adressiert. Er ist leichtgewichtig und trotzdem voll produktionsfähig mit ausgereifter Unterstützung für Hochverfügbarkeit und hervorragender Performance. Seine letzte Version ist GlassFish v3, die Referenzimplementierung von Java EE 6.

Die Java EE 6 Spezifikation setzt weiter den Prozess der Vereinfachung und Flexibilisierung der Java EE Plattform fort und führt mit dem Konzept von Profiles (im allgemeinen) und dem Web-Profile (im speziellen) zum ersten Mal die Möglichkeit ein, standardisierte Subsets als Plattform für die eigenen Anwendungen zu benutzen, auf unnötige Funktionalität zu verzichten und somit eine bessere Performance und Ausnutzung vorhandener Ressourcen zu erreichen.

Die hohen Anforderungen an Flexibilität und Erweiterbarkeit, die Java EE 6 mit sich bringt, haben zu einer neuen Architektur in GlassFish v3 geführt, die auf ein neues Modulsystem und OSGi basiert und dadurch neue, unerahnte Möglichkeiten für GlassFish-Anwender anbietet, die weit über Java EE 6 hinausgehen. Es ist sehr wahrscheinlich, dass auch andere Java EE Application Server Hersteller für ihre künftige Java EE 6 Implementierung einen ähnlichen Weg wie GlassFish v3 einschlagen.

Im folgenden wird auf einige der interessantesten Merkmale und Technologien von GlassFish v3 eingegangen, die ihn klar als ein Application Server der nächsten Generation positionieren.

Architektur, Modularität, Erweiterbarkeit

Die Hauptbestandteile in der Architektur von GlassFish v3 sind: der HK2-Kernel, die Dienste, die er bereitstellt und die Container, die von diesen Diensten Gebrauch machen:

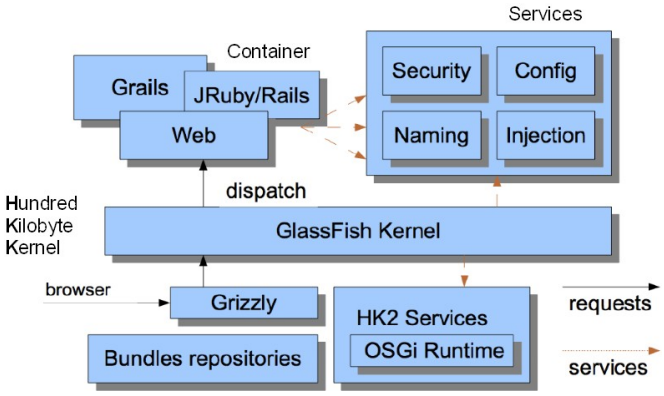

Abb. 1: GlassFish v3 Architektur

HK2 steht für **H**undred **K**ilobyte **K**ernel und das Modul-System, das GlassFish v3 intern verwendet. Das HK2 Modul-System hat eine Schnittstelle, über die andere Modul-Systeme angebunden werden können, wie z.B. über OSGi, das als Standardimplementierung mitgeführt wird. HK2 hat aber auch eine eigene OSGi unabhängige Implementierung, die vom GlassFish v3 im Embedded-Mode verwendet wird. Typischerweise ist ein HK2-Modul ein OSGi-Bundle, das zusätzliche Meta-Informationen enthält.

Die Gesamtfunktionalität von GlassFish v3 wird in der Form von Diensten und entsprechend bewährten Methoden einer Service-orientierten Architektur bereitgestellt. Diese Dienste können unterschiedliche Schnittstellen-Arten anbieten: den META-INF/services Mechanismus von Java SE, eine OSGi- oder eine HK2-Schnittstelle. Um System-Ressourcen zu schonen, wird Lazy-Loading verwendet, so dass Services nur dann geladen werden, wenn sie benötigt werden. So wird zum Beispiel der EJB-Container erst beim Deployment der ersten EJB-Anwendung hochgefahren.

GlassFish v3 kann durch die Nutzung eigener APIs oder über OSGi erweitert werden. Mann kann beispielsweise einen eigenen Custom-Container über die Extension-API von Grizzly hinzufügen, der hochperformanten, NIO-basierten Netzwerk-Protokoll-Engine von GlassFish. Dieser Mechanismus wurde für die Implementierung der nativen Jruby und Jython Container verwendet.

GlassFish v3 wird zusammen mit dem Felix OSGi Container ausgeliefert, könnte jedoch auch mit anderen OSGi Container wie Equinox oder Knopflerfish unmodifiziert laufen. Eine Besonderheit der Integration mit OSGi besteht darin, dass die Visibilität von OSGi von den GlassFish-Entwickler auf die GlassFish-Benutzer ausgedehnt wird. Java EE Anwendungsentwickler können über Annotations und Ressource-Injection transparent auf über OSGi definierte Services zugreifen, ohne jegliche Verwendung von GlassFish- oder OSGi-spezifische APIs:

```
@Resource(mappedName="checkOsgiService")
CheckService checkService;
```

Das ermöglicht die Erstellung von sogenannten „converged“ Java EE Anwendungen, bei denen OSGi zur Auflösung von Abhängigkeiten herangezogen wird. Das GlassFish Team erwägt auch die andere Richtung: künftig Java EE Komponenten und Dienste als OSGi Services zugänglich zu machen.

Einfache Entwicklung

Entwickler, die zum ersten Mal mit GlassFish v3 arbeiten, werden eine angenehme Überraschung erleben. Der Application Server ist extrem leichtgewichtig und ressourcenschonend. Das Web-Profile ist etwa 50MB, das Full-Profile etwa 80MB groß. Die Start-Up Zeit beträgt nur wenige Sekunden – auf gängige Laptops unter 10. Das hängt damit zusammen, dass die Services erst bei Bedarf hochgefahren werden und wenn noch nichts deployed ist, ist auch kein Container gestartet. Die Dokumentation ist ausführlich und übersichtlich. Es gibt zahlreiche Optionen für Management und Monitoring, wie weiter unten beschrieben. Für Testzwecke gibt es noch ein Maven Plug-in für Embedded GlassFish mit Maven-Goals zum Starten/Stoppen und Installation/Deinstallation von Anwendungen.

GlassFish bietet diverse Deployment Optionen. Zu den weniger bekannten gehören automatisches Deployment, Dynamic Reloading und Deployment von Java EE Application Clients über Java Web Start. Eine Neuheit für Entwickler ist die sogenannte „Keep-Session-on-Redeploy“ Feature. Wenn konfiguriert, serialisiert GlassFish die User-Sessions zwischen Redeployments, so dass sie erhalten bleiben, selbst wenn die Anwendung inzwischen modifiziert wird.

Der Spaß mit GlassFish kommt aber erst dann richtig auf, wenn man dabei eine geeignete Entwicklungsumgebung verwendet. Für diejenigen, die mit den neuesten Java EE 6 Features arbeiten wollen ist NetBeans 6.9.1 die beste Wahl. In NetBeans kann man leicht GlassFish integrieren, verwalten und debuggen. Der hervorragende Support von Java EE 6 in der Form von Code-Completion und Wizards erleichtert den Einstieg und steigert die Produktivität. Gute Unterstützung für GlassFish v3 und Java EE 6 bietet auch das Oracle Enterprise Pack für Eclipse, das Tools Bundle für Eclipse und IntelliJ IDEA.

Embeddable GlassFish

GlassFish v3 bietet die Möglichkeit, eingebettet in einer beliebigen Java Anwendung zu laufen. Dafür gibt es APIs, über die man den Application Server verwalten kann: z.B. starten, stoppen, konfigurieren, Anwendungen installieren und deinstallieren. Er kann einfach als Bibliothek und Bestandteil einer Anwendung ausgeliefert werden, was ihn für OEM-Hersteller sehr interessant macht. Für solche Szenarien wird GlassFish in der Form von einem JAR-File, jeweils für das Web- oder Full-Profile, oder alternativ dazu als Maven Plug-in bereitgestellt. Das Maven-Plugin vereinfacht stark den Build- und Test-Prozess für Anwendungen, weil man dafür alles auf dem selben Rechner abwickeln kann und nicht mal eine GlassFish Installation braucht. Das folgende Code-Segment skizziert einen JUnit-Test, bei dem eine Anwendung getestet wird, nach dem sie auf GlassFish in Embedded-Mode deployed wurde:

```
@BeforeClass public static void initContainer() {
   Server.Builder builder = new Server.Builder();
   Server server = builder.build();
   ContainerBuilder b =
              server.createConfig(ContainerBuilder.Type.web);
   server.addContainer(b);
   ...
   File archive = new File("hello.war");
   server.getDeployer().deploy(archive);
}

@Test public static void pingApplication() {
   ...
}
```

Management und Monitoring

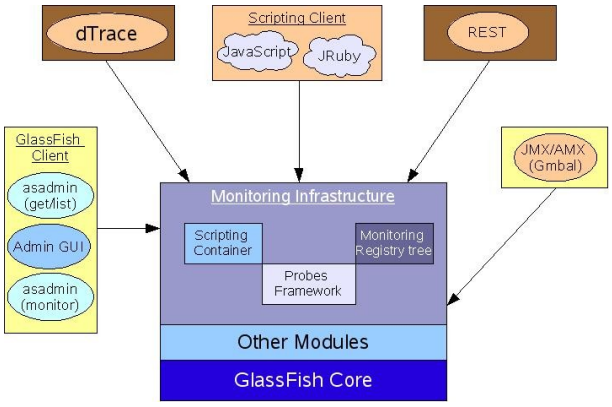

Abb. 2: GlassFish v3 Monitoring

Es gibt umfangreiche Möglichkeiten für Management und Monitoring von GlassFish v3 – über die Admin-Konsole, ein intuitives und gut durchdachtes Web Interface, über das Command Line Interface, über Ant-Tasks, über JMX oder die AMX-API, eine GlassFish spezifische API, die den Umgang mit JMX und den Mbeans von GlassFish vereinfacht.

Eine sehr interessante Möglichkeit für Management und Monitoring bietet das neue REST-Interface, das den Einsatz von GlassFish in Cloud-Umgebungen erleichtert. Jedes Konfigurationselement von GlassFish entspricht einem Knoten im Objekt-Graph einer GlassFish-Konfiguration, der als eine REST-Ressource über den URL `http://host:port/management/domain/<path-in-object-tree>` exponiert ist. Abhängig von dem Knoten-Typ ist die Zulässigkeit der HTTP Methoden GET, POST, DELETE, die zum Auslesen oder Manipulation des Konfigurationsgraphen dienen. Das REST-Interface macht es möglich, REST-basierte Management und Monitoring Clients für GlassFish v3 auch in anderen Sprachen zu schreiben. Das folgende Beispiel zeigt, wie man das bekannte cURL Utility benutzen kann, um über das REST-Interface eine JDBC-Ressource in GlassFish v3 zu deaktivieren:

```
curl -X POST -d "enabled=false" http://localhost:4848/management/
domain/resources/jdbc-resource/jdbc-sample
```

Das Montitoring in GlassFish v3 wurde komplett überarbeitet und erweitert. Es setzt auf ein BTrace-basiertes Framework auf, bei dem dynamisch und nicht-intrusiv von jeder GlassFish Runtime-Klasse Events generiert werden können. Die Events werden dann von dynamisch registrierten Listener gesammelt, statistisch ausgewertet und über diverse API für verschiedene Monitoring Clients bereitgestellt. Dank BTrace sind die BTrace-Probes, die als Ereignis-Quelle dienen, dem DTrace-Framework unter Solaris zugänglich. DTrace ist ein umfangreiches Framework und Tool für dynamisches Tracing, das im Solaris Kernel verankert ist, und non-intrusive die gleichzeitige Untersuchung von Anwendungen und Betriebssystem im laufenden Produktionsbetrieb ermöglicht. Mit DTrace kann man sich durch dynamische Instrumentierung ein komplettes Bild vom gesamten Produktionssystem(Anwendung + Betriebssystem) machen und gegebenenfalls schnell Performance- und sonstige Problem-Quellen lokalisieren. Mit anderen Worten: das neue Monitoring-Framework in GlassFish v3 ermöglicht eine feingranulare Analyse des Application Servers durch das mächtigste

Tracing-Tool unter Solaris und leistet dadurch einen großen Beitrag für höhere Verfügbarkeit und Qualität seiner Dienste in der Produktion.

Unterstützung für Scripting

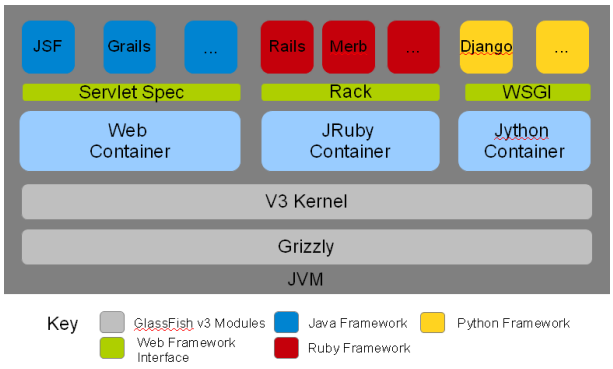

Abb. 3: Scripting Frameworks auf GlassFish v3

Dynamische Scripting-Sprachen wie Jruby, Groovy, Jython und Scala erfreuen sich zunehmend einer wachsenden Popularität und Verbreitung. Die Gründe dafür sind vielschichtig, aber zu den wichtigsten gehören die hohe Produktivität, der leichte Einstieg, die mächtigen und leicht verwendbaren Scaffolding-Frameworks, die sie begleiten und nicht zuletzt die pragmatische Überlegung, dass sie sich für ein bestimmtes Projekt als schnell und gut genug erweisen.

GlassFish v3 hat deutlich die Unterstützung von dynamischen Scripting-Sprachen verbessert und bietet native Container für Jruby und Jython. Weiterhin unterstützt GlassFish Groovy/Grails, Scala/Lift und PHP. Scripting Anwendungen, die auf GlassFish deployed werden, profitieren zunächst von der Infrastruktur eines Java EE Application Server – bessere Skalierbarkeit durch die Verwendung von Thread- und Datenbank-Connection-Pools, bessere Sicherheit und Wiederverwendung seiner Management und Monitoring Features. Vom großen Vorteil erweist sich aber auch die modulare und erweiterbare Architektur von GlassFish v3, da sie die Anbindung von anderen Container und die Erweiterung der Admin-Console, z.B. für Jruby-Spezifika, erleichtert. Über das GlassFish Update Center kann zu jedem Zeitpunkt eine bestehende GlassFish Installation mit den notwendigen Module für die gewünschte Scripting-Unterstützung nachgerüstet werden.

Zusammenfassung

GlassFish v3 ist ein sehr leichtgewichtiger, erweiterbarer, embeddable Application Server, dessen Unterstützung für Web 2.0 Anwendungen sich über Java hinaus auch auf andere moderne Scripting-Sprachen ausdehnt. GlassFish v3 bietet seinen Benutzern den vollständigen Support für Java EE 6, die benutzerfreundlichste, flexibelste und vollständigste aller bisherigen Java EE Plattformen. Mit GlassFish v3 können Unternehmen zwischen dem leichtgewichtigen Web-Profile und dem mächtigeren Full-Profile als standardisierte Laufzeitumgebungen für ihre Anwendungen wählen. Darüber hinaus können sie von den zusätzlichen GlassFish v3 spezifischen Produktivitätsvorteilen für Entwickler und Betreiber profitieren und somit die Komplexität und Kosten ihrer IT-Infrastruktur senken.

Kontaktadresse:

Peter Doschkinow
ORACLE Deutschland B.V. & Co. KG
Sonnenallee 1
D-85551 Kirchheim-Heimstetten

Telefon: +49 (0) 89-46008 2164
Fax: +49 (0) 89-46008 2666
E-Mail peter.doschkinow@oracle.com
Internet: www.oracle.com