GlassFish 社區奉獻的 Java EE 應用伺服器

Eduardo Pelegri-Llopart
Yutaka Yoshida
Alexis Moussine-Pouchkine
Sun Microsystems, Inc.

http://blogs.sun.com/theaquarium

最後更新 2007 年 9 月

在 JavaOne 2005 上，Sun 宣佈了 GlassFish 項目，這是一個關於開源的應用伺服器和 Java EE 的參考實現的項目(參看：GlassFish 項目)。這邁出了所有 Java 平臺開源的第一步，此外它還有著更多的意義。GlassFish 專案加速了 Java EE5 的採用，為開源社區在具備企業品質的應用伺服器方面增添了新的選擇，並促使 Sun 在應用伺服器產品的開發、測試、提供和改進方面進行方式轉變，以提供更優秀的產品。

在專案啟動一年半後，GlassFish 社區已經發佈了它的第一個正式版本，並且正在開發其第二個版本。這篇文章裏我們將就 GlassFish 社區的各個方面和 GlassFish 應用伺服器進行總體概述。

GlassFish 項目

Sun 於 2005 年 6 月 JavaOne 期間啓動了 GlassFish 項目，開源的 Java EE 5 參考實現和 Sun 的應用伺服器。在 JavaOne 2006 (2006 年 5 月)，不到一年的時間內就發佈了其第一個版本。第二個版本於 2007 年 9 月發佈。目前社區已將開發重點放在 GlassFish v3－羽量級的模組化的應用伺服器上。

什麼是 GlassFish

GlassFish 是一個社區和一個應用伺服器產品。GlassFish 社區提交的主要是與 Java EE 5 相容的應用伺服器－GlassFish 應用伺服器，以及 Java Persistence API 參考實現－TopLink Essentials。

GlassFish 社區提交了許多符合 Java EE 5 的 JCP 規範的可重用的元件，這其中包括 JAXB, JAX-WS, JAXP, StAX, JSP 和 JSF。GlassFish 還包括了幾個流行的 Web.Next 項目比如 jMaki, Phobos 和 dynaFaces，還有許多工具比如持續化的集成工具 Hudson，以及有用的架構比如基於 NIO 伺服器框架的 Grizzly。GlassFish 社區還為所有這些元件維護著一個 Maven 存儲庫(repository)。

Java 平臺和應用伺服器

Sun 於 1995 年發佈 Java 並於次年的第一屆 JavaOne 大會上展示了 Java Servlet API。到了 1999 年，Java Server Pages 和 Enterprise JavaBeans 連同 Java Servlets 被組裝到第一個 Java 企業平臺 J2EE 1.2 - 其版本號是為匹配相應的 J2SE。接下來發佈的版本是 2001 年的 J2EE 1.3 和 2003 年的

J2EE 1.4，在2006年，這一平臺的名稱最後變更為Java EE 5。

Java EE規範被廣泛地被包括Sun在內的軟體提供商和企業所接受，並在他們的應用伺服器中加以實現。多年來，Sun將參考實現(參看：什麼是參考實現)和商業的應用伺服器標成不同的品牌發佈。這兩者有著不同的用途：RI專注于開發和傳授規範，可以免費開發但不能在真正的生產環境下使用，但商業產品面向市場，並且側重在部署後的使用。7.0商業版(帶有SunOne的商標)第一次包括了J2EE 1.3 SDK的功能，自此兩者被統一起來，在其後以發佈Sun Java System商標發佈的8.x 和9.x延續了這一做法（參看：Sun的應用伺服器)。

Sun的應用伺服器

Sun爲其應用伺服器打上了多個商標。近期的主要發佈如下：

- iPlanet AS 6.0 –源自Netscape (Netscape AS)，相容J2EE 1.2
- SunOne AppServer 7.0 – J2EE 1.3參考實現中的第一個版本
- Sun Java System AppServer 8.0 –相容J2EE 1.4
- Sun Java System AppServer 9.0 –相容Java EE 5，GlassFish v1
- Sun Java System AppServer 9.1 –相容Java EE 5， GlassFish v2

到了J2EE 1.4，其參考實現和SJS AS 8.0 PE是相同的，可自由開發和部署，並且包含在為廣泛分發的J2EE 1.4 SDK中。

SJS AS 8.x產品提供3種不同的版本：Platform Edition (PE) 和 參考實現相同, Standard Edition (SE) 包括一些企業品質的特性比如集群Clustering和Failover, Enterprise Edition (EE) 提供給要求高可用性(99.999%的可用性）的市場。從9.x版開始，所有特性都被集中到單一的版本，這就是GlassFish社區在開源許可下所提供的版本。用戶可以自由地設定GlassFish所用的3個profile：developer, cluster和enterprise。

GlassFish 應用伺服器

2005年6月在JavaOne 2005上，Sun公司宣佈啓動GlassFish專案，不到一年的時間，2006年5月即下一個JavaOne大會上，GlassFish的第一個版本面世並且JavaEE 5的規範也正式推出。

這個應用伺服器產品可以在Sun網站通過下載Sun Java System AS 9.0 PE獲取，也可以在GlassFish社區通過下載GlassFish v1獲取，除了安裝程式，兩者是相同的。

GlassFish v1側重於Java EE 5規範的實現，一些企業級的特性並沒有包含在

這一版本中。此應用伺服器在 Sun 發佈中被標爲 PE，就反映了這一點。GlassFish v2 加上了所有企業級的特性，並且去除了 PE 標籤，同一個可執行檔可根據所期望的用途安裝成爲 developer，enterprise 或者 cluster 的 profile。

GlassFish v2 可以在 Sun 網站通過下載 Sun Java System AS 9.1 PE 獲取，它發佈於 2007 年的 9 月。

開源許可

OSI 組織(Open Source Initiative) 承認多種的許可。許可之間通常有著某種繼承關係。

ASL 許可出自 BSD 和 MIT 許可，它的使用非常靈活。

GPL, LGPL, GPLv2 和 GPL+CPE 都是 GNU 許可。 GNU 許可強調保留原始代碼的"自由性"。GPL 許可是基於項目的。

CDDL 許可派生自 MPL(Mozilla Public License)。不同於 GPL 許可，這些許可是基於檔的。

學習開源許可的好地方是 OSI 組織的網站(opensource.org)。

Java 平臺

Java EE 規範是一個 umbrella 規範，構建在相關 Java SE 規範（即 Java EE5 需要 Java SE 5）和許多其他 JCP 規範之上。GlassFish 社區反映了這點，並且包括了許多子專案作爲這些規範的官方參考實現。通過這些子專案，可以更容易被其他組織和專案重用。例如，JAXB 的實現被用在許多項目，這些項目大部分和 Sun 公司無關。

Java 標準介面是專家組通過 Java 社區流程(Java Community Process)開發的，許多專家組的領導本身就是 GlassFish 社區的成員。GlassFish 社區實現了這些介面，並在 JCP 的審核過程中提供回饋並對將來的版本提出需求。

關於 Java EE 6 的工作已經通過 JSR 316 展開，這是個 umbrella 專案，包括比如 EJB 3.1，JPA2.0 等規範。GlassFish v3 將實現 Java EE 6 規範。

開源和 GlassFish

Tomcat 是被廣泛採用的 Java 伺服器端項目。Tomcat 由 Apache 通過包括 Sun 和 JServ 開發人員構成的工作組發起，並且成為 Java Servlets 和 Java Server Pages 規範早期版本的參考實現(GlassFish 現在是這些規範最新的參考實現)。Tomcat 對於早期伺服器端的 Java 技術被認可和採用起到了關鍵的作用，因為它在開源許可下提供，它也為開源軟體在企業間的流行作出了貢

獻。

有許多不同的許可（參看“開源許可”），這裏並不打算詳細地描述它們。各類有著不同特性的許可吸引著不同的社區：許可中被某社區認為是優點的特性可能被另一個社區認為是缺點，例如，Apache 社區使用 ASL 許可而 GNU 社區使用 GPL 許可。

當 GlassFish 發佈之初，僅使用 Common Development and Distribution License(CDDL)，但是為了配合 Java SE 和 Java ME 在 2006 年 11 月所開始的開源， Sun 重新評估了來自不同社區的請求並決定對 Java ME 採用 GPLv2 許可，對 Java SE 採用 GPLv2 加 ClassPath 異常 (GPLv2+CPE)的許可。使用 GPL 許可增強了 GNU 的 ClassPath 和 Kaffe 社區，而 CPE 允許鏈結非 GPL 的構件。這樣，GlassFish 社區在 CDDL 的基礎上添加了 GPLv2+CPE 許可，成為雙許可。雙許可意味著可以使用其中任一個許可，這樣可以在為 GlassFish 的商業夥伴保留 CDDL 的優點的同時，增強和 GNU、Java SE 和 Java Me 社區的合作。在 GlassFish 所分發的軟體中不存在二進位許可。

採用 GlassFish 和 Java EE 5

Java EE 5 中的 WS Endpoints

下面的代碼顯示了一個 WS end point:

```
import javax.jws.WebService;
@WebService
public class MySimpleWS {
   public String sayHello(String s) {
       return “Hello” +s;
   }
}
```

這裏能夠如下面的代碼所示轉換一個 (transactional) Stateless Session Bean：

```
import javax.jws.WebService;
import javax.ejb.Stateless;
@WebService
@Stateless
public class MySimpleWS {
  public String sayHello(String s) {
     return “Hello” +s;
  }
}
```

開源的參與、產品級的品質和 Java EE 5 的參考實現使得 Java EE 5 的需求和供給都得到增加，從而加速了這個平臺的採用。此文寫作之時，GlassFish 已經有超過 1 百萬次的下載，這些從事開發和部署的人員感受到了 Java EE 5 所帶來的好處並要求增強它所支持的特性。GlassFish 應用伺服器滿足了其中一些要求，同時它還在幫助其他的 Java 應用開發商使用 GlassFish 的 具有產品品質的元件來更快且花銷更小地完成一個 Java EE 5 相容的實現。

使用 GlassFish 的應用開發商包括：TmaxSoft，Oracle，BEA，Jboss ，Jetty 和 Geronimo。TmaxSoft 是韓國最早獲取 Java EE 許可的公司，他們的最新產品 JEUS 6 使用了 JAXB， JAX-WS， Fast Infoset ， JSP ， JSF 和 TopLink Essentials。通過參與 GlassFish 社區，

TmaxSoft 提前獲取了許多元件，並確保這些元件可以很好地滿足他們的要求。這使得 TmaxSoft 的 JEUS 6 成為在 Sun 之後第一個通過 Java EE 5 認證的應用伺服器。

Oracle 尚沒發佈完全支持 Java EE 5 的應用伺服器，但他們為 GlassFish 社區貢獻了 TopLink Essentials 項目，並將其應用在自己的產品中。在上面提到的提供商中，BEA 使用了 JAXB 和 JAX-WS 技術，JBoss 使用了 JAXB 和 JSF 技術，Jetty 使用了 JSP 和 Grizzly 技術，Geronimo 使用了 JAXB 技術。

資源注射

J2EE 1.4 需要相當多的 template code。下面是典型的 EJB 引用：

```
Context initial = new InitialContext();
Context myEnv = (Context)initial....
Object objref =myEnv.lookup("...");
ConverterHome home = (ConverterHome)
PortableRemoteObject.narrow(...);
Converter currentConverter = home.create();
```

在 Java EE 5，通過約定和資源注射，我們只要說：

```
@EJB Converter currencyConverter;
```

Java EE 5 概覽

簡化開發是 Java EE 5 的主旋律，而主要實現這一目標的方法是借助 Java SE 5 的 annotations 技術使得可以基於 POJO(Plain Old Java Objects)進行編程。Annotation 用於許多方面包括方法和類的屬性(參看：Java EE 5 中的 WS Endpoints)，資源注射(參看：資源注射)，以及可移植的行為描述(參看：缺省的 annotation)。

主要的 Java EE 5 規範如下：

- JAX-WS 2.0 和 JAXB 2.0
- EJB 3.0 和 Java Persistence API
- JSF 1.2 和 JSP 2.1
- StAX

其他一些規範 (比如 JAXP 和 Servlet)也有變化相對較小的維護和發佈。這些變化在開發體驗上所帶來的總的效果是非常顯著的。

Java EE 5 和 GlassFish 緊密關聯：Java EE 5 來自規範的力量增加了 GlassFish 的價值，同時獲取 GlassFish 可幫助驗證 Java EE 5。我們希望將來能夠在 Java EE 平臺中保持二者的這種關係，為規範的不斷改進提供更有

效的回饋，通過早期獲取參考實現以促進社區更早地嘗試規範。

我們正在嘗試如何使用 annotation。例如，JAXB 2.0 規範由兩部分構成。一部分是描述如何映射 XML Schema 到 annotation，另一部分是描述這些 annotation 的語義。這意味著通過修改生成的代碼，仍可以將應用移植到相容的應用伺服器上運行。特殊情況下，需要插入和增加代碼以調用 setters 和 getters 方法，使之可以移植。

GlassFish 發佈

GlassFish 有 3 個處在不同開發階段的版本，它們是：V1，V2，V3。目前，關於 GlassFish 發佈的幾個週期性階段如下所示(目前仍是社區的早期階段，相關的定義仍將不斷完善)：

- 概念創建(Concept Creation) - 收集關鍵特性，粗略的時間表，原型。
- 活躍開發(Active Development) - 為里程碑和發佈正式版本進行實現。
- 維護(Maintenance) - 為正式版本進行錯誤修訂，創建更新版本。

GlassFish v1 目前處於維護階段，GlassFish v2 剛剛發佈並進入維護階段，而 GlassFish v3 正從概念創建轉向活躍開發階段。它將可能在實現其模組化特性後發佈。

GlassFish v1

GlassFish v1 目前處於維護階段。其最後一個版本發佈於 2006 年五月，正值 2006 年 JavaOne 大會之前，之後又發佈於 2006 年 10 月發佈了錯誤修訂的 v1 UR1 並於 2006 年 12 月發佈了第 2 個改動更小的 v1 UR1p1 。目前 v1 沒有更多的維護版本發佈的計畫，因為主要的開發已經轉移到 v2 上來。

GlassFish v1 也被作為官方正式的 Java EE 5 的參考實現。

同時，Sun 公司將 GlassFish v1 以 Sun Java System AS 9.0 PE 的名義分發，並將它包含在其他一些 Sun 公司所分發的包裝中，這其中包括 Java EE SDK, Java Application Platform SDK 和 NetBeans 5.5 Tools Bundle。GlassFish v1 和 SJS AS 9.0 PE 僅有的不同就是安裝程式。Sun 提供不同層次的商業支援給 GlassFish v1 (SJS AS 9.0 PE)的最後版本，並由社區來全力提供支持良好的支持。

GlassFish v2

GlassFish v2 的正式版本於 2007 年 9 月發佈。這個版本中最主要並且是最重要的特性是 clustering(包括 grouping, loadbalancing,data replication)。

GlassFish v2 繼承了 SJS AS 8.2 SE/EE 所有企業級的特性並經過 Java Enterprise System 中所有中間件產品的測試。GlassFish v2 支援 profile 的概念並支援將同一個可執行應用配置成 developer, enterprise 或是 cluster 的 profile。enterprise 的 profile 也可以被配置成使用 HADB (High Availability Data Base) 來達到高可用性 (99.999%)。

Sun 將 GlassFish v2 標以 Sun Java System AS 9.1 對外發佈並發佈在一系列的套裝軟體中(比如 Java ES 5.1)，並為其提供商業的和社區的支援。

GlassFish v3

關於 GlassFish v3 的演示在 JavaOne 2007 上得到廣泛關注。它缺省採用模組化(modular)架構，內核特別小（小於 100Kb，這使得它也適用於桌面甚至移動應用)，它的啟動時間少於 1 秒。容器被多個模組化的內核所支持，目前已集成有 Java Web, PHP (通過 Caucho 的 Quercus), jRuby on Rails, Phobos JavaScript。更多的模組將會推出，也歡迎你來開發自己的模組。

GlassFish v3 的預覽版本位於 http://glassfish.java.net ，相關文檔位於 http://wiki.glassfish.java.net 和 http://hk2.dev.java.net。

GlassFish v3 上的開發正在進行。相關計畫尚在制定中，其開發的模組化特性很可能使之發佈成一個在 Java EE 6 上完整提供特性的產品。

GlassFish v2 的特性

這一節描述了 GlassFish V2 的特性。對許多特性只是作了簡要的概述，更多的資訊將提供在網上。

Web 層是許多企業應用的重點，GlassFish 提供了豐富的 Web 層支援。

Web 層的 Java 規範

GlassFish 支援最新的 JCP 規範: JSP 2.1, JSF 1.2, Servlet 2.5 和 JSTL 1.2。

Java Server Faces(簡稱 JSF, 下一代基於標準的 MVC 架構) 同樣被加入到 Java EE 5 平臺中，JSF 在 Java 表現層提供了一個元件模型。JSF 能夠和 JavaServer Pages (JSP)或其他的如 facelets 一起使用。JSF 1.2 對早期版本做了很多改進，尤其是在統一運算式語言(unified expression language）方面。統一運算式語言目前已經可以被 JSF 和 JSP 所共用。JSF 1.2 還包括了一些其他方面的改進，比如請求的生命週期(request cycle)以及對 AJAX 的支持。

JSP 2.1 主要的改進是在統一運算式語言方面，Servlet 2.5 和 JSTL 同樣有新的變化但變化並不大。

GlassFish 還包括了不少在規範之外的重要改進。

GlassFish 的 JSF 實現在性能上有了顯著的提高並且解決了一些關鍵問題。其 JSF 的實現方法被 JBoss 和其他一些組織採用。Servlet 容器的實現源自 Tomcat，但現在由於穩定性和性能方面的原因，GlassFish 在獨立地維護它。

GlassFish 另一個顯著的改進發生在 Jasper(JSP 編譯器)上，它能夠利用 Java SE 6 中的編譯器介面 (JSR-199)來避免檔的讀寫並且編譯得更快 (據稱 10 倍以上)。Jasper 能夠被配置成使用 Eclipse 的 JDT 編譯器，雖然不如使用 JSR-199 時快 。JSF 的性能已經得到充分提高，雖然目前還沒有其 benchmark 的結果。

GlassFish 開始使用 Woodstock 專案中的高品質的 JSF 元件，許多開發人員對此感興趣。這些組件已於 http://woodstock.dev.java.net 開源。

性能提高

GlassFish v2 的性能在很多方面得到了提高。這其中包括 Web 層的 JSF, JSP 和 Grizzly，Web 服務方面的 JAXB 和 JAX-WS，以及 CORBA 和 EJB。

所有這些創造了 GlassFish v2 在 SPECjAppServer 2004 上的世界記錄。2007 年 7 月，Sun 宣佈了在 T2000 伺服器上創造的 883.66 JOPS@Standard 的位居第一的處理能力。這個結果比 GlassFish V1/SJSAS 9.0 快了 60％，比 BEA WebLogic 快了 10%，比 IBM WebSphere 6.1 快了 30％！另一項 benchmark 的結果(使用 PostgreSQL 資料庫獲得 813.73 JOPS@Standard 處理能力)顯示出 3 倍於 Oracle 和 HP 分數的 rice/perf ration。

很明顯，**你不須被迫在開源軟體和企業級特性中間作出選擇：你可以同時擁有二者**。

*說明：*SPEC 和命名爲 SPECjAppServer 2004 的 benchmark 被註冊爲 Standard Performance Evaluation 公司的商標。Sun Fire T2000 (1 芯 8 核) 1.4ghz 883.66 SPECjAppServer2004 JOPS@Standard。上述 benchmark 比較結果於 07/10/07 發佈於 www.spec.org。要獲得最新 SPECjAppServer 2004 benchmark 結果，請訪問 http://www.spec.org/.

提高 GlassFish 的啓動性能被予以特別的重視。通過簡化啓動和關閉的設計，分析啓動時的依賴關係，將非必須的服務的初始化時間被減少到最小。啓動和關閉的時間無論是對單個實例還是整個集群都被顯著地減少。

和流行架構的相容

多年來 Java 社區的核心力量之一就是它的多樣化，並且這種多樣化特別明顯地體現在 Web 層各種形式的框架上。這些框架有效地提高了開發效率。GlassFish 社區一個直接目標就是確保這些框架在 GlassFish 的不同版本上工作良好。類似的，一些流行的應用也被確保和 GlassFish 相容。

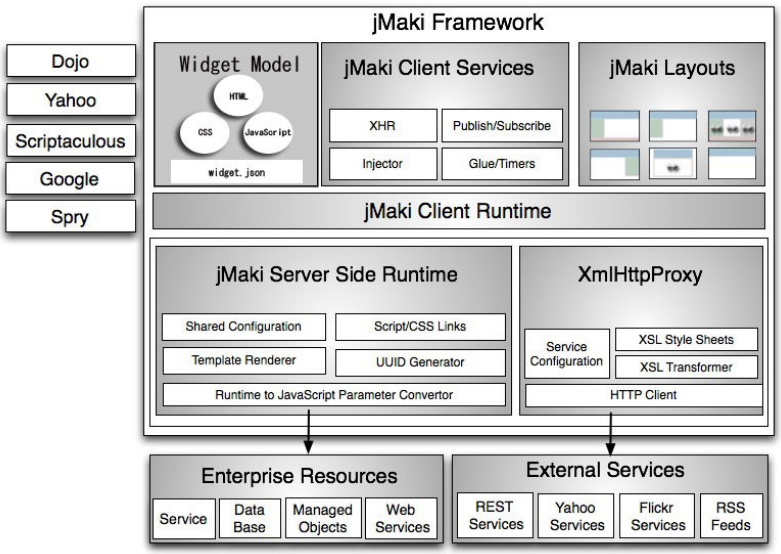
所相容的框架的清單非常龐大且在不斷增長，以下只是一個小的子集：

AppFuse, DWR, Facelets, IBATIS, JBoss Seam, Shale, Spring, Struts, Tapestry,WebWork, Wicket 等。

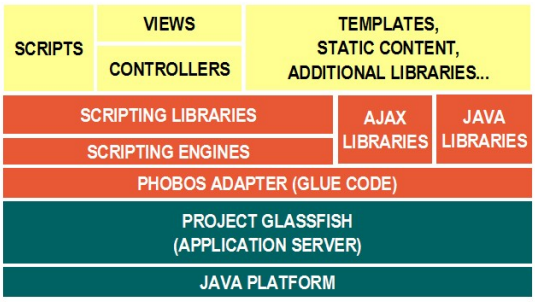
AJAX 支持

GlassFish 社區裏有許多專案是關於 AJAX 的，其中多數專案並不僅限於在 GlassFish 上使用，也有一些是依賴於 GlassFish 特有的擴展。

jMaki(ajax.dev.java.net)是一個提供了羽量級模型的框架，它支援使用 Java、PHP 和 Phobos 語言來創建以 JavaScript 為核心的支援 AJAX 的 web 應用。JMake 的模型基於 JavaScript、HTML 和 CSS，並且可以被方便地修改。jMaki 和許多 AJAX 的元件庫一起工作，包括 Dojo，Yahoo，Scriptaculous，Google 和 Spry 並可以方便地添加新的元件庫。jMaki 不太依賴伺服器端的模型，容易被採納和集成。



Phobos (phobos.dev.java.net)是一個羽量級的適合腳本的運行在 Java 平臺上的 web 應用環境，致力於滿足新興(emerging)開發者的需求。Phobos 的出發點是方便 JavaScript 的使用但設計時同樣考慮了其他動態語言的支援。比起 jMaki，Phobos 是一個更加雄心勃勃的專案，雖然只是早期的開發階段但它已引起廣泛的關注。通過伺服器端和用戶端的 JavaScript 腳本借助 JSON 和代碼重用來簡化內容傳輸。其 Java 腳本引擎 Rhino 非常堅實，並提供有 E4X 擴展和 bytecode 相容性。



Phobos Architecture

GlassFish 還擁有一些和 AJAX 相關的 JSF 項目，包括 Woodstock(一組包括了 AJAX 元件的 JSF 元件)和 Dynamic Faces(高品質將 AJAX 和 JSF 集成)。此外，還有象 DWR 和 AJAX4JSF 這樣的項目支持 AJAX。

Grizzly

GlassFish 的 Web 層的底層實現採用了 Grizzly 架構 (grizzly.dev.java.net)。這個架構用 Java 寫成，利用了 NIO 介面(可擴展的網路和檔 I/O)來提供擴展性，它高度可定制而且相當通用。Grizzly HTTP 連接器的角色類似 Tomcat 中的基於 Java 的 Coyote 連接器和基於 C 的 Apache Portable Runtime (APR)。一些初步的非正式的測試已經顯示出其具有很好的擴展性。

Grizzly 受到許多人的推崇因爲它通過利用 NIO 使得自身的使用非常簡單，比如， AsyncWeb 和 jetty 都在原型中集成了它。Grizzly 正在被改造得更加通用，以提供進一步的功能來滿足 GlassFish 社區內外各類組織的要求。

Grizzly 能夠被剝離出 GlassFish，單獨使用，例如它被 NetBeans 的 Phobos and Ruby 的 plugins 所使用的方式。Grizzly 的可擴展性能夠被用來爲“長時間持續的 HTTP 連接(long-lasting HTTP connections)”提供有效的支援，這被稱作是 Comet。Comet 技術可使用在比如聊天、線上日曆或文檔共用之類的應用中，持續地完成內容更新。

表現層的替代

GlassFish 社區一個基本的旋律就是擁抱多樣性。這樣做的理由有許多，從“競爭出品質”到“一個尺寸無法適用所有”到“創新隨處發生”。GlassFish 對於表現層上 Java 或其新的替代平臺上的典型架構所作的支持突出地體現出這一點。在某些情況下，這些替代平臺直接駐留在 GlassFish 應用伺服器上，但另一些情況下，它們和應用伺服器的關係不那麼緊密。

如前所述，伺服器端的 JavaScript 是由 Phobos 專案來支持的。在這種情況下，伺服器端的 JavaScript 代碼通過 Rhino 解釋器在 GlassFish 上執行。

基於 Ruby 的應用，比如，最著名的 Rails, 能夠通過兩個不同的方式執行。Rails 在 jRuby (jruby.codehaus.org)上執行，是基於 Java 平臺的方式。另一種方式，Rails 通過 native 的 Ruby 解釋器來執行， 並通過 CGI 介面來和 GlassFish 通訊。在 GlassFish 上運行 Rails(Rails on GlassFish)是特別有吸引力的想法而且得到了積極的探討，請繼續關注這方面的發展。

PHP 也能夠作爲外部的 native 解釋器在 GlassFish 執行。Quercus(由 Caucho 開發)是基於 Java 平臺的 PHP 5 實現，Caucho 和 GlassFish 社區正在一起致力於使得 Quercus 能在 GlassFish 上工作。

Web 服務 – *Metro*

GlassFish 的 Web 服務棧被稱作“Metro”。GlassFish v1 通過新的 JAX-WS 2.0 API(jax-ws.dev.java.net)在簡化 Web 服務開發方面作出巨大的改進。GlassFish v2 繼續這方面的改進，並支援 JAX-WS 2.1 規範 ，但這些改進主要來自多年來的演變和幾個重新的設計。Metro 的實現是經得起考驗的，其完整功能在基於 AXIS 2 stack (見下圖)的 benchmarks 測試中有 30%-100% 的性能提高。我們相信它是業界領先的技術。JAX-WS 2.0 同樣在 Sun 的 Java SE 6 參考實現中提供並且能夠升級到 JAX-WS 2.1。

SJS AS 8.2 有著對 JAX-RPC 1.1 的實現。JAX-WS 2.0 的初步實現在 GlassFish v1 中被使用，它鬆散地基於 JAX-RPC 1.1 之上，但有著兩方面的變化，這兩方面的變化是爲了應對兩個主要的挑戰。第一個挑戰是由新的規範帶來的：不同於 JAX-RPC 1.1，JAX-WS 2.0 將所有的資料綁定交由 JAXB 2.0 處理並且嚴格地支援非 HTTP 的協定。另一方面，Sun 意識到 XML 文本編碼上所帶來的性能損耗並且開始參與制定高效 XML 編碼的各類標準 (參看: Fast Infoset)。

結果是 JAX-WS 的實現被重構，它被清楚地分離成傳送(Transport),編碼(Encoding)和資料綁定(Data Binding)。這一新的 Metro 架構被 GlassFish v2 集成並且支持以下特性：

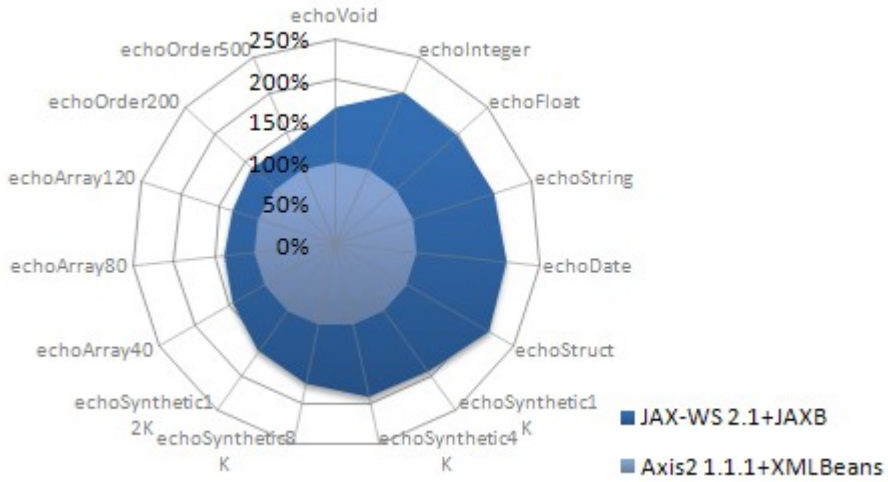
- 多種傳送(Multiple Transports)–實現基於 HTTP，JMS 和 TCP/IP 協議上的傳送並且對 SMTP 協議進行了設計考慮。
- 多種編碼(Multiple Encodings)–支援文本(Textual) XML ，MTOM 和 Fast Infoset 等多種方式編碼
- 資料綁定(Data Binding)–由 JAXB 控制資料綁定，小心地處理資料移交(handover)以避免資料拷貝。

這個架構被設計成支持多個可疊放(stackable)的處理實體，每個實體實現一個 WS-*規範，而不會導致資料拷貝。

其架構見下圖：


invoke
return
Stub
Message
WS-TX
WS-RM
WSA
security
HTTP
Message
01010001...
11001010...

性能也是一個目標，JAX-WS 2.0 的實現性能比原先的 JAX-RPC 1.1 棧(曾被認爲具有高性能和高競爭性)高出一倍。關於最近的 JAX-WS 2.1 棧和 Axis 2 性能測試的比較結果顯示 GlassFish 團隊提供了業界最好的棧。



然而性能並不是考量生產力的唯一要素，在某種程度上擴展性更加重要，業務集成將要求能夠處理成千上萬的事務，其中一些是長時間啓動的(long-lived)。通過任務共用執行線程池，只需很少的線程就可完成大量的事務處理。

WSIT 和項目 Tango

Web 服務的需求之一就是搭建橋樑，實現 Java 平臺和 Microsoft 平臺之間互操作。Tango 項目是 GlassFish 社區發起的關於這類互操作的方案，其正式名稱是 WSIT(Web Services Interoperability Technologies， wsit.dev.java.net)。它是 Metro (metro.dev.java.net)的子集。

項目 Tango 的詳細描述，可以參見文章： http://wsit.dev.java.net/docs/tango-overview.pdf

WSIT 利用了 GlassFish v2 中新的 WS 架構，通過 WS 管道(pipeline)中的元素去處理不同的 WS-* 標準。WSIT 透明地工作，開發人員可以使用標準的 JAX-WS 開發模型並通過 NetBeans 增加 WS-*屬性(要包含一個 NetBeans 的

plugin)。WSIT 保證了和其他提供商的互操作性，主要是 Microsoft，這一互操作方案已在許多 WS Plugfests 上測試過。

下列標準目前被支援：

Boostrapping: WS-MetadataExchange, WS-MetadataExchange WSDL, WS-Transfer

Reliable Messaging: WS-ReliableMessaging, WSReliableMessaging Policy

Atomic Transactions: WS-Coordination and WSAtomic Transaction

安全：WS-Security, WS-SecurityPolicy, WS-Trust, WS-SecureConversation

策略: WS-Policy, WS-PolicyAttachment

XML 處理

GlassFish 對關於 XML 處理的 JAXP 和 StAX 介面提供支援。

The Streaming API for XML(StAX) 由 BEA 領導是新的 XML 解析介面，並且它已成為 Java 平臺的一個新的部分，包含在 Java EE 5 和 Java SE 6 平臺中。StAX 能夠被用來替代 SAX 和 DOM，並且有著不同的性能表現和介面特性。DOM 是一個駐留在記憶體中以文檔為核心的開發介面。它將文檔解析成一組能夠以任何次序訪問的物件，但這種靈活性的獲取是以性能受到顯著影響為代價的，這種影響包括了記憶體的消耗和解析的性能。SAX 是一種事件驅動的開發介面，可以在 XML 解析到新元素時進行回調（callbacks）。 SAX 比 DOM 有著更好的性能特性，但當事件回應要求狀態進行更新時，它會導致控制反轉（an inversion of control）。

Fast Infoset

在某些情况下 XML 文档相当冗以致于妨碍了它的用，不是因文档大小，而且因文档的花。 Gzip □ □ □ □ 了大小但需花更多的在上，Fast Infoset ANSI/ISO □ 准在方面提供了更好的平衡。

目前有 4 □ 不同的 Fast Infoset 商，但其它准也会主个方向 (例如 W3C EXI 工作也在注个域)，所以 GlassFish Web Services stack 被成支持多。它目前支持 MTOM 和 Fast Infoset □ □ （除了文本)，但当其它准出它也会支持。

Fast Infoset 能和 WSIT 一起使用，Windows Communication Foundation (WCF) 提供了隔离(encoding isolation)，一些厂家其展并了 FI 的支持。

拉式解析器(pull parser)具有了 SAX 的性能優勢，同時主控代碼保留了對需要時事件的控制。近來 GlassFish 社區在和 Woodstox 社區(Codehaus.org，提供有高度優化的 StAX 實現)進行合作開發。

最新的 JAXP 規範是 JAXP 1.4。JAXP 1.4 是 JAXP 1.3 以來的一個維護版本。JAXP 1.4 包括了一系列的改進，最主要的變化是和 StAX 的集成。 JAXP 1.4 被包含在 Java SE 6 中，J2SE 5.0 包含的只是 JAXP 1.3。 GlassFish 可以和 J2SE 5 或 Java SE 6 一起工作。

JAXP 1.4 的實現包括了 JAXP 1.4 和 StAX，但 StAX 的實現還可以被單獨提供。所有這些實現由 GlassFish 社區完成，並通過不同的發佈分發。

XML 數據綁定

在 Java EE 中 XML Data Binding 是通過 JAXB API 實現的，Java EE 5 中的版本是 JAXB 2.0 (jaxb.dev.java.net) ，類似與 JAX-WS，JAXB 相比於 J2EE 1.4 中的規範同樣提供了顯著的改進。這一改進得到 Java 社區的認可並且很快將成爲一個新的規範。

JAXB 2.0 的實現是高品質的快速的且功能完整。JAXB 爲其他社區的開發人員接納，並因此被許多組織採用，這其中包括 JBoss, Apache Axis, Apache XFire, TmaxSoft, ServiceMix 和 ActiveSOAP。

Java EE 5 之後

Java EE 5 規範要求的是 JAXB 2.0 和 JAX-WS 2.0，並且這些規範由 GlassFish v1 和 Java SE 6 提供了實現。經過在多種情況下的使用，包括與 Microsoft 的 Windows Communication Framework 所嘗試的 Web 服務的互操作，導致了 JAXB 2.1 和 JAX-WS 2.1.x.規範的形成，以改善相容性，而這些規範的實現是 GlassFish v2 所要完成的。

TopLink Essentials 和 JPA

Java EE 5 規範中一個巨大的功能提升是 Java Persistence API (JPA) 和新的 Enterprise JavaBeans 規範－EJB 3.0。這些開發介面源自早期的 EJB 2.1 ，但利用了 Java 語言的新特性比如 annotation，並吸納了來自社區的經驗比如 Hibernate, Oracle TopLink 和 JDO。

兩套 APIs 能夠被單獨使用。JPA 爲實體的持久化提供了簡單的編程模型，它可脫離 Java EE 5 提供，用在 J2EE 1.4 甚至 Java SE 中。它還包括了一套 API 來插入開發商的不同的持久化方案。EJB 規範因爲受益於使用 JPA 而變得簡單。JPA 和 EJB 3.0 都已被很好地採納和接受。

TopLink Essentials (TLE) 是 GlassFish 社區所採用的 JPA 實現。它是 Oracle 公司給與社區的巨大貢獻，TLE 基於 Oracle 在 Top Link 產品上的長期經驗，其現已成爲 Oracle 和 Sun 的聯合開發的專案。TLE 的初始實現比如 GlassFish v1 所包含的，只是爲了和規範相容，但 TLE 所試圖成爲的是一個完整的實現，Oracle 的計畫是將持續地增強 TopLink 的特性並將其貢獻給更廣泛的社區。TopLink Essentials 被用在許多 Oracle 的商業產品上，包括其應用伺服器，此外還包括 Sun 公司的 GlassFish 和 NetBeans， Apache Tomcat, TmaxSoft's JEUS, Spring 2.0 和 EasyBeans。

JPA 的可插拔性意味著可以非常容易地替換不同的 JPA 實現並放置到 GlassFish 應用伺服器上。比如，替換成 Apache 的 OpenJPA 和 Hibernate 的 JPA 方案，而且這樣的方案非常受歡迎。

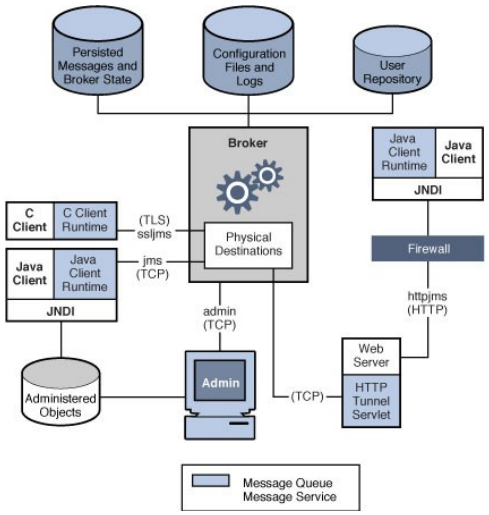
CORBA

GlassFish 帶有全面地 CORBA 實現。這種實現已經被持續地改進了多年，例如 GlassFish v2 現在可以動態生成 stubs 和 skeletons，並應用 NIO 架構。GlassFish v2 還將包括許多關鍵性能上的提高。

OpenMQ Messaging

GlassFish 包括了產品品質的 Message Queue 實現 (mq.dev.java.net) ，它提供以下特性：

- Message buffering between enterprise system components
- 可擴展的 message servers (broker clustering)分佈性
- 集成的 SOAP / HTTP 消息
- Java 和 C 的用戶端 API
- 可擴展的 JCA 1.5 相容資源適配器
- 企業 Administration 特性
- Extensive JMX 支持



GlassFish 還包括了 JMS 通用資源適配器(Generic Adaptor for JMS) ，它能夠被用在許多不同的 Messaging 的解決方案中，包括： Tibco, WebSphere MQ 6, Active MQ 和 JBoss Messaging。

Java Business Integration

GlassFish v2 還包括對 Java Business Integration (JBI) 介面的支援。

JCP 規範(JSR-208)定義了面向服務的集成匯流排和元件架構的核心。JBI 標準化了通用消息路由(routing)架構，服務引擎和綁定的插入介面，以及組合多個服務到單個可執行和審計的工作單元的機制。

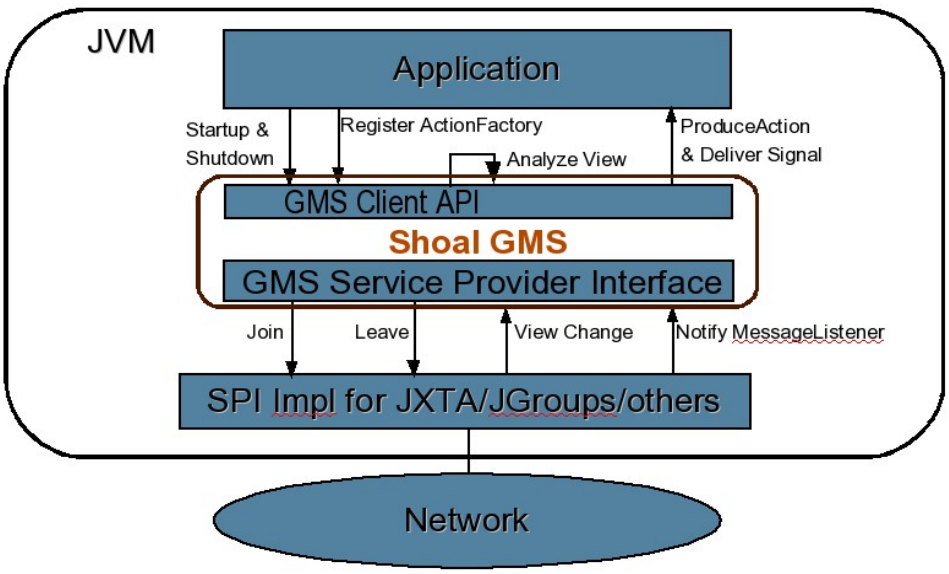
GlassFish v2 所使用的 JBI 實現來自項目 OpenESB ，這種實現也被用來在 Sun 公司其他的企業級別的產品中。 OpenESB 的實現通過 Java EE 服務引擎(JavaEE Service Engine)被集成進 GlassFish，相同介面也可用于集成其他的實現，比如 ServiceMix。Enterprise Pack of NetBeans 爲此提供了一流的開發工具，包括 Service Assembly 編輯器, 圖形化的 WSDL 和 XSLT 編輯器，和智慧消息處理器(an Intelligent Event Processor)。

Clustering 和状态复制(State Replication)

GlassFish v2 支援集群(clustering)，並支援動態集群(dynamic clustering)和記憶體複製(memory replication)。動態集群的特性由專案 Shoal (shoal.dev.java.net)提供，該項目利用了 JXTA 技術的通用框架，使得一個應用程式可以動態地成爲一個預定義的集群的成員，並同樣訂閱到集群的事件，比如：

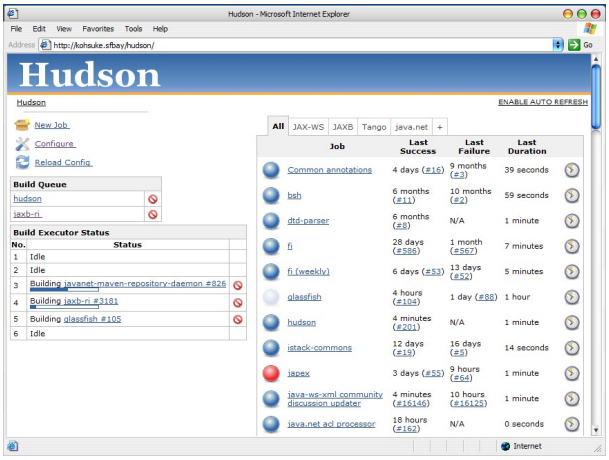
- 成員加入(Member join), 計畫關閉(planned shutdown)，失敗發生 (failures)
- 選擇恢復成員
- 自動委派恢復(delegated recovery initiation)

GlassFish v2 還支援 In-Memory Replication，這是一個爲 HTTP 會話持久化和 Stateful Session Bean 持久化提供的羽量級的低成本的方案。此方案對於大多數程式已經足夠，但對於那些需要達到 99.999% 可提供性的應用，仍可能需要使用 High Availability Data Base (HADB) 的持久化方案。



工具

GlassFish 和 NetBeans 社區合作來開發和實現 Java EE 5 的規範，NetBeans 是第一個支援 GlassFish 的集成開發環境(IDE)。



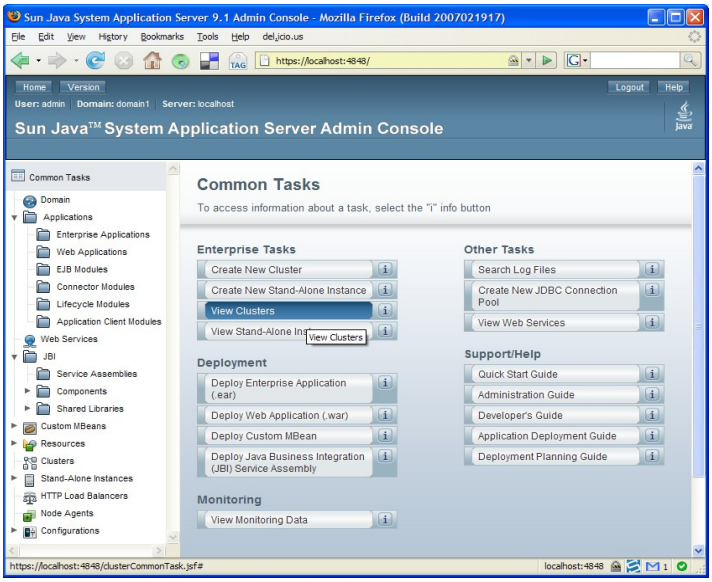
之後，IDEA、JBuilder 和 MyEclipse 也增加了對 GlassFish 的支持。

從另一個角度看，GlassFish 社區自身也包括了許多工具，其中一些在其所屬領域已經相當流行。

值得一提的是 Hudson，一個可持續進行集成的工具，被用來完成構建、執行測試和其他任務。Hudson 的構建非常簡單，支持多個從屬(slave)機器並擁有可插拔的架構來進行擴展。

管理

管理是 GlassFish v2 的強項之一。GlassFish 包括一個圖形化管理介面，它功能全面且由任務驅動，另外還包括一個命令行介面，它支援 TCL（Tool Command Language）解釋器，並應用“do what I mean”機制來修正拼寫錯誤。GlassFish 還支援基於 JMX 的管理，並且支援對集群和負載均衡(Load Balancing)的管理。



安裝體驗

所前所述，GlassFish v2 有一個單一的安裝程式，它可用來安裝下面 3 種用戶 profile 中的任一種： developer, enterprise 或 cluster。使用網站

http://glassfish.java.net 所得到的二進位檔進行安裝過程，分為兩步：安裝和配置。注意，可以從 developer 的 profile 升級到 cluster 的 profile。

GlassFish v2 安裝包(大約 55MB)通過借助 Pack200 技術大大壓縮了大小(這一技術同樣被用在 Java SE 6 的安裝中)。

Sun Java System Application Server 的安裝程式將略為高級和圖形化。

Ubuntu 上，安裝可以通過 apt-get 提供。

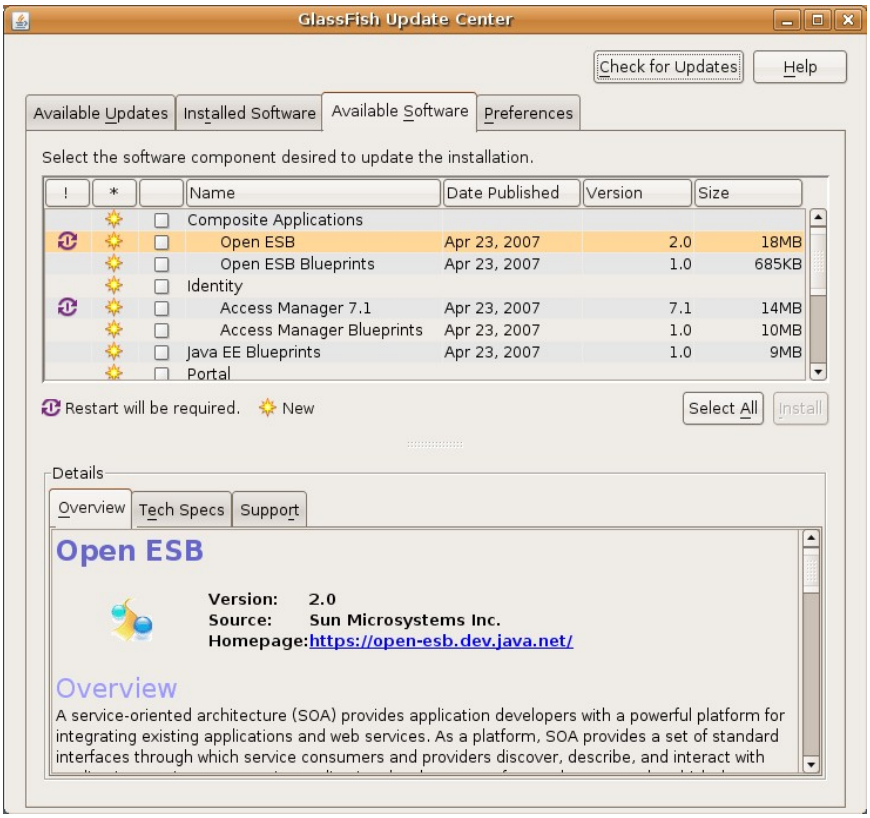
Sun Java System Application Server 9.1 同樣被 Java Enterprise System 5 Update 1 提供和支援。

升級中心

升級中心負責發佈 GlassFish 附加的元件和更新並提供一個機制來給社區發佈貢獻。現在各種附加的元件都可以通過升級中心發佈。

GlassFish 升級中心是個開源專案 http://updatecenter.dev.java.net。GlassFish V2 上升級中心的初始版本注重給 GlassFish 的附加應用提供簡單的下載和安裝體驗。

GlassFish 升級中心的功能被在兩層實現。在伺服器端，有更新中心模組和相應的 XML 目錄檔。在用戶端，包含在 GlassFish V2 的安裝中。當啓動時，更新中心的用戶端(位於 **GlassFish 目錄/updatecenter**) 會連接到一個預先定義的更新中心伺服器的 URL 位址，然後下載目錄檔，將其和本地註冊的已安裝元件比較，然後顯示可安裝或升級的元件。 最後，根據用戶的選擇，進行裝元件的下載和安裝。



這一機制被用來部署新的 GlassFish 特性、框架庫、甚至完整的 Web 應用程式。你可以創建你自己的更新中心附加模組，將其打包，測試，並通過簡單

地添加更新中心的定義使它們可以通過更新中心用戶端得到。

GlassFish 產品

開源專案普遍的一個益處是用戶可以根據對技術的理解，並結合自身所需特性和風險狀況作出自己的決定，從而加速了產品採用的週期。在 GlassFish v1 甚至在 GlassFish v2 中，我們正在見到這種加速採用的情況。部署案例處於回饋環(feedback loop)最後的階段，它對於 GlassFish 的成功至關重要，我們正在將這些案例收集到 http://blogs.sun.com/stories。

一個採用的例子是 Peerflix (peerflix.com)，一個媒體共用社區(media-sharing community)，最近重新設計了他們的架構並從 Microsoft 架構轉移到了 GlassFish v1。新站點使用 JSF 的 Facelets 和 Apache Shale 技術來實現表現層，並採用 Kodo JDO 在 MySQL 資料庫上完成持久化，並運行在 Solaris 10。網站重構至今的體驗是非常正面的。另一個很好的例子是高容量的澳大利亞第三大商業站點 wotif.com，選用了 GlassFish，而不是 JBoss。

開發 GlassFish

GlassFish 是通過一種"透明"的方式來進行社區的參與和其他社區的合作。社區和開發流程對開發結果有著關鍵的影響，較之早期的用傳統方法和開發流程所開發出的產品，它們創造出更加匹配客戶需求的產品，並通過更快更精確的迴圈回饋得到了更高的產品品質，以及更快更敏捷的開發週期。

有些影響是超出期望的，比如，對外部的通訊工具的使用(web 網站, 博客, 郵件列表等)使得能夠使用一些工具比如外部的搜索引擎、博客閱讀器和郵件列表歸檔，對用戶在 Web 上行爲加以利用。其結果是在由各類不同團隊、個人、組織構成的 GlassFish 社區上的資訊流動，較之同一個公司的同類組織之間的流動更加順暢，即使這個社區在地理上分佈是分散的跨時區的。

未來方向

GlassFish 正在開發之中。我們期望社區和產品能夠繼續進步。社區正在規範它的管理並將繼續加以完善。我們期望社區的構造在多方面得以繼續增強，比如，JavaOne 2007 之前的 **GlassFish 技術日**。我們將啓動一些 GlassFish 用戶組。請注意， 如果 Sun 技術日 (http://developers.sun.com/events/techdays/)在您附近的城市舉行，很可能那也會有一個 GlassFish 技術日。

社區已經提交了高品質的 GlassFish v2 特性，短期目標是增加對其的採用和部署。我們還希望看到更多的應用和架構在 GlassFish 上工作，並且增強與其他組織和社區的關係。

中期目標是 v3，一個高度模組化的容器，代號 HK2。它將啓動非常快速且只需少量的資源，只在需要執行需要管理的程式時才啓動相應的服務。這樣的容器將適用在許多場合，從非常羽量級的使用到高可靠性的應用。GlassFish v3 的原型提供在 http://hk2.dev.java.net。

Jersey - RESTful Web Services

JAX-RS 是項由 JCP 開發 (JSR 311)的新技術，用來為 RESTful Web 服務定義基於 POJO 編程模型。 資源被簡單地標注成 POJOs，而 HTTP verbs 被映射成 java 方法（參看： JAX-RS annotations 和 POJOs）。 其 Consumer 和 producer 標注可以進一步簡化 MIME 類型的操作。

相關的規範制定工作位於 http://jsr311.dev.java.net/ (其早期草稿已經提供)。JAX-RS 列舉了即將到來的 Java EE 6 規範(JSR 316)的潛在組件。

Jersey (http://jersey.dev.java.net) 是 JSR 311 的參考實現，並緊

JAX-RS annotations 和 POJOs

下面是典型的 JAX-RS 資源使用 POJO：

```
@UriTemplate("/helloworld")
public class HelloWorldResource {
@HttpMethod("GET")
@ProduceMime("text/plain")
public String getMessage() {
    return "Hello World";
}
}
```

資源一旦部署，將可通過 URI 路徑"/helloworld" 訪問到。

密跟隨規範的演變。Jersey 運行時可以被部署到各類環境，包括：GlassFish servlet container, Metro JAX-WS endpoint, Grizzly, Java 6 (使用羽量級 Web 伺服器), Tomcat 等等...

Jersey 是一組庫，並且可以通過 GlassFish 的更新中心獲取其在 GlassFish 上的安裝。NetBeans 6.0 支持從 POJO 創建資源以及為測試 JAX-RS 資源而創建基於 AJAX 的用戶端。

Sailfin - GlassFish 的 SIP Servlets 技术

項目 SailFin (http://sailfin.dev.java.net) 是基於由 Ericsson 貢獻的穩定的可擴展的 SIP Servlets 技術 和 GlassFish v2 應用伺服器。

SIP (Session Initiation Protocol) 和 SIP Servlets 在後端支援著許多流行的服務比如 Voice-over-IP (VoIP) 電話服務、即時通訊、參與者和好友列表管理和 web 會議。SIP Servlets 被期望在下一代電信服務中扮演更加重要的角色。

結合 Java EE 平臺，SIP Servlets 可被用來添加富媒體(rich media)和企業應用之間的交互。 SIP Servlet 1.1 (JSR 289) 更新了 SIP Servlets API 並且定義了一個標準的應用編程模型來組合 SIP Servlets 和 Java EE 元件。通過利用 Java EE 服務比如 Web 服務、持久化、安全和交易，將可以更快地開發出更智慧的通訊應用。 Sailfin 專案的目標是在其第一個版本中支持 JSR 289。

Sailfin 的每日構建和里程碑構建都可以在 SailFin 的網站得到 (http://sailfin.dev.java.net)，它是自包含的(self-contained)，其安裝和 GlassFish 的安裝類似。其網站還提供包含文檔說明的常式。SailFin 的第一個版本計畫於 2008 中期發佈。

現在開始

跟蹤 GlassFish 社區活動的好去處是水族館 — TheAquarium (blogs.sun.com/theaquarium)。這是一個關於社區日常重要活動的熱點報導。另一個相關的 blog 是 Stories (blogs.sun.com/stories) ，這裏收集了關於採用 GlassFish 的各類案例。

GlassFish 社區的主頁是 java.net (glassfish.java.net)；文檔、郵件列表、論壇和 wiki 都放置在這裏。所有關於 GlassFish 的構建也在這裏提供。

GlassFish 以不同的方式提供其構建：Nightly 和 Weekly builds 是從源代碼構建常規的構建；Milestone builds 經歷了穩定的開發週期並且通常含有新的特性 – 特別的細節由相關的 Highlight 文本描述；final builds 是最穩定的也是我們所推薦用於部署的版本，Sun 爲 final builds 提供支援。

保持聯繫

我們希望這篇文章能夠勾勒出 GlassFish 社區和其應用伺服器的優勢。我們希望你能嘗試並發掘它的用途。。。

我們會一直珍視您所提供的任何回饋。希望在 http://glassfish.java.net/ 和 http://blogs.sun.com/theaquarium 上見到你。

關於作者

Eduardo Pelegri-Llopart 是 Sun 公司的 Distinguished Engineer 並且是 GlassFish 社區的總負責人。

Eduardo 為多個開源項目工作並且於 1995 年就參與了 Java 社區。他居住在 San Francisco Bay Area。

Yutaka Yoshida 是 GlassFish 專案的工程師。

Alexis Moussine-Pouchkine 是一位 GlassFish 技術傳道士，居住在法國，巴黎。

索引