# Java WebServer
# Tomcat, JBoss, JRun, JOnAS

Michaël TRANCHANT
Décembre 2008

Veille technologique dans le cadre de l'UE NFE107
Architecture et Urbanisation de Systèmes d'Informations

**Sommaire**

1. Brefs rappels de notions à connaître ..... 3

1.1 J2EE ..... 3
1.2 JSP ..... 4
1.3 Servlets ..... 4
1.4 EJB ..... 5
1.5 Un serveur d'application ..... 5
1.6 Serveur web / serveur d'application ..... 6

2. Tomcat ..... 7

3. JBoss Application Server ..... 9

4. JRun ..... 11

5. JOnAS ..... 13

6. Comparaison ..... 15

Conclusion ..... 16

Sources ..... 17

# 1. Brefs rappels de notions à connaître

### 1.1. J2EE : Java 2 Platform, Enterprise Edition

J2EE est une plate-forme Java, conçu pour du mainframe, typique de l'informatique des grandes entreprises. Sun Microsystems (en collaboration avec des partenaires de l'industrie tels que IBM) a conçu J2EE pour simplifier le développement d'applications en environnement client léger. J2EE simplifie le développement d'applications et permet au programmeur le développement normalisé de composants modulaires réutilisables.
Inclus le JDK, la technologie «*Write Once Run Anywhere*» (portabilité), communique avec CORBA (Common Object Request Broker Architecture), le JDBC (Java Database Connectivity), permet EJB, XML, Servlet, JSP.

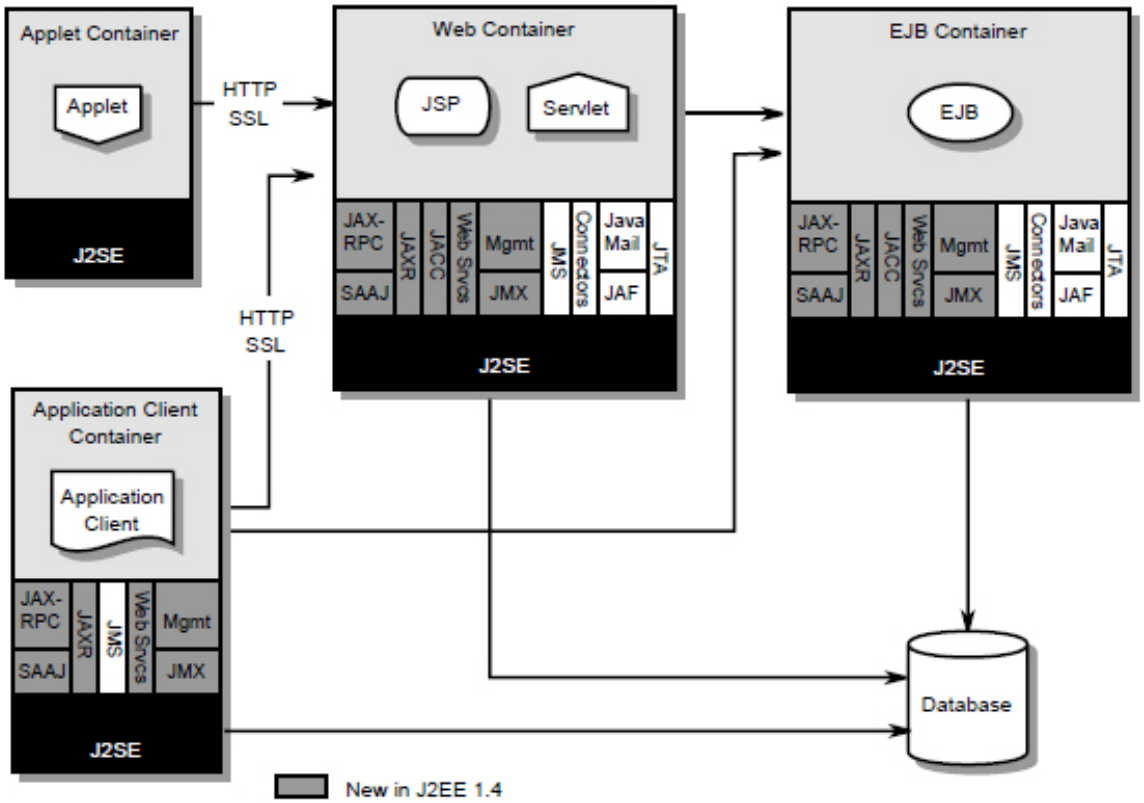


**Fig. 1: Architecture J2EE 1.4** (Source: Sun Microsystems)

### 1.2. JSP : Java Server Page

JSP est une technologie pour le développement de pages Web incluant du contenu dynamique. Contrairement à une page HTML qui ne contient que du contenu statique qui reste par définition toujours le même, JSP peut changer selon l'identité du visiteur, de son navigateur Internet, de l'heure, de la configuration du système, des actions du visiteur, etc.
Une page JSP contient des balises standards, comme du HTML (ou du WML, XML…), comme toute page web normale. Pourtant, une page JSP contient aussi des éléments JSP spécifiques (scriptlets), permettant au serveur l'insertion dynamique de contenu (contenu de BDD, préférences du visiteur…)
Lorsqu'un utilisateur accède à une page JSP, le serveur exécute les éléments JSP, fusionne les résultats avec les parties statiques de la page, et envoie le tout au navigateur.

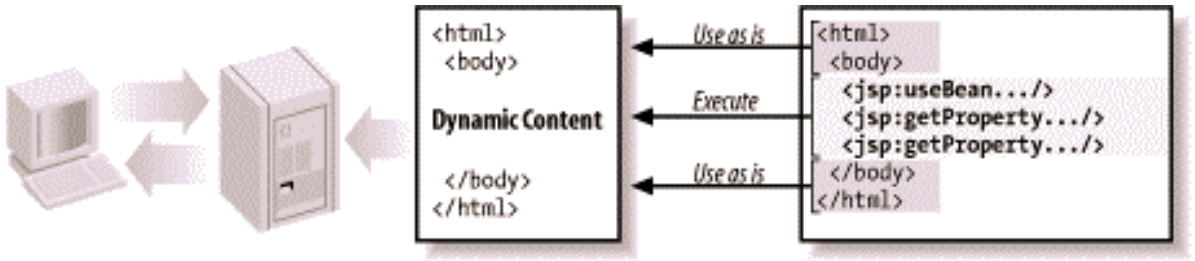


**Fig. 2: Principe des JSP**

JSP définit des outils standards utiles pour n'importe quelle application web comme l'accès à des composants JavaBeans, les contrôles d'accès entre pages, le partage d'information entre requêtes, pages et utilisateurs.

```
<html>
<head>
<title>Greetings !</title>
</head>
<body>
<% for (int i=0;i<5;i++) { %>
      <h1>Hello World !</h1>
<% } %>
</body>
</html>
```

**Code 1: Exemple de JSP**

### 1.3. Servlet

Technologie Java utilisée pour effectuer des traitements coté serveur en réponse aux requêtes provenant en général de poste clients distants.
Bien que les Servlets puissent répondre à n'importe quel type de requête, elles sont généralement employées pour répondre à des requêtes de type HTTP et qui permettent de retourner dynamiquement des pages HTML.

```
import java.io.*;
import javax.servlet.*;
import java.servlet.http.*;
public class SimpleServlet extends HttpServlet {
    public void doGet(ServletRequest req,ServletResponse resp) throws IOException,ServletException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.println("<html><head><title>Simple Servlet Output</title></head>");
        out.println("<body><h1> SimpleServletOutput</h1>");
        out.println("</body></html>");
    }
}
```

**Code 2: Exemple de SERVLET**

### 1.4. EJB : Enterprise JavaBeans

Cette architecture propose un cadre pour créer des composants distribués (déployés sur des serveurs distants) écrit en langage de programmation Java hébergés au sein d'un serveur applicatif permettant de représenter des données (EJB dit *entité*), de proposer des services avec ou sans conservation d'état entre les appels (EJB dit *session*), ou encore d'accomplir des tâches de manière asynchrone (EJB dit *message*). Notions d'annotations.
EJB est grossièrement équivalent à COM/DOM de Microsoft dans son architecture, mais comme toute architecture basée sur Java, elle peut être déployée sur quasi-tous les OS.

### 1.5. Un serveur d'application

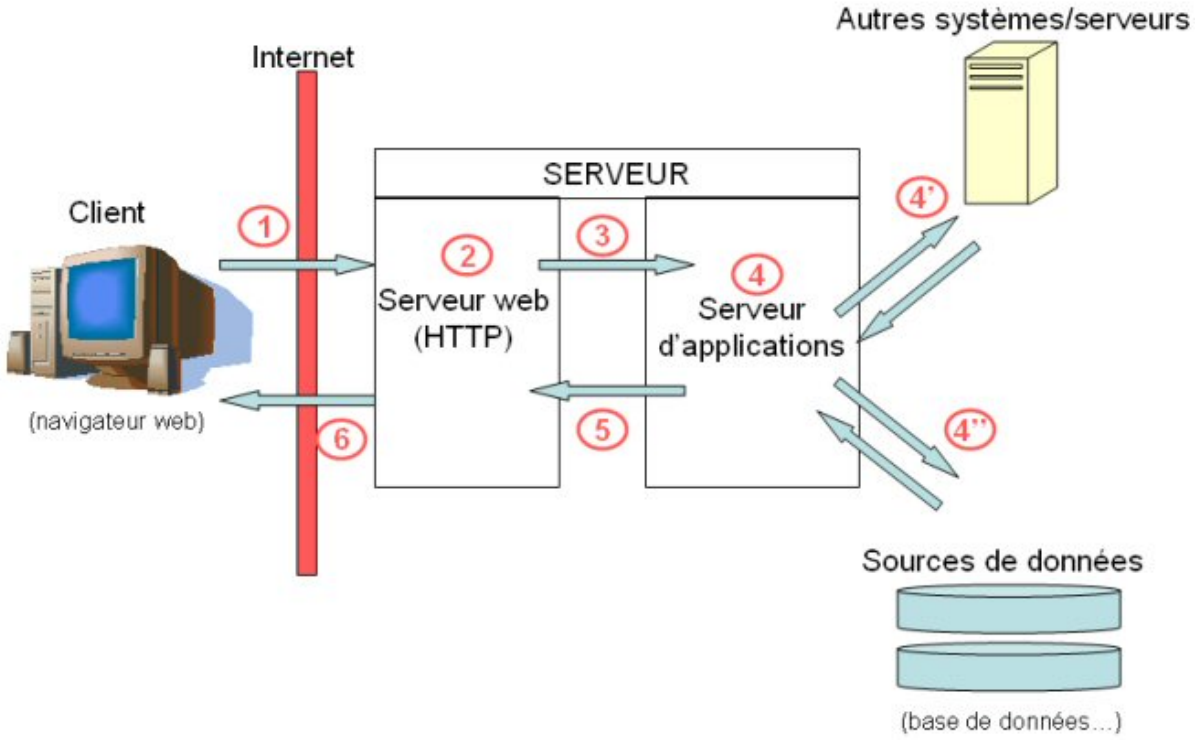


**Fig. 3 : Fonctionnement d'un Serveur d'Application** (Source: Université Paris-Est Marne-la-Vallée)

1) Le client émet une requête (i.e. appelle une URL) pour demander une ressource au serveur. Exemple : `http://leserveur.com/welcome`. Il ne sait pas ici si la réponse qui va lui parvenir est statique (page HTML simple) ou dynamique (générée par une application WEB). Dans notre cas, il s'agit d'une application répondant à l'adresse `welcome` sur le serveur `leserveur.com`.

2) Côté serveur, c'est le serveur web (exemple : Apache) qui traite les requêtes HTTP entrantes. Il traite donc toutes les requêtes, qu'elles demandent une ressource statique ou dynamique. Seulement, un serveur HTTP ne sait répondre qu'aux requêtes visant des ressources statiques. Il ne peut que renvoyer des pages HTML, des images,... existantes.

3) Ainsi, si le serveur HTTP s'aperçoit que la requête reçue est destinée au serveur d'applications, il la lui transmet. Les deux serveurs sont reliés par un canal, nommé connecteur.

4) Le serveur d'applications (exemple : Tomcat !) reçoit la requête à son tour. Il est, lui, en mesure de la traiter. Il exécute donc le morceau d'application (la Servlet) auquel est destinée la requête, en fonction de l'URL. Cette opération est effectuée à partir de la configuration du serveur. La Servlet est donc invoquée, et le serveur lui fournit notamment deux objets Java (Tomcat est un serveur d'applications Java) exploitables : un représentant la requête, l'autre représentant la réponse. La Servlet peut maintenant travailler, et générer la réponse à la demande. Cela peut passer par la consultation de sources de données, comme des

bases de données (4" sur le schéma). Ou bien par l'interrogation d'autres serveurs ou systèmes (4' sur le schéma), l'environnement Java web permettant de se connecter à de nombreux systèmes.

5) Une fois sa réponse générée, le serveur d'applications la renvoie, par le connecteur, au serveur web. Celui-ci la récupère comme s'il était lui-même allé chercher une ressource statique. Il a simplement délégué la récupération de la réponse, et celle-ci a été générée, mais ce n'est plus le problème.

6) La réponse est dorénavant du simple code HTML, compréhensible par un navigateur. Le serveur HTTP peut donc retourner la réponse au client.

### 1.6. Serveur web / serveur d'applications

Dans le schéma précédent (fig. 3), le serveur web et le serveur d'applications sont séparés. Ces deux composants sont en effet nécessaires côté serveur, puisqu'ils se complètent : le serveur d'applications ne sait pas traiter une requête HTTP, le serveur web ne sait pas exécuter d'applications !

Si ces deux composantes sont indispensables, elles ne sont pas nécessairement séparées. Tomcat inclut ainsi un serveur web, et est donc capable de fonctionner en autonomie (*StandAlone*), pour traiter à la fois les requêtes HTTP simples (ressources statiques) et les applications web. Le principe est de changer de connecteur (par rapport à notre schéma en haut de la page), pour en utiliser un comprenant les requêtes HTTP et non plus les requêtes triées venant du serveur web.

Dans certains cas, cette possibilité est extrêmement intéressante. Elle permet de proposer un serveur web complet en installant le minimum de logiciels. Néanmoins, ce n'est pas une fin en soi. Pour des besoins de production plus importants, il est intéressant de scinder les deux activités, ne serait-ce que pour alléger la tâche de chacun des serveurs. Tomcat peut ainsi se concentrer uniquement sur l'exécution des applications. Cela économise également sa mémoire et augmente ses performances, en lui permettant notamment de créer moins d'objets... En outre, dans certains cas, comme la mise en place d'un serveur sécurisé (SSL), on préférera gérer la sécurisation sur le serveur web, et pas sous Tomcat, même si celui-ci est capable de mettre en place un environnement sécurisé.

## 2. TOMCAT

**Apache Tomcat** est un conteneur libre de Servlet Java EE. Issu du projet Jakarta, Tomcat est désormais un projet principal de la fondation Apache. Tomcat implémente les spécifications des Servlets et des JSP de Sun Microsystems. Il inclut des outils pour la configuration et la gestion, mais peut également être configuré en éditant des fichiers de configuration XML. Comme Tomcat inclut un serveur HTTP interne, il est aussi considéré comme un serveur HTTP (web).
Tomcat peut être utilisé en autonomie avec son propre serveur web, ou en collaboration avec d'autres comme IIS par exemple.

Le projet Tomcat a été lancé comme implémentation de référence des Servlets par James Duncan Davidson, architecte logiciel chez Sun. Il a contribué à rendre le projet libre et a joué un rôle majeur dans sa donation par Sun à la fondation Apache. Le projet Tomcat fait partie d'un collectif Open Source connu sous le nom de **Jakarta**.

Il est constitué de composants
- **Catalina** est le container Servlets, et implémente les spécifications de Sun pour les Servlets et les JSP;
- **Coyote** est le connecteur HTTP: il écoute le trafic entrant, dirige les requêtes au moteur de Tomcat, processe la requête et renvoie la réponse au client ;
- **Jasper** est le moteur JSP. Il parse les fichiers JSP pour les compiler en tant que Servlets (gérable par Catalina). Il est capable de détecter les modifications des fichiers et de les recompiler à la volée.

Avantages de Tomcat :
- Tomcat est simple, beaucoup plus que les serveurs d'application Open Source « complets »
- Il est donc plus simple d'administrer une instance Tomcat qu'un serveur d'applications complet.
- Il n'occupe que 2 ports sur la machine (8080 et 8009), alors que les autres en prennent une dizaine
  - 8080 : port propre de Tomcat
  - 8009 : port de communication entre Apache et Tomcat (protocole AJP13)

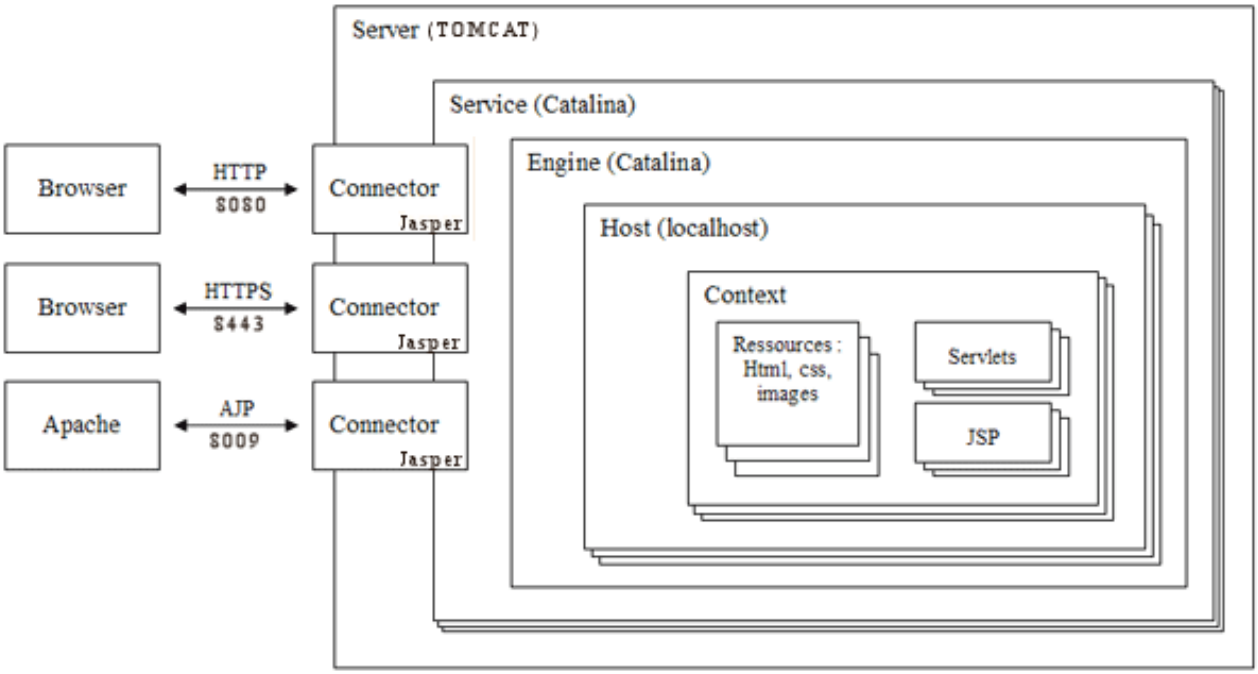


**Fig. 4 : Fonctionnement de Tomcat** (Source: Université Paris-Est Marne-la-Vallée)

Le schéma met en évidence les conteneurs `Context`, qui représentent les applications, dans un conteneur `Host`, d'un point de vue de l'organisation du fichier `server.xml`, les conteneurs `Context` apparaîtront comme éléments XML enfants du conteneur `Host`.

Première version libre : 3.0.X (1999), ***dernière version : 6.0.18 (juillet 2008)***

Dans la même catégorie, on retrouve **Jetty**.

**Jetty** est un serveur HTTP et un moteur de Servlets Java. En raison de sa petite taille, il convient parfaitement pour fournir des services web une fois embarqué dans une application Java. La première version date de 1995.
Il peut être utilisé en autonomie, comme serveur web traditionnel, comme server de contenu dédié derrière un Apache utilisant `mod_proxy`, ou intégré dans une application Java (JOnAS, Geronimo, JBoss, JFox)

*Dernière version : 7.0.0 (septembre 2008)*

Autre conteneur existant : **Resin Server**.

Peut servir du PHP ou du Java
2 versions (Open Source GPL et Professionnelle)
La Professionnelle permet le clustering et ses performances sont accrues.

Spécifications :
Servlet 2.4, JSP 2.0, JSTL 1.01, Portlet 1. 0, XSLT 1. 0, XPath 1. 0, JAXP 1. 2, EJB 2.0

*Dernière version : 3.1.8 (novembre 2008)*

## 3. JBoss Application Server

JBoss Application Server est un serveur d'applications J2EE libre entièrement écrit en Java.

Les développeurs du cœur de JBoss ont tous été employés par une société de services appelée « JBoss Inc. ». Le projet est sponsorisé par un réseau mondial de partenaires et utilise un business model fondé sur le service. En effet, JBoss est maintenu par le Groupe JBoss gratuitement, mais toute customisation et tous services consultants sont facturés. Ce groupe est une division de RedHat depuis avril 2006.

La DGI (Direction Générale des Impôts) utilise JBoss.

JBoss est similaire à Weblogic de BEA ou à WebSphere d'IBM dans sa complexité.

Compatible avec les standards :

- CORBA OTS (Object Transaction Service),
- JTA/JTS (Java API Transaction/Service),
- WebServices

**Caractéristiques :**

- Supporte les Sun JDK 1.5 et 1.6
- Version 5GA est en passe d'être certifiée 1.5
- **Clustering** : Failover (including sessions) / Load balancing
- Distributed caching (using JBoss Cache, a standalone product)
- Distributed deployment (farming)
- Deployment API
- Management API
- Aspect-Oriented Programming(AOP)-support
- **JSP/Servlet 2.1/2.5** (Tomcat)
- JavaServer Faces 1.2 (Mojarra)
- **Enterprise Java Beans version 3 and 2.1**
- JNDI (Java Naming and Directory Interface)
- Hibernate-integration (for persistence programming; JPA)
- JDBC
- JTA (Java Transaction API)
- Support for Java EE-Web Services like JAX-RPC (Java API for XML for Remote Procedure Call), JAX-WS, JAXB
- SAAJ (soap with Attachments API for Java)
- JMS (Java Message Service) integration / JavaMail
- RMI-IIOP (JacORB, alias Java and CORBA)
- JAAS (Java Authentication and Authorization Service)
- JCA (Java Connector Architecture)-integration
- JACC (Java Authorization Contract for Containers)-integration
- Java Management Extensions

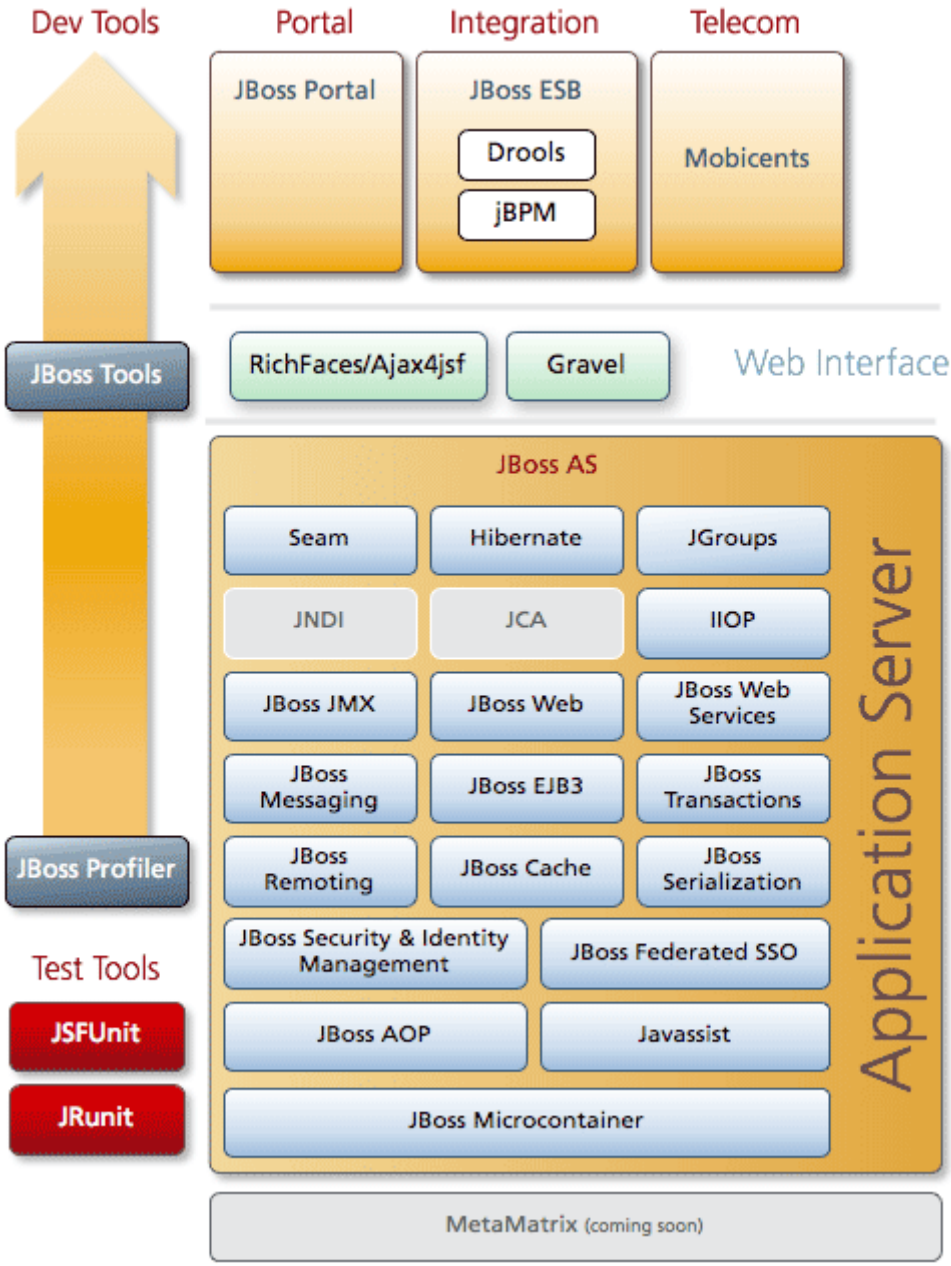


**Fig. 5 : Structure de JBoss AS 5.0.0 GA** (Source: jboss.org)

***Dernière version stable : 5.0.0 GA (décembre 2008)***

Un mot sur Seam.
C'est une « extension » très orientée Web 2.0 (AJAX), EJB3, JPA (Java Persistance API), étends JSF (Java Server Face) notamment au niveau du support XML. Pour l'utiliser plus facilement, il existe une perspective qui lui est dédiée dans Eclipse.
*Dernière version stable : 2.1.1 (Octobre 2008). Licence LGPL.*

Vous pouvez voir qu'il existe beaucoup d'autres produits édités par jboss.org :

| | | | | |
|---|---|---|---|---|
| JBoss AS | JBoss jBPM | JBoss Tools | JBoss Web | JBoss Transactions |
| JBoss Profiler | Mobicents | JBoss WS | JBoss JRunit | JBoss Microcontainer |
| RESTEasy | JSFUnit | JBoss ESB | DNA | Red Hat Messaging |
| RichFaces | JBoss IIOP | Gravel | Ajax4jsf | JBoss SOA Governance |
| JBoss Messaging | Drools | JGroups | Jopr | Jboss Federated SSO |
| Portlet Swap | JBoss Cache | Hibernate | Portal | JBoss Seam |
| JBoss S&IM | JBoss Forums | Portlet Bridge | Javassist | JBoss EJB3 |
| JBoss JMX | JBoss Wiki | JBoss Blog | JBoss AOP | […] |

## 4. JRun

JRun est un serveur d'application de Macromedia, basé sur Microsystem Java 2 Platform, Entreprise Edition (J2EE). Macromedia a été récemment racheté par Adobe.

JRun consiste en Java Server Page (JSP), Servlets Java, Enterprise JavaBeans (EJB), de Java Transaction Service (JST), et de Java Message Service (JMS).

JRun fonctionne avec la plupart des serveurs Web, comme Apache, IIS, et de manière générale, tout webserver supportant Internet Server Application Program Interface (ISAPI) ou les Common Gateway Interface (CGI).

Il existe 4 versions de JRUN : Développeur, Professionnelle, Avancée et Entreprise.
Donnant chacune des prestations différentes.

- Développeur : Toute option, mais uniquement pour le développement, et limitée à 3 connexions simultanées
- Avancée : Prévue pour le déploiement de JSP et de Servlets, en environnement en cluster.
- Professionnelle : pour les entreprises hébergeant des Servlets et des applications à base de JSP, sur un seul serveur.
- Entreprise : pour les entreprises créant et déployant des applications Java de e-commerce.

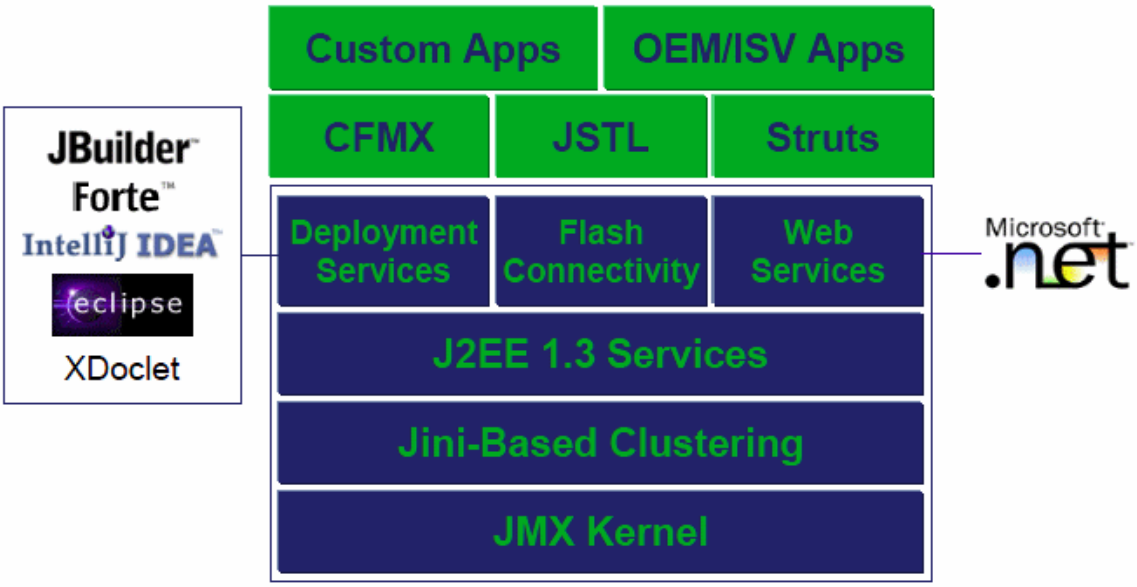


**Fig. 6: Architecture de JRun4** (Source: Jr4_technical_wp)

Il est réputé pour être le plus rapide du marché. Voici un graph présenté par Adobe, à la sortie de JRun (d'où le Tomcat 4) :

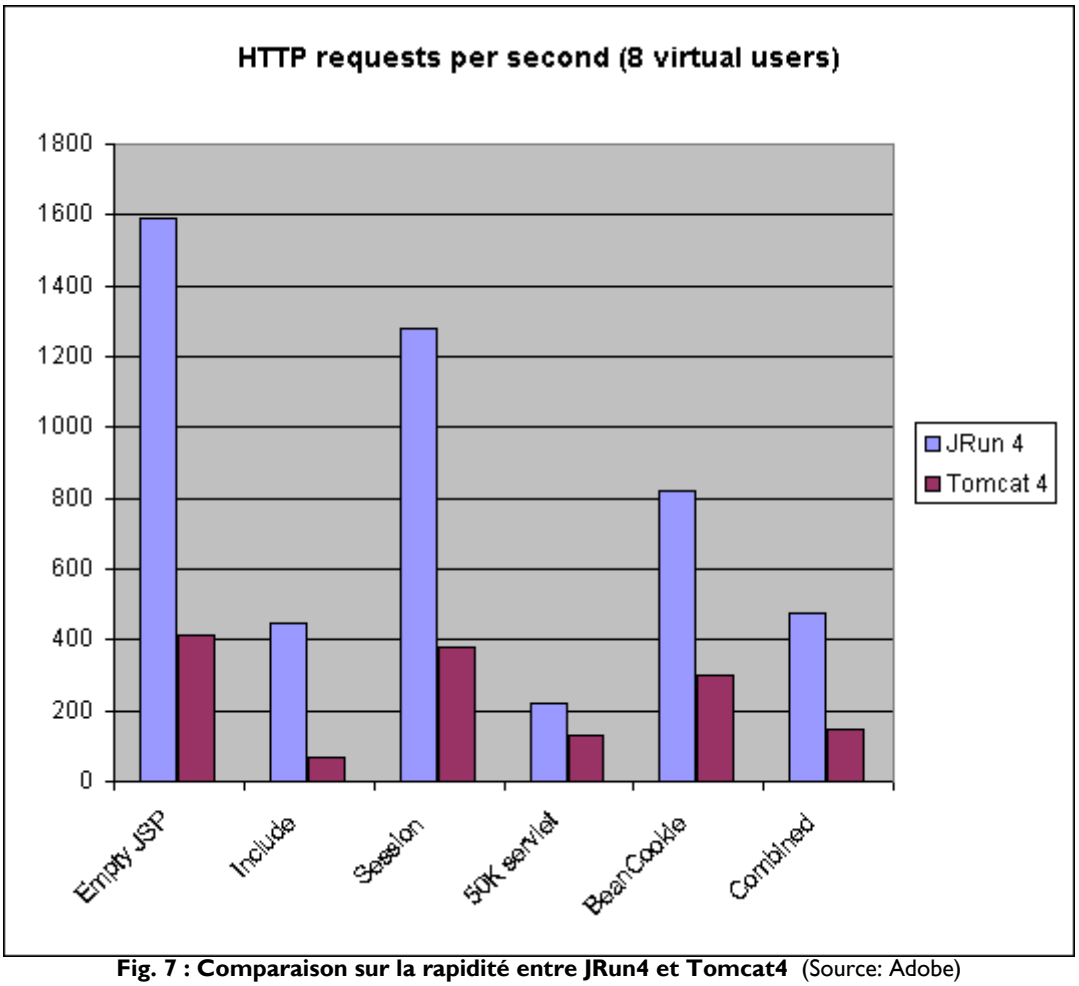


**Fig. 7 : Comparaison sur la rapidité entre JRun4 et Tomcat4** (Source: Adobe)

**Caractéristiques :**

- Supporte les JDK de Sun 1.3, 1.4, 1.5 et 1.6
- Certifié J2EE 1.3
- JSP 1.2
- JDBC 3.6
- JNDI 1.2
- JAAS 1.0
- JAXP 1.1
- Servlet 2.3
- JTA 1.0.1
- Java RMI 1.0
- JavaMail 1.1
- EJB 2.0
- JMS 1.0.2
- RMI/IIOP 1.0
- JMX 1.0

***Dernière version en date : 4.7 – Novembre 2007***

## 5. JOnAS Java Open Application Server

Le projet a démarré en 1998, sous la direction de Bull. Il est à présent développé par le consortium OW2, anciennement nommé ObjectWeb, qui est formé de plusieurs membres comme l'INRIA, France Telecom, Bull, Thalès, ou RedHat. De nombreuses entreprises participent à l'aventure OW2.

JOnAS est un Serveur d'Application libre, écrit en Java. Depuis février 2005, avec sa version 4.3.4, JOnAS est certifié J2E1.4.

Ainsi, le projet permet le développement du commerce électronique, en s'appuyant sur le registre EBXML, à compléter avec le registre d'annuaires de services web UDDI.

L'architecture de la base UDDI offre en effet, pour la mise en œuvre des processus d'affaires, des possibilités d'accès à différents types d'entités métier, de services applicatifs métier, d'informations techniques sur les services web ainsi que le référencement aux modèles techniques nécessaires au déploiement (OSGI)

**Fig. 8 : Présentation de JOnAS** (Source: Site JOnAS)

Le framework OSGI (Open Services Gateway initiative) définit un grand nombre de couches, d'API, et de services : journalisation, gestion de configurations, de préférence, service HTTP, analyse syntaxique XML, administration de paquetage, administration de permissions, gestion des utilisateurs, connecteurs E/S, exportateur UPNP (JINI), pistage applicatif, signature des paquets, services déclaratifs, gestion énergie, gestion des dispositifs, politiques de sécurité, diagnostic/contrôle et organisation en couches du framework.

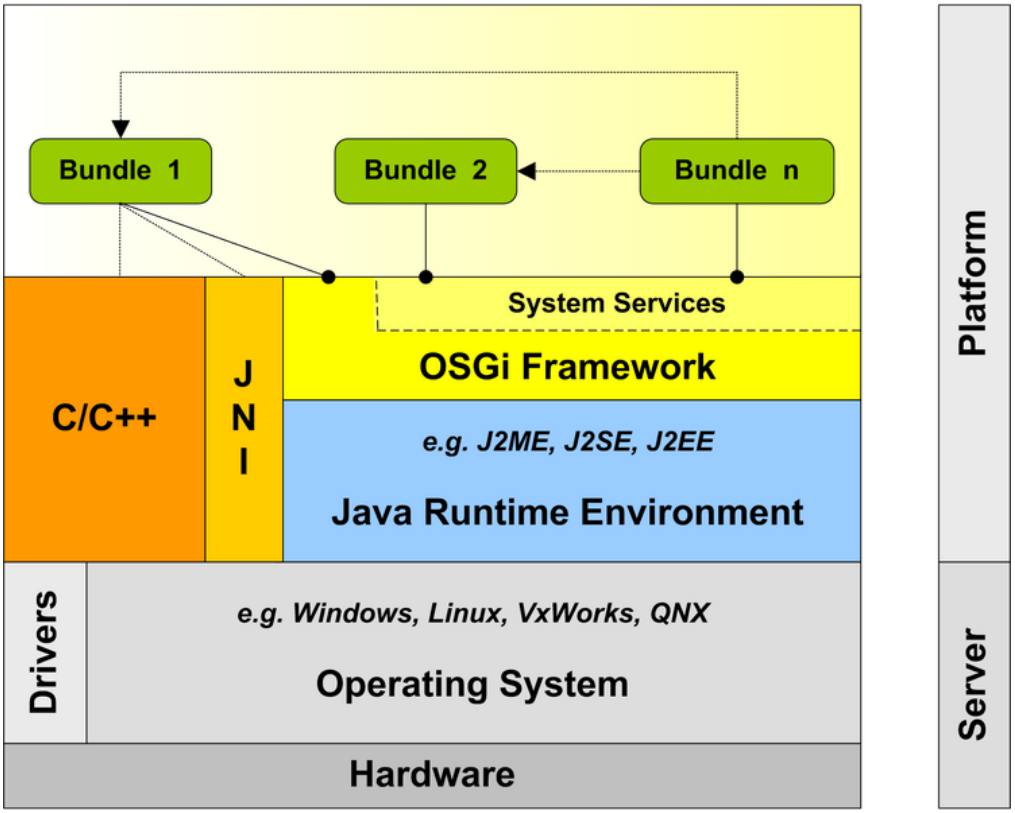


**Fig. 9 : Emplacement de la couche du framework OSGI** (Source: Wikipédia)

Par défaut, JOnAS est proposé de trois manières :

- Avec Tomcat 5.5.26
- Avec Jetty 5.1.10
- Sans web container

***Dernière version en date : 4.10.3 – Mai 2008***

## 6. Comparatif

| | Tomcat | JBoss | JRun | JOnAS | Geronimo | NetWeaver Application Server | GlassFish | WebSphere Application Server (WAS) |
|---|---|---|---|---|---|---|---|---|
| **Licence**<br>**Prix** | OpenSource | OpenSource | Payant<br>1'079€HT/proc. | OpenSource | OpenSource | Payant<br>*NC* | OpenSource | Payant<br>≈15'000€HT |
| **Dernière version stable** | 6.0.18<br>(Juil 2008) | 5.0.0 GA<br>(Déc 2008) | 4.7<br>(Nov 2007) | 4.10.3<br>(Mai 2008) | 2.1.3<br>(Sept 2008) | 7.1<br>(*NC*) | 2 UR2<br>(Avril 2008) | 7.0<br>(Sept 2008) |
| **Dernière version beta** | x | x | Pas de beta | 5.1-M2<br>(Déc 2008) | 2.2 | Pas de beta | 2 v3 | Pas de beta |
| **Certifié Java** | « implémente 5 » | 5<br>« compatible 6 » | 3 | 4 | 5 | 5 | 5 | 5 |
| **Editeur** | A.S.F.[1] | Redhat | Adobe<br>(Macromedia) | OW2<br>(Object Web) | A.S.F. | SAP | Sun Microsys.<br>(GlassFish Community) | IBM |

**Caractéristiques des dernières versions stables**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Servet / JSP** | 2.5 / 2.1 | 2.5 / 2.1 | 2.3 / 1.2 | 2.4 / 2.0 | 2.5 / 2.1 | 2.5 / 2.1 | 2.5 / 2.1 | 2.5 / 2.1 |
| **JNDI**[2] | 1.2.1 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 | 1.2 |
| **JAAS**[3] **/ SAAJ**[4] | *A intégrer* | 1.3 / 1.2 | 1.0 / 1.1 | 1.0 / 1.2 | 1.3 / 1.2 | 1.3 / 1.2 | 1.3 / 1.2 | 1.3 / 1.2 |
| **JavaMail / JMS** | *A intégrer*<br>(1.4.1 / 1.1) | 1.4 / 1.1 | 1.1 / 1.0.2 | 1.3 / 1.1 | 1.4 / 1.1 | 1.4 / 1.1 | 1.4 / 1.1 | 1.4 / 1.1 |
| **EJB** | *A intégrer* | 3.0 | 2.0 | 2.1 | 3.0 | 3.0 | 3.0 | 3.0 |
| **Notes** | Web Container de référence | 1er serveur application open-source | Est réputé être le plus rapide. | 2nd serveur application open-source | | Inclus toute une suite avec BI, studio de développement, Portail,… | | Intègre Tomcat/Geronimo en tant que serveur de contenu. |

[1] Apache Software Foundation
[2] Java Naming and Directory Interface : Service permettant aux programmes clients Java de découvrir les données et objets via un nom.
[3] Java Authentification and Authorization Service : Framework Java de sécurité, centré sur l'utilisateur. (SSL/TLS, PKI, ACL, Crypto)
[4] SOAP with Attachments for Java : API Java permettant de produire et lire des messages SOAP.

## 7. Conclusion

**SOA : Service Oriented Architecture (Architecture Orientée Service)**

Ces serveurs d'applications s'inscrivent résolument dans le cadre de la stratégie des applications orientées services (SOA), qui seront "la pierre angulaire des systèmes d'information ouverts et flexibles de demain", selon François Exertier, Manager R&D chez Bull et chef du projet JOnAS à ObjectWeb.

Ces technologies mettent en application une partie des principes d'urbanisation :

- Cohérence interne forte
- Couplages externes faibles

**Bien d'autres Serveurs d'Application pour J2EE:**

Borland ES (Borland), ColdFusion (Adobe), Cosminexus (Hitachi), Sun JSAS (Sun, dérivé de Glassfish), Trifork (Trifork), Weblogic Server (Oracle), WebObjects (Apple), WebOTX (NEC)

# Sources

Les informations étant clairsemées, et pour recouper les informations, il y a beaucoup de sources…


**Rappels**

http://java.developpez.com/faq/javaee/
http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci283984,00.html
http://www-igm.univ-mlv.fr/~dr/XPOSE2003/tomcat/tomcat.php?rub=25

**Tomcat**

http://www.journaldunet.com/developpeur/java-j2ee/
http://tomcat.apache.org/
http://en.wikipedia.org/wiki/Apache_Tomcat
http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci868204,00.html
http://www.jmdoudoux.fr/java/dej/chap061.htm#chap_61
http://www-igm.univ-mlv.fr/~dr/XPOSE2003/tomcat/tomcat.php?rub=3
« Apache Tomcat 6 » Edition ENI
http://www.editions-eni.fr/_Download/e5353b66-34fb-44bc-82e2-b641ee247083/apache-Tomcat-6_(Extrait-du-livre).pdf

**Jetty**

http://en.wikipedia.org/wiki/Jetty_(web_server)
http://jetty.mortbay.com/jetty/

**Resin**

http://www.caucho.com/resin-3.0/
https://www.gopay.com.cn/resin-doc/features/overview.xtp

**JBoss**

http://fr.wikipedia.org/wiki/JBoss
http://en.wikipedia.org/wiki/JBoss
http://www.jboss.com/fr/index
http://alunasoft.com/JBoss5_New_Features-v002.pdf

**JRun**

http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci783004,00.html
http://coldfused.blogspot.com/2007/11/jrun-updater-7-released.html
http://www.adobe.com/products/jrun/
http://www.adobe.com/support/documentation/en/jrun/4/updater7/releasenotes_4_updater7.pdf
White Paper « jr4_technical_wp.pdf »

**JOnAS**

http://fr.wikipedia.org/wiki/JOnAS
http://fr.wikipedia.org/wiki/OSGi
http://wiki.jonas.objectweb.org/xwiki/bin/view/Main/WebHome

**Comparatif** (sources supplémentaires)

http://www.application-servers.com/
http://java.sun.com/javaee/community/glassfish/
http://www.sap.com/platform/netweaver/index.epx
http://www.itrportal.com/
http://java.sun.com/javase/technologies/security/
http://en.wikipedia.org/wiki/Comparison_of_application_servers
http://en.wikipedia.org/wiki/Comparison_of_web_servers
http://java.sun.com/javaee/overview/compatibility.jsp
http://java.sun.com/j2ee/compatibility_1.4.html
http://www-01.ibm.com/software/webservers/appserv/was/
http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf
http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf