

Java EE 6

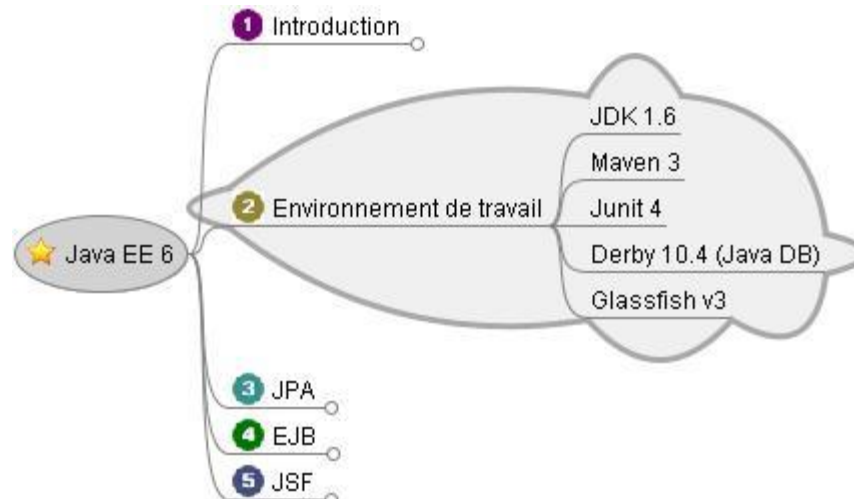
Développer des applications d'entreprise

- **Chapitre 2**
Environnement de travail



Agenda

Planning de la formation



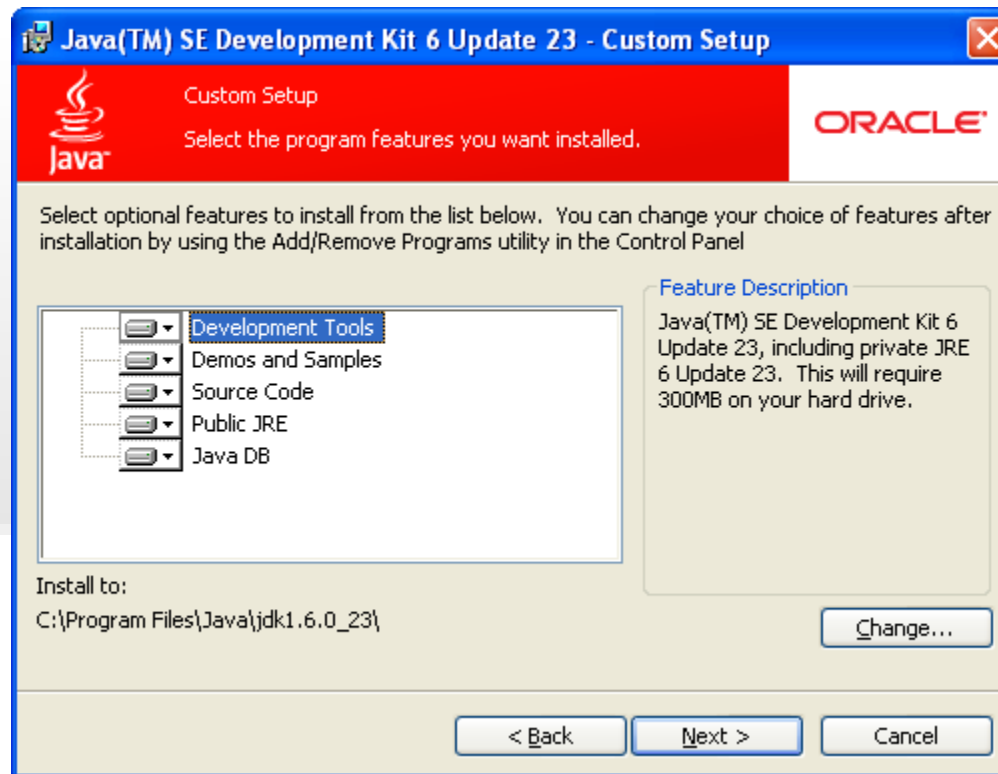
Java 6

Java Development Kit

- Contient :
 - Un compilateur (javac),
 - Une machine virtuelle Java,
 - Un outil de génération de documentation (javadoc),
 - Des outils de monitoring (Visual VM),
 - Un outil pour le packaging (jar),
 - ...

Java 6

Java Development Kit



Maven 3

Construction

- Construire une application Java EE consiste à :
 - Créer du code et des ressources,
 - Compiler les classes de code et de test,
 - Packager le code dans des archives (jar, ear, war, ...) avec éventuellement d'autres librairies externes
- Faire ce travail manuellement est trop long et source d'erreur et il faut automatiser ce qui peut l'être

Maven 3

Construction

- Le premier outil développé par la communauté Java était Apache Ant pour créer des scripts portable car Ant est lui-même écrit en Java
- Mais Ant a vite été poussé dans ses limites pour les projets complexes avec de nombreuses dépendances à d'autres projets
- En 2002, Apache Maven a été créé, au départ pour les besoins internes d'un projet Apache

Maven 3

Construction

- Maven 3 peut faire ce que faisait Ant, mais va beaucoup plus loin aussi :
 - Construction de projet,
 - Gestion des librairies avec dépendances et versions,
 - Plateforme extensible par plugins : contrôle qualité, documentation, travail en équipe, ...

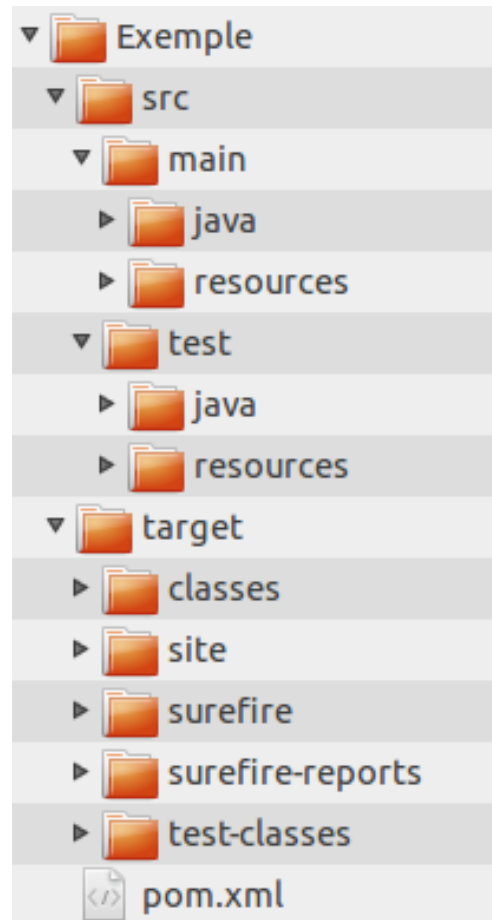
Maven 3

Construction

- Maven 3 est basé sur le principe de "convention plutôt que configuration"
- Maven 3 apporte un système de description standard des projets et différentes conventions comme une arborescence standard pour les projets
- Les plug-ins (appelés mojos) permettent de choisir les services mis en œuvre

Maven 3

Construction



Maven 3

Project descriptor - POM

- Un projet Maven a besoin de suivre des standards et de définir ses besoins spécifiques dans un fichier de description (project descriptor) appelé POM (Project Object Model)
- Le fichier POM est un document xml (pom.xml) placé à la racine du projet
- Au minimum ce fichier contient la description de l'identité du projet, c'est à dire son groupId, artifactId, version et type de packaging

Maven 3

Project descriptor - POM

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" ↵
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ↵
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 ↵
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.apress.javaee6</groupId>
    <artifactId>chapter01</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
</project>
```

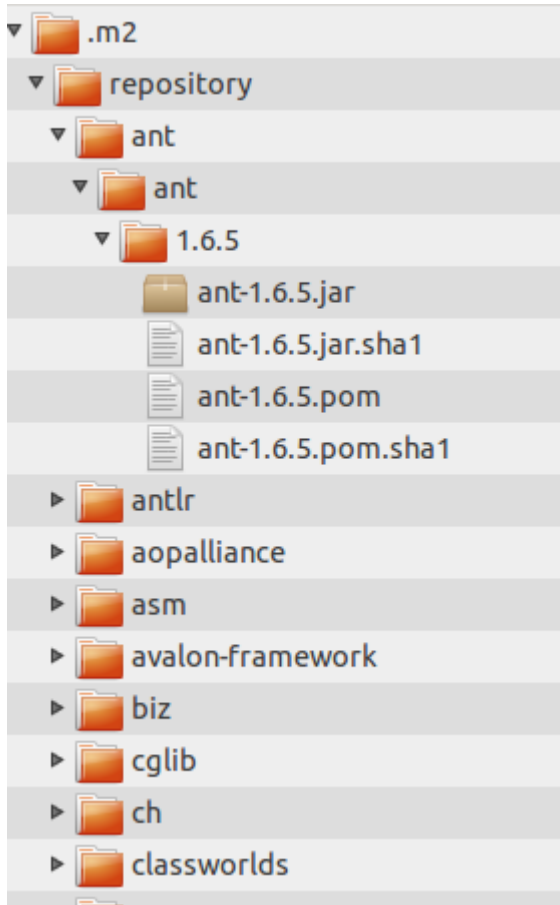
Maven 3

Gestion des artefacts

- Au delà de la construction des artefacts, Maven permet des les archiver et de les partager
- Maven utilise un repository local sur votre disque dur (par défaut dans %USER_HOME%/.m2/repository) pour ranger tous les artefacts manipulés par les pom
- Maven télécharge les artefacts qui lui manquent, par défaut dans le repository à l'adresse <http://repo1.maven.org/maven2>

Maven 3

Exemple de repository local



Maven 3

Dépendances

- Les dépendances entre artefacts sont déclarées dans les pom grâce à leur identité
- Maven 2 gère automatiquement les dépendances transitives et télécharge les artefacts et plug-ins nécessaires

Maven 3

Dépendances

pom.xml

```
...
<dependencies>
  <dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>1.1.0</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.glassfish</groupId>
    <artifactId>javax.ejb</artifactId>
    <version>3.0</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
...
```

Maven 3

Dépendances

- Les dépendances ont une portée appelée scope :
 - test : la librairie est utilisée pour la compilation et pour jouer les tests mais n'est pas packagées dans l'artefact
 - provided : la librairie est fournie par l'environnement (serveur de persistance, serveur d'application, ...) et sert seulement à compiler le code
 - compile (valeur par défaut) : la librairie est nécessaire pour la compilation et l'exécution
 - runtime : la librairie est nécessaire pour l'exécution mais est exclue de la compilation (composant JSF, taglib JSTL, ...)

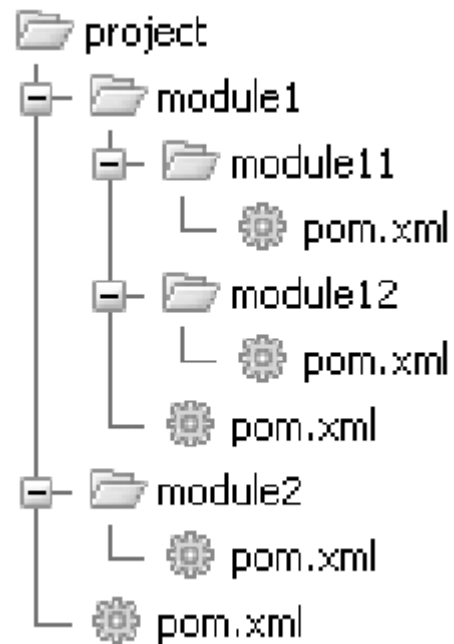
Maven 3

Modularité

- Maven fournit un mécanisme basé sur les modules :
 - Chaque module est un projet Maven,
 - Maven est capable de gérer les dépendances entre modules pour la construction du projet,
 - Pour faciliter la réutilisation de paramètres communs, les POM peuvent hériter de POM de projets parents

Maven 3

Modularité



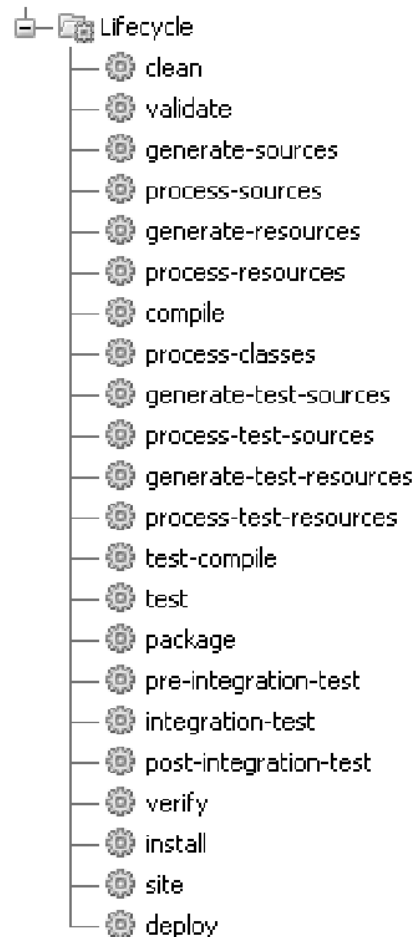
Maven 3

Plug-ins et cycle de vie

- Maven définit un cycle de vie avec différentes phases :
 - nettoyage des ressources,
 - validation du projet,
 - génération de sources,
 - compilation de classes,
 - exécution de tests,
 - packaging,
 - installation dans le repository

Maven 3

Plug-ins et cycle de vie



Maven 3

Plug-ins et cycle de vie

- Le cycle de vie est la colonne vertébrale sur laquelle les plug-ins Maven se branchent
- Suivant le type de projet, les plug-ins (appelés mojos) peuvent être différents
- Dans le POM, vous pouvez brancher de nouveaux plug-ins à une étape particulière, changer la configuration d'un plug-in et ainsi de suite

Maven 3

Installation

- Télécharger la dernière version sur <http://maven.apache.org>
- Dézipper l'archive zip quelque part
- Ajouter dans le path le chemin vers le dossier bin de l'archive dézippée
- Ne pas oublier que pour fonctionner, Maven a besoin d'Internet pour télécharger les artefacts et plug-ins

Maven 3

Usage

- On peut lancer une phase du cycle Maven ou un plug-in spécifique si besoin
- Les commandes principales sont :
 - mvn clean : nettoie toutes les classes
 - mvn compile : compile les classes java du main
 - mvn test-compile : compile les classes de test
 - mvn test : compile et exécute les tests
 - mvn package : compile, teste et package
 - mvn install : compile, teste, package et installe l'artefact dans le repository
 - mvn clean install : nettoie puis installe

JUnit 4

Présentation

- C'est un framework open source pour écrire et exécuter des tests rejouables.
- Le framework inclut :
 - Des "assertions" pour tester les résultats escomptés
 - Des moyens de partager des jeux de données
 - Des "runners" pour exécuter les tests
- JUnit est le standard de fait du langage Java
- Il consiste en un simple jar à télécharger à <http://www.junit.org> (ou utiliser une dépendance Maven)

JUnit 4

Historique

- JUnit a été écrit à l'origine par Erich Gamma (Gang Of Four) et Kent Beck (XP) en 1998, s'inspirant du framework SUnit pour Smalltalk écrit par Kent Beck
- Il est rapidement devenu le framework de tests unitaires le plus utilisé dans le monde java
- JUnit a inspiré toute la famille des xUnit comme NUnit (.Net), pyUnit (Python), CppUnit (C++), dUnit (Delphi), ...

JUnit 4

Fonctionnement

- Depuis la version 4, l'écriture de tests unitaires est simplifiée par l'utilisation des annotations, des imports statiques et autres nouvelles fonctionnalités de Java.

JUnit 4

Fonctionnement

Customer.java

```
public class Customer {  
    private Long id;  
    private String firstName;  
    private String lastName;  
    private String email;  
    private String phoneNumber;  
    private Date dateOfBirth;  
    private Date creationDate;  
    // Constructors, getters, setters  
}
```

CustomerHelper.java

```
public class CustomerHelper {  
    private int ageCalcResult;  
    private Customer customer;  
  
    public void calculateAge() {  
        Date dateOfBirth = customer.getDateOfBirth();  
        Calendar birth = new GregorianCalendar();  
        birth.setTime(dateOfBirth);  
        Calendar now = new GregorianCalendar(2001, 1, 1);  
        ageCalcResult = now.get(Calendar.YEAR) -  
            birth.get(Calendar.YEAR);  
    }  
    // Not implemented yet  
    public Date getNextBirthDay() {  
        return null;  
    }  
    public void clear() {  
        ageCalcResult=0;  
        customer=null;  
    }  
}
```

JUnit 4

Fonctionnement

CustomerHelperTest.java

```
import java.util.Calendar;
import java.util.GregorianCalendar;

import org.junit.Before;
import org.junit.Ignore;
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class CustomerHelperTest {
    private CustomerHelper customerHelper = new
CustomerHelper();

    @Before
    public void clearCustomerHelper() {
        customerHelper.clear();
    }

    @Test
    public void notNegative() {
        Customer customer = new Customer();
        customer.setDateOfBirth(new
GregorianCalendar(1975, 5, 27).getTime());
        customerHelper.setCustomer(customer);
        customerHelper.calculateAge();

        int calculatedAge =
customerHelper.getAgeCalcResult();
        assert calculatedAge >= 0;
    }

    @Test
    public void expectedValue() {
        int expectedAge = 33;
        Calendar birth = new GregorianCalendar();
        birth.roll(Calendar.YEAR, expectedAge * (-1));
        birth.roll(Calendar.DAY_OF_YEAR, -1);
        Customer customer = new Customer();
        customer.setDateOfBirth(birth.getTime());
        customerHelper.setCustomer(customer);
        customerHelper.calculateAge();
        assertEquals(expectedAge,
customerHelper.getAgeCalcResult());
    }

    @Ignore("not ready yet")
    @Test
    public void nextBirthDay() {
        // some work to do
    }
}
```

JUnit 4

Règles d'écriture

- La classe de test n'a pas à hériter de quoi que ce soit, mais doit posséder au moins une méthode annotée avec `@Test`
- Les méthodes de test sont annotées avec `@Test`, retournent `void` et n'ont pas de paramètre
- On peut ajouter des paramètres à l'annotation, par exemple `expected` pour déclarer que le test attend une exception

JUnit 4

Règles d'écriture

- Pour ignorer un test on peut ajouter l'annotation `@Ignore` avant ou après l'annotation `@Test`
- Le test runner affichera le nombre de tests ignorés par cette annotation

JUnit 4

Méthodes assert...

- La classe Assert fournit un certain nombre de méthodes statiques pour tester les résultats :
 - AssertEquals,
 - AssertTrue
 - ...
- Vous pouvez utiliser la syntaxe Assert.assertEquals ou importer statiquement Assert
- Vous pouvez aussi utiliser le mot clé assert de java mais il faudra ajouter une option à l'exécution des tests (-ea)

JUnit 4

Fixtures

- Les fixtures sont des méthodes pour initialiser ou nettoyer des objets communs aux méthodes de tests
- Les annotation `@Before` et `@After` sont exécutées avant et après chaque méthode de test
- Les annotation `@BeforeClass` et `@AfterClass` sont exécutées une seule fois par classe de test. Ces méthodes doivent être uniques et statiques

JUnit 4

Lancement des tests

- En ligne de commande :
 - Ajouter le jar junit au CLASSPATH (ou ajouter la dépendance Maven)
 - `java -ea org.junit.runner.JUnitCore`
`com.apress.javaee6.CustomerHelperTest`
 - Ajouter l'option `-ea` si le mot clé `assert` est utilisé

JUnit 4

Lancement des tests

Console

```
JUnit version 4.5
..E.I
Time: 0.016
There was 1 failure:
1) expectedValue(com.apress.javaee6.CustomerHelperTest)
java.lang.AssertionError: at
CustomerHelperTest.expectedValue(CustomerHelperTest.java:52)
FAILURES!!!
Tests run: 3,  Failures: 1
```

JUnit 4

Intégration

- JUnit est très bien intégré à la plupart des IDE (IntelliJ IDEA, Eclipse, NetBeans, ...) aussi bien pour l'exécution que pour la création
- JUnit est intégré à Maven grâce au plugin Surefire. C'est ce plugin qui est lancé quand on exécute la commande `mvn test`

Derby 10.4

Présentation

- Initialement appelé Cloudscape, Derby est une base de données développée en Java et donnée à la fondation Apache par IBM
- Sun a créé sa propre distribution appelée JavaDB
- La base se caractérise par une très faible empreinte mémoire (2 Mo)
- C'est une base de données relationnelle très fonctionnelle, qui supporte les transactions et peut très facilement être embarquée

Derby 10.4

Installation

- Elle est installée par défaut avec le JDK 1.6 si vous avez coché l'option durant l'installation
- Elle peut être téléchargée séparément depuis <http://db.apache.org>
- Une fois installée (dézippée), positionner la variable d'environnement DERBY_HOME avec pour valeur le dossier de derby, et ajouter à la variable PATH %DERBY_HOME%\bin sous windows et \$DERBY_HOME/bin sous linux

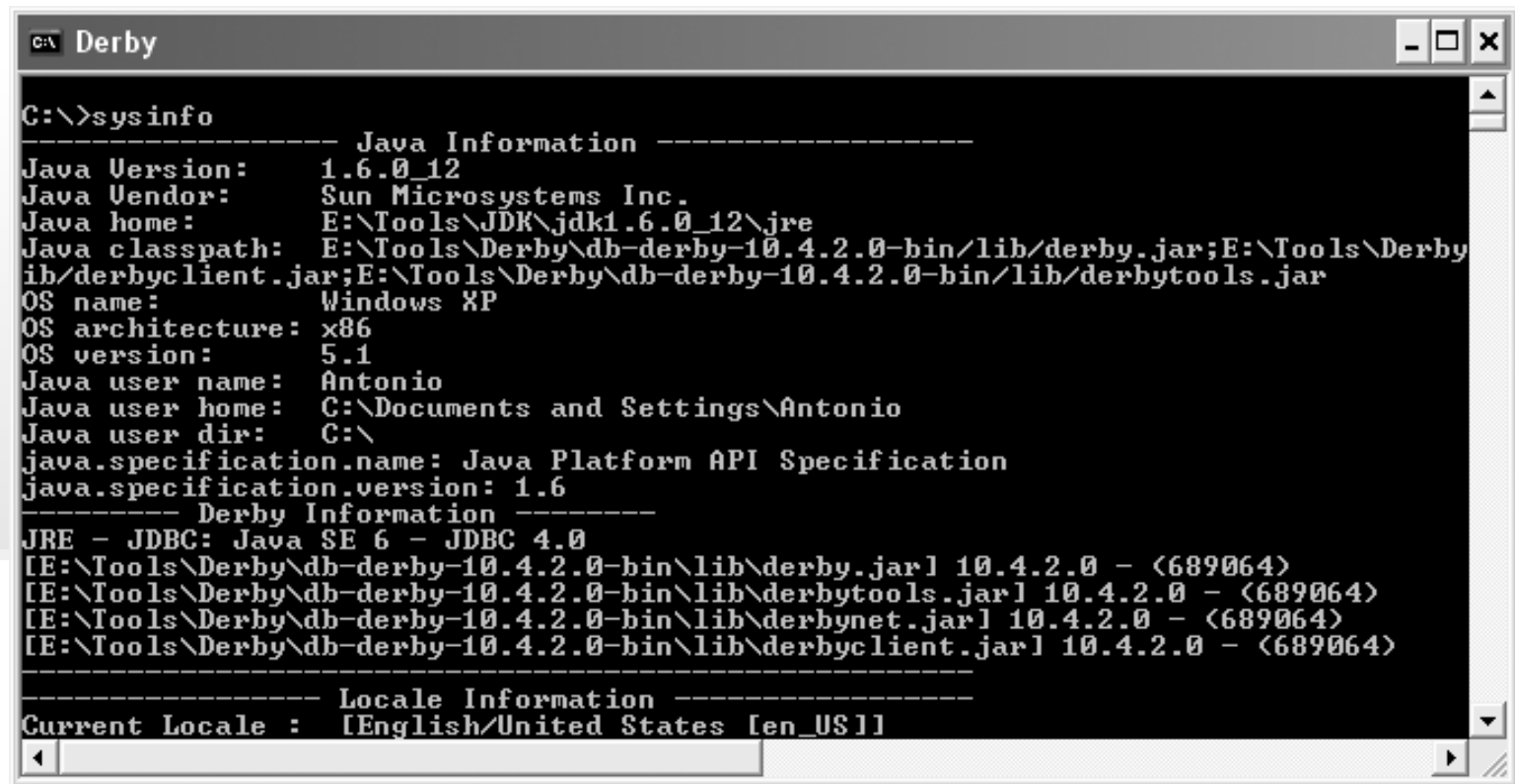
Derby 10.4

Outils

- Le dossier bin contient plusieurs commandes de lancement et des utilitaires
- Parmi les utilitaires, la commande sysinfo affiche des informations sur l'environnement d'exécution de java

Derby 10.4

Outils



```
C:\>sysinfo
----- Java Information -----
Java Version:      1.6.0_12
Java Vendor:       Sun Microsystems Inc.
Java home:         E:\Tools\JDK\jdk1.6.0_12\jre
Java classpath:    E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derby.jar;E:\Tools\Derby
ib\derbyclient.jar;E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derbytools.jar
OS name:           Windows XP
OS architecture:  x86
OS version:        5.1
Java user name:    Antonio
Java user home:    C:\Documents and Settings\Antonio
Java user dir:     C:\
java.specification.name: Java Platform API Specification
java.specification.version: 1.6
----- Derby Information -----
JRE - JDBC: Java SE 6 - JDBC 4.0
[E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derby.jar] 10.4.2.0 - (689064)
[E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derbytools.jar] 10.4.2.0 - (689064)
[E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derbynet.jar] 10.4.2.0 - (689064)
[E:\Tools\Derby\db-derby-10.4.2.0-bin\lib\derbyclient.jar] 10.4.2.0 - (689064)
----- Locale Information -----
Current Locale : [English/United States [en_US]]
```

Derby 10.4

Utilisation

- La commande ij permet de saisir des commandes SQL

Console

```
ij
ij> connect 'jdbc:derby://localhost:1527/maBase;create=true';
ij> create table customer (custId int primary key, firstname varchar(20), lastname
varchar(20));
ij> describe customer;
COLUMN_NAME | TYPE_NAME | DEC& | NUM& | COLUM& | COLUMN_DEF | CHAR_OCTE& | IS_NULL&
-----
CUSTID      | INTEGER  | 0     | 10  | 10      | NULL        | NULL        | NO
FIRSTNAME   | VARCHAR  | NULL  | NULL | 20      | NULL        | 40          | YES
LASTNAME    | VARCHAR  | NULL  | NULL | 20      | NULL        | 40          | YES
```


Derby 10.4

Utilisation

Console

```
ij> insert into customer values (1, 'Fred', 'Chene');
ij> insert into customer values (2, 'Sylvain', 'Verin');
ij> insert into customer values (3, 'Robin', 'Riou');
ij> select count(*) from customer;
1
-----
3
ij> select * from customer where custId=3;
CUSTID      |FIRSTNAME                |LASTNAME
-----
3           |Robin                    |Riou
ij> exit;
```

Derby 10.4

Utilisation

- La commande dblook permet de récupérer la DDL d'une base (Data Definition Language)

Console

```
dblook -d 'jdbc:derby://localhost:1527/maBase'
-- Source database is: Chapter01DB
-- Connection URL is: jdbc:derby://localhost:1527/Chapter01DB
-- appendLogs: false
-- -----
-- DDL Statements for tables
-- -----
CREATE TABLE "APP"."CUSTOMER" ("CUSTID" INTEGER NOT NULL, "FIRSTNAME" →
VARCHAR(20), "LASTNAME" VARCHAR(20));
-- -----
-- DDL Statements for keys
-- -----
-- primary
ALTER TABLE "APP"."CUSTOMER" ADD CONSTRAINT "SQL0903154616250" →
PRIMARY KEY ("CUSTID");
```

GlassFish v3

Présentation

- Relativement nouveau, ce serveur d'applications est néanmoins très utilisé et possède un grand nombre de développeurs
- C'est l'implémentation de référence de Java EE
- C'est lui que vous téléchargez quand vous téléchargez le Java EE SDK de Sun
- C'est un serveur d'application robuste pouvant travailler en production
- C'est un outil open source <http://glassfish.org>

GlassFish v3

Historique

- A l'origine, Tomcat a été donné par Sun à la fondation Apache
- Depuis, Sun a toujours continué d'intégrer Tomcat dans ses produits
- En 2005, Sun a créé le projet GlassFish pour développer un serveur d'application certifié JavaEE
- La version 1.0 est sortie en mai 2006
- Le conteneur web était très proche de Tomcat

GlassFish v3

Historique

- GlassFish v2 est sorti en septembre 2007
- GlassFish v3 vient de sortir fin 2009
- La différence entre la version communautaire et la version commerciale consiste uniquement en la mise à disposition de patches et de d'outils supplémentaires de monitoring (GlassFish Enterprise Manager)
- Les versions sont disponibles depuis <http://glassfish.org> et <http://www.sun.com/appserver>

GlassFish v3

Historique

- GlassFish v2 était l'implémentation de référence de Java EE 5
- GlassFish v3 est l'implémentation de référence de Java EE 6
- Début 2010, GlassFish v3 est le seul serveur certifié Java EE 6

GlassFish v3

Architecture

- GlassFish v3 est construit autour d'un noyau OSGI, ici l'implémentation Felix de la fondation Apache
- OSGI est un standard pour la gestion et la découverte de composants dynamiques. On peut installer, démarrer, arrêter, mettre à jour et désinstaller des composants sans redémarrage.
- OSGI n'est pas visible par les développeurs JEE

GlassFish v3

Architecture

- La modularité et l'extensibilité de GlassFish se traduit par le fait qu'il peut "grossir" suivant les besoins.
- Démarrant en moins de 5 secondes, c'est un simple serveur web écoutant des commandes d'administration
- En déployant un war, le conteneur web est chargé et démarré et l'application est déployée. Démarrer le conteneur web prend environ 3 secondes et le déploiement environ 1 seconde

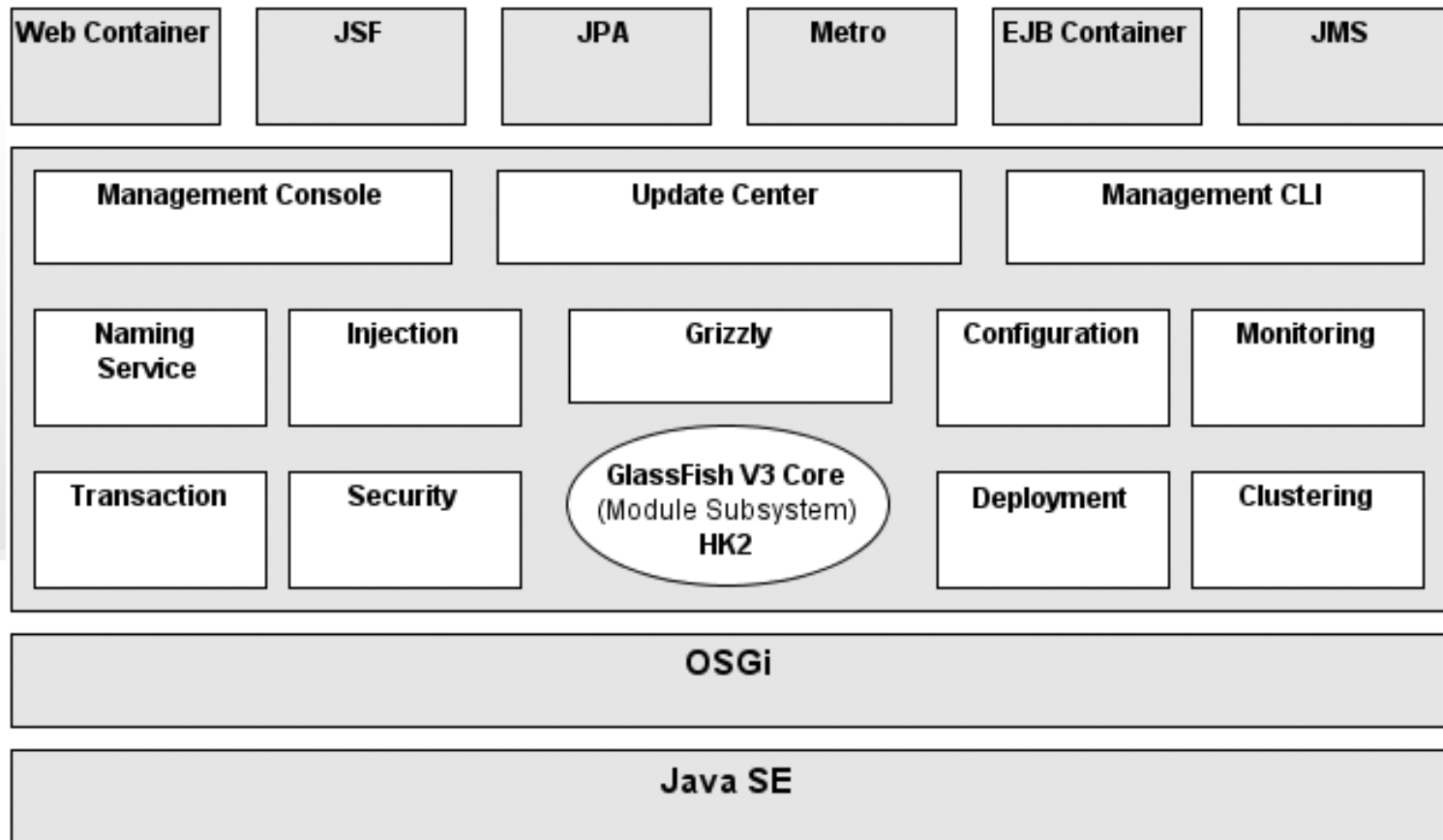
GlassFish v3

Architecture

- En déployant un EJB jar, le conteneur d'EJBs est de la même façon chargé et démarré pour déployer l'application et ce de façon toujours aussi rapide
- Cette souplesse et rapidité en fait un serveur extrêmement pratique en période de développement également
- La console d'administration, la ligne de commande et le fichier de configuration central sont tous extensibles

GlassFish v3

Architecture



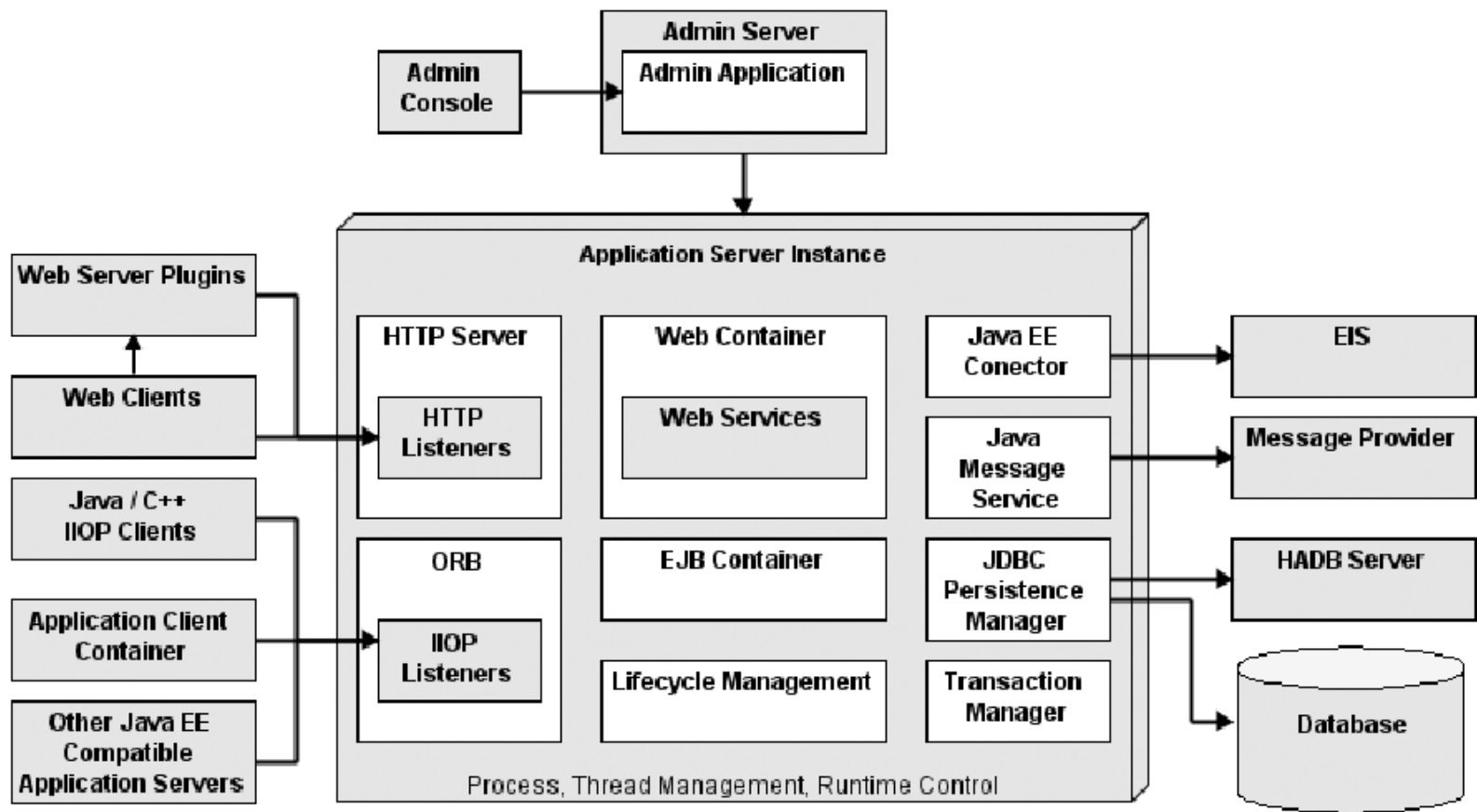
GlassFish v3

Mises à jour

- Le fonctionnement modulaire de GlassFish permet de choisir et mélanger les modules, comme pour les IDE ou extensions Firefox
- L'Update Center permet de gérer en ligne de commande (commande pkg) ou de façon graphique (console) l'environnement d'exécution
- L'Update Center permet de mettre à jour et de rajouter ou enlever des fonctionnalités (support Grails, conteneur de portlets, ...)

GlassFish v3

Sous-projets



GlassFish v3

Sous-projets

- Toutes les parties de GlassFish sont développées dans le cadre de sous-projets qui peuvent parfois être réutilisés de façon indépendante dans d'autres projets
- Par exemple
 - OpenMQ est l'implémentation de JMS, <http://openmq.dev.java.net>
 - Metro est l'implémentation des web services, <http://metro.dev.java.net>
 - Mojarra est l'implémentation de JSF, <http://mojarra.dev.java.net>

GlassFish v3

Administration

- GlassFish propose 2 modes d'administration : en ligne de commande ou via la console graphique. Ces deux modes s'appuient fortement sur JMX
- Presque toute la configuration est stockée dans le fichier `domain.xml` du domaine (dans `domains/domain1/config`), mais ce fichier ne doit pas être édité à la main, utiliser la console ou les outils en ligne de commande

GlassFish v3

Console d'administration

- C'est une interface web
- Elle est utile à la fois aux développeurs et aux administrateurs système
- On peut accéder à tous les paramètres, aux logs, au cache, à l'état du système, à l'annuaire JNDI, pools JDBC, ...
- GlassFish v3 peut aussi être mis en cluster
- Par défaut, la console est disponible à <http://localhost:4848>

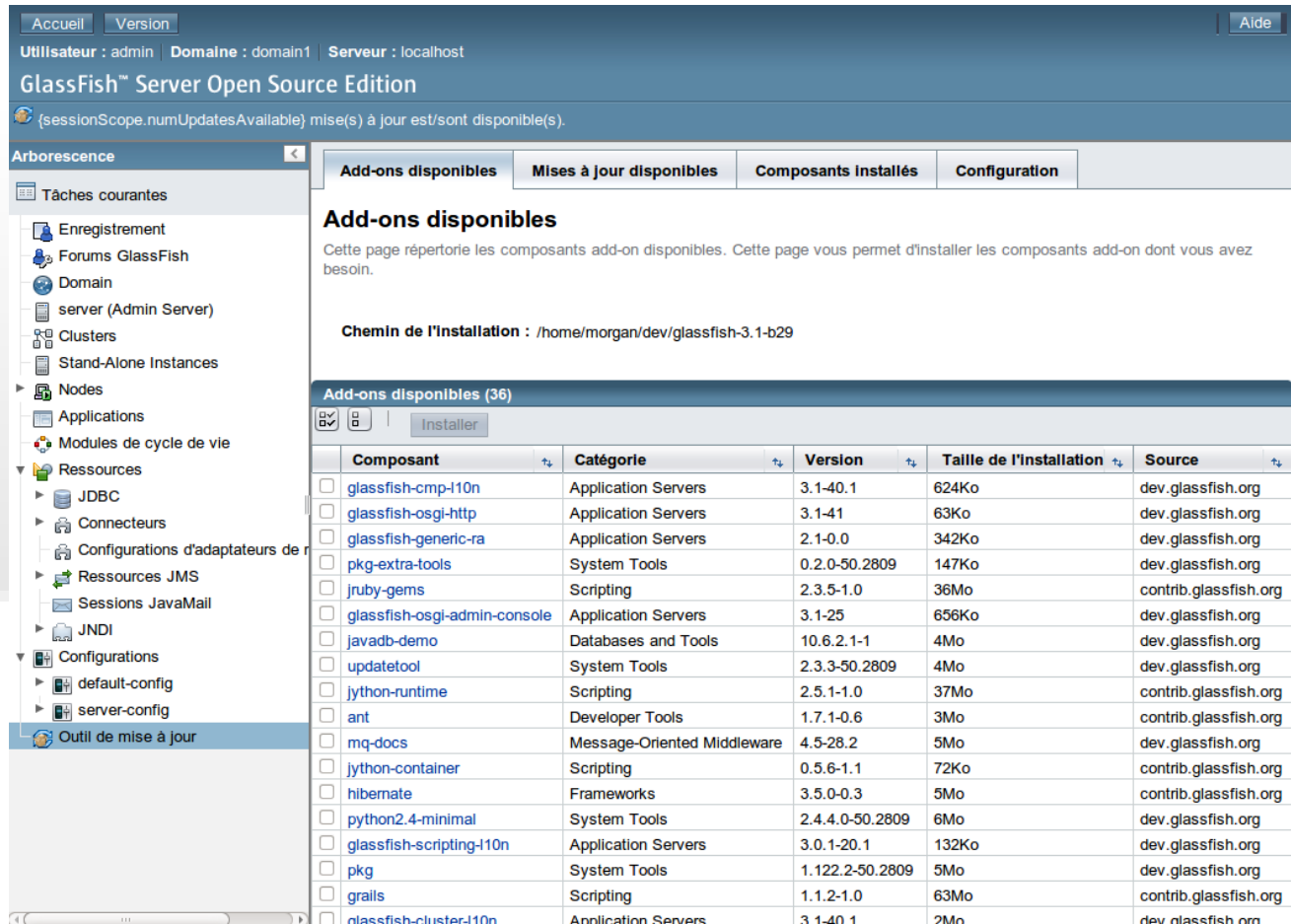
GlassFish v3

Console d'administration

- On peut choisir d'installer GlassFish avec ou sans utilisateur pour se connecter à la Console
- Par défaut, admin/adminadmin est utilisé

GlassFish v3

Console d'administration



Accueil Version Aide

Utilisateur : admin Domaine : domain1 Serveur : localhost

GlassFish™ Server Open Source Edition

{sessionScope.numUpdatesAvailable} mise(s) à jour est/sont disponible(s).

Arborescence

- Tâches courantes
- Enregistrement
- Forums GlassFish
- Domain
- server (Admin Server)
- Clusters
- Stand-Alone Instances
- Nodes
- Applications
- Modules de cycle de vie
- Ressources
 - JDBC
 - Connecteurs
 - Configurations d'adaptateurs de
 - Ressources JMS
 - Sessions JavaMail
 - JNDI
- Configurations
 - default-config
 - server-config
- Outil de mise à jour

Add-ons disponibles Mises à jour disponibles Composants installés Configuration

Add-ons disponibles

Cette page répertorie les composants add-on disponibles. Cette page vous permet d'installer les composants add-on dont vous avez besoin.

Chemin de l'installation : /home/morgan/dev/glassfish-3.1-b29

Add-ons disponibles (36)

Composant	Catégorie	Version	Taille de l'installation	Source
<input type="checkbox"/> glassfish-cmp-110n	Application Servers	3.1-40.1	624Ko	dev.glassfish.org
<input type="checkbox"/> glassfish-osgi-http	Application Servers	3.1-41	63Ko	dev.glassfish.org
<input type="checkbox"/> glassfish-generic-ra	Application Servers	2.1-0.0	342Ko	dev.glassfish.org
<input type="checkbox"/> pkg-extra-tools	System Tools	0.2.0-50.2809	147Ko	dev.glassfish.org
<input type="checkbox"/> jruby-gems	Scripting	2.3.5-1.0	36Mo	contrib.glassfish.org
<input type="checkbox"/> glassfish-osgi-admin-console	Application Servers	3.1-25	656Ko	dev.glassfish.org
<input type="checkbox"/> javadb-demo	Databases and Tools	10.6.2.1-1	4Mo	dev.glassfish.org
<input type="checkbox"/> updatetool	System Tools	2.3.3-50.2809	4Mo	dev.glassfish.org
<input type="checkbox"/> jython-runtime	Scripting	2.5.1-1.0	37Mo	contrib.glassfish.org
<input type="checkbox"/> ant	Developer Tools	1.7.1-0.6	3Mo	contrib.glassfish.org
<input type="checkbox"/> mq-docs	Message-Oriented Middleware	4.5-28.2	5Mo	dev.glassfish.org
<input type="checkbox"/> jython-container	Scripting	0.5.6-1.1	72Ko	contrib.glassfish.org
<input type="checkbox"/> hibernate	Frameworks	3.5.0-0.3	5Mo	contrib.glassfish.org
<input type="checkbox"/> python2.4-minimal	System Tools	2.4.4.0-50.2809	6Mo	dev.glassfish.org
<input type="checkbox"/> glassfish-scripting-110n	Application Servers	3.0.1-20.1	132Ko	dev.glassfish.org
<input type="checkbox"/> pkg	System Tools	1.122.2-50.2809	5Mo	dev.glassfish.org
<input type="checkbox"/> grails	Scripting	1.1.2-1.0	63Mo	contrib.glassfish.org
<input type="checkbox"/> glassfish-cluster-110n	Application Servers	3.1-40.1	2Mo	dev.glassfish.org

GlassFish v3

Ligne de commande

- La commande asadmin est très puissante et permet de créer des instances, des ressources, de déployer des applications et de fournir des données de monitoring
- C'est la commande principalement utilisée dans un environnement de production
- Elle est disponible dans le dossier bin de GlassFish

GlassFish v3

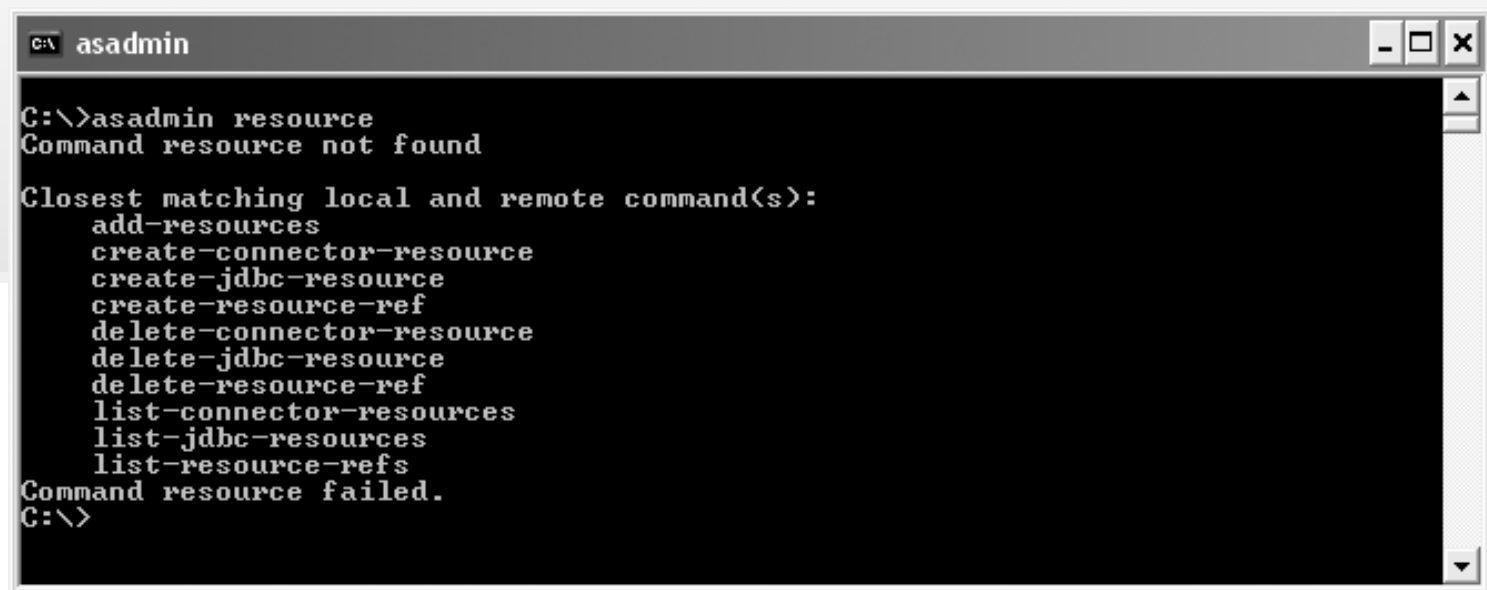
Ligne de commande

- Les commandes principales sont :
 - `asadmin help`
 - `asadmin create-domain`
 - `asadmin start-domain`
 - `asadmin stop-domain`
 - `asadmin deploy`
 - `asadmin deploydir`
 - `asadmin undeploy`

GlassFish v3

Ligne de commande

- Asadmin gère aussi la complétion et propose les commandes proches en cas d'erreur de saisie



```
C:\>asadmin resource
Command resource not found

Closest matching local and remote command(s):
  add-resources
  create-connector-resource
  create-jdbc-resource
  create-resource-ref
  delete-connector-resource
  delete-jdbc-resource
  delete-resource-ref
  list-connector-resources
  list-jdbc-resources
  list-resource-refs
Command resource failed.
C:\>
```

GlassFish v3

Installation

- Le plus simple est d'installer le Java EE SDK. On peut choisir entre JavaEE complet ou le Web Profile
- On peut aussi installer un ensemble complet avec NetBeans et Java EE

GlassFish v3

Démarrage

- NetBeans sait démarrer GlassFish quand on lui demande de tester ou déployer un composant
- On peut aussi démarrer manuellement le domaine avec la commande

`asadmin start-domain domain1`

ou si on n'a qu'un domaine

`asadmin start-domain`

- On peut alors lancer la console à l'adresse <http://localhost:4848> et le serveur par défaut est à l'adresse <http://localhost:8080>

GlassFish v3

Domaines

- Vous pouvez créer des domaines qui sont des instances logiques exécutant vos applications avec la commande
asadmin create-domain monNouveauDomaine
- GlassFish peut exécuter simultanément plusieurs domaines mais il faut que les ports soient différents

GlassFish v3

Voir les logs

- Les logs de GlassFish sont dans le fichier
domains/domain1/logs/server.log
- Vous pouvez aussi afficher les logs dans la console de démarrage du domaine
asadmin start-domain --verbose

Conclusion

Ce qu'il faut retenir

- Il faut différents outils pour développer un projet Java EE 6
- Maven permet de gérer la construction du projet
- JUnit permet de créer des tests unitaires
- Derby permet d'avoir une base embarquée (pour les tests)
- GlassFish permet d'exécuter l'application créée

