	Année Universitaire : 2025|2026 **Enseignante : AYED Asma** AUDITOIRE : 3 ème année MDW - DSI

Atelier L'Architecture Orientée Services (SOA)

I. Partie théorique

1- Le concept Service

- **Composant logiciel** qui exécute une action pour le compte d'un client
- Il traduit le **niveau logique** d'accès aux traitements, plutôt que le niveau physique d'implémentation,

2- Orchestration des services

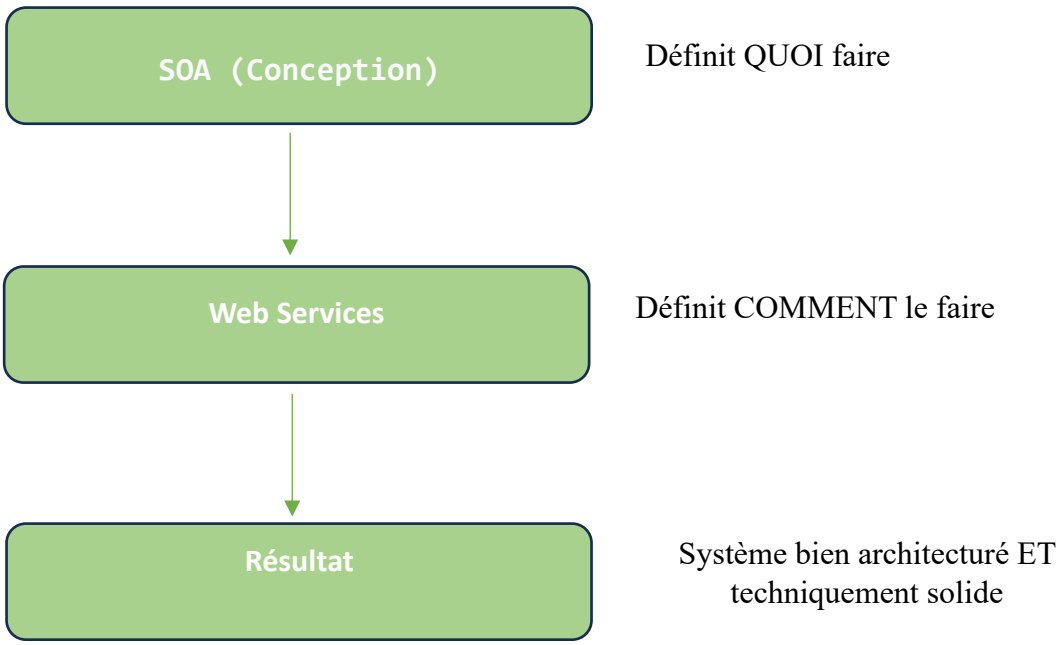
- Les services peuvent être composés (agrégés) dans le but de réaliser un processus donné
- L'orchestration leur permet de **communiquer sans avoir à se connaître** pour préserver leur couplage lâche (leur indépendance)
- **Un moteur d'orchestration se charge d'appeler les services** selon l'enchaînement désiré

3- Qu'est-ce que SOA ?

- « Une vision d'un système destinée à traiter toute application comme un fournisseur de services »
- Tout traitement applicatif est vu comme un service potentiellement utilisable par un client
- Le client est découplé de l'architecture technique du service qui 'il invoque
- SOA véhicule des Messages et non des objets
- Pour adapter les technologies hétérogènes, on fait appel à des Connecteurs/Adaptateurs
- L'Architecture Orientée Services (SOA) est une approche architecturale dans laquelle des composants logiciels appelés services sont utilisés pour répondre aux besoins des utilisateurs finaux ou d'autres services via une communication réseau. Chaque service dans SOA est une unité distincte qui exécute une fonction spécifique et qui est

indépendant des autres services, tout en étant conçu pour être utilisé, réutilisé et combiné à d'autres services dans différents contextes.

- La notion de « service » est le concept important. Les Services Web sont juste un moyen de les implémenter.
- Une bonne SOA est une histoire de conception pas de technologie.
- Les services web forment un très bon ensemble de spécifications pour construire une SOA : indépendance, sécurité, transaction, orchestration, …
- Le concept de **services web** s'inscrit dans l'architecture orientée services (**SOA**, *Service-Oriented Architecture*),
- Un modèle de conception de logiciels qui permet d'organiser les systèmes en services autonomes, modulaires et réutilisables.
- Ces services sont indépendants et communiquent entre eux via des protocoles standards, souvent sur un réseau.
- Composant logiciel qui exécute une action pour le compte d'un client.
- Il traduit le niveau logique d'accès aux traitements, plutôt que le niveau physique d'implémentation (EJB, Servlet…)



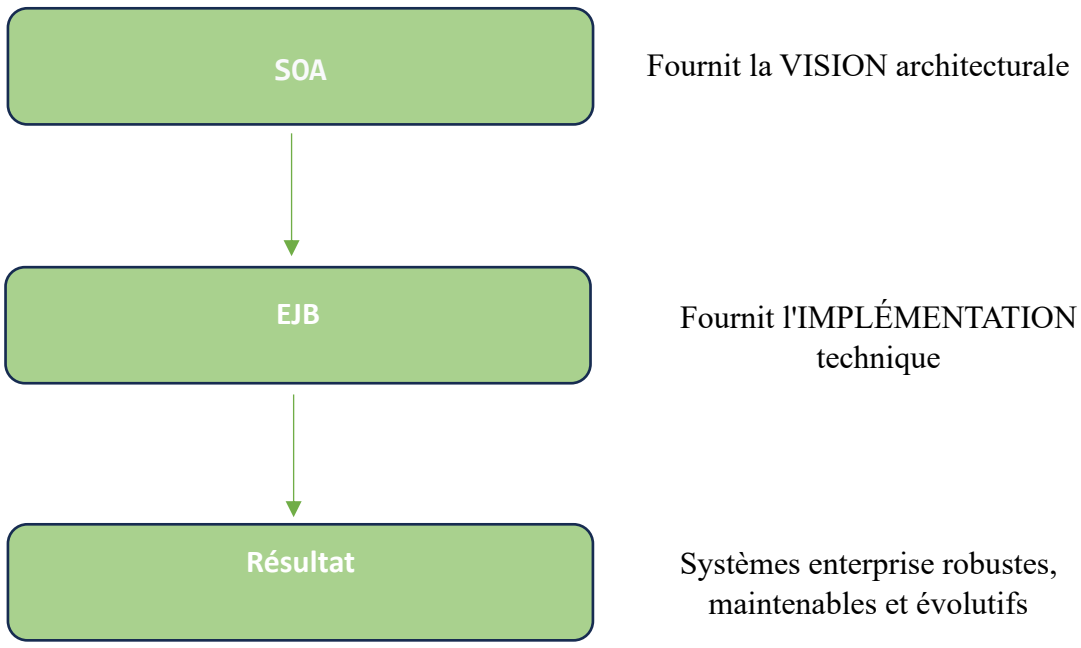
SOA et Web Services sont COMPLÉMENTAIRES mais DISTINCTS :

SOA sans Web Services → Bonne architecture mais interopérabilité limitée

Web Services sans SOA → Interopérabilité mais architecture potentiellement médiocre

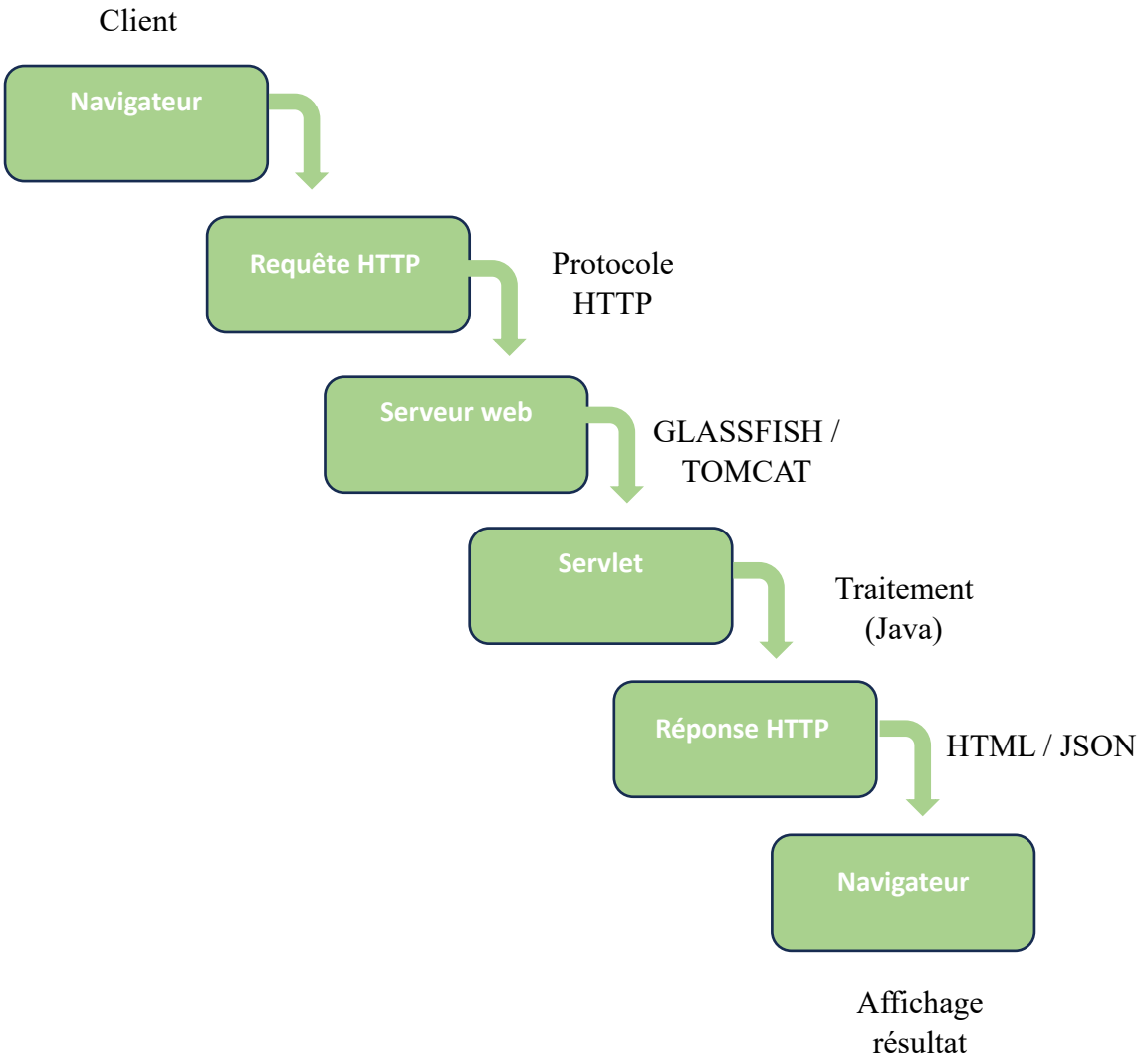
4- Les EJB (Entreprise Java Bean)

- La technologie Enterprise JavaBeans (EJB) est l'architecture de composants côté serveur de Java Platform, Enterprise Edition (Java EE). Elle permet le développement rapide et simplifié d'applications distribuées, transactionnelles, sécurisées et portables basées sur Java.
- Le conteneur d'EJB propose un certain nombre de services qui assurent la gestion :
    - du cycle de vie du bean
    - de l'accès au bean
    - de la sécurité d'accès
    - des accès concurrents
    - des transactions



Une Servlet est un composant Java qui s'exécute sur le serveur web et traite les requêtes HTTP pour générer des réponses dynamiques.

Comme un "contrôleur" dans une application web


Client
Navigateur
Requête HTTP
Protocole HTTP
Serveur web
GLASSFISH / TOMCAT
Servlet
Traitement (Java)
Réponse HTTP
HTML / JSON
Navigateur
Affichage résultat

II. Partie pratique

1- Installation logicielle

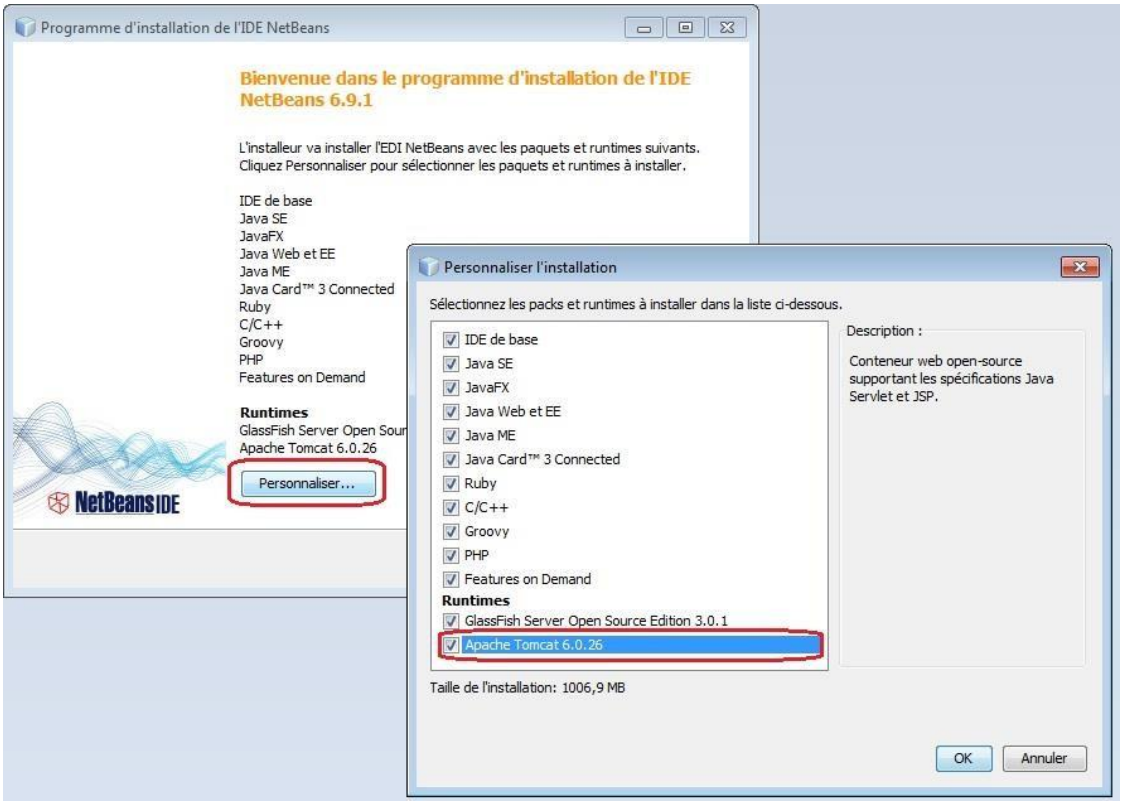
Logiciels obligatoires :

- **un JDK version 1.8 ou supérieur ET Netbeans 6.9.x**
  Le JDK : On a besoin d'un JDK et non pas d'un JRE.

  Installez la version la plus récente à partir du site d'Oracle (février 2011 : Java SE 6 update 23)

  un JDK plutôt qu'u **Netbeans 6.9.x** : On a besoin d'une version qui contient les **serveurs glassfish** et **tomcat**.

  Installez la version (en français si vous préférez) la plus complète sur le site de Netbeans



NOTE : les photos d'écrans sont toutes avec Glassfish v5.0.1 prelude et avec netbeans 6.7.1, comme nous allons utiliser glassfish v5.0.1 final et netbeans 6.9, il se peut qu'il y ait de petites différences.

NE CHANGEZ AUCUN MOT DE PASSE !

Note : si jamais les liens ci-dessus ont changé, google est votre ami pour récupérer des versions à jour.

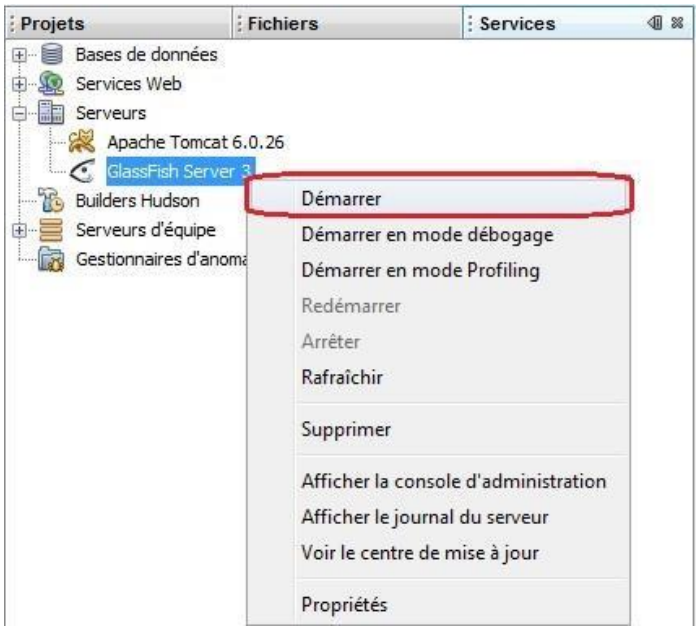
Installez donc le JDK si vous ne l'avez pas, puis netbeans, et passons à la suite. JE CONSEILLE DE DESINSTALLER LA VERSION PRECEDENTE DE NETBEANS ET DE GLASSFISH SI VOUS AVIEZ DEJA UNE ANCIENNE VERSION !

1.1. Vérification de l'installation

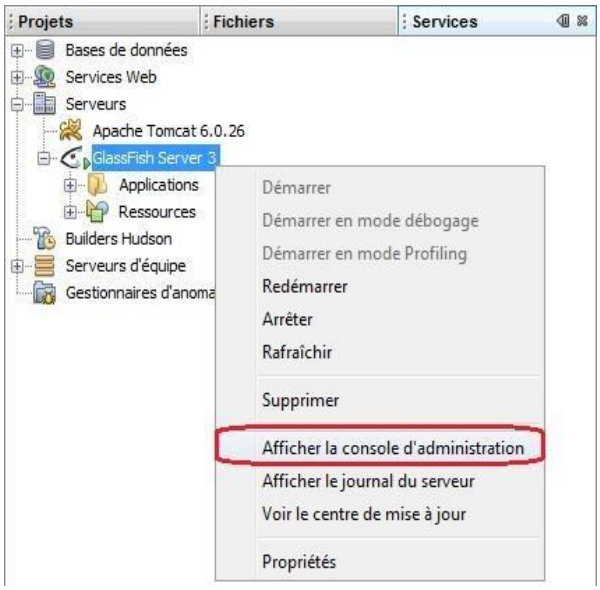
Lancez netbeans une fois l'installation terminée. Cliquez sur l'onglet "services" et ouvrez l'item serveurs. Vous devez voir que tomcat et glassfish sont installés (si vous n'avez pas installé tomcat, ce n'est pas grave)

Vérifions maintenant que tout est ok... L'archive que nous avons installée comprend l'outil de développement netbeans mais aussi le serveur d'application de Sun, appellé Glassfish. Lancez netbeans, puis allez dans l'onglet "services".

Nous allons lancer le serveur pour voir si tout est ok:

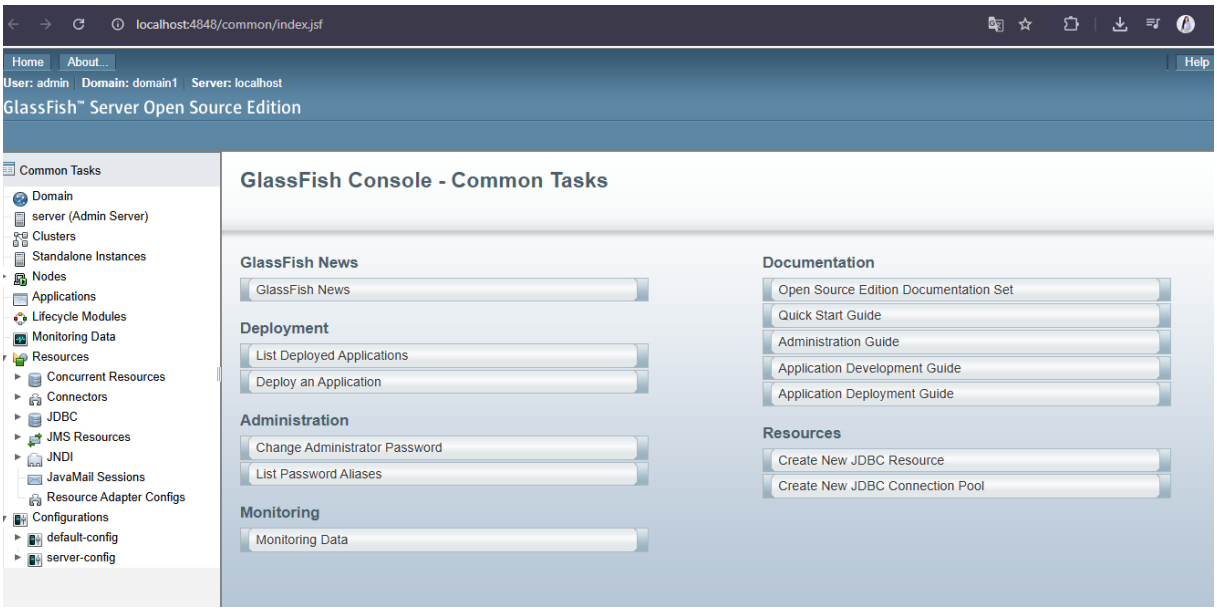


Puis, en demandant à voir la console d'administration, cela va ouvrir la page d'admin dans votre navigateur. En entrant le mot de passe *adminadmin*, une belle page devrait s'afficher. J'espère que vous n'avez pas modifié le mot de passe proposé par défaut, **car dans une classe de TP, les statistiques prouvent que la moitié des élèves ne se souviennent plus de leur mot de passe très rapidement!**



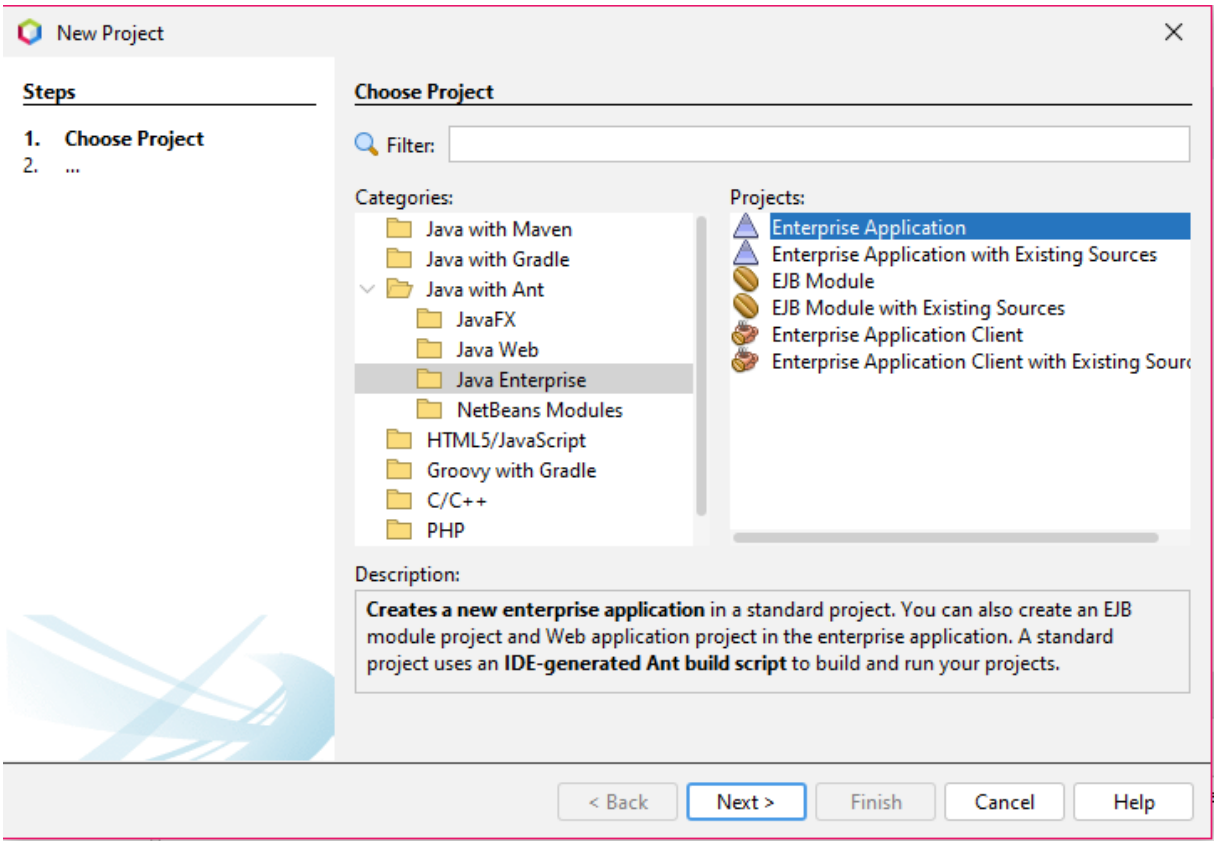
Le screenshot ci-dessous présente la console d'administration. Nous en reparlerons plus tard.

Le fait qu'elle s'affiche dans votre navigateur prouve que l'installation est ok.

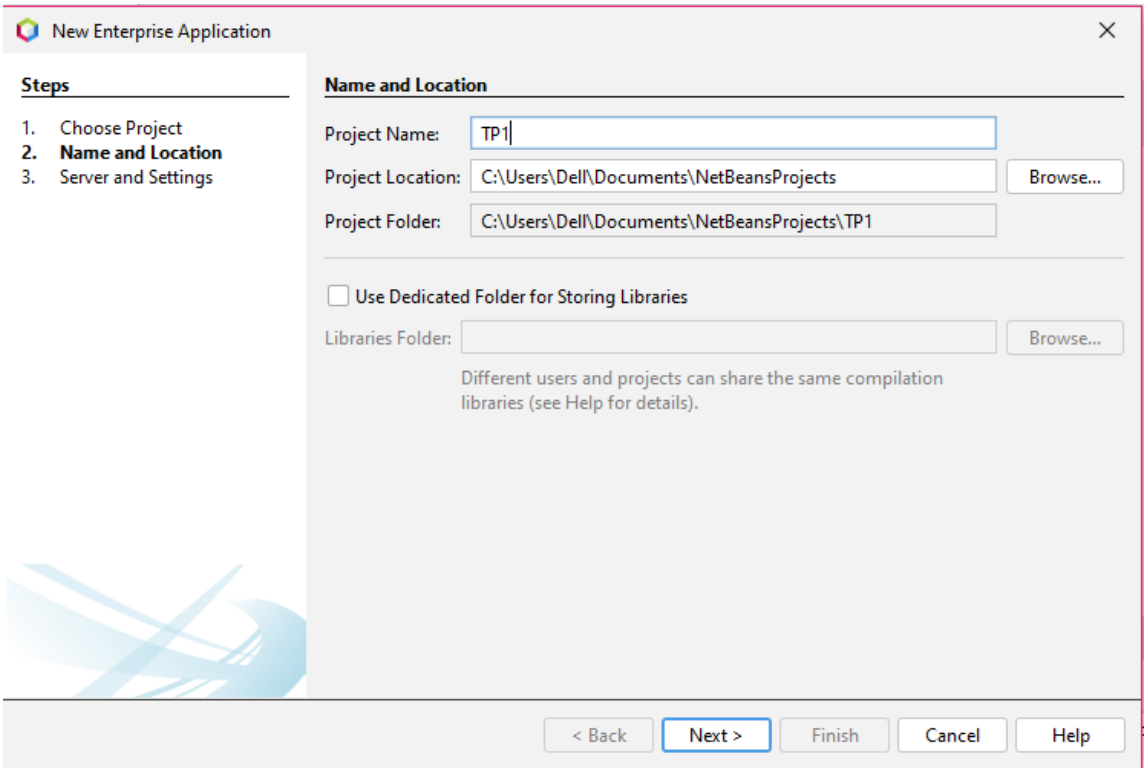


2. Création d'un nouveau projet

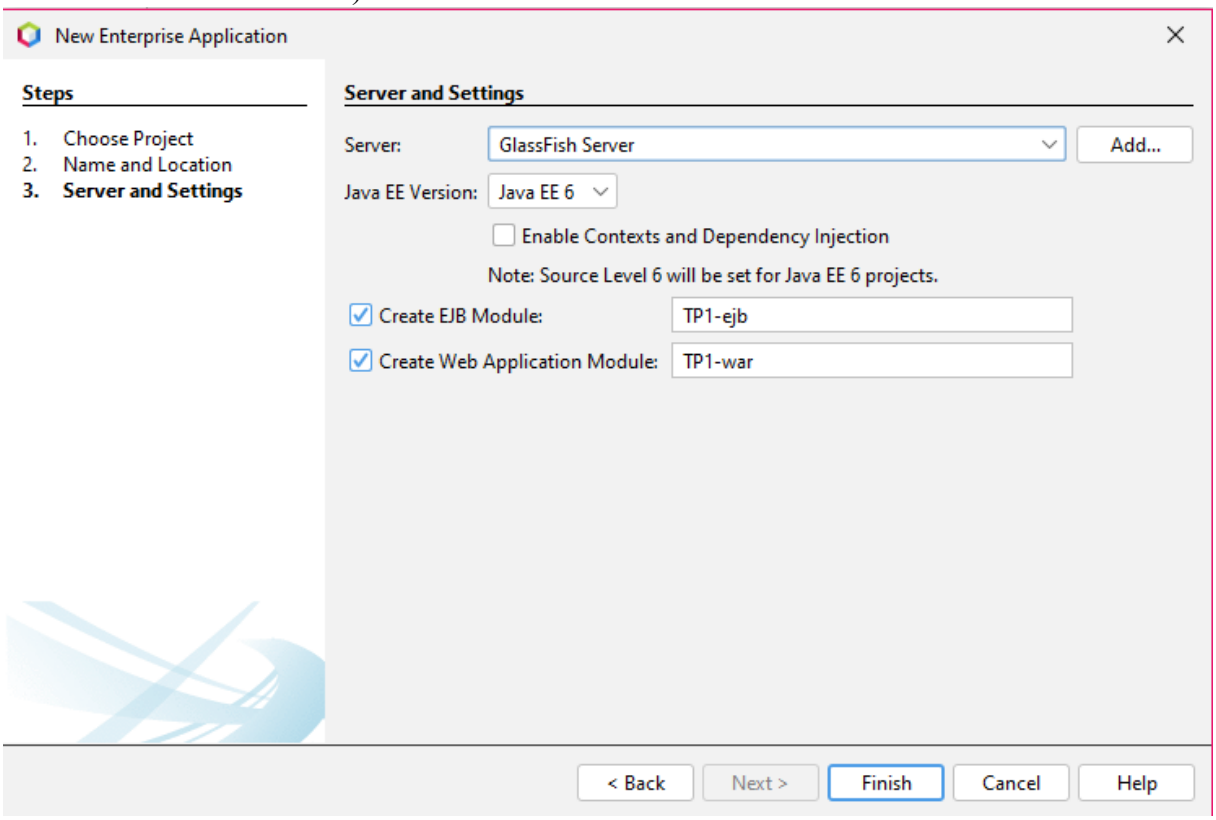
Nous allons créer maintenant un projet pour le TP d'aujourd'hui. Allez dans le menu file /new project, puis dans la fenêtre suivante, sélectionnez Java Entreprise comme catégorie de projet et « enterprise application » comme projet:



Cliquez sur « next », puis donnez un nom à votre projet, cliquez sur next ou suivant...



Indiquez maitenant que nous allons utiliser le serveur d'application de Sun (glassfish v5.0.1 et le mode Java EE6)



Une fois terminé, vous verrez dans l'onglet project « trois sous projets » :

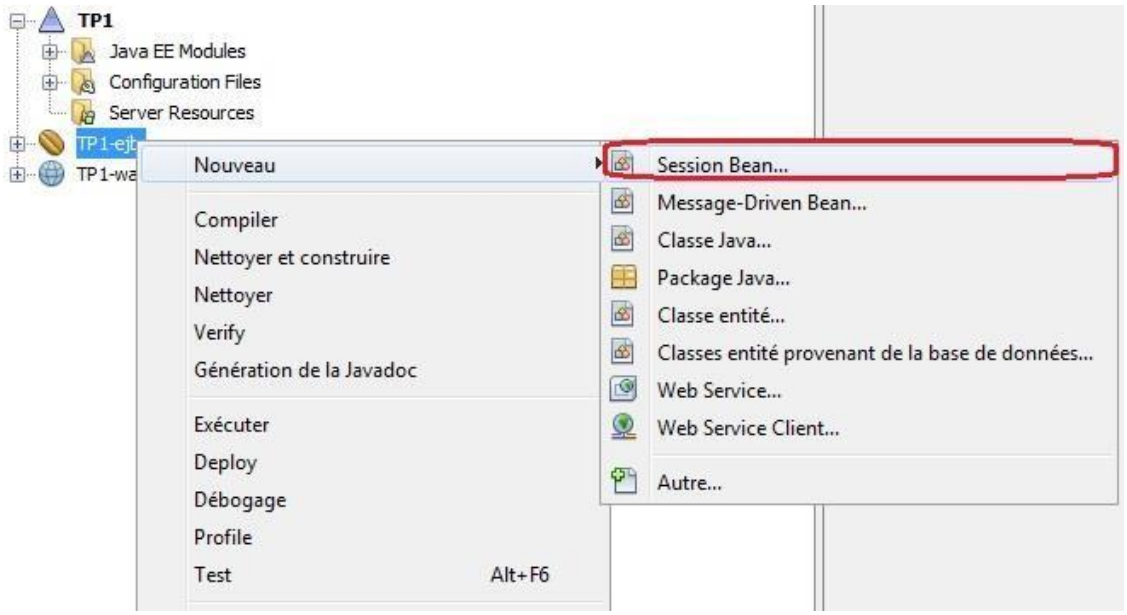
1. Le premier, "TP1", correspond au projet « global » qui contient en fait les archives des deux autres (il correspond au fichier .ear du cours),
2. le second, "TP1-ejb", correspond au .jar qui va contenir tous vos ejbs
3. le troisième, "TP1-war", correspond au .war qui va contenir vos servlets, jsps et pages html.

Le fichier .ear est en fait une archive qui contient les deux autres fichiers : le .jar et le .war, il permet de déployer l'ensemble de la web application d'un seul coup.

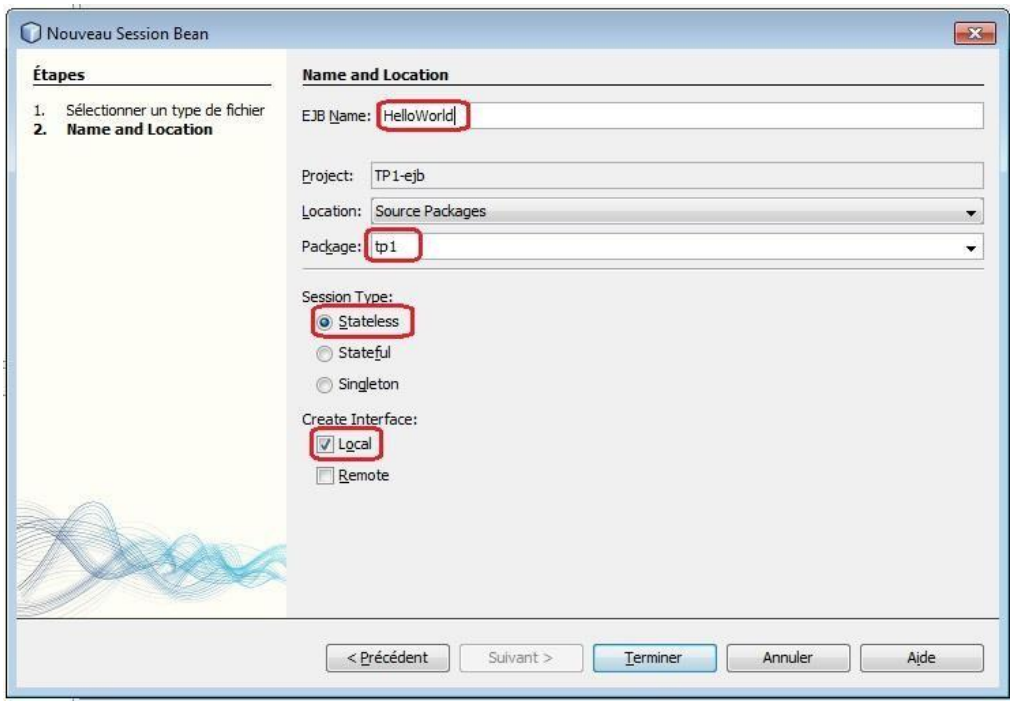
Nous verrons par la suite qu'on peut aussi mettre des EJBs dans une simple application web (nouveauté Java EE 6).

3.Création d'un premier ejb de type session bean stateless : HelloWorld

Allez sur le sous projet TP1-ejb (celui avec l'icone en forme de haricot), qui correspond à la partie "ejb" du projet) et faites bouton droit/new/session bean :



Puis dans la fenêtre suivante, indiquez le nom de votre Ejb, son type, et un nom de package (obligatoire) :



Une fois terminé, deux fichiers sont ajoutés dans le sous-projet : HelloWord.java et HelloWorldLocal.java, qui correspondent à la classe du bean et à son interface « locale »

(un bean avec un interface locale ne peut être utilisé que par un client tournant dans la même

JVM : une jsp, une servlet ou un autre ejb).

NOTE : si jamais on ne coche pas d'interface, ce qui n'est possible qu'avec un serveur Java EE6, cela suppose que le bean dispose par défaut d'une interface locale.

Il est toujours nécessaire de créer explicitement une interface remote dans le cas d'une application distribuée.

Ajoutons une méthode dans ce bean, la méthode getMessage() qui renvoie la chaîne de

caractères «Hello World» :

```
public String getMessage () {
    System.out.println("Hello World qui va s'afficher dans la console du
    serveur, pour tr
    return "Hello World" ;
    }
```

Dans l'éditeur, une petite lampe jaune doit apparaître sur la gauche de l'en-tête de la méthode. Lorsqu'on clique dessus, une ou des actions sont proposées.

Choisissez "Expose method in local business interface HelloWorldLocal" pour indiquer que cette méthode sera accessible au travers de l'interface locale du bean

(ce n'est pas une méthode interne du bean).

Cette action ajoute automatiquement la déclaration de la méthode getMessage() dans l'interface HelloWorldLocal.java.

(Si la petite lampe jaune n'apparaît pas, on peut obtenir le même résultat en déclarant manuellement la méthode getMessage() dans l'interface.)

```
@Local
        public interface HelloWorldLocal {
        java.lang.String getMessage () ;
}
```

Rappel : les lignes commençant par @ sont des "attributs de code" ou encore des

"annotations de code". Ce sont en fait des méta-données… Elles serviront au compilateur à générer du code contextuel.