4.1 Introduction

Dans le chapitre précédent nous avons introduit les JavaBeans comme des composants java réutilisables s'exécutant coté client et représentent des entités applicatives qui interagissent entre eux.

Les Entreprise Java Beans (EJBs) sont un autre type de composants qu'on peut concevoir avec le langage Java. Mais néanmoins, les EJBs n'ont rien à voir avec les JavaBean.
Ces composants logiciels qui s'exécute coté serveur forment un outil de la plateforme Java EE qui facilite le développement et le déploiement d'applications réparties avec une attention particulière sur la logique métier.

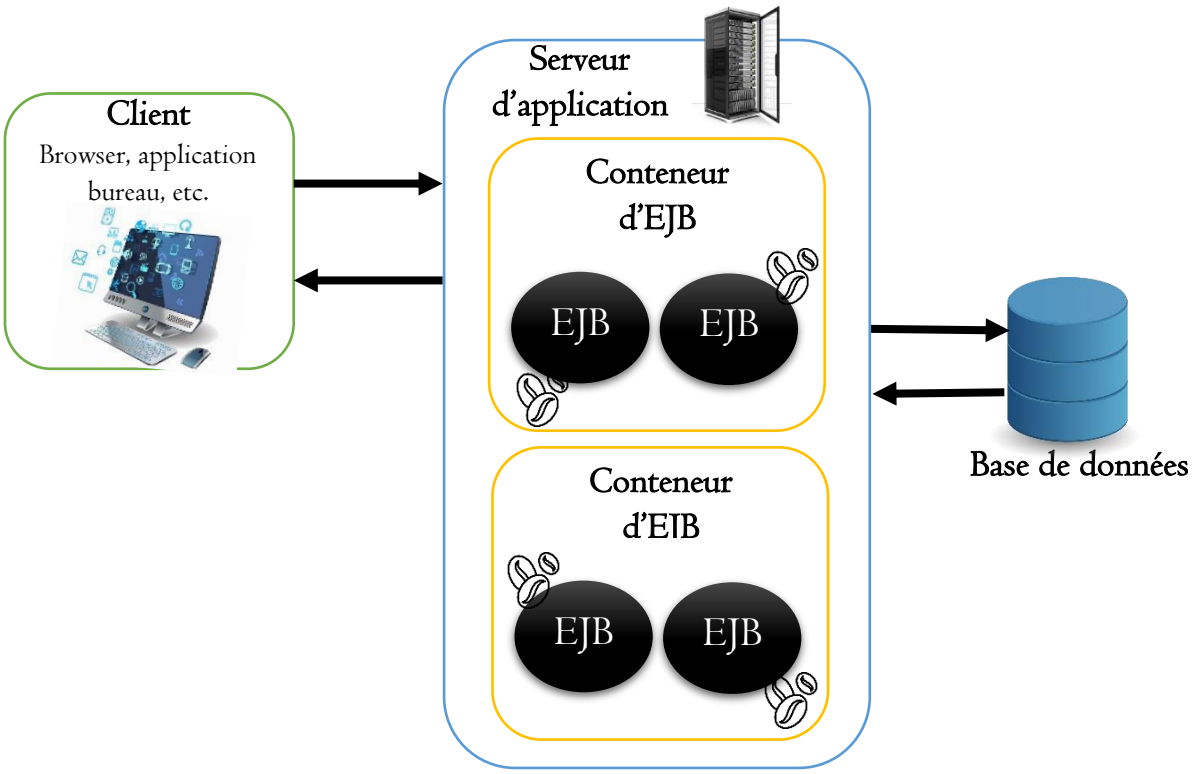
Selon Wikipédia[1], un EJBs sont vus comme des "composants distribués (c'est-à-dire déployés sur des serveurs distants) écrits en langage de programmation Java hébergés au sein d'un serveur applicatif".

4.2 L'architecture à base d'EJBs

Dans cette architecture, on trouve quatre (04) éléments de base :

- Le serveur d'applications
- Les EJBs (les briques logicielles)
- Le conteneur d'EJBs
- Le client

[1] https://fr.wikipedia.org/wiki/Enterprise_JavaBeans



FigI : Un fichier manifest

Notes

- Un serveur peut accueillir un ou plusieurs conteneurs d'EJBs.
- Le rôle de la base de données dans cette architecture se focalise sur la garantie de la persistance des informations.

4.3 Caractéristiques d'EJBs

Parmi les caractéristiques d'un EJB on peut citer :

- S'exécute dans un serveur.
- D'êtres exécutable sur chaque plateforme qui est dotée d'une JVM.
- Réutilisabilité (Écrit une fois, s'exécute partout).
- L'assemblage avec d'autres EJBs pour le développement d'applications.
- Les services d'un EJB sont offerts sous la forme de méthodes invocable localement (dans la même JVM) ou à distance (appel RMI par exemple).

- Assurent les tâches relatives aux différentes parties impliquées dans la conception d'une application.
- La concentration sur les activités qui concernent le coté :
    - Développement où le développeur s'intéresse exclusivement à la mise en place de la logique métier de l'application indépendamment des autres taches assurées par le serveur.
    - Déploiement (sans recompilation ou modification), et exécution sur indépendamment de la plateforme de déploiement où il sera utilisé.

4.4 Le conteneur d'EJBs

Un serveur d'EJB est le logiciel d'infrastructure qui gère la manipulation des EJBs et offre le contexte approprié pour leur exécution. Cette mission est assurée précisément grâce aux conteneurs d'EJB.
De nombreux services sont assurées par le conteneur d'EJB, à savoir :

- La prise en charge du cycle de vie d'un EJB à travers la création (L'Instanciation), la destruction, la passivation ou l'activation des EJBs en fonction des besoins.
- L'accès aux EJBs.
- La gestion des transactions (à travers aux interfaces) entre les clients et l'EJB.
- La gestion et le contrôle des droits d'accès aux EJBs.
- La prise en charge des services de bas niveau tels que le nommage, sécurité, persistance, etc.
- L'implémentation des mécanismes qui permettent le suivi des appels entrants au Bean.

Notes

- Les clients n'interagissent pas directement avec les EJBS, mais indirectement à travers les conteneurs d'EJBs. Par exemple, pour créer ou détruire un EJB, le client ne fait jamais appel par lui-même aux méthodes create () et remove

() de l'EJB ; mais c'est le conteneur d'EJBs qui déclenche indirectement ces appels.

- Le client demande au serveur via le conteneur d'EJBs un EJB particulier par son nom logique (JNDI), et une fois la demande acceptée, l'ensemble des méthodes de cet EJB seraient prêt à être invoqué.

4.5 Les types d'EJBs

Selon la version 3.0 de JavaEE, il existe trois catégories d'EJBs, à savoir :

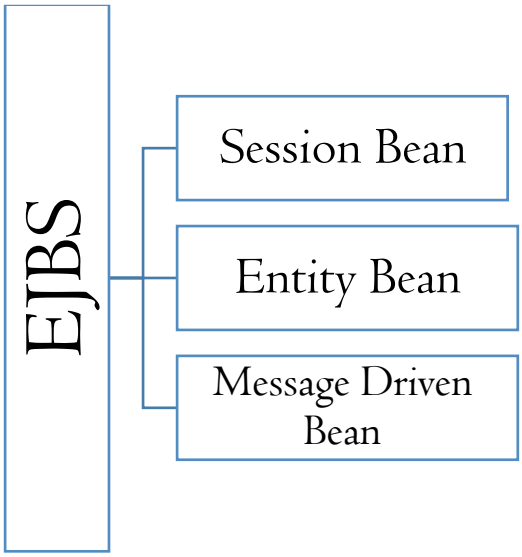


Fig2 : Les Types d'EJBs.

4.5.1 Les EJBs session

Les EJB de session (suite d'interactions) représentent généralement des services sous forme de méthodes qu'un serveur d'application offre à ses clients. Ces méthodes destinées pour implémenter les traitements qui concernent la logique métier (business logic) de l'application.
Ce genre d'EJBs offrent des services sans ou avec conservation d'état entre les appels. De ce fait, on peut distinguer deux types de EJB de session, à savoir :

4.5.1.1 Stateless (sans état) EJBs

- Ils ne conservent pas leurs états (les données) entre deux appels successifs.
- Une instance de cet EJB peut être utilisé simultanément par un ou plusieurs clients (requêtes), c-à-d partageable.
- Ce n'est pas nécessaire de protéger notre code source contre les accès multiples lors de la création de tels EJBs, c-à-d ils sont adaptés pour qu'ils soient traités de façon synchrone.
- Ils implémentent l'interface er l'interface javax.ejb.SessionBean.
- Ils sont détruits (la perte des données) à la fin de la session (la conversation avec le client) ou l'arrêt (ou une panne) du serveur.
- Annotés @Stateless.

Exemple

- Une conversion de devise.
- Le catalogue des produits.
- Les opérations de calculs.

```
1  @Stateless
2  public class StatelessSessionBeanImpl implements StatelessSessionBean {
3
4      public String sayHello() {
5          return ("Peace!");
6      }
7  }
```

Fig3 : Un EJB Stateless.

4.5.1.2 Statefull (avec état) EJBs

- Ils conservent (et mettre à jours) leurs états (les données) entre deux appels successifs d'un même client.
- Une instance de l'EJB est attribuée au seul client demandeur durant toute la période de leur conversation, c-à-d chaque client à sa propre instance d'EJB.

- Tout comme les EJBs précédents, ejbCreate () et ejbRemove () est invoquées par le conteneur lors de la création ou la suppression d'un EJB successivement.
- Il est obligatoire au serveur d'assurer la persistance (sérialiser) de tels EJBs. Cette tâche est effectuée par le conteneur à travers l'appel des méthodes ejbActivate () et ejbPassivate ().
- Ils font le bon choix aux cas des interactions qui doivent doit être suivies.
- Ils implémentent l'interface er l'interface javax.ejb.SessionBean.
- Ils sont détruits (la perte des données) à la fin de la session (la conversation avec le client) ou l'arrêt (ou une panne) du serveur.
- Annotés @Stateful.

Exemple

- Le remplissage d'un panier de produits (Livres par exemple) à partir d'un catalogue sur Amazone.

```java
@Stateful
public class StatefulSessionBeanImpl implements StatefulSessionBean {

    private String user;

    public void login(String user) {
        this.user = user;
    }

    public String sayHello() {
        if (user == null) {
            return ("Assalamo Alykom !");
        }
        return ("Peace " + user + " !");
    }
}
```

Fig4 : Un EJB Statefull.

4.5.2 Les EJBs Entity

Les Entity Bean sont des composants conçus pour qu'ils soient persistants afin de représenter des entités (données) manipulées par une application et qui doit être sauvegardée sur un support physique (une base de données).
Parmi les caractéristiques de ces EJBs on peut citer :

- Une entité (donnée) qu'on peut sauvegarder dans une base de données.
- Sont dotés des mécanismes (des méthodes telles que ejbStore, ejbLoad, etc.) qui assurent leur persistance.
- Plusieurs clients peuvent partager la même données grâce à leurs instances d'EJB. (Attention : on dit que les clients partagent le même EJB et chacun reçoi son instance de cet EJB).
- Ils implémentent l'interface EntityBean.
- Restent exister et préservent leurs états aux pannes du serveur.

Exemple

- Un produit à travers son enregistrement dans la base de données (son nom, prix, qualité, quantité, fournisseur, etc.)

4.5.3 Les EJBs Message Driven

Les EJB orientés message assurent aux clients une interaction par la transmission de messages de façon asynchrone.

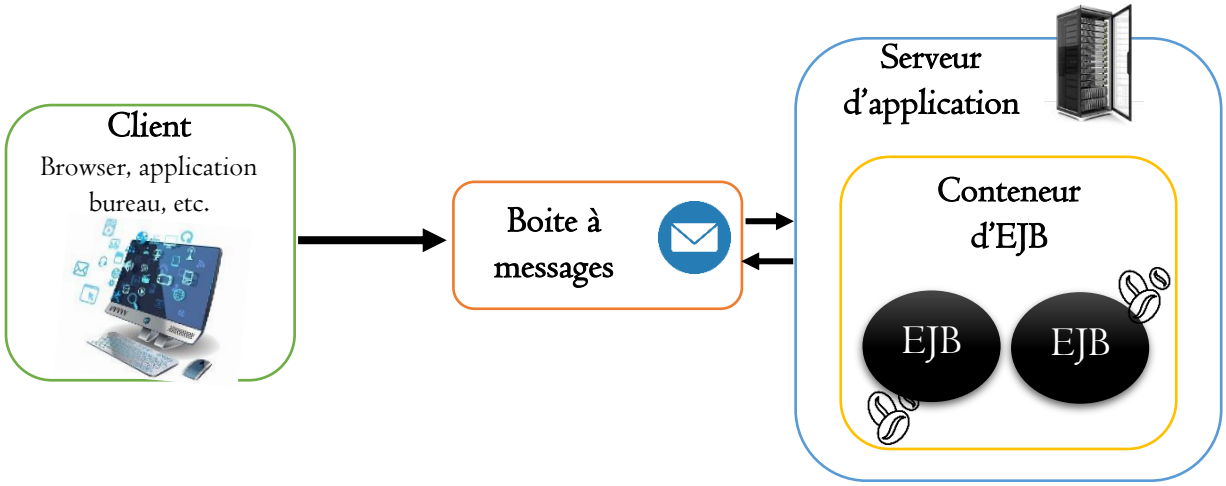


Fig5 : Les Types d'EJBs.

Parmi les caractéristiques de ces EJBs on peut citer :

- La boite à message (la queue) est l'intermédiaire entre le client et l'EJB, c-à-d le client ne s'adresse pas directement à l'EJB.
- Le serveur select un EJB pour traiter un message posté par un client.
- Ils assurent la garantie de livraison (sa fiabilité) dont les déconnexions temporaires du serveur ne font pas mal.
- Annotés @MessageDriven.

Exemple

- MOM (Message Oriented Middleware) et JMS (Java Messaging Service).

4.6 TP

On crée une application dans laquelle on utilise un EJB Stateless qui nous permet de faire une simple opération d'addition entre deux entiers. Cet EJB va ensuite être incorporé dans une Servlet, laquelle on va appeler dans une page JSP.

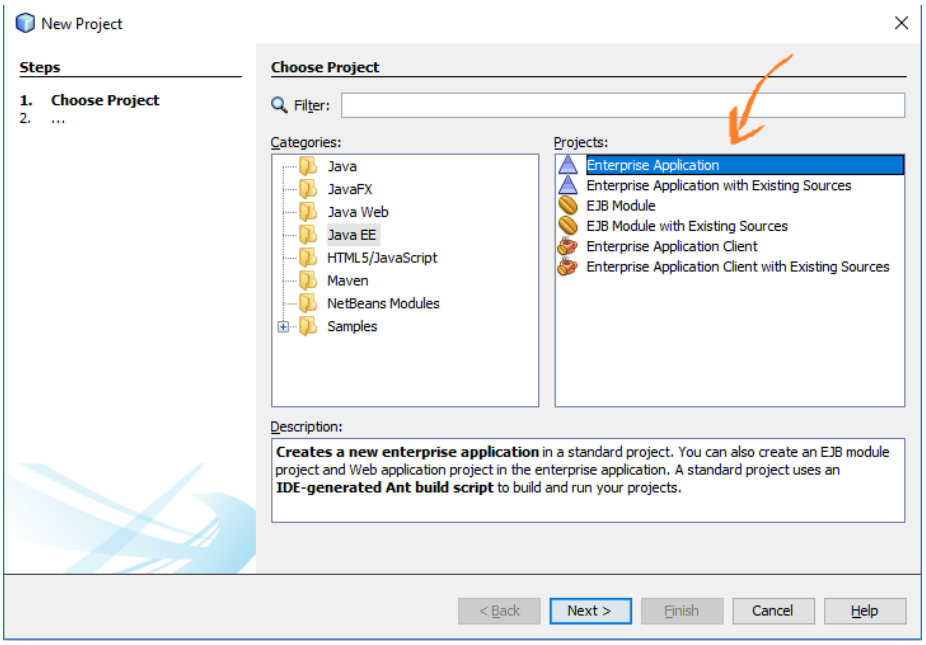


Fig6 : Choisir un projet de type Entreprise Application.

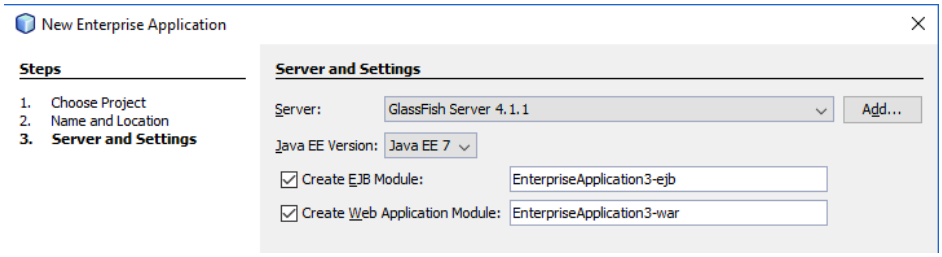


Fig7 : Choisir un serveur.

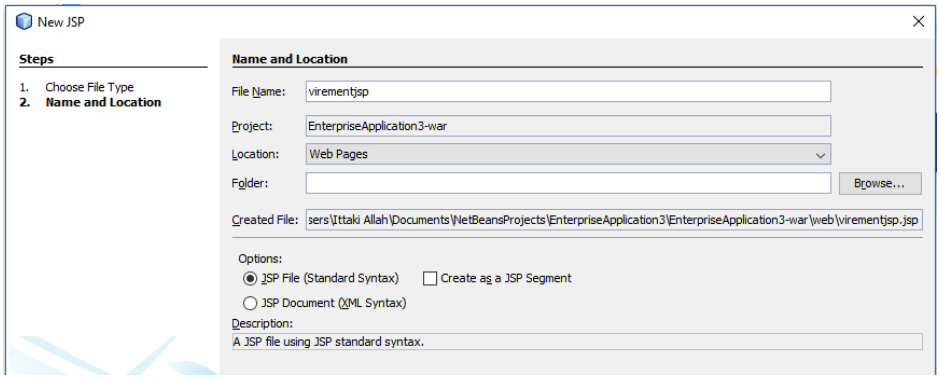


Fig8 : Créer une page JSP.

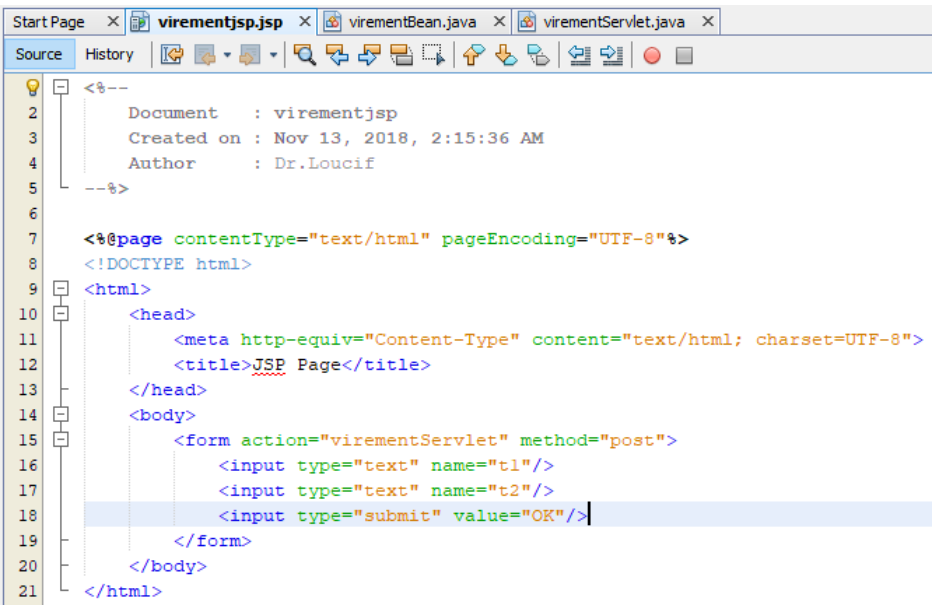


Fig9 : La page JSP.

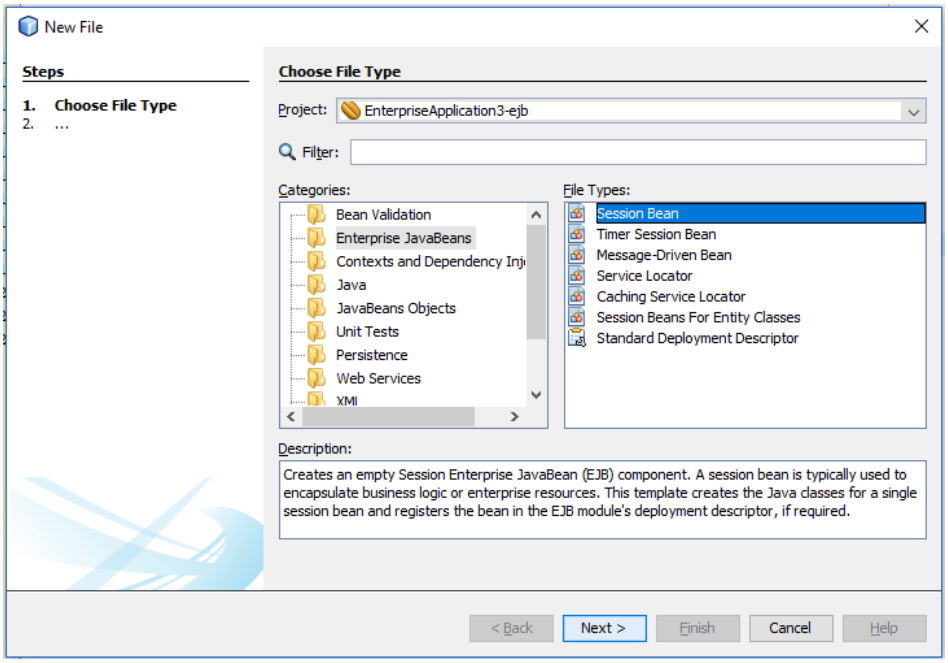
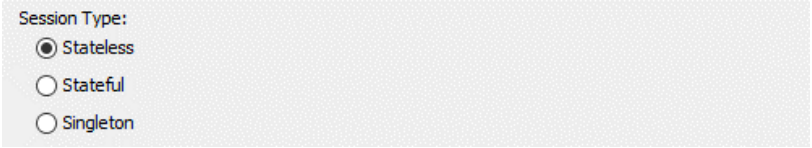
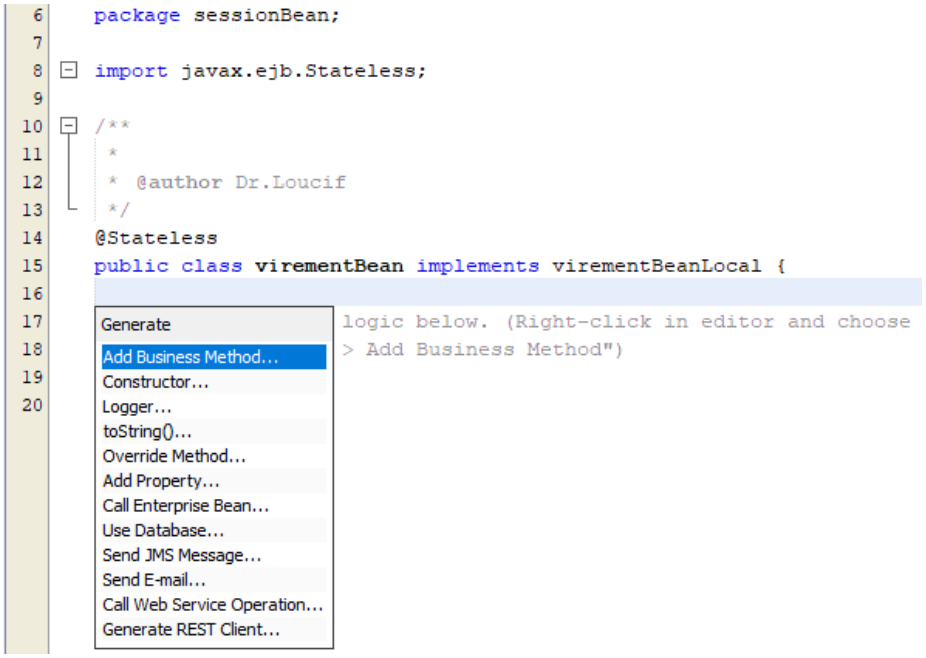


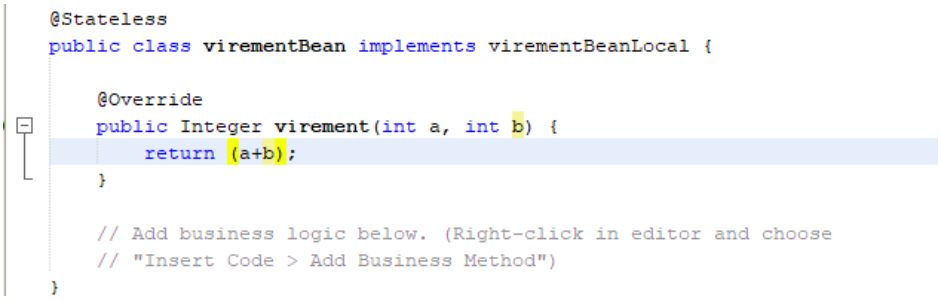
Fig10 : Choisir un EJB Session.



FigI1 : Choisir Stateless.



FigI2 : Ajouter de la logic métier à L'EJB.



FigI3 : la logic métier sous la forme d'une addition.

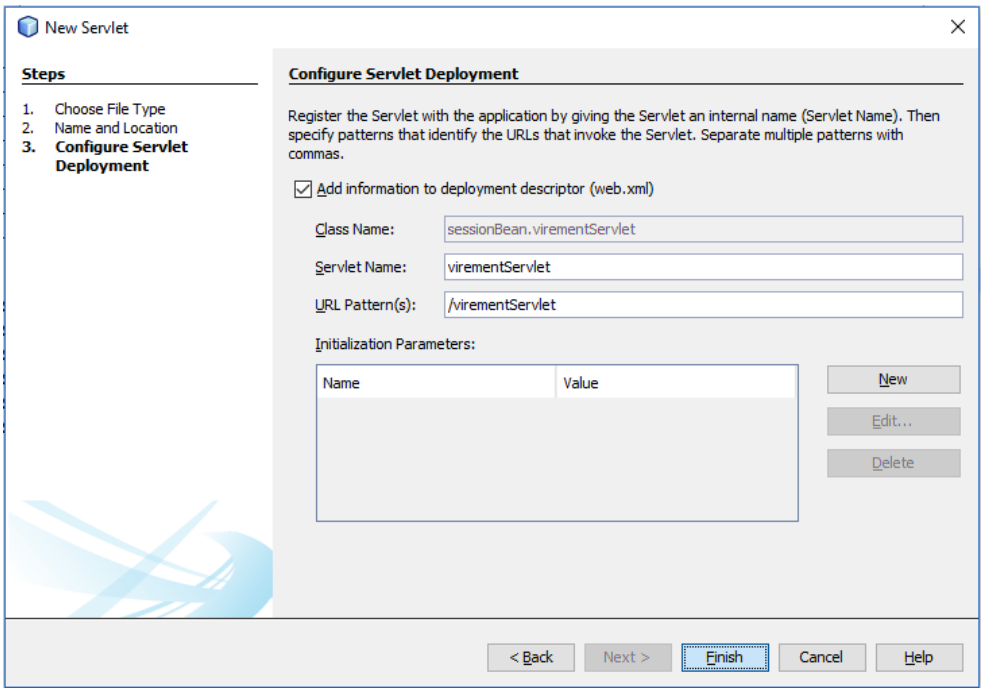


Fig13 : Création d'une Servlet.

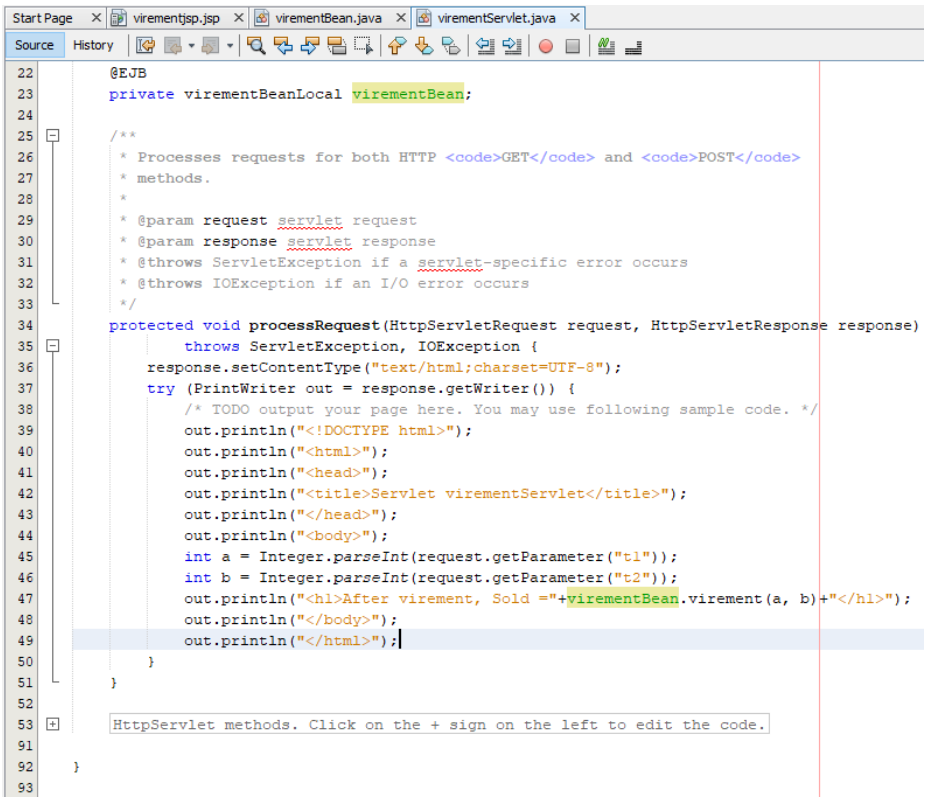
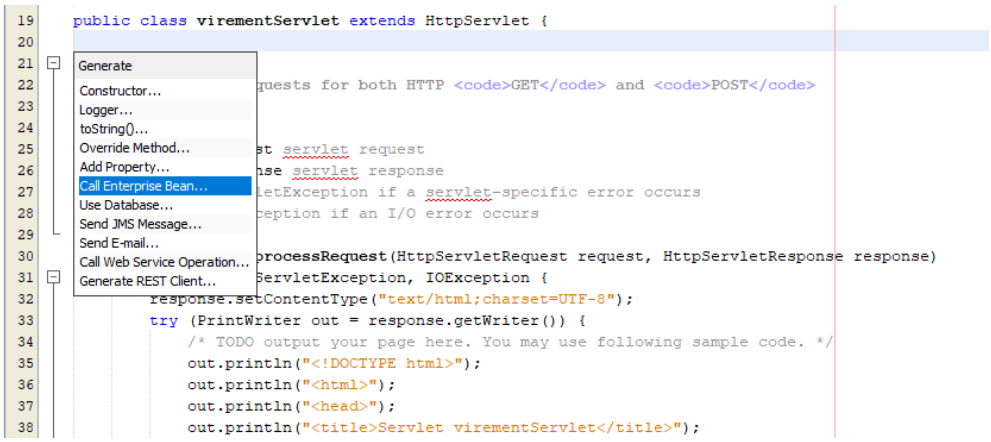
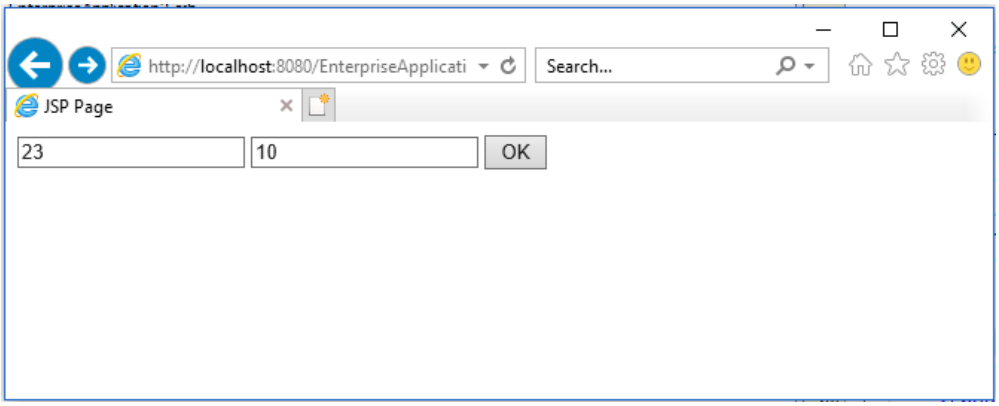


Fig15 : Le code de la Servlet.



FigI6 : L'appel de L'EJB dans la Servlet.



FigI7 : L'exécution de l'application.

FigI8 : L'exécution avec succès.