

Les Entreprise JavaBeans 3.0 (EJB 3.0)



Jean-Marc Farinone

`farinone@cnam.fr`

**Maître de Conférences
Conservatoire National des Arts et Métiers
CNAM Paris (France)**

Spécification EJB 3.0

- Simplification pour la manipulation de l'EJB : élimination des deux interfaces de spécification home et remote (component) de la spécification 2.0 au profit d'une seule interface en 3.0 (remote ou business).
- Simplification de la classe d'implantation : élimination de la nécessité d'implémenter une sous interface de l'interface `javax.ejb.EnterpriseBean`.
- Introduction des annotations Java 1.5 (metadata) comme alternative aux descripteurs de déploiement.
- ...

Le contenu d'un EJB 3.x

- Une interface de manipulation : Business Interface
 - contient les déclarations de méthodes applicatives de l'EJB
- Une classe d'implémentation : Enterprise Bean class
 - implémente les méthodes déclarées dans la business interface
 - implémente les méthodes nécessaires à son cycle de vie
- Les classes et interfaces annexes : Helper classes
 - Toute classe spécifique utilisée par le bean : classes d'exception, utilitaires, ...
- On empaquette (package ?) ces classes et interfaces dans une archive EJB : un EJB JAR. C'est un module contenant l'EJB.
- Une application Java EE formée de plusieurs modules EJB est packagée dans une archive EAR.

L'accès aux EJB session par des clients

- Seulement par des méthodes déclarées dans l'interface de l'EJB
- Les méthodes d'implémentation et les configurations de déploiement sont inaccessibles directement par/pour le client
- Pour définir un EJB session il faut savoir si les clients de cet EJB sont "remote" ou "local".
- Un client remote peut être exécuté sur une machine physique (ou virtuelle = JVM) distincte de celle de l'EJB. Dans ce cas l'interface de manipulation doit être annotée `@Remote`.
- Un client local doit être exécuté sur la même machine virtuelle (= JVM) (donc la même machine physique) que celle utilisée par l'EJB. Dans ce cas l'interface de manipulation doit être annotée `@Local`.
- En cas de doute mettre Remote. On peut avoir les deux.



**Un environnement (gratuit)
pour les EJB :
Java EE 5 SDK**

Chargement et installation de Java EE 5 SDK (1/5)

- Le télécharger à partir de `java.sun.com/javaee`



The screenshot shows a web browser window displaying the Sun Developer Network (SDN) website. The address bar shows the URL `http://java.sun.com/javaee/downloads/index.jsp`. The page features a navigation menu with links for Java, APIs, Downloads, Technologies, Products, Support, Training, and Sun.com. A search bar is visible with the text "search tips" and a "Search" button. The main content area is titled "Java EE Downloads" and includes a banner for "Streamline application development." with the text "A simpler programming model maximizes productivity." and a "Get the Java EE 5 SDK Now" button. Below the banner, there are tabs for Overview, Technologies, Reference, Community, Support, Dev Kit, and Downloads. The "Downloads" tab is selected, showing the "Java EE 5 SDK | J2EE 1.4 SDK" section. This section includes a "Java Application Platform SDK Update 2" announcement with links for "Docs & Resources", "Legal Notices", and "Tools Legal Notices". A list of items is shown, including "All contents of Java EE 5 SDK Update 2", "Sun Java System Access Manager 7.1 Beta", and "Portlet Container 1.0 Beta". There are "Download" and "Download with JDK" buttons. On the right side, there are sections for "Regional Downloads" (Chinese and Japanese) and "Key Resources" (Training and Support).

Chargement et installation de Java EE 5 SDK (2/5)

Il n'y a pas de problèmes pour l'installation.

- Il faut avoir installer un SDK Java (1.5)
- Donner un mot de passe pour l'administrateur (et cocher "Don't Prompt for Admin User Name")

- cocher la case "Add bin directory to PATH" pour accéder facilement aux scripts de Application Server

The screenshot shows the 'Admin Configuration' step of the Java EE 5 SDK Installation Wizard. On the left, a navigation pane lists 'Sun Java™ System', 'Application Server', 'Java EE 5 SDK', and 'Beta'. The main area contains the following fields and options:

- Admin User Name:
- Password (min 8 chars.):
- Re-enter Password:
- Don't Prompt for Admin User Name
The admin user name and password will be stored in a user preference file and will not have to be provided when performing admin functions.
- Prompt for Admin User Name
The admin user name and password must be provided when performing all admin functions.
- Admin Port:
- HTTP Port:
- HTTPS Port:

At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Help'.

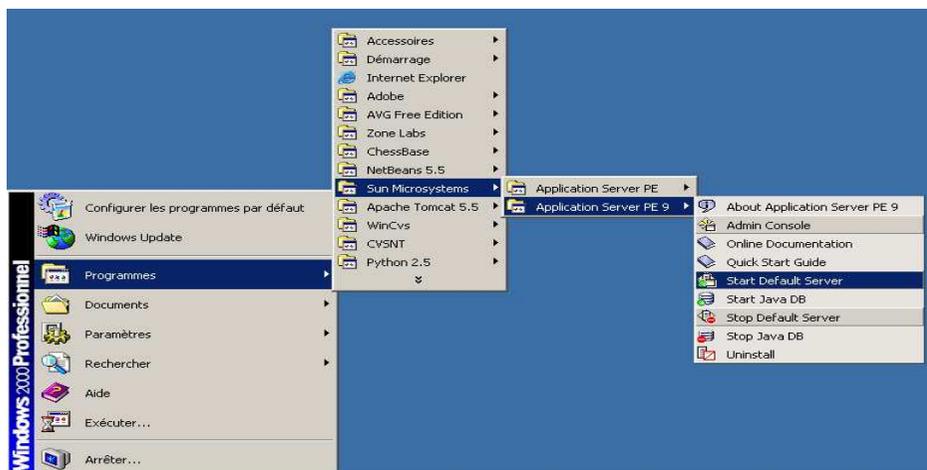
The screenshot shows the 'Installation Options' step of the Java EE 5 SDK Installation Wizard. On the left, the navigation pane is the same as in the previous screenshot. The main area contains the following options:

- Register Application Server
A registration form is displayed after installation.
- Create Samples Server
A second server instance will be created and preconfigured to automatically deploy the bundled sample applications.
- Upgrade from Previous Version
The configuration settings from a previous application server installation can be transferred to this installation.
- Create Desktop Shortcut to Autodeploy Directory
Items can be dragged into this directory for automatic deployment to the server.
- Add bin directory to PATH
Adding this directory simplifies running the server and tools from the command line.
- Create Windows service

At the bottom, there are buttons for '< Back', 'Next >', 'Cancel', and 'Help'.

Chargement et installation de Java EE 5 SDK (3/5)

- Après avoir lancé le serveur web (par Démarrer|Programmes|Sun Microsystems|Applications Server PE| Start Default Server),



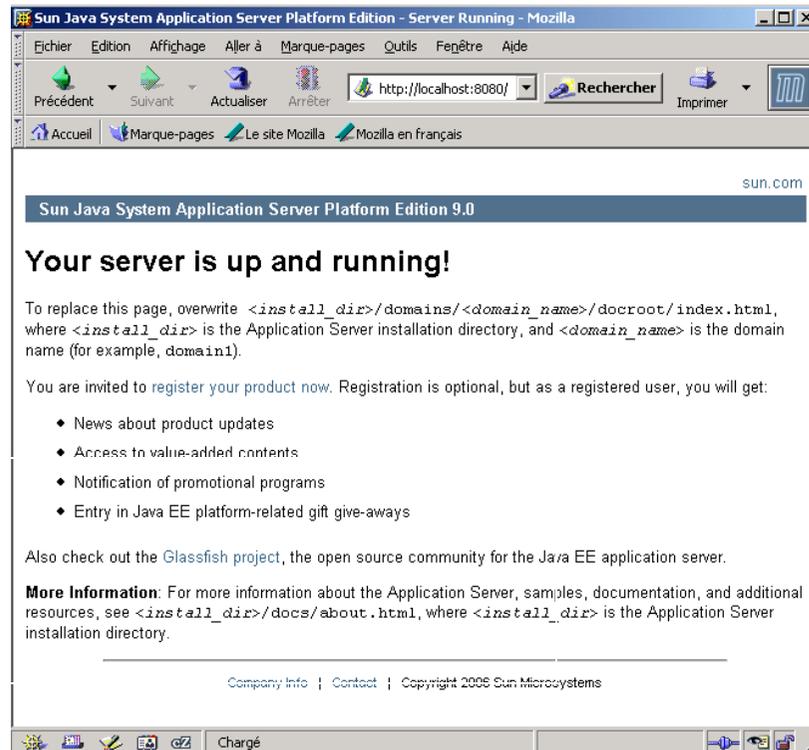
- vous obtenez :

```
Start Default Server
Starting Domain domain1, please wait.
Log redirected to C:\Applications\Sun\AppServer\domains\domain1\logs\server.log.

Domain domain1 is ready to receive client requests. Additional services are being
started in background.
Domain [domain1] is running [Sun Java System Application Server Platform Edition
9.0 (build )] with its configuration and logs at: [C:\Applications\Sun\AppServer\
domains\
Admin Console is available at [http://localhost:4848].
Use the same port [4848] for "asadmin" commands.
User web applications are available at these URLs:
[http://localhost:8080 https://localhost:8181 ].
Following web-contexts are available:
[/web1 /asadmin ].
Standard JMX Clients (like JConsole) can connect to JMXServiceURL:
[service:jmx:rmi:///jndi/rmi://FARINONE:8686/jmxrmi] for domain management purpo
ses.
Domain listens on at least following ports for connections:
[8080 8181 4848 3700 3820 3920 8686 ].
Appuyez sur une touche pour continuer... _
```

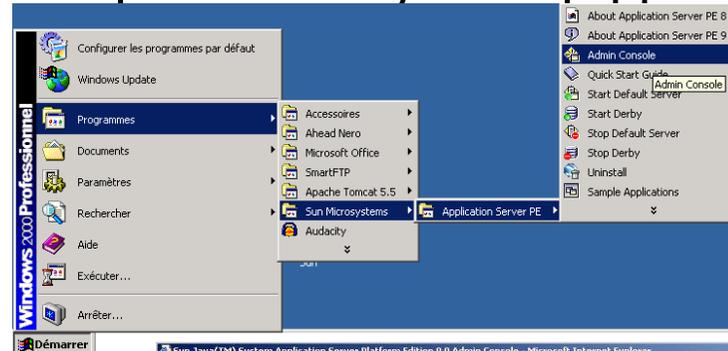
Chargement et installation de Java EE 5 SDK (4/5)

■ Puis se connecter à localhost:8080, pour avoir :

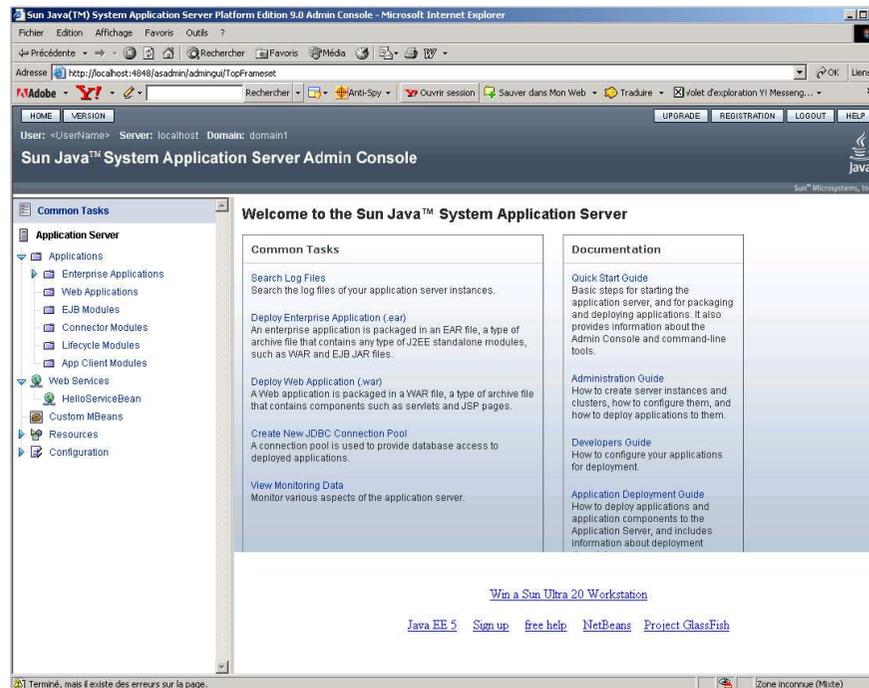


Chargement et installation de Java EE 5 SDK (5/5)

Par Démarrer|Programmes|Sun Microsystems|Applications Server PE| Admin Console,



on obtient :

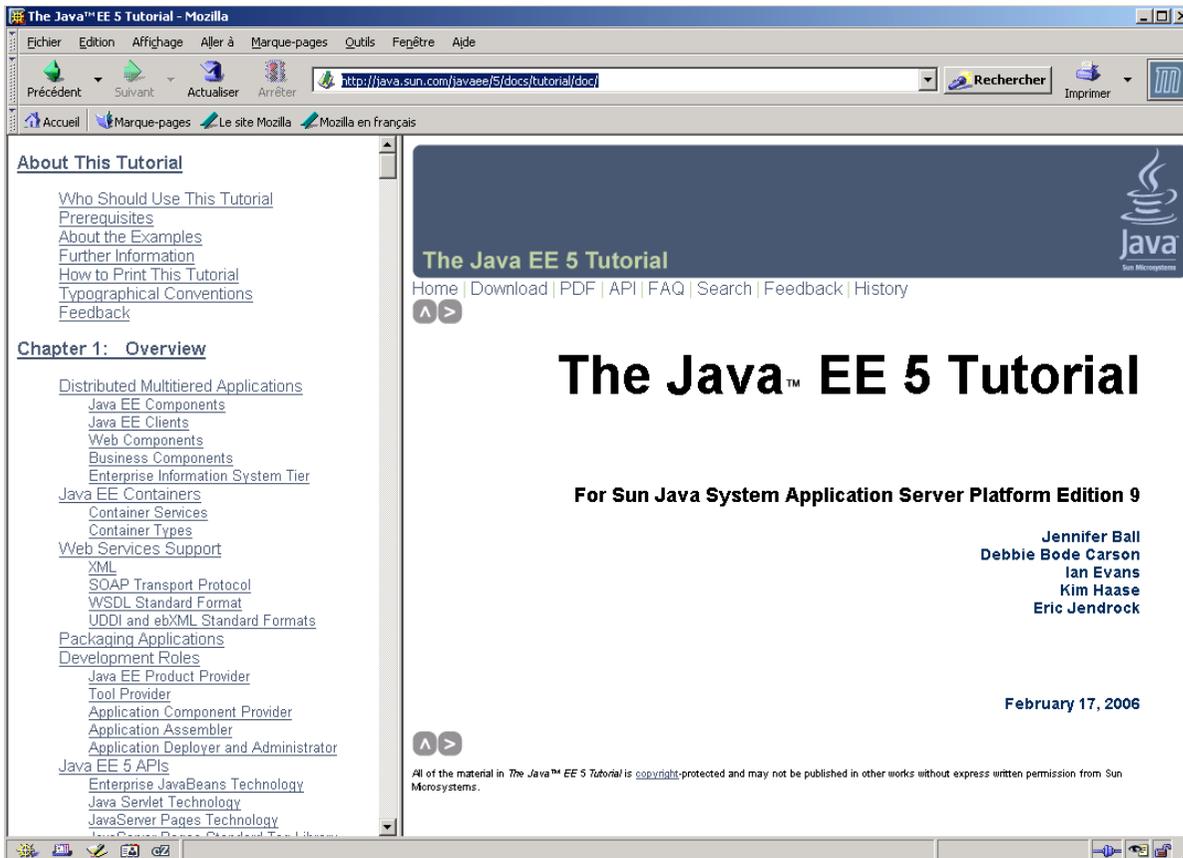




**Un tutorial (gratuit) pour
les EJB :
Java EE 5 Tutorial**

Le Java EE 5 Tutorial

- En ligne à <http://java.sun.com/javae/5/docs/tutorial/doc/>
- On peut le télécharger en cliquant sur Download pour obtenir `j2ee-5_0-beta-doc-tutorial.zip`



The screenshot shows a Mozilla browser window displaying the Java EE 5 Tutorial page. The browser's address bar shows the URL <http://java.sun.com/javae/5/docs/tutorial/doc/>. The page content includes a navigation menu on the left with links such as "About This Tutorial", "Chapter 1: Overview", and "Java EE 5 APIs". The main content area features the title "The Java™ EE 5 Tutorial" and the subtitle "For Sun Java System Application Server Platform Edition 9". Below the title, the authors are listed: Jennifer Ball, Debbie Bode Carson, Ian Evans, Kim Haase, and Eric Jendrock. The date "February 17, 2006" is also displayed. At the bottom of the page, there is a copyright notice: "All of the material in The Java™ EE 5 Tutorial is copyright-protected and may not be published in other works without express written permission from Sun Microsystems."

- Après ouverture de ce `.zip`, on a les exemples du tutorial.

Configurations pour les exemples du tutorial

- Lire section "About the examples" du chapitre "About this tutorial" du Java EE 5 tutorial (18 Février 2006)

- En gros :

- Pour pouvoir utiliser les outils (asant, ...), dans le fichier

- <INSTALL_TUTORIAL>/javaeetutorial5/examples/common/build.properties

- - positionner la propriété `javaee.home` au répertoire d'installation de Java EE 5 SDK par exemple : `javaee.home=c:/Applications/Sun/AppServer` sous windows

- - positionner la propriété `javaee.tutorial.home` au répertoire d'installation du tutorial par exemple :

- `javaee.tutorial.home=C:\JeanMarc\Java\javaeetutorial5`

- sous windows

- - positionner le mot de passe de l'administrateur J2EE à la propriété

- `AS_ADMIN_PASSWORD=leMotDePasse` dans le fichier

- <INSTALL_TUTORIAL>/examples/common/admin-password.txt

Un exemple complet d'un EJB session stateless : le convertisseur de monnaies

- Source Java EE 5 tutorial (4 janvier 2007) chapitre 21
- Les différentes étapes de développement
 - 1°) Créer l'EJB session stateless : ConverterBean.
 - 2°) Créer l'application cliente : ConverterClient.
 - 3°) Faire un package ConverterApp (packager ?) pour l'EJB.
 - 4°) Déployer l'EJB dans le serveur.
 - 5°) Exécuter le client ConverterClient.
- Voir les sources à
`<INSTALL_TUTORIAL>/javaeetutorial5/examples/ejb/converter/src`

Le convertisseur de monnaies :

1°) Créer l'EJB session stateless (1/4)

- Il faut coder une interface de manipulation (Remote) Converter et une classe d'implantation ConverterBean.
- Compiler ces deux parties
- 1°) interface de manipulation (Remote) Converter.java

```
package converter.ejb;

import java.math.BigDecimal;
import javax.ejb.Remote;

@Remote()
public interface Converter {
    public BigDecimal dollarToYen(BigDecimal dollars);
    public BigDecimal yenToEuro(BigDecimal yen);
}
```

- Nouveau en 3.0 : @Remote() et on n'hérite plus de javax.ejb.EJBObject

Le convertisseur de monnaies :

1°) Créer l'EJB session stateless (2/4)

- 2°) La classe d'implémentation `ConverterBean.java`
- Le type d'EJB session (stateful ou stateless) doit être indiqué.
- On utilise les annotations `@stateless()` OU `@stateful()`.
- Les descripteurs de déploiement peuvent être utilisés comme alternative.

Le convertisseur de monnaies :

1°) Créer l'EJB session stateless (3/4)

```
package converter.ejb;

import java.math.BigDecimal;
import javax.ejb.*;

@Stateless()
public class ConverterBean implements Converter {
    private BigDecimal yenRate = new BigDecimal("112.58");
    private BigDecimal euroRate = new BigDecimal("0.0070");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        BigDecimal result = yen.multiply(euroRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
}
```

Le convertisseur de monnaies :

1°) Créer l'EJB session stateless (4/4)

- Il faut compiler l'interface `Converter.java` et la classe d'implémentation `ConverterBean.java`
- On a des outils !!
- `asant ~ ant ~ make`
- configuration par `build.xml`

- Dans le répertoire du tutorial i.e.
`<INSTALL_TUTORIAL>/javaeetutorial5/examples/ejb/convertter,`
lancer `asant`
- Création de 5 arborescences commençant par les répertoires `build` et `dist`, `converter-app-client`, `converter-ejb`, `converter-war`
- Fin de la création et de son packaging dans un `.ear` de l'EJB session.

Le convertisseur de monnaies :

3°) Déployer l'EJB dans le serveur.



- asant deploy

Le convertisseur de monnaies :

2°) Créer l'application cliente : ConverterClient

- Doit être écrite en Java ou techniques Java (JSP, servlet, EJB, ...) ou pour un EJB session, des web services.
- L'essentiel du code est :

```
import javax.ejb.EJB;
...
public class ConverterClient {
    @EJB
    private static Converter converter;
    ...
    BigDecimal param = new BigDecimal ("100.00");
    BigDecimal amount = converter.dollarToYen(param);
    System.out.println("$" + param + " is " + amount + " Yen.");
    ...
}
```

- C'est l'outil de lancement des clients (`appclient`) qui indique où trouver l'EJB.

Le convertisseur de monnaies :

4°) Exécuter le client ConverterClient



- `asant run`
- on obtient :
... \$100.00 is 11531.00 Yen.

Le convertisseur de monnaies :

5°) Créer l'application cliente web : une JSP (1/3)

- Un formulaire html demandera à lancer la JSP. L'essentiel de son code est :

```
<%@ page import="converter.ejb.Converter,  
    java.math.*, javax.naming.*"%>  
  
<%!  
    private Converter converter = null;  
    public void jspInit() {  
        try {  
            InitialContext ic = new InitialContext();  
            converter = (Converter) ic.lookup(Converter.class.getName());  
        } catch (Exception ex) {  
            System.out.println("Couldn't create converter bean."+ ex.getMessage());  
        }  
    }  
  
    public void jspDestroy() {converter = null;}  
%>
```

- Par la suite dans cette jsp on lance :
`converter.dollarToYen(...)`

Le convertisseur de monnaies :

5°) Créer l'application cliente web : une JSP (2/3)

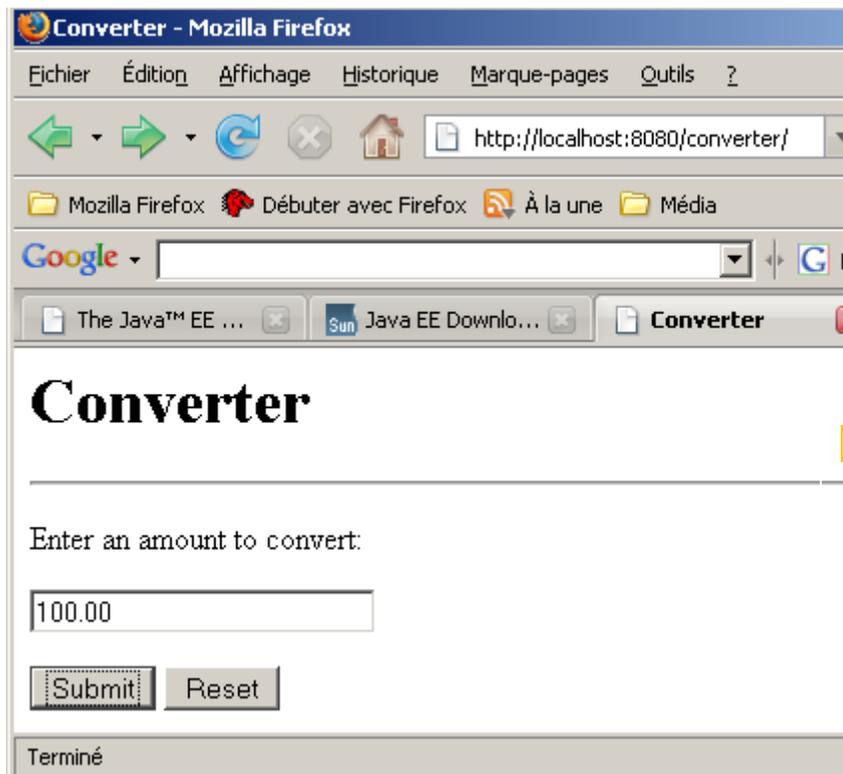
- En fait le formulaire et la JSP sont une seule page (JSP) : très classique. L'astuce est :

```
<body bgcolor="white">
  <h1>Converter</h1>
  <p>Enter an amount to convert:</p>
  <form method="get">
    <input type="text" name="amount" size="25"> <br>
    <p>
      <input type="submit" value="Submit"> <input type="reset" value="Reset">
    </p>
  </form>
  <%
    String amount = request.getParameter("amount");
    if ( amount != null && amount.length() > 0 ) {
      // faire les calculs et retourner la conversion
      // ...
    }
  %>
  ...
  <%
    } // fin du if (si pas de amount initialisé
  %>
</body>
```

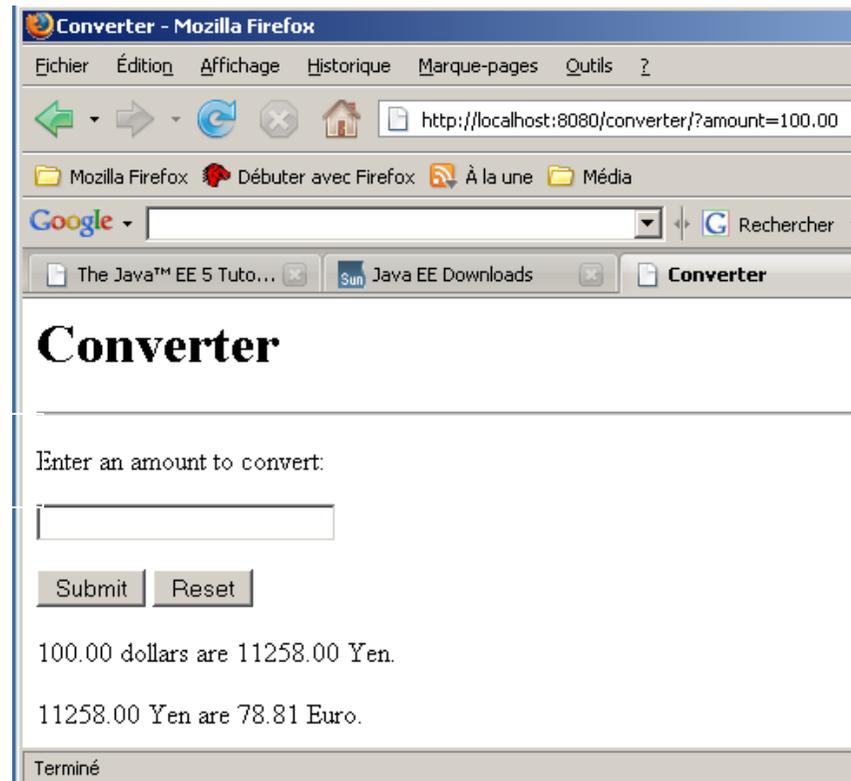
Le convertisseur de monnaies :

5°) Créer l'application cliente web : une JSP (3/3)

■ Le résultat est :



=>



Le convertisseur de monnaies : les codes clients : un mystère dévoilé

■ Le code :

```
InitialContext ic = new InitialContext();  
converter = (Converter) ic.lookup(Converter.class.getName());
```

de la JSP, indique que, par défaut pour les EJB 3.0, les EJB sont enregistrés dans le service de nommage à l'aide du nom de sa classe.



Les EJB Message-Driven 3.0

Message-Driven EJB =

- Ce sont des EJB qui sont abonnés à des "systèmes d'envoi" de messages
- Des clients publient sur ces systèmes.
- L'EJB s'abonne à ces systèmes.
- ~ traitement asynchrone d'événements
- => il n'y a pas de connexion directe entre un client et un message-driven bean
- => le client ne sait pas quel message-driven EJB va traiter son message
- => le message-driven EJB ne sait pas quel client a posté le message

- On utilise les concepts et la technologie JMS (Java Message Service)

JMS =



- Java Message Service
- Traitement asynchrone ou/et événementiel. Les récepteurs peuvent recevoir des messages même s'ils étaient inactifs au moment de l'émission
- Spécification de Sun. Trouver des implémentations

- Notion de file d'attente (queue) et sujet (topic)
- Exemple d'utilisation :
 - Un appel d'offres est lancé sur une liste de diffusion (= un sujet de discussion)
 - Les candidats répondent sur une file de réponses

JMS : les files (d'attente) et les sujets

- File d'attente (queue) : comme à la poste. Une file commune. On passe à un seul guichet = on est traité par une seule personne
- Sujet (topic) : une liste de diffusion. On s'abonne à une liste de diffusion. On reçoit, comme tous les abonnés, les bulletins publiés sur cette liste.
- Sujet = publication/abonnement

EJB Message Driven : un peu de code

Le code du client (1/2)

- Source Java EE 5 Tutorial chapitre 23 EJB MD
- `SimpleMessageClient` envoie des messages sur une file d'attente. L'EJB MD lira dans cette file d'attente.
- Tout d'abord associer un nom à une fabrique de connexion et un nom à une file d'attente par :

```
@Resource(mappedName="jms/ConnectionFactory")
private static ConnectionFactory connectionFactory;
@Resource(mappedName="jms/Queue")
private static Queue queue;
```

- Le client crée ensuite une connexion, une session et un producteur de message (bidouille JMS)

```
connection = connectionFactory.createConnection();
session = connection.createSession(false,
    Session.AUTO_ACKNOWLEDGE);
messageProducer = session.createProducer(queue);
```

EJB Message Driven : un peu de code

Le code du client (2/2)

- Enfin le client envoie des messages sur la file d'attente.

```
message = session.createTextMessage();
for (int i = 0; i < NUM_MSGS; i++) {
    message.setText("This is message " + (i + 1));
    System.out.println("Sending message: " +
        message.getText());
    messageProducer.send(message);
}
```

EJB Message Driven : un peu de code

Le code de l'EJB MD (1/2)

- Les spécifications EJB 3.0 précisent qu'un EJB MD :
 - doit implémenter l'interface `javax.jms.MessageListener` (si on utilise JMS)
 - doit utiliser les annotations `MessageDriven` si il n'utilise pas un descripteur de déploiement
 - doit être une classe public et doit contenir un constructeur public sans argument
 - ne doit pas être une classe abstraite ou finale et ne doit pas définir la méthode `finalize()`.
 - N'a pas de local ou remote interface (≠ EJB session ou entité)
 - Pas de méthodes métiers. La méthode qui sera invoquée lors de l'utilisation de l'EJB est `onMessage(...)`.

EJB Message Driven : un peu de code

Le code de l'EJB MD (2/2)

- Le code utilise les noms JNDI commun avec le client :

```
@MessageDriven(mappedName="jms/Queue")
```

```
public class SimpleMessageBean implements MessageListener {  
    ...}
```

- Le code de la méthode `public void onMessage(Message msg)`

```
static final Logger logger = Logger.getLogger("SimpleMessageBean");  
public void onMessage(Message inMessage) {  
    TextMessage msg = null;  
  
    try {  
        if (inMessage instanceof TextMessage) {  
            msg = (TextMessage) inMessage;  
            logger.info("MESSAGE BEAN: Message received: " + msg.getText());  
        } catch (JMSEException e) {  
            ...  
        }  
    }  
}
```

Démonstration (1/2) :

A) Créer les ressources JMS

- Dans

`<INSTALL_TUTORIAL>/javaeetutorial5/examples/ejb/simplemessage`

- Il faut tout d'abord :

- Une fabrique JMS de connexion
- Une ressource JMS de destination
- Une destination physique correspondant à ces ressources (noms logiques)

- obtenu par les commandes :

```
asant create-cf
```

```
asant create-queue
```

- Ces commandes

- construisent une fabrique de connexion nommée `jms/ConnectionFactory`
- crée physiquement une file d'attente nommée `jms/Queue`

- A ne faire qu'une fois dans l'architecture Application Server PE 9

Démonstration (2/2) :

B) Construction du client et de l'EJB, archivage, déploiement, exécution du client et de l'EJB MD

- On construit l'archive `simplemessage.ear` ainsi que l'application cliente par
`asant`
- On déploie l'EJB par
`asant deploy`
- On construit un `.jar` du client par
`asant client-jar`
- On lance le client par
`asant run-client`
Le client indique les messages envoyés sur la file d'attente
Voir le fichier
`<install_dir_javaee>/domains/domain1/logs/server.log`
pour les écritures des messages recus par l'EJB

Bibliographie EJB

- La littérature est énorme mais les bibles se trouvent à :
- Site originel <http://java.sun.com/javaee/>
- Tutorial à
<http://java.sun.com/javaee/5/docs/tutorial/doc/>