**Université de Montpellier**
**Faculté des Sciences**
**Département Informatique**

**Filière** : Master Informatique **Intervenant :** Chouki TIBERMACINE

# TP EJB 2

## Objectifs du TD

- Développer un composant MessageDriven Bean (MDB)
- Utiliser le composant MDB avec d'autres composants (Web et session stateful)
- Déployer une application avec un MDB dans un serveur d'application

**Exercice.**

- L'objectif de cet exercice est d'étendre l'application écrite dans la première partie du TP EJB 1 pour que la conversion soit effectuée dans toutes les monnaies disponibles, et que les résultats soient transmis par courrier électronique à l'utilisateur s'il le souhaite. Dans ce dernier cas, il doit indiquer sur l'interface Web une adresse email.
- Modifier le composant Web pour permettre à l'utilisateur d'indiquer en plus du montant à convertir, et de la monnaie cible, son adresse email
- Lorsqu'une adresse email est indiquée, le composant Web enverra un message sur une file de messages (*Queue*), écoutée par un composant MDB, qui va solliciter le bean session pour effectuer les conversions
- Modifier le bean session pour convertir le montant dans toutes les monnaies disponibles (le bean session devra donc fournir deux méthodes)
- Écrire le composant MDB qui récupère le message dès sa disponibilité dans la file, invoque la méthode du bean session, puis envoie un email avec les résultats de conversion
- Déployer toute l'application dans le serveur d'application GlassFish et la tester.

**Procédure à suivre.**

- Démarrer Netbeans, puis le serveur d'application (depuis Netbeans) :
  - Sélectionner dans le menu à gauche sur l'interface de Netbeans l'onglet "Services"
  - Cliquer avec le bouton droit sur le serveur GlassFish puis choisir "Start"

- Modifier le composant Web de l'application (Converter) du TP précédent :
  - Créer une page JSP (index.jsp) pour afficher le formulaire permettant la saisie d'un montant en euros à convertir, le choix d'une monnaie cible, et l'édition d'une adresse email. Lors de la soumission de ce formulaire, c'est ce même script qui va s'exécuter
  - Au début de la page JSP (après la balise <%@page ... %>) placer le code HTML du formulaire : mettre comme valeur de l'attribut action du formulaire "index.jsp"
  - Ajouter le code JSP qui permet de récupérer les données saisies dans ce même formulaire (comme dans le TP précédent). Précédez cependant ce code par des tests permettant de savoir si c'est le premier exécution du script (car c'est le même script qui affiche le formulaire, qui s'exécutera lors de la soumission de ce dernier) :

```
String amount = request.getParameter("amount"); // ...
if((amount!=null)&&(amount.length()!=0)) {
  // Convertir le montant en double ...
  // Effectuer la conversion du montant dans la monnaie cible
  // en sollicitant le bean session, puis afficher le résultat
  String email = request.getParameter("email");
  if(email != null && email.length() != 0) {
    // Demander au MDB de déclencher la demande de conversion
    // dans toutes les monnaies par le bean session
    // le MDB va ensuite envoyer un email avec toutes
    // ces informations au format HTML (dans un tableau HTML)
    // (voir plus loin ce que doivent faire les beans) ...
  }
}
```

- Ajouter au bean session une méthode métier (`euroToOtherCurrencies(double amount)`) qui retourne une `Map<Monnaie,Double>` avec comme « clés » les monnaies et comme « valeurs » les résultats de conversion du montant dans chaque monnaie. On représentera les monnaies par des objets créés à partir d'une classe (Monnaie) qui comporte quatre attributs : le nom du pays, le nom complet de la monnaie, le code de la monnaie, et le taux de change actuel. Pour obtenir le taux de change, nous allons utiliser le même document XML que dans le TP précédent, et pour obtenir les noms des pays correspondants à un code de monnaie et les noms complets des monnaies, nous allons analyser le document XML qui se trouve sur le site officiel ISO suivant : http://www.currency-iso.org/dam/downloads/lists/list_one.xml

  Pour ne pas relancer l'analyse du document XML des taux de change une deuxième fois dans cette nouvelle méthode, nous pouvons déclarer les variables SAXBuilder et Document comme attributs de la classe du bean. Dans ce cas, il faudrait déclarer le bean session comme stateful : remplacer l'annotation `@Stateless` par `@Stateful`. Ceci permet d'avoir un état (les objets SAXBuilder et Document) sauvegardé entre les invocations des deux méthodes du bean. En fait, dans le cas des beans session sans état, le serveur d'application crée un pool d'instances de ce bean, lors du déploiement, et à chaque invocation d'une méthode du bean, il peut prendre une instance différente, d'où la perte potentielle de l'état enregistré entre deux invocations.

- Créer le composant MDB (MailerMDB) et la file de messages :
  - Cliquer sur l'icône du module EJB ConverterMailer-ejb de l'application avec le bouton droit, puis choisir New... > Message Driven Bean ...
  - Donner un nom à votre bean : MailerMDB
  - Donner un nom de package à la classe du MDB (même nom que le bean session)
  - Dans « Project Destinations », cliquer sur « Add » pour créer une file de messages
  - Donner un nom à la file (jms/MailContentQueue)
  - Laisser comme type de destination de messages la valeur 'Queue'
  - Ceci permet de créer automatiquement la file de message (jms/MailContentQueue) et la fabrique de connexions (jms/MailContentQueueFactory). Cette dernière nous permettra de créer des messages et les déposer dans la file
  - Nous allons éditer la classe du bean (MailerMDB.java) plus tard

- Dans le script JSP, ajouter l'envoi de message dans la file créée ci-dessus :
  - Après le code permettant la récupération de l'adresse email, ajouter le code suivant (ce code a besoin de quelques imports javax.naming.*, javax.jms.*, ...) :
    - Récupérer le contexte initial dans le serveur de noms JNDI :

```
Context jndiContext = new InitialContext();
```

    - Obtenir une instance de la fabrique de connexions qui a été créée précédemment :

```
javax.jms.ConnectionFactory connectionFactory =
(QueueConnectionFactory)jndiContext.lookup("jms/MailContentQueueFactory");
```

    - Créer une connexion à l'aide de la fabrique de connexions :

```
Connection connection = connectionFactory.createConnection();
```

    - Créer une session sans transactions et avec des accusés de réception :

```
Session sessionQ =
connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
```

    - Créer un message de type texte utilisant l'objet session :

```
TextMessage message = sessionQ.createTextMessage();
```

    - Mettre le texte nécessaire dans ce message :

```
message.setText(amount+"#"+email);
```

    - Obtenir une référence vers une instance de la file de messages :

```
javax.jms.Queue queue =
   (javax.jms.Queue) jndiContext.lookup("jms/MailContentQueue");
```

    - Créer un objet de type producteur de messages sur la file de messages à l'aide de l'objet session :

```
MessageProducer messageProducer=sessionQ.createProducer(queue);
```

    - Envoyer le message à l'aide de cet objet producteur de messages :

```
messageProducer.send(message);
```

- Éditer la classe du MDB :
  - Ajouter un attribut qui référencera le bean session :

```
@EJB
Converter converter;
```

    La valeur de cet attribut est injecté automatiquement par le serveur d'application lors du déploiement (donc connexion automatique entre MDB et bean session)
  - Télécharger le JAR « mail.jar » se trouvant sur la page Web du cours (TP 2 > Librairie)
  - Ajouter ce JAR comme librairie du composant EJB
  - Importer les classes des packages javax.mail, javax.mail.internet, javax.jms et java.util
  - Utiliser un compte Mail à vous (les paramètres SMTP doivent être récupérés auprès de votre fournisseur de service de messagerie. Pour Gmail, voir ci-dessous).
  - Dans le corps de la méthode onMessage(...) du bean, compléter ce qui suit avec le code nécessaire à la mise en forme des résultats et indiquer votre compte Mail et votre mot de passe :

```
try {
  if (message instanceof TextMessage) {
     TextMessage mesg = (TextMessage) message;
     String content = mesg.getText();
     // Récupérer le montant à convertir :
    String s = content.substring(0,content.indexOf("#"));
    double amount = Double.parseDouble(s);
    // Demander au bean session de faire toutes les conversions ...
    Map<Monnaie,Double> map = converter.euroToOtherCurrencies(amount);
```

```
        Properties p = new Properties();
        p.put("mail.smtp.host", "smtp.gmail.com");
        p.put("mail.smtp.ssl.enable", "true");
        p.put("mail.smtp.auth", "true");
        p.put("mail.smtp.starttls.enable","true");
        javax.mail.Session session = javax.mail.Session.getInstance(p);
        javax.mail.Message msg = new MimeMessage(session);
        try {
          // Préparation du mail
          msg.setFrom(new InternetAddress("<user>@gmail.com"));
          String dest = content.substring(content.indexOf("#")+1);
          msg.setRecipient(javax.mail.Message.RecipientType.TO,
                           new InternetAddress(dest));
          String sujet = "Conversions de monnaie";
          msg.setSubject(sujet);
         // Mettre en forme les résultats retournés par le bean session (Map)
         // dans une chaine de caractères contenant les balises HTML
         // nécessaires pour construire un tableau HTML (variable content)
         // Voir la capture d'écran ci-dessous (Figure 1)
          msg.setContent(content,"text/html;charset=utf8");
          msg.setSentDate(Calendar.getInstance().getTime());
          // Préparation de l'envoi du mail
          Transport transport = session.getTransport("smtp");
          transport.connect("smtp.gmail.com",587,"<user>","<mot-de-passe>");
          // Envoi du mail
          transport.sendMessage(msg,msg.getAllRecipients());
          transport.close();
          System.out.println("Email envoyé à "+dest); // une petite vérif.
        }
        catch(MessagingException e){e.printStackTrace();}
      }
    } catch (JMSException ex) {ex.printStackTrace();}
```

| Currency | Actual Rate | Converted Amount |
|---|---|---|
| INDIA (Indian Rupee : INR) | 84,77 | 25 430,4 |
| AUSTRALIA (Australian Dollar : AUD) | 1,44 | 431,07 |
| BULGARIA (Bulgarian Lev : BGN) | 1,96 | 586,74 |
| ISLE OF MAN (Pound Sterling : GBP) | 0,85 | 255,57 |
| NAMIBIA (Rand : ZAR) | 13,55 | 4 065,06 |
| KOREA, REPUBLIC OF (Won : KRW) | 1 464,88 | 439 464 |
| MARSHALL ISLANDS (US Dollar : USD) | 1,38 | 413,31 |

**Figure 1:** Capture d'écran du courrier électronique envoyé à l'utilisateur

- Déployer et tester votre application (le message du println ci-dessus s'affichera sur l'onglet GlassFish Server en bas de la fenêtre de Netbeans)

- Ne pas oublier de dé-déployer l'application et arrêter le serveur d'application à la fin de la séance

Chouki Tibermacine