

Tutoriel Eclipse / Jboss

ENTITY BEANS EJB 3

Résumé

Mise en oeuvre des entity beans (EJB3) sous Eclipse Europa et Jboss 4.2.

Description

Il s'agit de réaliser un petit exemple trivial, de type « chat » : Une fenêtre permet de saisir un message et affiche les messages reçus. Pour que différents Clients puissent voir tous les messages, et pour garantir la pérennité de ces messages, ils sont placés dans des EJB entity. On crée donc une classe *Message* de type Entity. Pour gérer la création de nouveaux messages et l'affichage des messages reçus on met en frontal un EJB Session (stateless) *Chat*, qui dispose de deux méthodes : *ajoutMessage*, et *listeMessages*.

Au sein de l'EJB Session Chat on

Et bien sûr pour le frontal web, on utilise un servlet *ChatServlet* et un JSP.

Création du projet

On crée un projet EJB, comme d'habitude...

Création de l'entity

On crée un classe java standard appelée *Message*. On déclare les deux attributs (ID + texte), les getters / setters correspondants (*Astuce : dans la fenêtre Outline on fait un clic droit sur la classe Message, et on choisit source/generate Getters and Setters...*), on ajoute les tags nécessaires. Au final on obtient la classe suivante :

```
package chat.ejb;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Message {
    private int msgID;
    private String msgTxt;

    @Id @GeneratedValue
    public int getMsgID() {
        return msgID;
    }
    public void setMsgID(int msgID) {
        this.msgID = msgID;
    }
    public String getMsgTxt() {
        return msgTxt;
    }
    public void setMsgTxt(String msgTxt) {
        this.msgTxt = msgTxt;
    }
}
```

Définition de l'unité de persistance

On ajoute le fichier persistence.xml, qui définit l'unité de persistance, dans le répertoire META-INF du projet. Ce fichier indique avec quelle source de données l'entity manager doit communiquer pour persister ou charger des entités.

```
<persistence>
  <persistence-unit name="monUnite">
    <jta-data-source>java:/DefaultDS</jta-data-source>
    <properties>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.HSQLDialect"/>
      <property name="hibernate.hbm2ddl.auto" value="update"/>
    </properties>
  </persistence-unit>
</persistence>
```

Vous pouvez maintenant faire un test de déploiement. Démarrez Jboss, puis publiez votre projet. Vous devez obtenir le message suivant dans la console Jboss :

```
10:53:58,688 INFO [JmxKernelAbstraction] creating wrapper delegate for:
org.jboss.ejb3.entity.PersistenceUnitDeployment

...

10:53:58,728 INFO [NamingHelper] JNDI InitialContext
properties:{java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory,
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces}
10:53:58,729 INFO [EJB3Deployer] Deployed: file:/usr/java/jboss-
4.2.1.GA/server/default/deploy/Chat.jar
```

Après avoir vérifié que le déploiement se passait bien, enlevez votre projet de la liste des projets déployés (sinon les redéploiements automatiques perturberont la suite du codage...).

Création de l'EJB Session

Notre EJB session propose deux méthodes : une pour ajouter un message, l'autre pour récupérer la liste des messages. On crée donc une interface (locale) dont le code est :

```
package chat.ejb;

import java.util.List;

import javax.ejb.Local;

@Local
public interface Chat {
    public void ajoutMessage(String txt);
    public List<String> listeMessages();
}
```

On crée ensuite la classe du bean, dont le code est :

```
package chat.ejb;

import java.util.ArrayList;
import java.util.List;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceUnit;
import javax.persistence.Query;
```

```

@Stateless
public class ChatBean implements Chat {
    @PersistenceContext(unitName="monUnite")
    EntityManager em;

    public void ajoutMessage(String txt) {
        Message msg=new Message();
        msg.setMsgTxt(txt);
        em.persist(msg);
    }

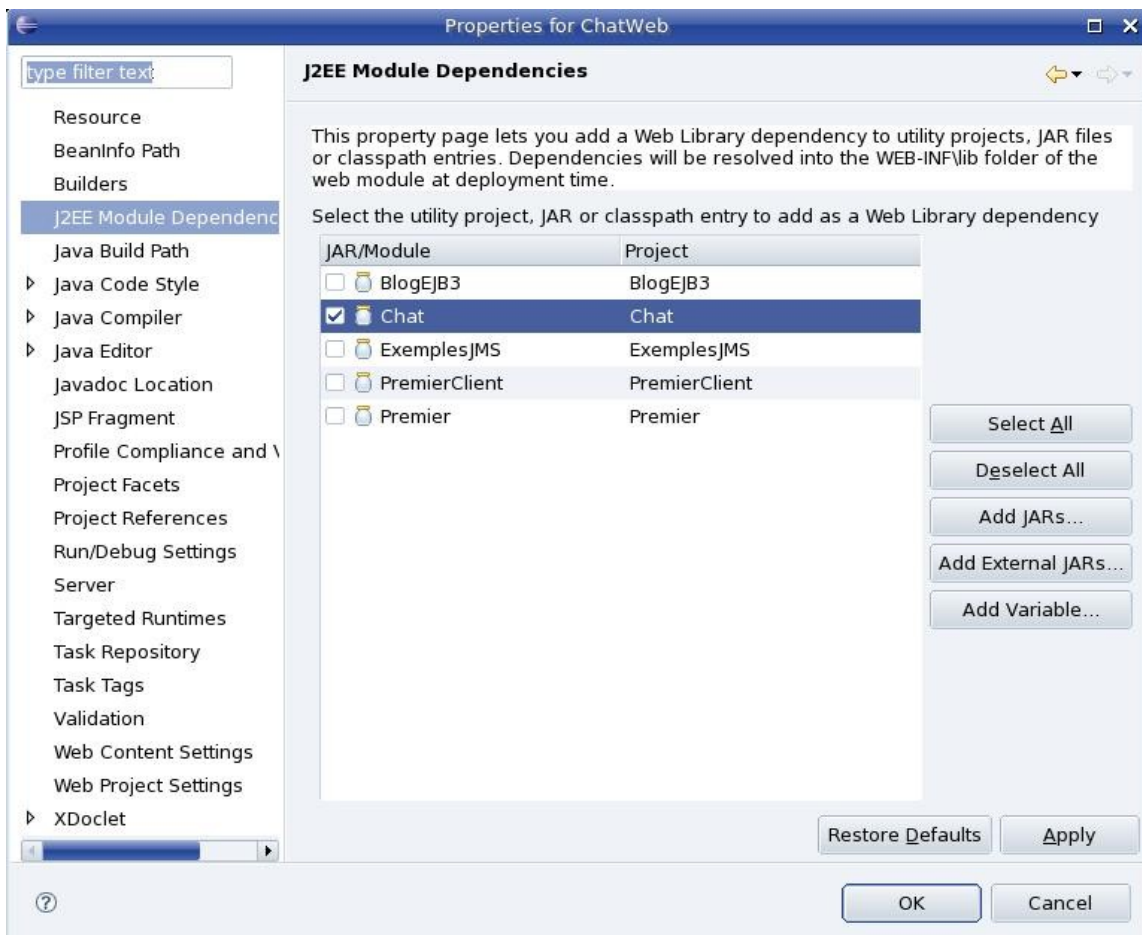
    public List<String> listeMessages() {
        Query q=em.createQuery("From Message");
        List<Message> lm=(List<Message>)q.getResultList();
        List<String> ret=new ArrayList<String>();
        for (Message m:lm) {
            ret.add(m.getMsgTxt());
        }
        return ret;
    }
}
}

```

On peut maintenant déployer le projet Chat.

Création du Projet Web

On crée un projet de type Web dynamique. On va ensuite modifier les propriétés de ce projet (clic droit sur ce projet + propriétés) : dans la rubrique J2EE module dependencies, on coche le projet EJB.



On crée ensuite le servlet :

```
package chat.web;

import java.io.IOException;
import java.util.List;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import chat.ejb.Chat;

public class ChatServlet extends javax.servlet.http.HttpServlet implements
javax.servlet.Servlet {
    static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        try {
            Context ctx=new InitialContext();
            Chat chat=(Chat) ctx.lookup("ChatBean/local");
            String msg=request.getParameter("NouveauMessage");
            if (msg!=null) {
                chat.ajoutMessage(msg);
            }
            List<String> lm=chat.listeMessages();
            request.setAttribute("messages", lm);
            getServletContext().getRequestDispatcher("/chat.jsp").forward(request,
response);
        } catch (NamingException ne) {
            ne.printStackTrace();
        }
    }
}
```

Et la jsp :

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.*"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Chatons sous la pluie</title>
</head>
<body>
<% List lm=(List)request.getAttribute("messages"); // A noter : pas de cast au
niveau du type dans les jsp sous jboss
Iterator it=lm.iterator();
while (it.hasNext()) {
    String s=(String) it.next();
    out.println(s+"<br>");
}
```

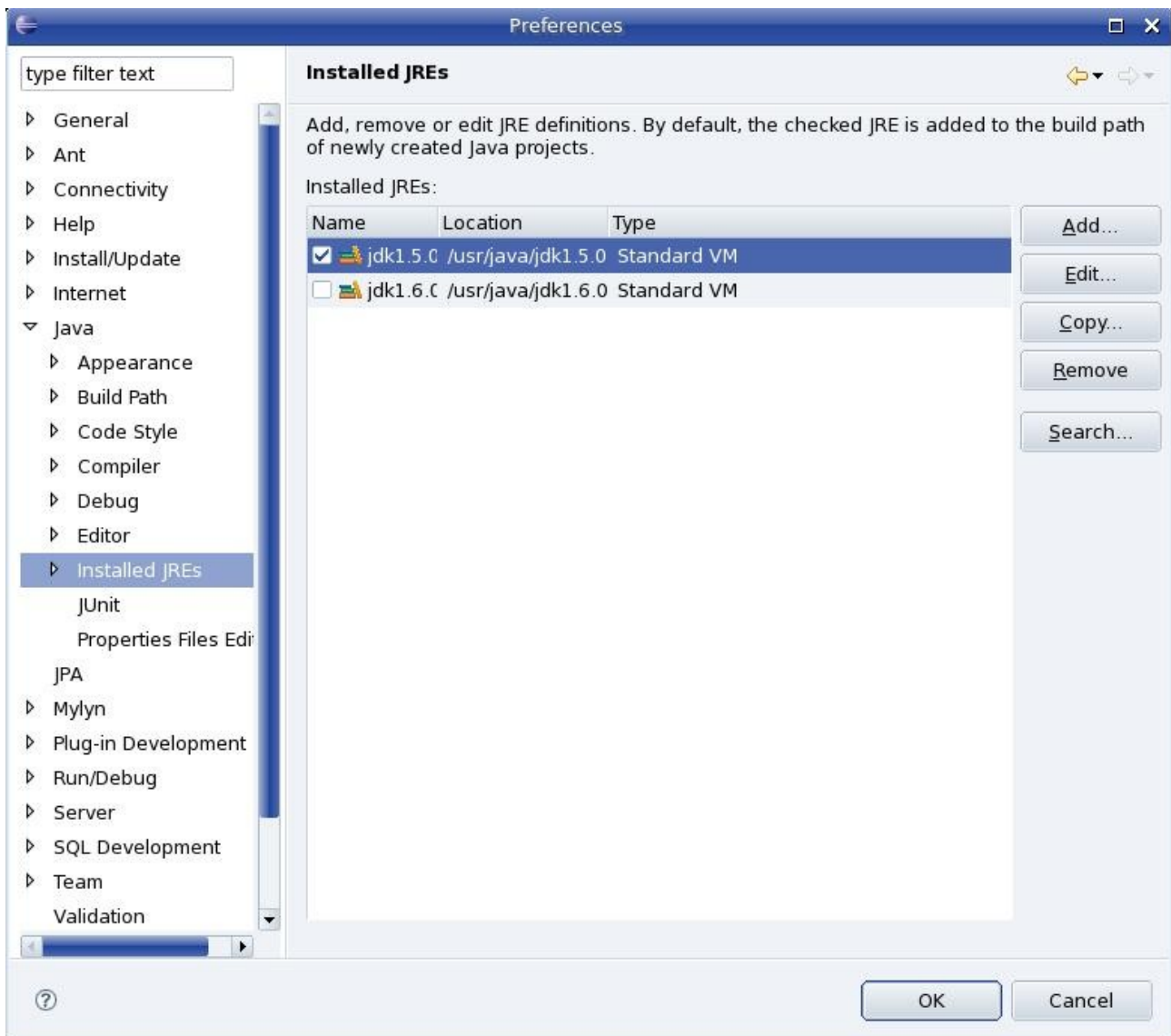
```
}  
%>  
<form method="Post" action="ChatServlet">  
<input type="text" name="NouveauMessage" size=100>  
<button>Valider</button>  
</form>  
</body>  
</html>
```

Il ne reste plus qu'à déployer et à tester...

ANNEXES : Configuration d'Europa

Effectuez scrupuleusement cette configuration, sous peine d'être guetté(e) par bien des tourments...

Lancer Europa. Cliquer sur Windows/preferences.



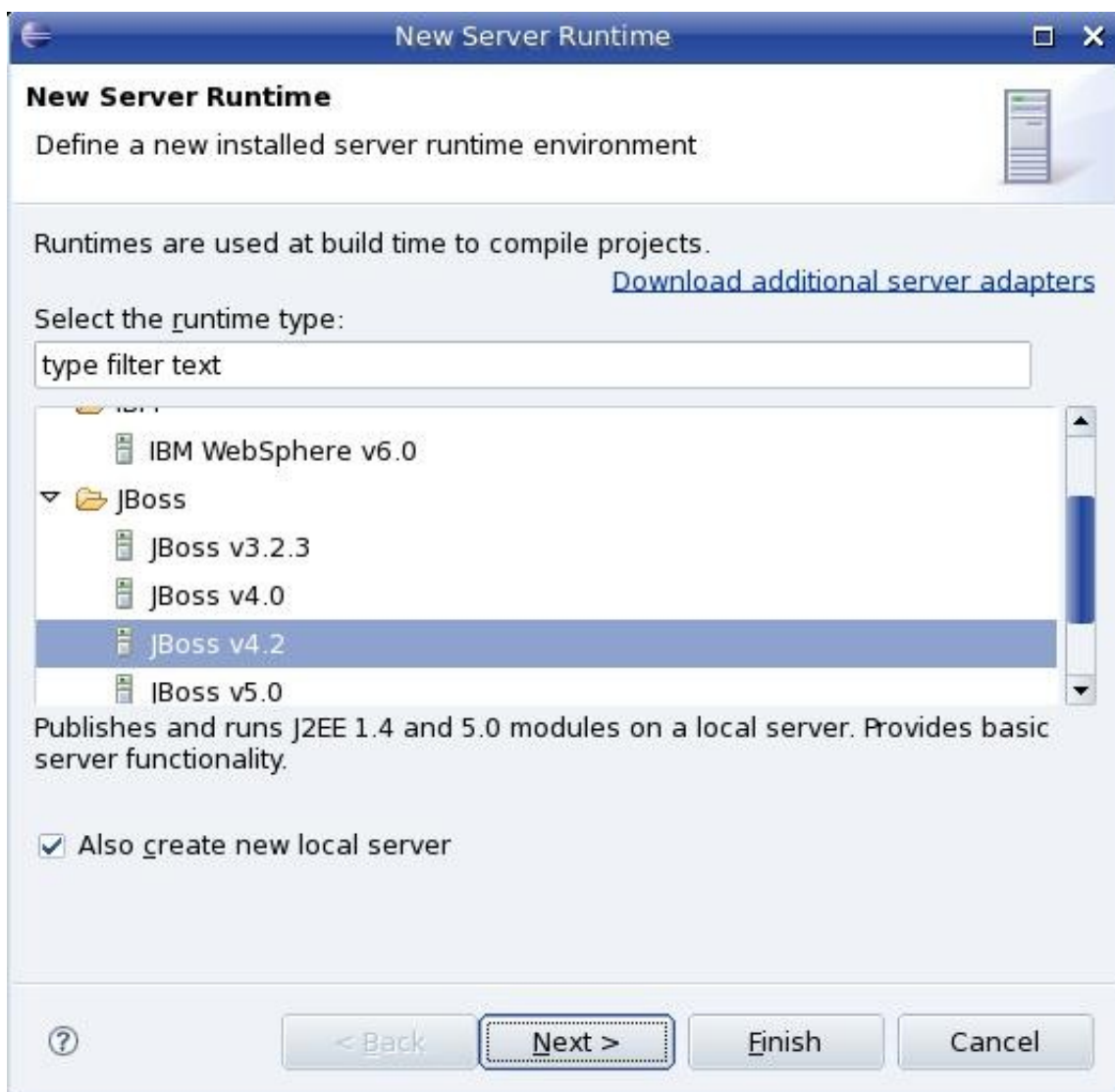
Configurer Java : Vérifier les JRE installés. Au besoin, ajouter un jdk 5 et le choisir comme jdk par défaut (dysfonctionnements de jboss avec jdk 6).

Dans java/compilers, indiquer que le niveau de compatibilité java est 5.0 (par défaut, il s'agit de 6.0).



A ce point, cliquez sur ok pour valider ces deux modifs, puis de rouvrez la fenêtre des préférences pour vérifier que les deux modifs ont bien été prises en compte.

Ajouter ensuite Jboss dans la liste des serveurs disponibles (Server/installed runtime).



Sans oublier de préciser ensuite le répertoire d'installation de jboss...



Cliquez sur finish et fermez la fenêtrés des préférences en cliquant sur OK.

Création d'un projet

Créez un projet de type EJB.



Indiquez le nom du projet.

New EJB Project

EJB Project
Create an EJB Project and add it to a new or existing Enterprise Application.

Project name: Premier

Project contents:
 Use default

Directory: /home/exbrayat/workspaceEuropa2/Premier

Target Runtime
JBoss v4.2

Configurations
Default Configuration for JBoss v4.2
A good starting for working with JBoss v4.2 runtime. Additional facets can later be installed to add new functionality to the project.

EAR Membership
 Add project to an EAR

EAR Project Name: EAR

Cliquez sur Next jusqu'à la dernière fenêtre (pour s'assurer du paramétrage correct de Jboss), puis sur Finish.