J2EE Dalam Aplikasi Enterprise + CD
Pengarang : M. Shalahuddin dan Rosa A.S.
Penerbit: INFORMATIKA
Rp. 38.000

3.4 Contoh Bean Managed Persistence

Fungsi dari contoh *entity bean* BMP yang akan dibuat adalah mengambil data semua *user* dari basis data dan mengirimkan hasilnya ke klien. Seperti pada *session bean*, *entity bean* juga memiliki *framework* atau struktur file dalam paket .JAR. Contoh *entity bean* BMP pada subbab ini diberi nama `contoh_bmp_UserBMP`.

Struktur direktori yang harus dibuat adalah sebagai berikut:

- Direktori `contoh_bmp_UserBMP`
  i. Direktori `jboss`
     1. Direktori `META-INF`
        a. file `jboss.xml`
  ii. Direktori `src`
     1. Direktori `contoh`
        a. Direktori `bmp`
           i. File *source code*
  iii. File `build.xml`

File `jboss.xml` adalah sebagai berikut :

```
<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>UserBMPBean</ejb-name>
      <jndi-name>UserBMPBean</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/MySqlDS_UserBMP</res-ref-name>
        <resource-name>java:/MySqlDS_UserBMP</resource-
```

```
name>
      </resource-ref>
    </entity>
  </enterprise-beans>
</jboss>
```

File `jboss.xml` digunakan untuk memetakan ejb yang dibuat. Bagian `<ejb-name>UserBMPBean</ejb-name>` dan `<jndi-name>UserBMPBean</jndi-name>` merupakan pemetaan nama bean, bagian `<res-ref-name>jdbc/MySqlDS_UserBMP</res-ref-name>` dan `<resource-name>java:/MySqlDS_UserBMP</resource-name>` merupakan pemetaan dari penamaan basis data yang akan digunakan pada bean.

Pada tag `resource-ref-name`, isi dengan `jdbc/<nama deklarasi basis data pada file mysql-ds.xml>` dan `resource-name` dengan `java/<nama deklarasi basis data pada file mysql-ds.xml>`.

File-file *source code* adalah sebagai berikut :

Remote Home Interface

Nama file : `UserBMPHome.java`

```
package contoh.bmp;

import java.util.Collection;
import java.rmi.RemoteException;
import javax.ejb.*;


public interface UserBMPHome extends EJBHome {
    public UserBMP create(String id, String nama, String
password)
                throws RemoteException, CreateException;

    public UserBMP findByPrimaryKey(String id)
        throws FinderException, RemoteException;

    public Collection findAll()
        throws FinderException, RemoteException;
}
```

Pada *remote home interface* dideklarasikan tiga *method* yaitu `create`, `findByPrimaryKey`, dan `findAll` dimana *method* `create` adalah *method* untuk memanggil bean, `findByPrimaryKey` adalah *method* untuk mencari data berdasarkan *primary key*-nya, dan `findAll` adalah *method* untuk mengambil semua data.

Remote Interface
Nama file : `UserBMP.java`

```
package contoh.bmp;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;


public interface UserBMP extends EJBObject {

    public String getNama() throws RemoteException;

    public String getPassword() throws RemoteException;
}
```

Method yang dapat diakses oleh klien hanya yang terdapat pada *remote interface* yang dalam contoh di atas adalah method `getNama` dan `getPassword`, `getId` tidak diperlukan karena BMP telah memiliki method `getPrimaryKey()` yang selalu dapat diakses oleh klien walaupun tidak dideklarasikan pada *remote interface*.

Bean Class
Nama file : `UserBMPBean.java`

```
package contoh.bmp;

import java.sql.*;
import javax.sql.*;
import java.util.*;
import javax.ejb.*;
import javax.naming.*;
import java.rmi.RemoteException;


public class UserBMPBean implements EntityBean {
```

```
   private final static String dbName =
"java:/MySqlDS_UserBMP";
   private String id;
   private String nama;
   private String password;
   private EntityContext context;
   private Connection con;

   public String getNama() throws RemoteException {
        return nama;
   }

   public String getPassword() throws RemoteException {
        return password;
   }

   public String ejbCreate(String id, String nama, String
Password)
               throws CreateException {

        try {
           insertRow(id, nama, password);
        } catch (Exception ex) {
           throw new EJBException("ejbCreate: " +
ex.getMessage());
        }

        this.id = id;
        this.nama = nama;
        this.password = password;

        return id;
   }

   public String ejbFindByPrimaryKey(String primaryKey)
        throws FinderException {
        boolean result;

        try {
            result = selectByPrimaryKey(primaryKey);
        } catch (Exception ex) {
            throw new EJBException("ejbFindByPrimaryKey: "
            + ex.getMessage());
        }

        if (result) {
            return primaryKey;
        } else {
            throw new ObjectNotFoundException("Baris dengan
```

```
id "
            + primaryKey + " tidak ditemukan.");
        }
   }

   public void ejbRemove() {
       try {
           deleteRow(id);
       } catch (Exception ex) {
           throw new EJBException("ejbRemove: " +
ex.getMessage());
       }
   }

   public void setEntityContext(EntityContext context) {
        this.context = context;
   }

   public void unsetEntityContext() {
   }

   public void ejbActivate() {
        id = (String) context.getPrimaryKey();
   }

   public void ejbPassivate() {
        id = null;
   }

   public void ejbLoad() {
        try {
            loadRow();
        } catch (Exception ex) {
            throw new EJBException("ejbLoad: " +
ex.getMessage());
        }
   }

   public void ejbStore() {
        try {
            storeRow();
        } catch (Exception ex) {
            throw new EJBException("ejbStore: " +
ex.getMessage());
        }
   }

   public void ejbPostCreate(String id, String nama,
       String password){
```

```
   }

   private void makeConnection() {
        try {
            InitialContext ic = new InitialContext();
            DataSource ds = (DataSource) ic.lookup(dbName);

            con = ds.getConnection();
        } catch (Exception ex) {
            throw new EJBException("Koneksi Basis Data
Gagal " +
                 ex.getMessage());
        }
   }

   private void releaseConnection() {
        try {
            con.close();
        } catch (SQLException ex) {
            throw new EJBException("releaseConnection: "
                + ex.getMessage());
        }
   }

   private void insertRow(String id, String nama, String
password)
               throws SQLException {
        makeConnection();

        String insertStatement =
            "insert into tbl_user values ( ? , ? , ?)";
        PreparedStatement prepStmt =
           con.prepareStatement(insertStatement);

        prepStmt.setString(1, id);
        prepStmt.setString(2, nama);
        prepStmt.setString(3, password);

        prepStmt.executeUpdate();
        prepStmt.close();
        releaseConnection();
    }

   private void deleteRow(String id) throws SQLException {
     makeConnection();

     String deleteStatement = "delete from tbl_user where
id = ? ";
     PreparedStatement prepStmt =
```

```
      con.prepareStatement(deleteStatement);

      prepStmt.setString(1, id);
      prepStmt.executeUpdate();
      prepStmt.close();
      releaseConnection();
   }

   private boolean selectByPrimaryKey(String primaryKey)
       throws SQLException {
       makeConnection();

       String selectStatement =
          "select id " + "from tbl_user where id = ? ";
       PreparedStatement prepStmt =
       con.prepareStatement(selectStatement);

       prepStmt.setString(1, primaryKey);

       ResultSet rs = prepStmt.executeQuery();
       boolean result = rs.next();

       prepStmt.close();
       releaseConnection();

       return result;
   }

   private void loadRow() throws SQLException {
       makeConnection();

       String selectStatement =
          "select nama, password "
          + "from tbl_user where id = ? ";
       PreparedStatement prepStmt =
       con.prepareStatement(selectStatement);

       prepStmt.setString(1, this.id);

       ResultSet rs = prepStmt.executeQuery();

       if (rs.next()) {
           this.nama = rs.getString(1);
           this.password = rs.getString(2);
           prepStmt.close();
       } else {
           prepStmt.close();
           throw new NoSuchEntityException("Baris dengan id
" + id
```

```
                + " tidak ditemukan.");
        }

        releaseConnection();
    }

    private void storeRow() throws SQLException {
         makeConnection();

         String updateStatement =
             "update tbl_user set nama =  ? ," +
             "password = ? " + "where id = ?";
         PreparedStatement prepStmt =
         con.prepareStatement(updateStatement);

         prepStmt.setString(1, nama);
         prepStmt.setString(2, password);
         prepStmt.setString(3, id);

         int rowCount = prepStmt.executeUpdate();

         prepStmt.close();

         if (rowCount == 0) {
             throw new EJBException("Menyimpan baris dengan
id "
                       + id + " gagal.");
         }

         releaseConnection();
    }

    public Collection ejbFindAll()
         throws FinderException {
         Collection result;

         try {
            result = selectAll();
         } catch (Exception ex) {
            throw new EJBException("ejbFindAll " +
ex.getMessage());
         }

         return result;
    }

    private Collection selectAll()
         throws SQLException {
         makeConnection();
```

```
        String selectStatement =
            "select id " + "from tbl_user";
        PreparedStatement prepStmt =
              con.prepareStatement(selectStatement);

        ResultSet rs = prepStmt.executeQuery();
        ArrayList a = new ArrayList();

        while (rs.next()) {
            String id = rs.getString(1);

            a.add(id);
        }

        prepStmt.close();
        releaseConnection();

        return a;
    }
}
```

Bagian :

```
        try {
            InitialContext ic = new InitialContext();
            DataSource ds = (DataSource) ic.lookup(dbName);

            con = ds.getConnection();
        } catch (Exception ex) {
            throw new EJBException("Koneksi Basis Data
Gagal " +
                ex.getMessage());
        }
```

pada *method* `makeConnection` merupakan proses mengaitkan basis data dengan ejb sedangkan bagian

```
        PreparedStatement prepStmt =
         con.prepareStatement(insertStatement);
```

adalah persiapan dalam melakukan pengaksesan basis data dengan menggunakan *query* dan bagian `prepStmt.setString(1, id)` merupakan *method* yang telah ada dalam *library* yang berarti mengganti ? pertama dari *query* yang didefinisikan dengan isi dari variabel `id`.

Method yang harus ada pada *Bean Class* walaupun tidak berisi apapun adalah :

- `setEntityContext (EntityContext entityContext)`
  Method untuk mengeset *entity context* sehingga dapat mengaitkan bean dengan objek (*binding*).
- `unsetEntityContext()`
  Method untuk menghapus *entity context*.
- `ejbRemove()`
  Method yang dijalankan saat bean mengalami terminasi.
- `ejbLoad()`
- `ejbStore()`
  Method untuk melakukan update pada basis data.
- `ejbActivate()`
- `ejbPassivate()`
- `ejbFindByPrimaryKey()`
  Method untuk mencari *record* berdasarkan *primary key*.
- `ejbCreate()`
  Method untuk membuat sebuah *record* pada basis data.

- `ejbPostCreate()`
  Parameter *method* ini harus sama dengan method `ejbCreate()`.

Primary Key Class
Nama file : `UserBMPPk.java`

```
package contoh.bmp;

import java.util.Random;

public class UserBMPPk implements java.io.Serializable {
  public String id;

  public UserBMPPk ( String id ) {
    this.id = id;
  }

  public int hashCode() {
       return this.id.hashCode();
  }
```

```
  public boolean equals( Object obj ){
    if( obj instanceof UserBMPPk ){
      return( id == ( (UserBMPPk )obj ).id );
    }
    return false;
  }
}
```

`UserBMPPk.java` merupakan kelas yang mendefinisikan *primary key* dari sebuah entity bean yang bertipe BMP. *Method* yang harus ada pada kelas tersebut adalah :

- `hashCode()`
  Method ini berfungsi untuk mengembalikan nilai id atau *primary key* yang unik.
- `equals( Object obj )`
  Method ini berfungsi untuk mengecek keunikan dari *primary key* atau id.

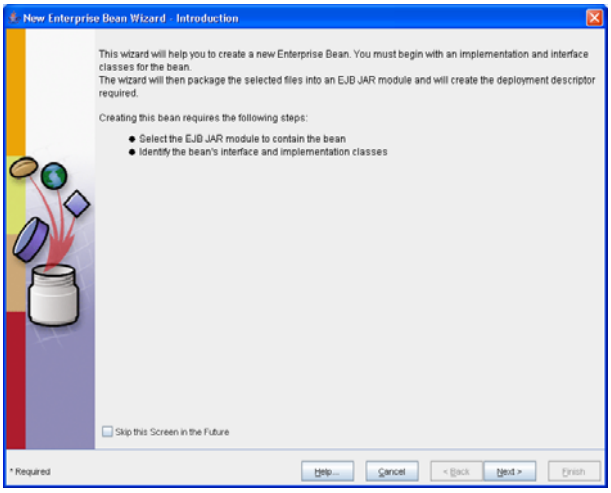
Membuat file JAR

1. Buka jendela command prompt, masuk ke direktori `contoh_bmp_UserBMP` misalnya : `C:\Projects\j2eetutorial\examples\ejb\contoh_bmp_UserBMP>`.
2. ketik perintah `$> asant build`. Perintah asant build digunakan untuk mengkompilasi *source code*, jika *source code* telah lulus kompilasi maka akan muncul pesan sebagai berikut :

```
Buildfile: build.xml

init:

prepare:

build:
    [javac] Compiling 3 source files to
C:\Project\j2eetutorial\examples\ejb\c
ontoh_bmp_UserBMP\build
```

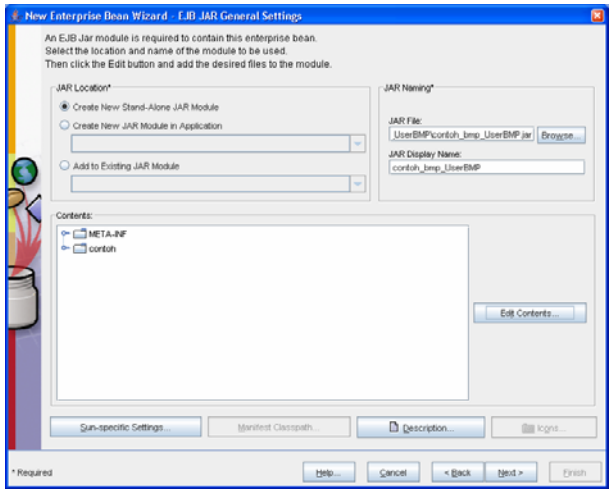
```
    [javac] Note:
C:\Project\j2eetutorial\examples\ejb\contoh_bmp_UserBMP\src
\
contoh\bmp\UserBMPBean.java uses unchecked or unsafe
operations.
    [javac] Note: Recompile with -Xlint:unchecked for
details.

BUILD SUCCESSFUL
Total time: 5 seconds
```

3. Buka Deploytool dari Sun untuk melakukan packaging menjadi file JAR. `Klik Start->All Program->Sun Microsystems->Application Server PE->Deploytool`.
4. Klik `File->New->Enterprise Bean`. Hingga muncul jendela seperti pada gambar 15.



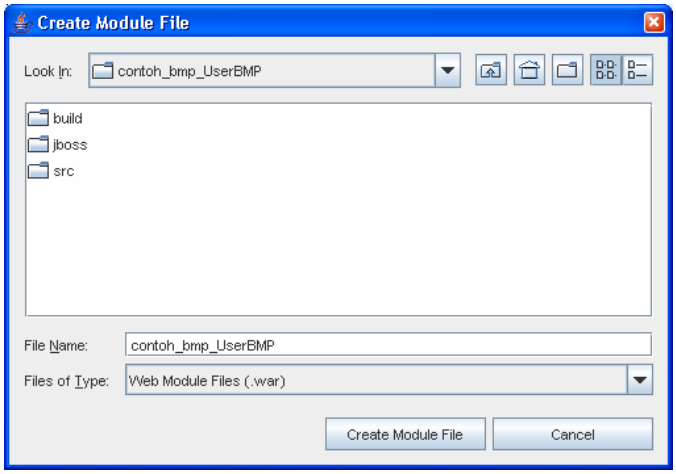
Gambar 15 Jendela Awal New->Enterprise Bean

5. Klik `Next`. Hingga muncul jendela seperti pada gambar 16.



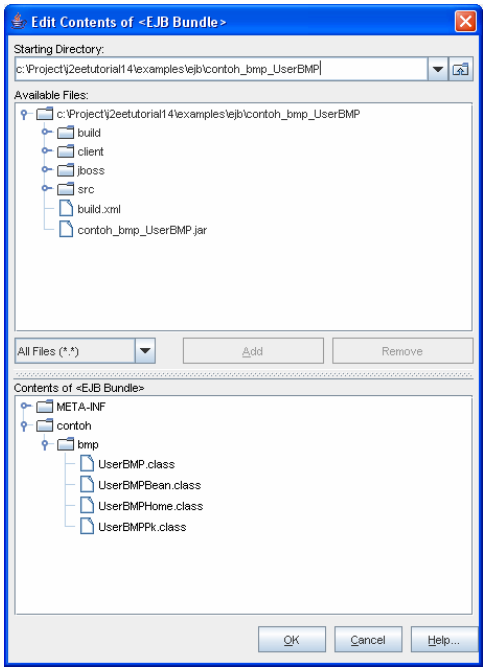
Gambar 16 Jendela Setting Pembuatan EJB

Pilih `Create New Stand-Alone JAR Module` dan isi *field* JAR File dengan mengeklik `browse` untuk mencari direktori `contoh_bmp_UserBMP`, dan isi file name dengan `contoh_bmp_UserBMP` seperti gambar 17 dan klik `create module file`.



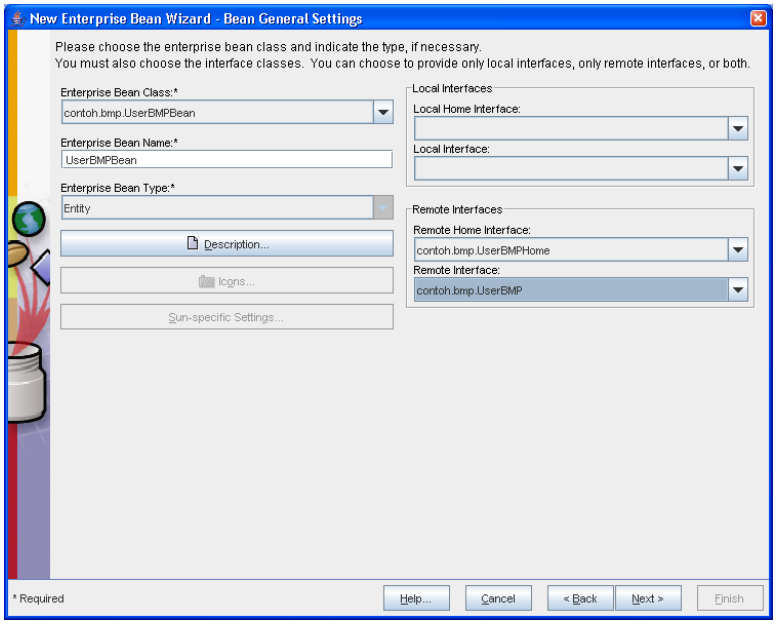
Gambar 17 Jendela Browse

Klik edit contents pada jendela seperti gambar 16 hingga muncul jendela seperti gambar 18. Tambahkan direktori `contoh` pada `available file` dengan mengeklik direktori `contoh` dan klik tombol `add` kemudian klik `OK`. Kemudian klik `next`.



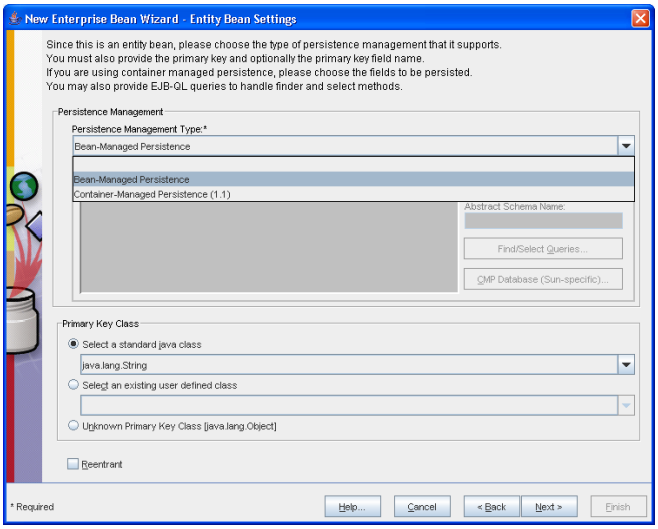
Gambar 18 Jendela Add File

6. Kemudian akan muncul jendela seperti gambar 19. Isi field yang ada dengan mengeklik tanda panah, dan pilih pilihan yang ada, maka beberapa field akan terisi dengan sendirinya, dan pilih tipe entity bean. Klik `next`.



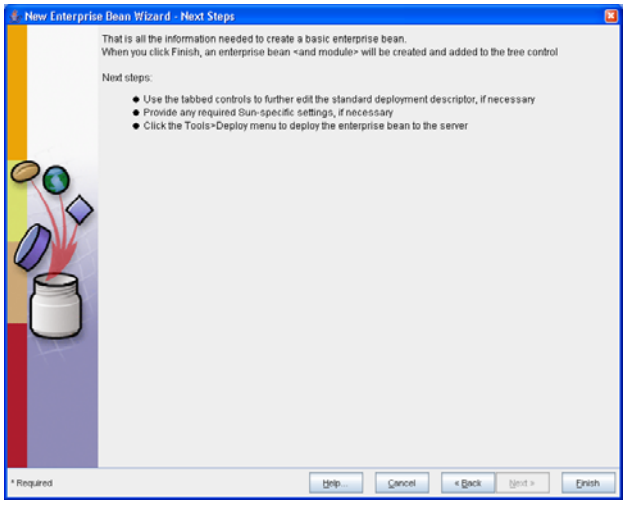
Gambar 19 Jendela Setting Pembuatan EJB

7. Pada jendela seperti pada gambar 20 pilih Management type Bean-Managed Persistence. Pilih Select a standard java class untuk primary key, karena primary key BMP yang dibuat bertipe string. Kemudian klik `next`.



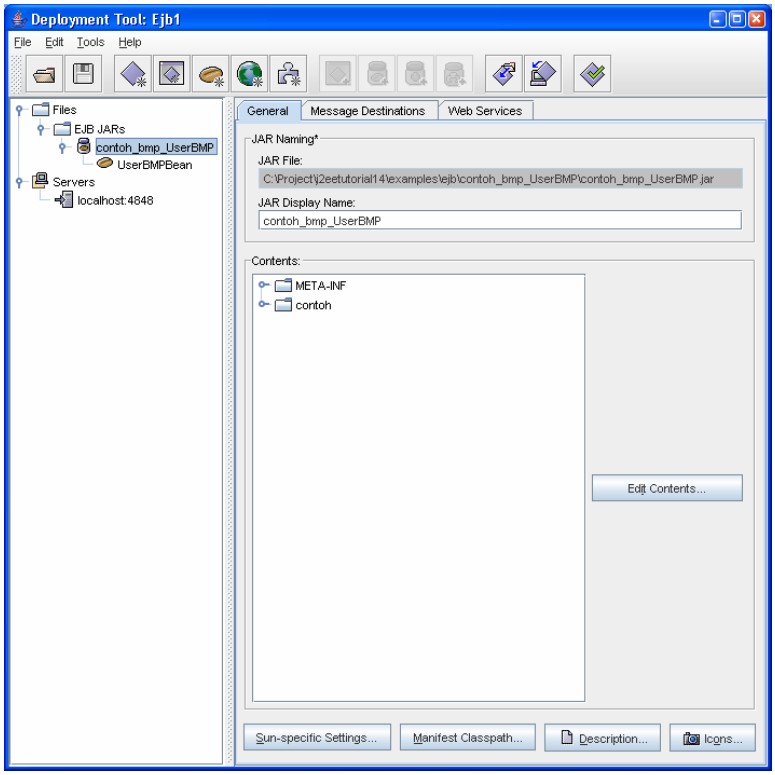
Gambar 20 Jendela Setting pembuatan Entity Bean

8. Pada jendela seperti pada gambar 21 klik `Finish`



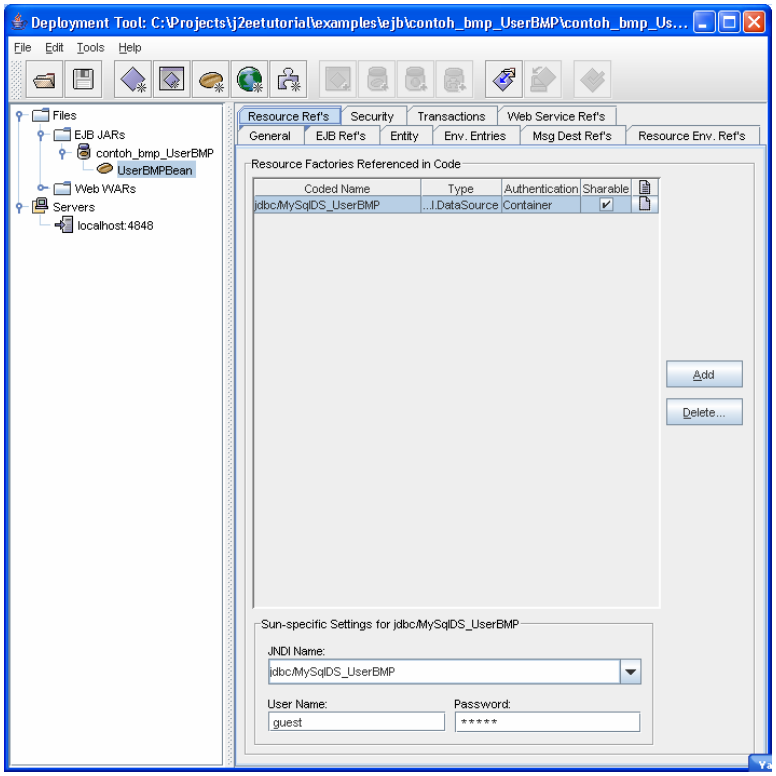
Gambar 21 Jendela Konfirmasi Pembuatan EJB

Kemudian akan muncul jendela seperti pada gambar 22.



Gambar 22 Jendela Deploy tool

Isi *field* `Resource Ref's` seperti pada gambar 23, klik `Add`, isi `Coded Name` dengan `jdbc/MySqlDS_UserBMP` dengan tipe `DataSource`, isi `JNDI Name` dengan `jdbc/MySqlDS_UserBMP`, isi `User Name` dan `Password` dengan `guest` lalu tekan enter.



Gambar 23 Jendela Deploy tool -> Resource Ref's

9. Klik tombol save yang bergambar disket, maka file JAR akan terbentuk di dalam direktori contoh_bmp_UserBMP.
10. Copy file JAR ke dalam direktori jboss yang ada di dalam direktori contoh_bmp_UserBMP. Masuk ke direktori jboss pada command prompt. Ketik perintah `$> jar –xf conton_bmp_UserBMP.jar`. Kemudian ketik perintah `$> jar –cf contoh_bmp_UserBMP.jar META-INF contoh`.
11. Jalankan jboss dengan mengeklik run.bat pada direktori bin tempat server jboss di-copy pada komputer.
12. Copy `contoh_bmp_UserBMP.jar` dalam `<INSTAL>/server/default/deploy` hingga muncul pesan sebagai berikut

```
11:09:17,943 INFO  [EjbModule] Deploying UserBMPBean
11:09:18,453 INFO  [EJBDeployer] Deployed: file:/C:/jboss-
4.0.2/server/default/deploy/contoh_bmp_UserBMP.jar
```

Jika pesan di atas telah muncul maka entity bean telah berhasil di-*deploy* oleh JBoss