JPA Partea a 2-a: JPA in aplicatii JavaEE

(numai daca e cazul -> https://docs.oracle.com/javaee/6/tutorial/doc/gfiud.html)

STUDIUL APLICATIEI "ROSTER" DIN TUTORIALUL JAVAEE.

In **ROSTER.zip** sunt 2 proiecte din tutorialul javaEE5: roster-ejb si roster-app-client. Din aceste proiecte o sa copiem ce avem nevoie.

Proiectele pe care le cream noi le vom denumi similar dar cu majuscule pentru a evita confuzia: roster-EJB respectiv roster-CLIENT.

Se creeaza pe rand cele 2 proiecte.

1. Proiectul de tip: Modul EJB

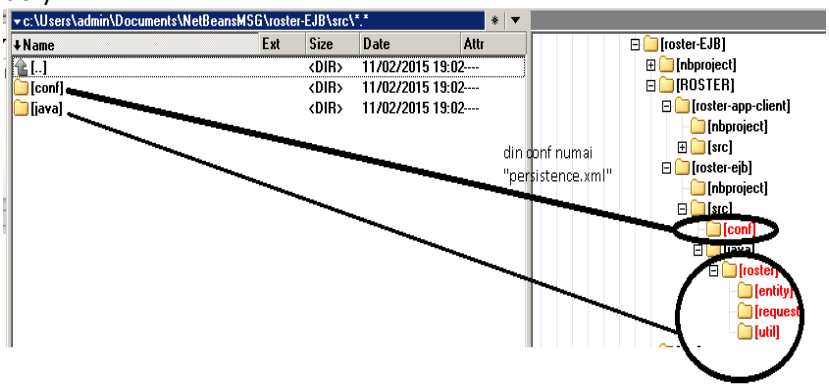
In Netbeans:

a) File->New Project...->JavaEE : EJB Module. Introducem numele proiectului: "roster-EJB". Next si Finish

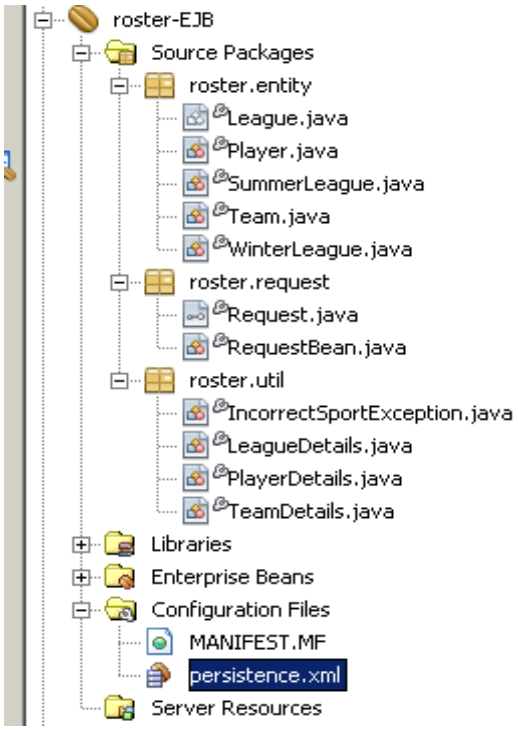
b) In directorul unde s-a creat proiectul se gaseste “src” si cele 2 subdirectoare ale sale: conf si java

In “conf” copiem fisierul “persistence.xml” din directorul cu acelasi nume din sursele atasate (din roster-ejb).

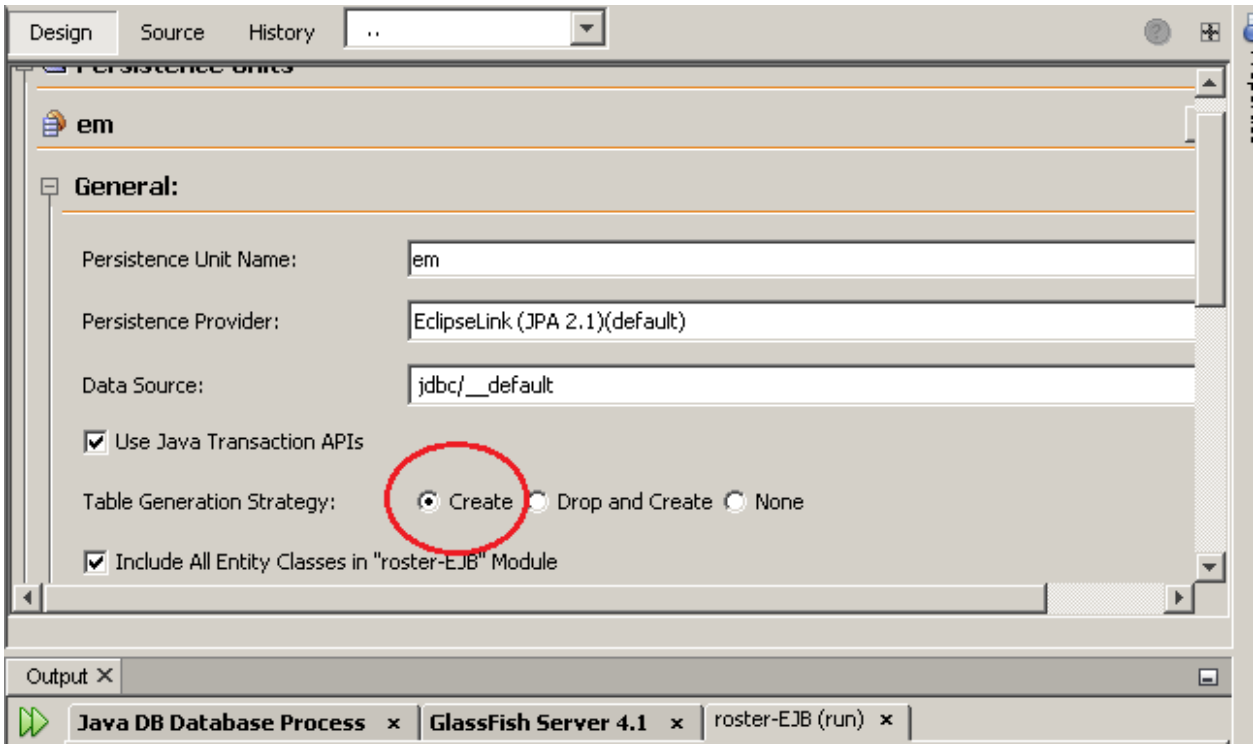
In “java” copiem tot ce este in “java” din roster-ejb(roster cu subdirectoarele entity, request si util)



Acum, in Netbeans, in tab-ul cu proiecte trebuie sa avem proiectul roster-EJB ca mai jos :



DataBase care se foloseste in proiect : trebuie sa avem un fisier de configurare JPA(JavaPersistenceAPI): fisierul persistence.xml (l-am copiat la inceput in src/conf).
https://docs.jboss.org/jbossas/docs/Server_Configuration_Guide/4/html/ch01s02s01.html
Daca fac dublu-click pe persistence.xml , continutul lui apare in dreapta. Putem sa-l vizualizam in 2 variante: “Design” sau “Source” (vezi cele 2 Tab-uri). Daca e selectat “Design” si expandam “General”:



Am facut o singura modificare: am selectat “Create” pentru “Table Generation Strategy”. Ca efect, atunci cand instalam modulul pe server(la Deploy), tabelele din baza de date se vor genera automat (ce tabele si ce structura au?: o sa o vedem cand ne uitam la clasele “entity”).

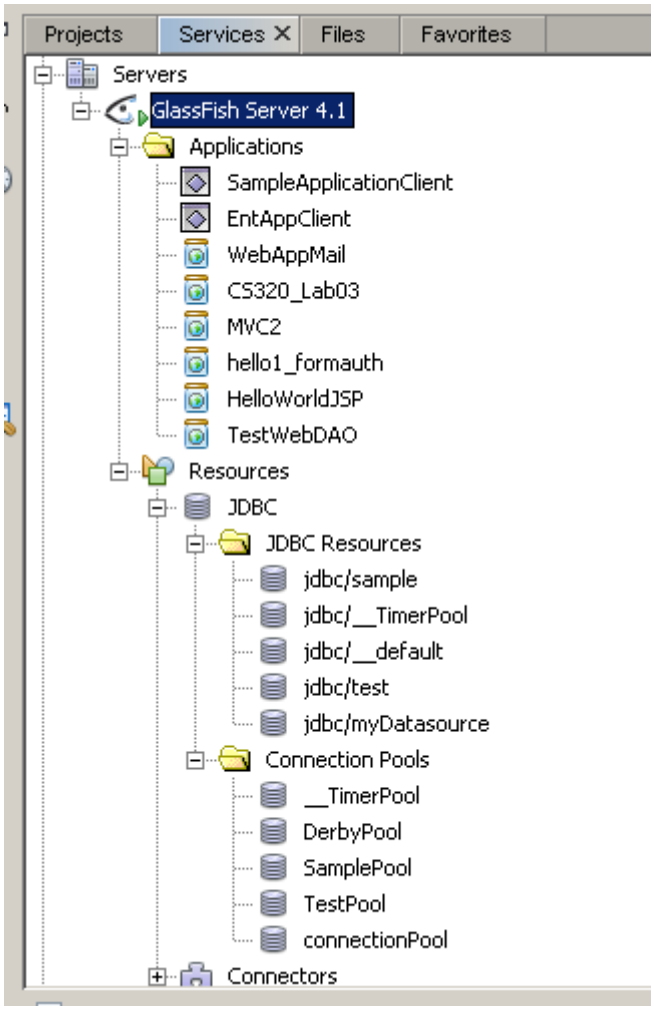
Daca selectam “Source” vedem varianta xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
 <persistence-unit name="em" transaction-type="JTA">
   <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
   <jta-data-source>jdbc/__default</jta-data-source>
   <properties>
     <property name="toplink.ddl-generation" value="drop-and-create-tables"/>
     <property name="javax.persistence.schema-generation.database.action" value="create"/>
   </properties>
 </persistence-unit>
</persistence>
```

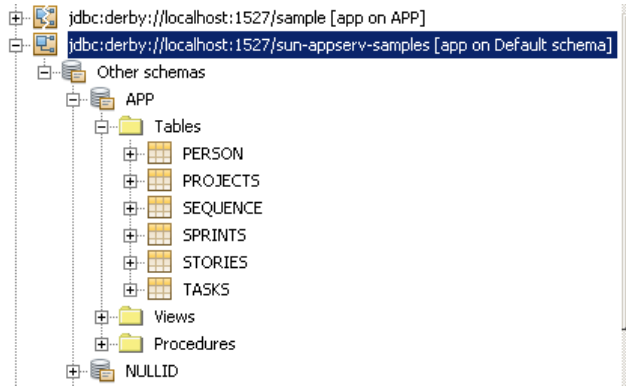
Precizari:
- Spre deosebire de cazul utilizarii JPA pe java SE(laboratorul de data trecuta cand in fisierul de configurare aveam : transaction-type=”ROSOURCE_LOCAL”) , in cazul Java EE se foloseste intotdeauna : transaction-type="JTA"
- Numele unitatii de persistenta este “em” dar in proiectul nostru nu e nevoie sa-l referim explicit
- Important: <jta-data-source>jdbc/__default</jta-data-source> . Prin aceasta se identifica baza de date folosita. Aici, “jdbc/__default” este un nume JNDI care identifica o resursa JDBC, resursa care este deja creata pe server (de catre administrator) . Pentru a vedea ce reprezinta “jdbc/__defaul”(atentie este ‘_’ de doua ori), in Netbeans procedam ca mai jos:

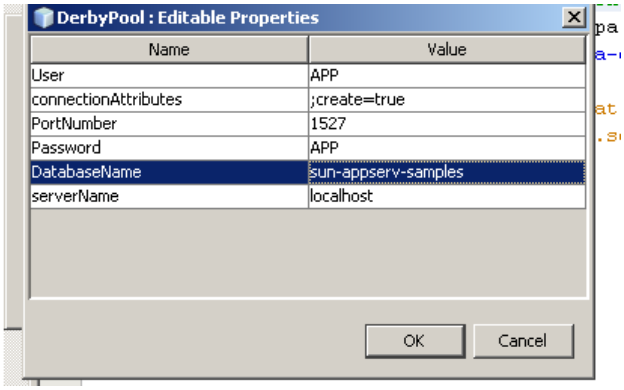
Ne asiguram ca serverele GlassFish si JavaDB sunt pornite. Pentru asta selectam tab-ul “Services” (este langa “Projects”). Expandam “Servers” si trebuie sa vedem “GlassFish Server 4.1”(sau 4.0 functie de versiunea instalata). Facem click-dreapta si daca “Start” este activ il selectam si asteptam cateva secunde pana porneste serverul. In mod normal se starteaza si JavaDB. Se verifica astfel: tot in “Services” , sus este “Databases”; se expandeaza si click-dreapata pe “JavaDB”. Daca este deja pornit “Start” este inactiv.
Revenim la “Servers”: expandam (daca nu este deja) “GlassFish Server 4.1”. Apar doua subdirectoare: Applications si Resources.
In Applications : toate aplicatiile instalate pe server (eventual dam right-click si “Refressh”)

Expandam “Resources” si apoi “JDBC”. Vedem “JDBC Resources” si “Connections Pool”



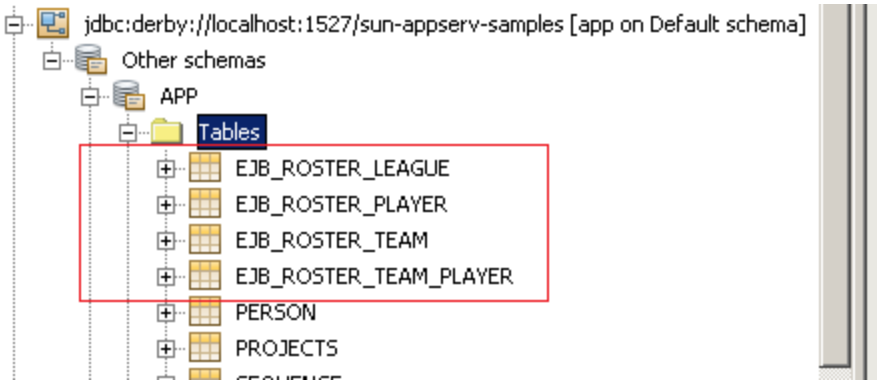
Facem click-dreapa->Properties pe numele resursei JDBC care apare in persistence.xml: "**jdbc/__default**". In fereastra de proprietati apare ca foloseste un "pool" cu numele "DerbyPool" (am amintit la curs ;-) ce-I un "pool" de conexiuni). Acum facem click-dreapta pe "DerbyPool" de sub "Connection Pools". Din fereastra de proprietati(in dreapta in poza de mai jos) obtinem informatii despre baza de date unde se creeaza tabelele din proiect. Dupa ce m-am conectat la baza de date (parola "APP") observam tabelele deja existente(in "Tbales"):





Dupa ce am facut cu succes DEPLOY, click-dreapta pe "Tables" si, daca am respectat setarile discutate

mai sus, trebuie sa-mi apara tabele din proiectul roster-EJB iar in Applications, dupa Refresh,va apare “roster-EJB”.
Pentru instalarea modului se revine in tablul “Projects”, click-dreapta pe numele proiectului(roster-EJB), selectez “Clean and Build” si daca e ok, din nou click-dreapta pe “roster-EJB” si acum selectez “Deploy”. Cand s-a terminat (in cateva secunde) merg din nou in “Services” si dau refresh la “Tables”. Trebuie apara tabelele noastre:



Click-dreapta pe fiecare tabel, apoi pe fiecare coloana si “Properties”=> se afla proprietatile care ne intereseaza

Informatiile despre tabele generate, inclusiv legaturile dintre ele, se pot afla studiind adnotarile din clasele “entity”: toate clasele din package “roster.entity”.

RECOMAND ca inainte de a crea al 2-lea proiect (prin care accesam persistenta) sa studiem adnotarile utilizate clasele entity:

```
@Entity, @Table, @Id, @ManyToOne, @ManyToMany, @JoinTable
```

Documentatie din tutorial:https://docs.oracle.com/javaee/6/tutorial/doc/giqsq.html

Alte documentatii:

https://www.lucidchart.com/pages/ER-diagram-symbols-and-meaning
http://plugins.netbeans.org/plugin/53057/jpa-modeler

Ce mai pot face din Netbeans:

- Click dreapta pe numele roiectului (in ferestra Projects)
- New->Other…->Persistence->apare o lista

In Parte a 3-a vom accesa datele din aplicatia client