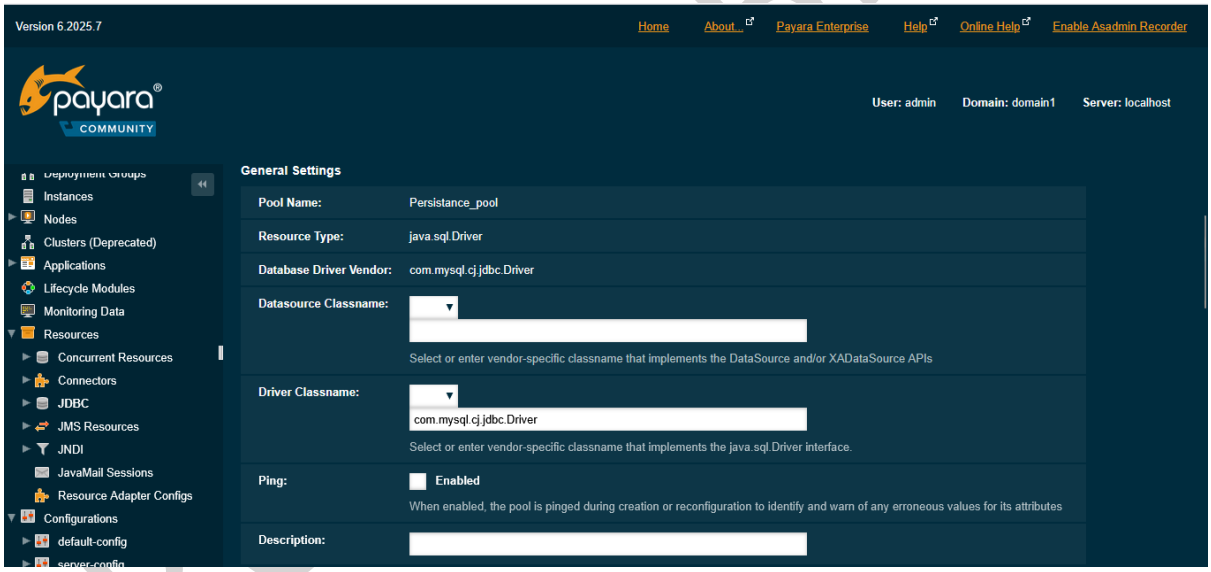
ENTERPRISE JAVA

➡ **Java Persistance API :-**

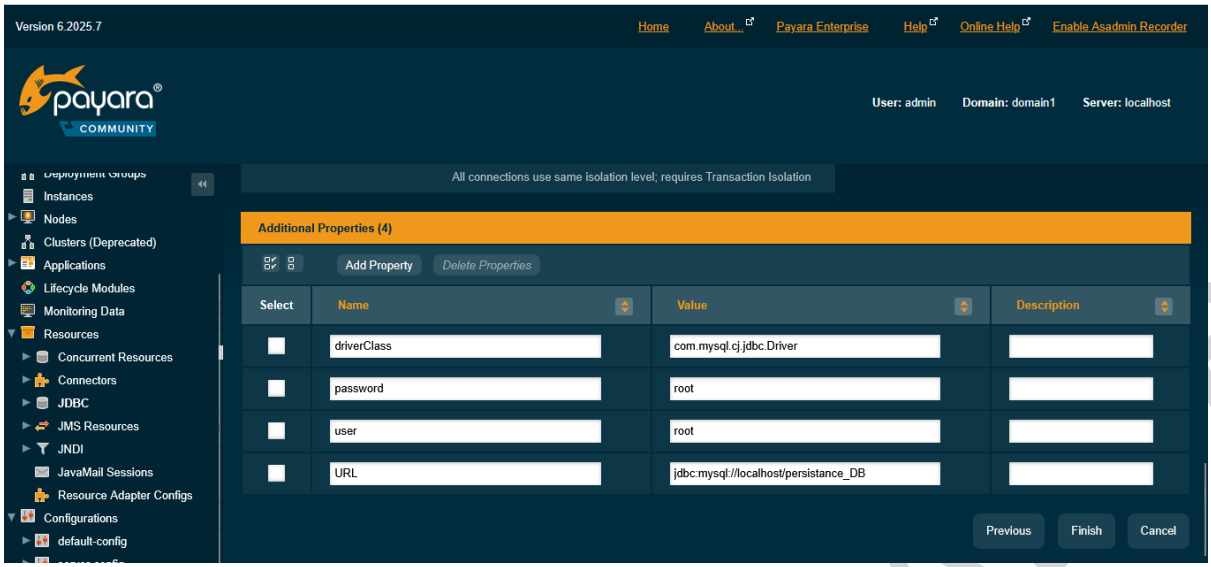
JDBC Connection Pool :--

Go to Payara server -> Right Click -> View Domain Admin Console -> http://localhost:4848/common/index.jsf

In Domain Admin Console -> JDBC -> JDBC Connection Pool -> new ->Pool Name -> name(Persistance_pool) -> Resource Type : java.sql.Driver -> Database Driver Vender : com.mysql.cj.jdbc.Driver -> Next -> Driver Classname : com.mysql.cj.jdbc.Driver -> Additional Properties -> Password : (phpMyAdmin password) , user : root , URL : jdbc:mysql://localhost/persistance_DB



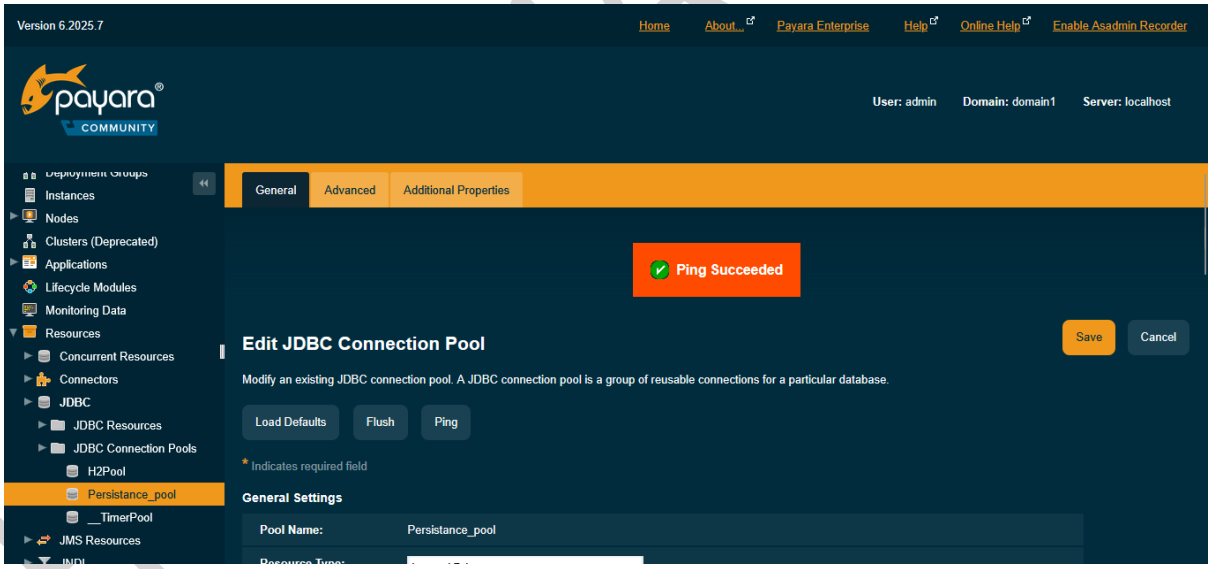
➡ Then go to phpMyAdmin -> create new Database -> name same as above : persistence_DB -> then return Payara Domain Admin console -> add new property -> driverClass :com.mysql.cj.jdbc.Driver



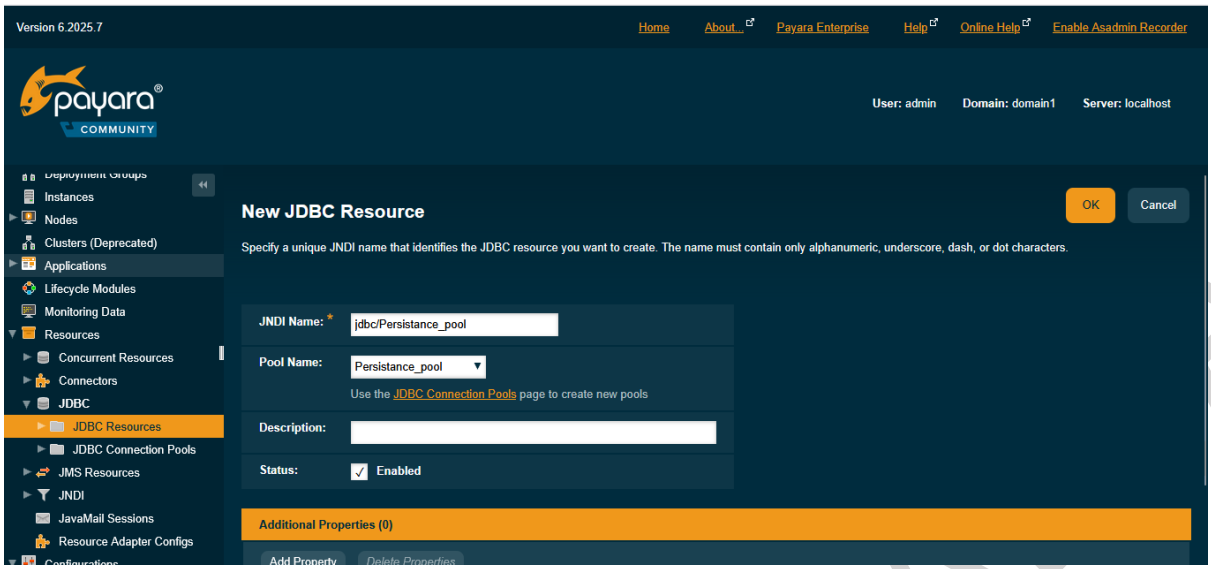
➡ Note that

- **payara6\glassfish\domains\domain1\lib\mysql-connector-j-8.0.29**

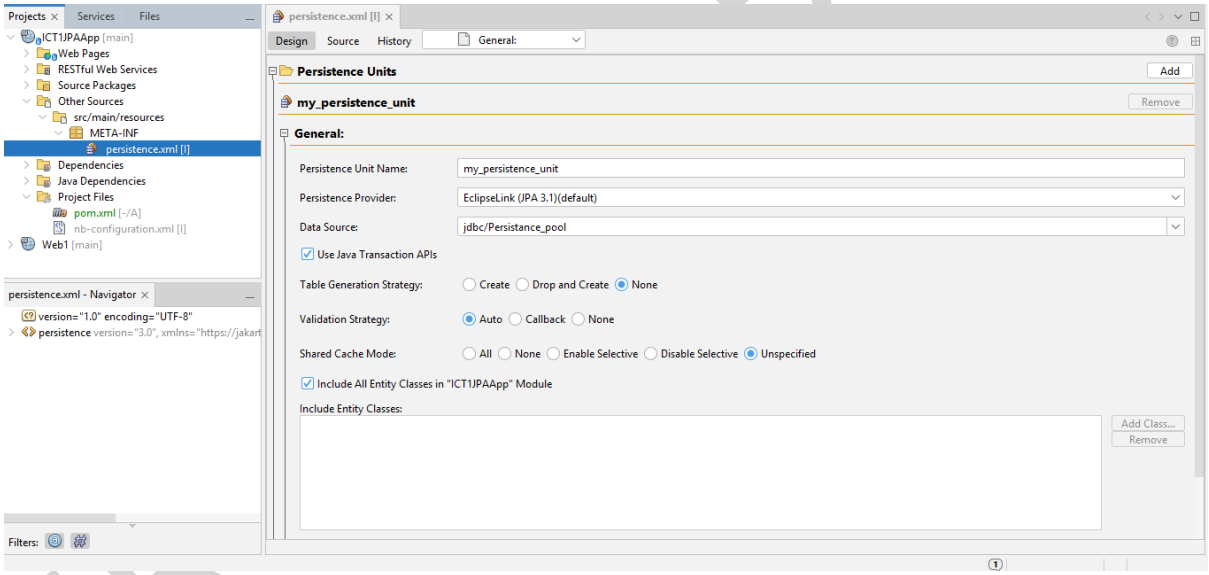
above in this location jar file compulsory -> restart payara server -> netbeans ->



After Ping Suceeded -> go to JDBC Resources -> New JDBC Resource -> JNDI Name : jdbc/Persistance_pool -> Pool Name : Also Select Your Pool Name (Persistance_pool)



In netbeans -> persistence.xml :-



- ➡ IN netbeans -> Source Packages -> new -> Entity Class -> name(Theater) -> package (entity) -> Finish
- ➡ Then creating Entity(movie)

JSF (Java Server Faces):-

- ➡ For designing purpose need to create xhtml pages

How to create :--

- In .xhtml readymade form designing just Alt + Insert -> open JSF table From entity -> select entity -> Managed Bean property automatically selected. -> Templates styles must be Standard JavaServer Faces -> OK

What is JPA?

- JPA (Java Persistence API) is a standard way to store, update, delete, and fetch Java objects from a database (like MySQL, PostgreSQL, etc.) — without writing raw SQL.

Why it's useful :-

- Because in enterprise apps (like yours with EJBs or Servlets), you often need to save Java data into a relational database.

- **JPA automates all of that.**
  You work with simple Java classes (called **entities**), and JPA handles the SQL behind the scenes.

How JPA works :-

Entity Class — A normal Java class mapped to a database table:

EntityManager — The main JPA object that handles database operations:

Persistence Unit — Defined in persistence.xml (inside META-INF/):

Why we use it :-

- Simplifies database code (no JDBC boilerplate).
- Works across multiple databases (only config changes).
- Integrates easily with **EJB**, **Servlets**, **Spring**, etc.
- Supports **object relationships** (@OneToMany, @ManyToOne, etc.)