

---

# L7. TUTORIAL Eclipse Java Persistence API

IDE:	Eclipse Luna/ <b>Mars</b> JEE [4.4/ <b>4.5</b> ] Distribution: <b>OEPE 12.1.3.x</b>
Implementare JPA:	JPA 2.1 / <b>EclipseLink</b> 2.5.2
SGBD:	PostgreSQL9
Driver JDBC:	postgresql-9.jdbc

---

# Plan

---

- 1. Creare și configurare Eclipse
  - 2. Declarare entități JPA și creare structuri de mapare relaționale
  - 3. Test tranzacții JPA: salvare și interogare
-

# 1. Configurare Eclipse

---

- 1.1 Configurare mediu Eclipse
    - Distribuția de bază Eclipse Luna/Mars.
    - Plug-in-uri necesare JEE-OEPE:
      - Eclipse Data Tools Platform;
      - Eclipse Web Tools Platform;
      - Oracle Database Tools;
      - Eclipse Link Project: Java Persistence API 2.1.
    - Configurare server ApacheTomcat cu suport EclipseLink.
  - 1.2 Configurare proiect Eclipse JPA
    - 1.2.1 Creare proiect JPA cu alegere distribuție JPA
    - 1.2.2 Creare conexiune la baza de date
    - 1.2.3 Configurare proiect JPA
-

# Instalare server Apache Tomcat

---

- 1. În laboratoarele FEAA, serverul ApacheTomcat *preconfigurat* se găsește în folderul local:
  - E:\\_Programare2\\_apache\_tomcat
- Pentru alte locații (laptopuri, notebookuri personale):
  - Descărcare *arhivă-kit apache-tomcat-7* de pe *portal*.
  - Dezarhivare kit în directorul-gazdă pentru serverul de aplicații Web Tomcat:
    - E:\\_Programare2\\_apache\_tomcat
  - Localizare biblioteci (jars) corespunzătoare JPA în
    - E:\\_Programare2\\_apache\_tomcat\lib
- 2. Adăugare (locăție) server Apache Tomcat în contextul workspace-ului Eclipse.

# Kitul ApacheTomcat preconfigurat de pe portal

---

**Portal FEAA**

Lucrari practice (laborator) ▸ L7.POO.JPA

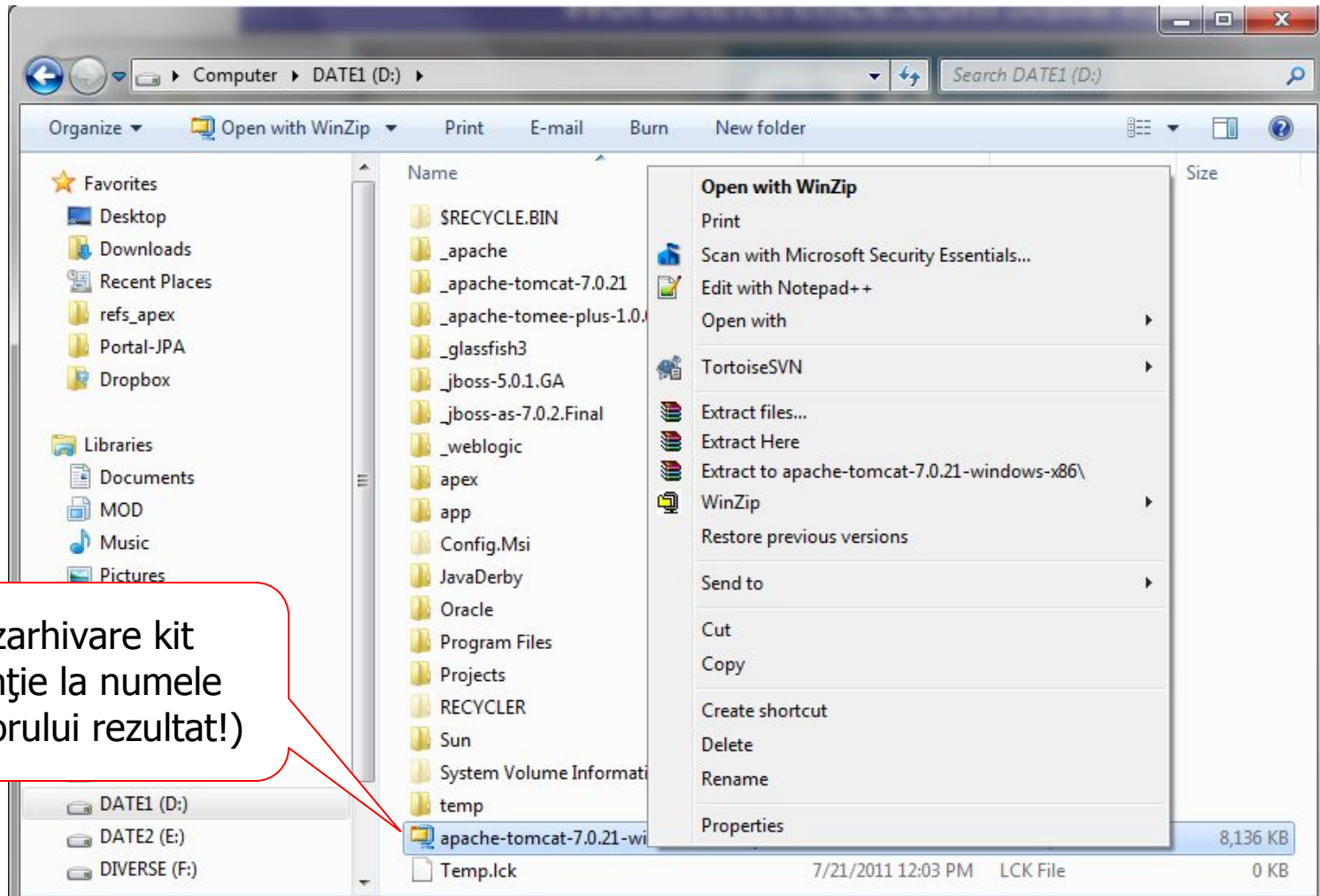
⊕ new document or drag files here

All Documents ...

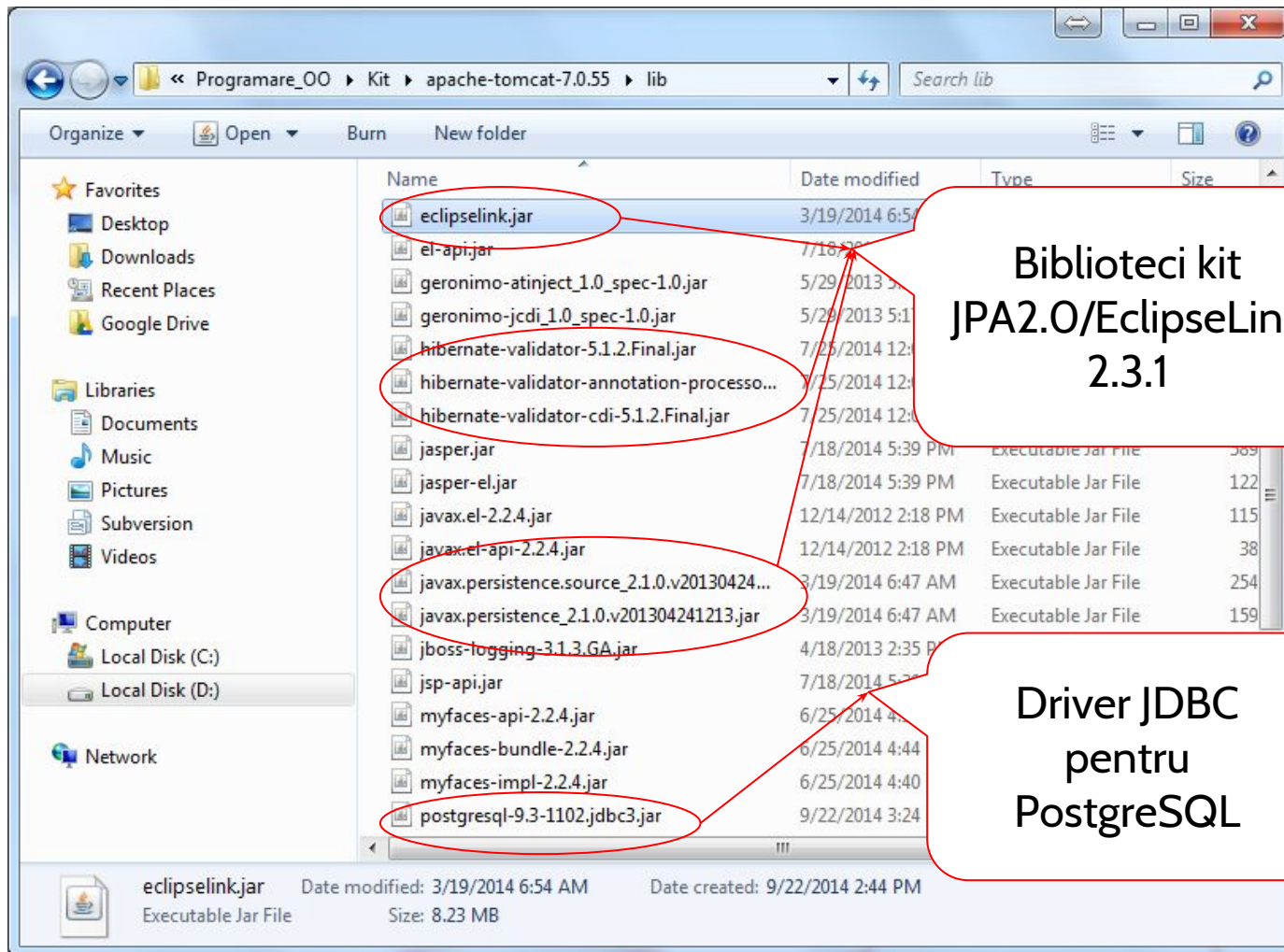
✓	📄	Name	Modified	Modified By	Checked Out To
	📁	apache-tomcat-7.0.55-plus ✎	4 minutes ago	☐ Catalin STRÎMBEI	

Descărcare kit

# Kit Apache Tomcat preconfigurat descărcat



# Localizare biblioteci JPA preconfigurate

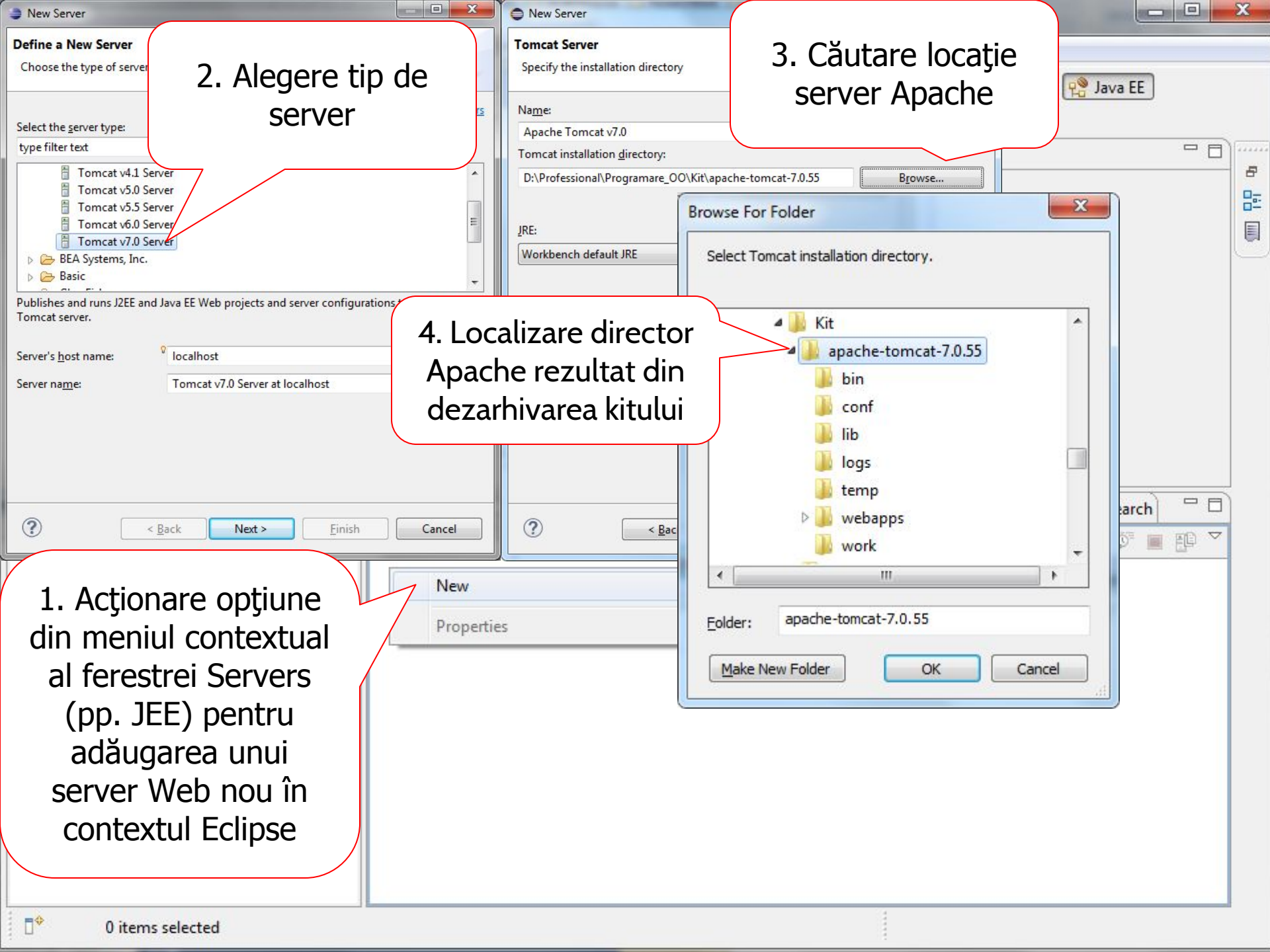


# Configurare Apache Tomcat în Eclipse

---

1. Activarea ferestrei *Servers*, din *perspectiva Java EE* și acționare opțiune adăugare server.
  2. Selecție tip server Apache Tomcat 7.
  3. Activare fereastră de configurare Apache Tomcat 7.
  4. Selecție locație (director) runtime Apache Tomcat.
-





2. Alegere tip de server

3. Căutare locație server Apache

4. Localizare director Apache rezultat din dezarhivarea kitului

1. Acționare opțiune din meniul contextual al ferestrei Servers (pp. JEE) pentru adăugarea unui server Web nou în contextul Eclipse

## 1.2.1 Create project JPA

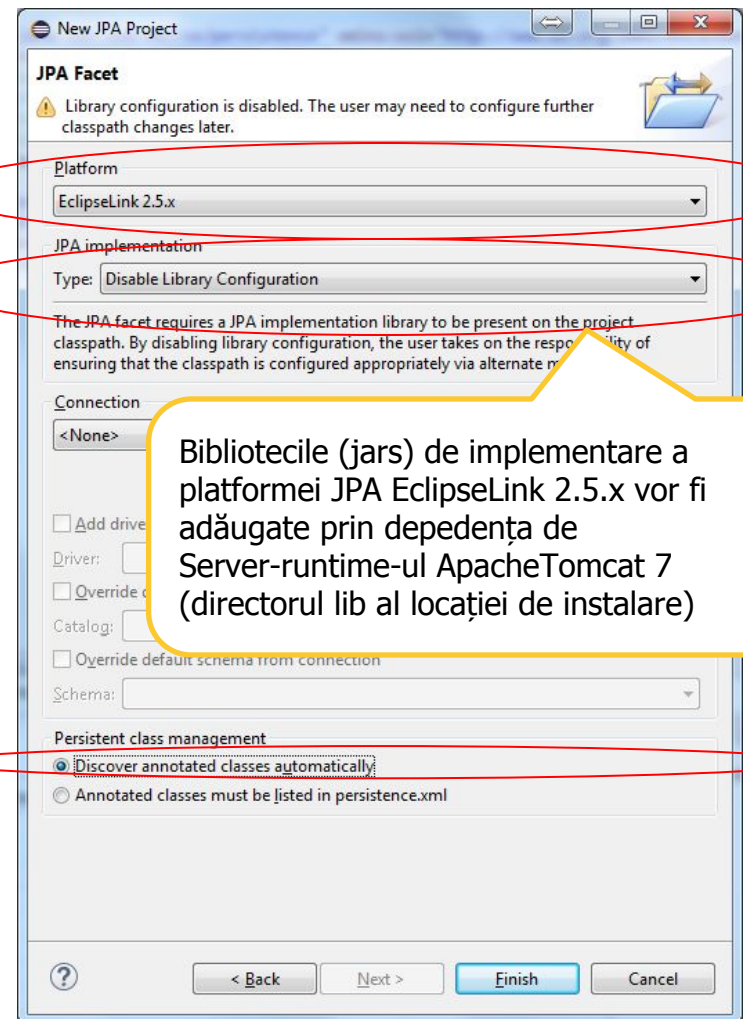
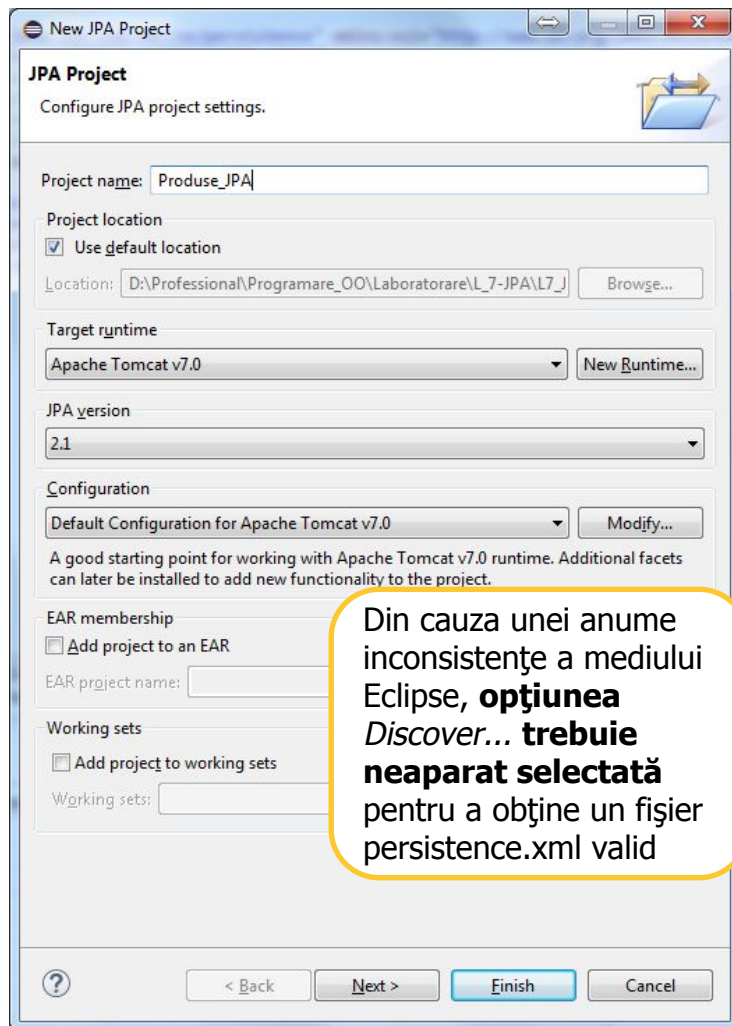
### File – New – JPA Project *wizard*

---

- **Pas 1:** JPA Project [Configure JPA Project Settings]
    - Project Name: **ProduseJPA**
    - Target Runtime: **Apache Tomcat v7.0**
    - JPA Version: **2.1**
    - Configuration: **Default Configuration**
  - **Pas 2:** Java [Configure project for building a Java application]
    - *Defaults*
  - **Pas 3:** JPA Facet [Configure JPA settings]
    - Platform: *Eclipse Link 2.5.x*
    - Connection: *Add connection*, optional
    - Persistent class management : *discover annotated classes automatically*
-

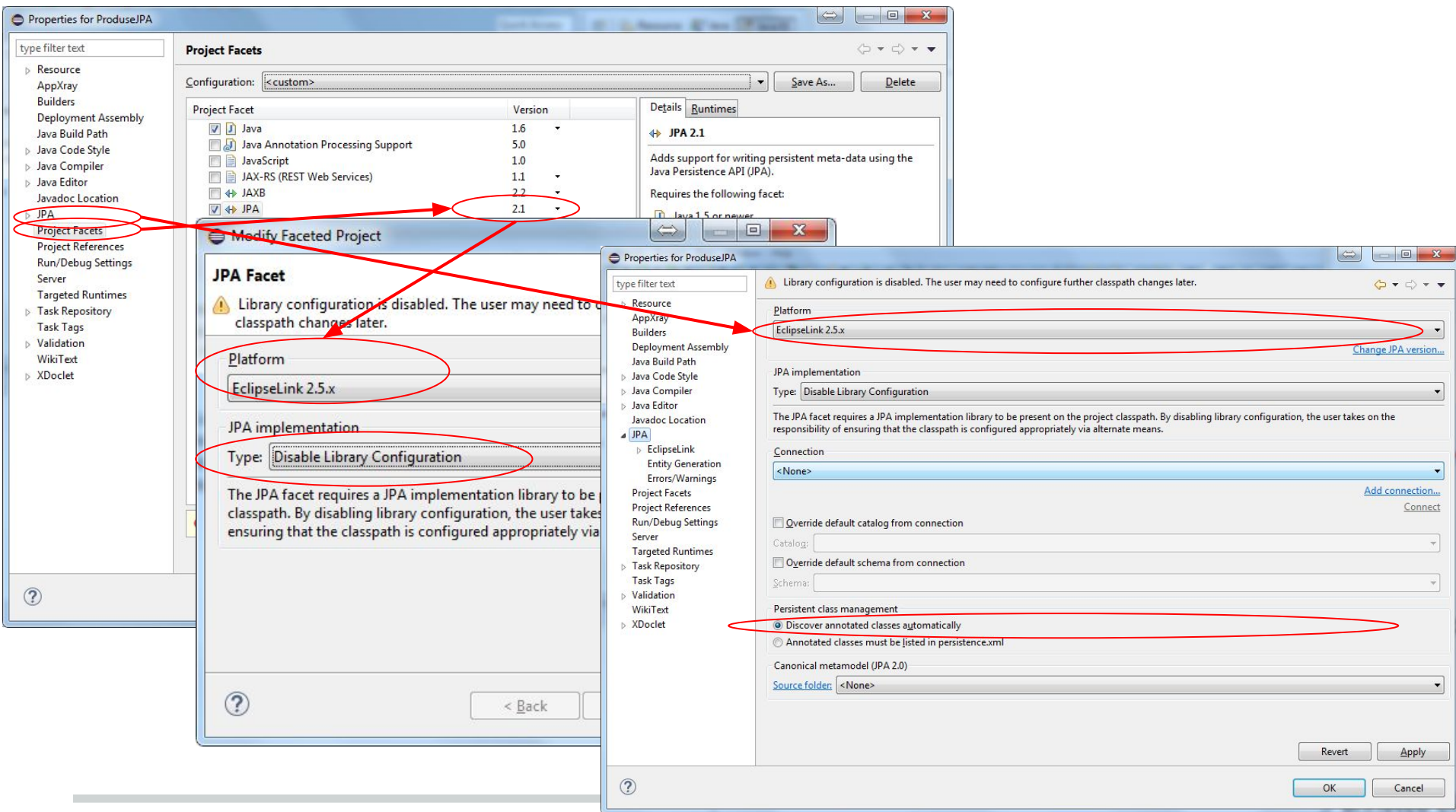
# [Pas1: New JPA Project]

## [Pas3: Configurare JPA Facet: Eclipse 2.5.x]



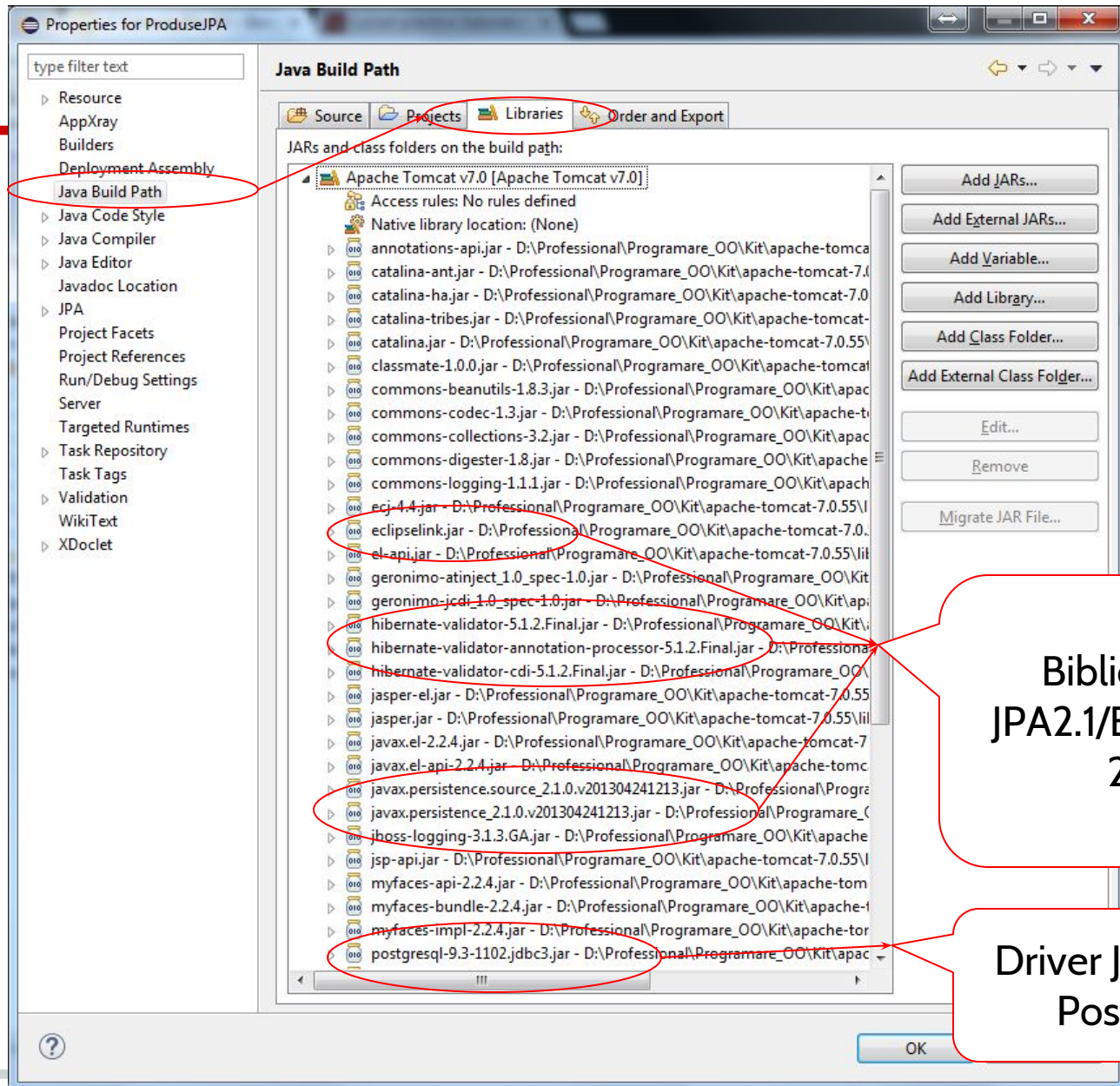
# [Pas4: Project Properties (menu contextual): Verificare Project Facet with:

- JPA 2.1 checked;
- JPA Category with Eclipse Link 2.5.x].





# Verificare bibliotecii JPA/EclipseLink din Fereastra Properties (pp. Java, meniul contextual al proiectului, opțiunea Properties)



In cazul in care bibliotecile JPA/EclipseLink lipsesc din Fereastra Properties (lipsind de fapt dependenta catre runtime-ul serverului Apache Tomcat), procedura de urmat este următoarea:

**Pas 1: Lansare secventa adaugare biblioteca**

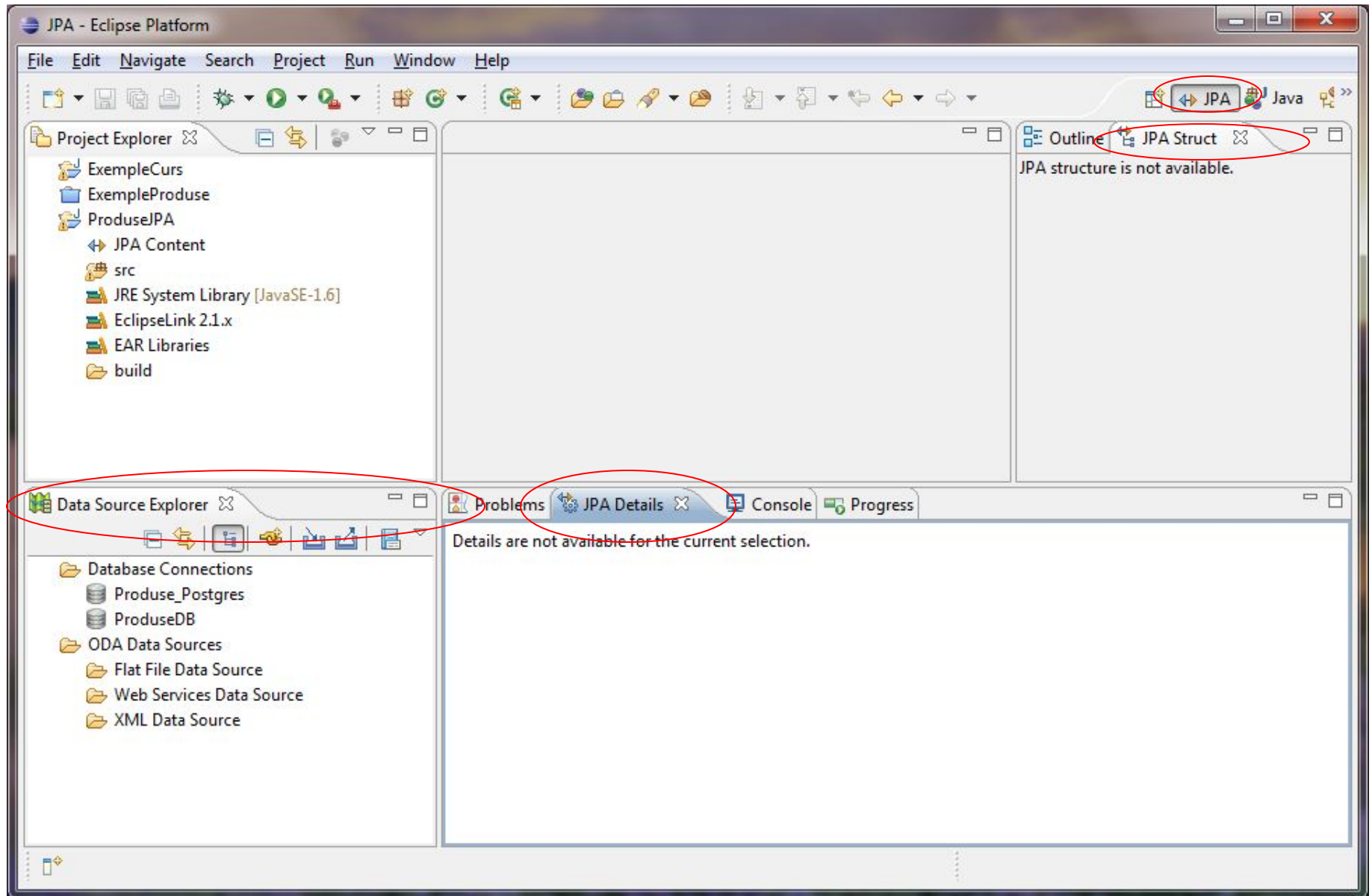
**Pas 2: Alegere tip biblioteca: Server Runtime**

**Pas 3: Selectare Server Apache al carui runtime contine si JARurile JPA**

# Organizare mediu Eclipse - Perspectiva JPA

---

- Ferestre generice
    - Project Explorer
    - Problems
    - Outline
  - Ferestre specifice
    - Data Source Explorer
    - JPA Structure
    - JPA Details
-





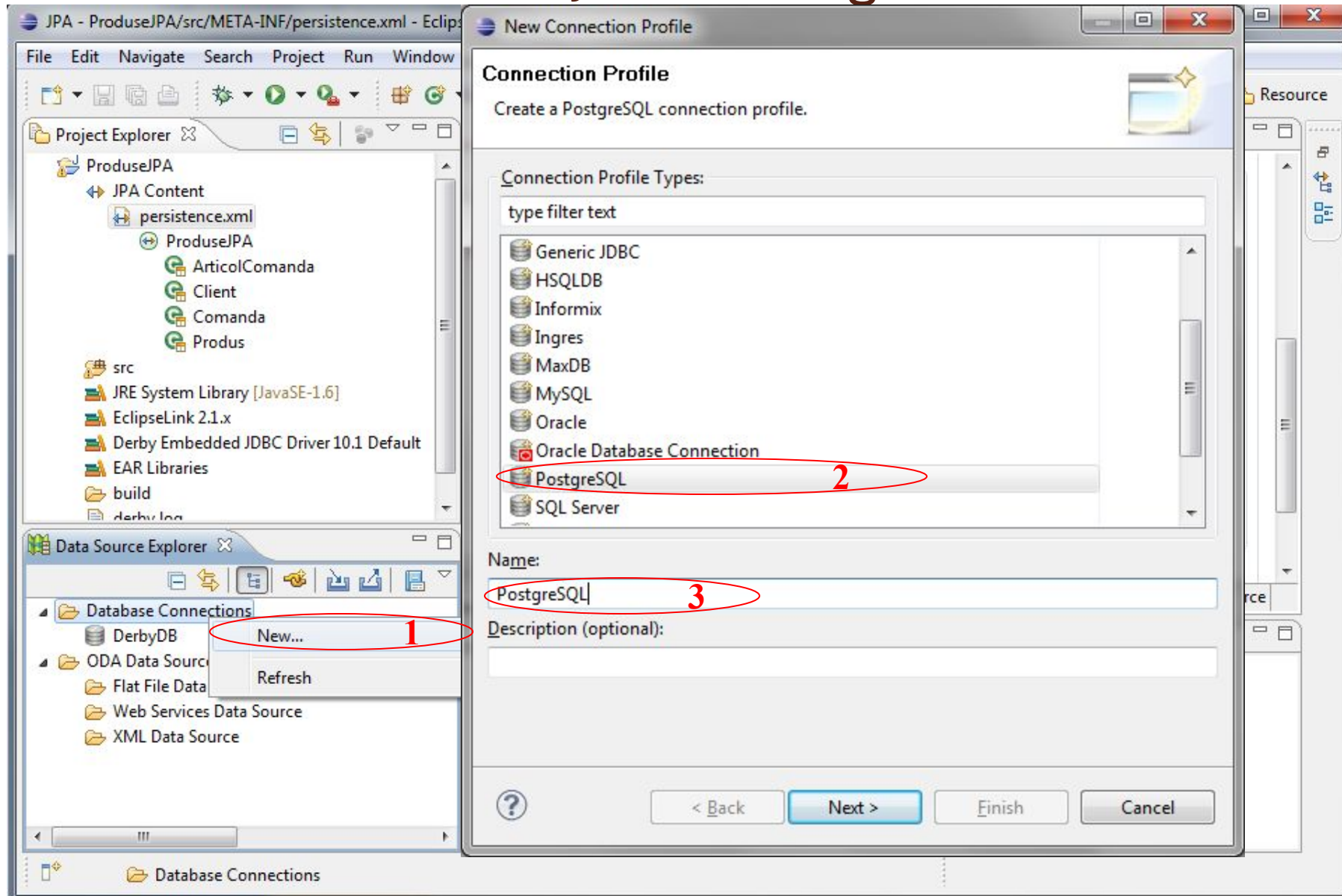
## 1.2.2 Conexiune server BD

### Perspectiva JPA – Data Source Explorer

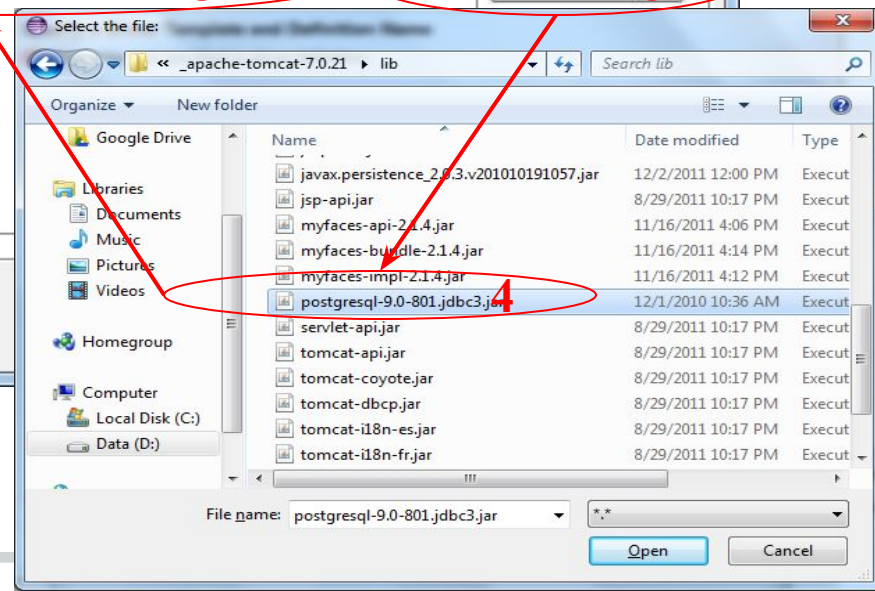
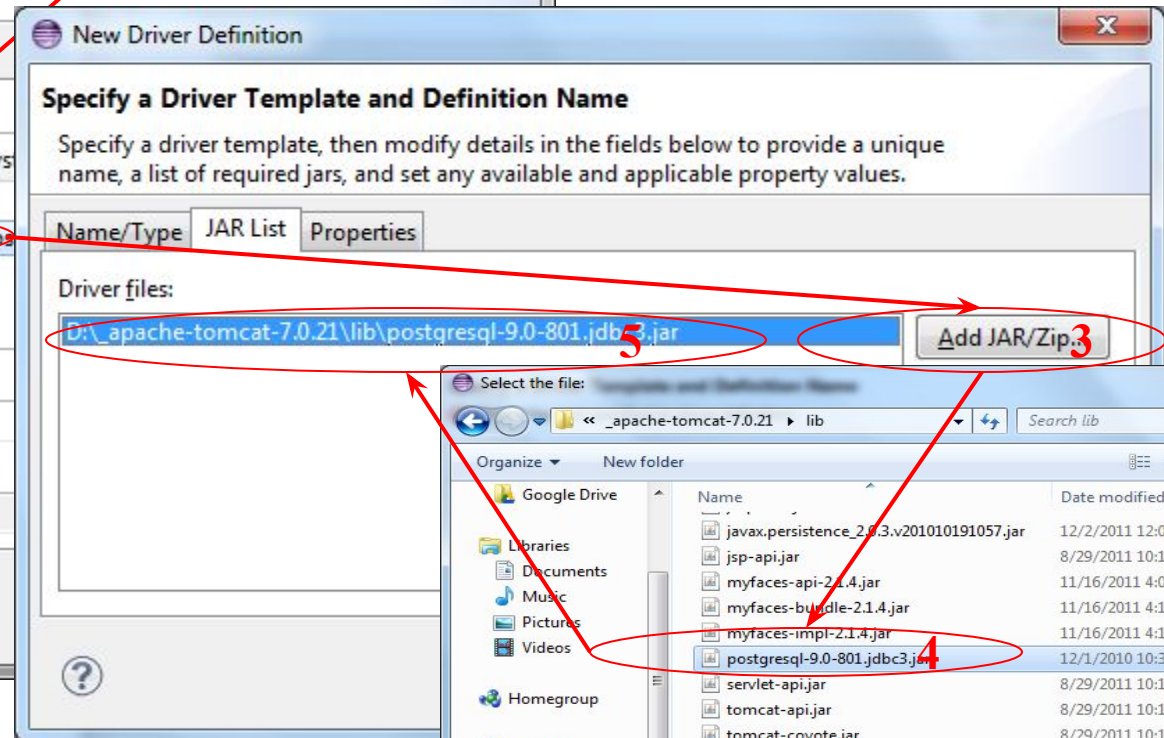
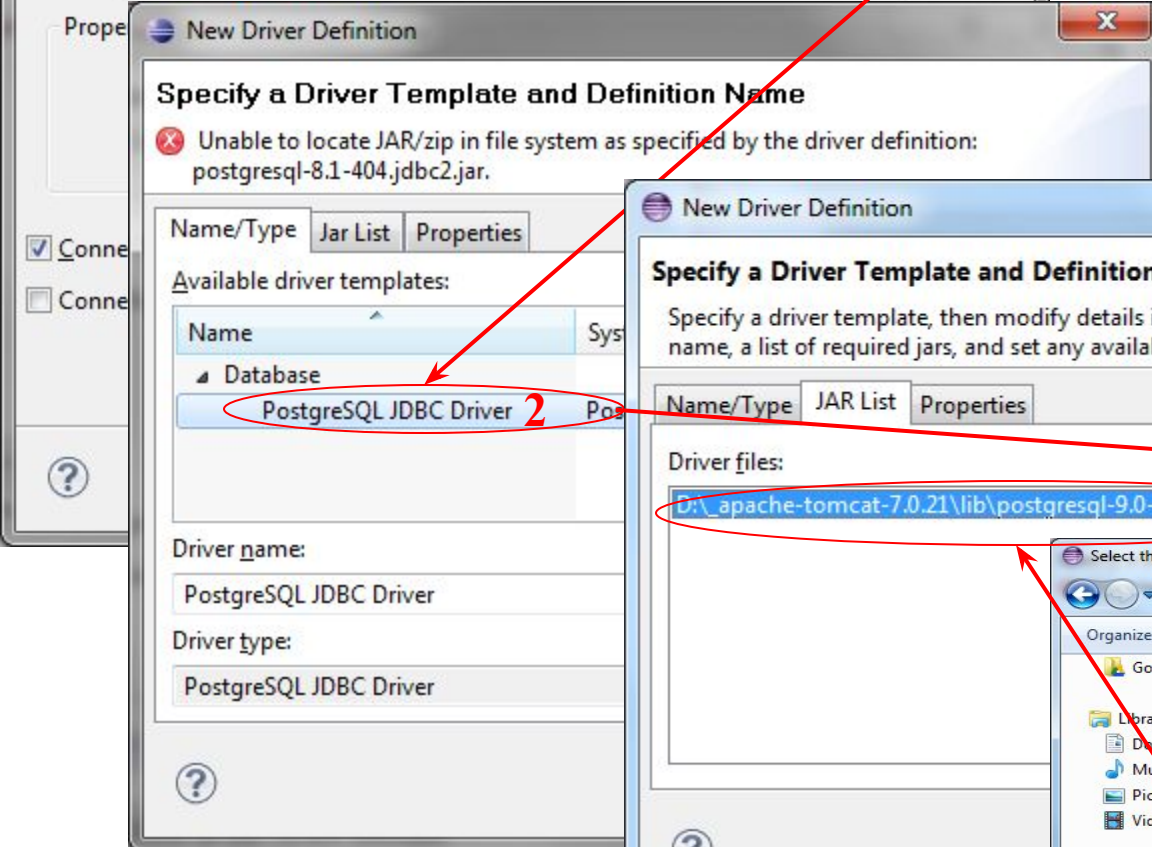
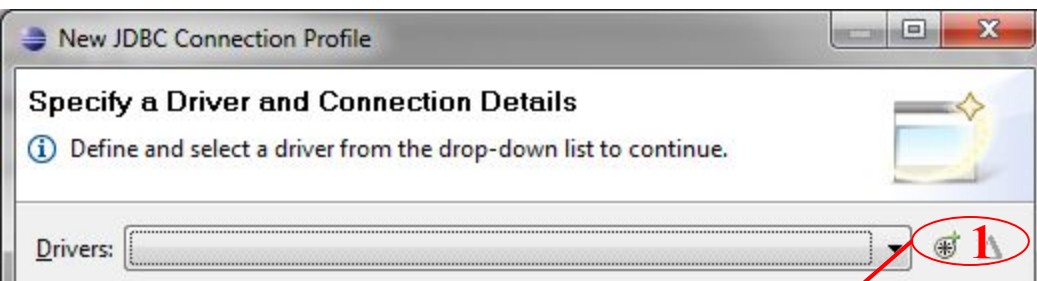
---

- Definire conexiune bază de date Postgres
    - Creare conexiune nouă JDBC
    - Completare definiție driver JDBC-PostgreSQL
    - Specificare date de conectare PostgreSQL
  - Verificare conexiune în Data Source Explorer
-

# Creare nouă conexiune JDBC - PostgreSQL



Completare definiției driverului  
Postgres care punctează către  
biblioteca postgres-jdbc.jar  
descărcată anterior



# Specificare date de conectare PostgreSQL: adresa serverului Postgres, numele schemei/contului pentru conectare

New JDBC Connection Profile

**Specify a Driver and Connection Details**

Select a driver from the drop-down and provide login details for the connection.

Drivers: PostgreSQL JDBC Driver

Properties

General Optional

Database: database

URL: jdbc:postgresql://85.122.22.12:5432/postgres

User name: produse

Password: ••••••

☒ Save password

☒ Connect when the wizard completes

☐ Connect every time the workbench is started

Test Connection

< Back Next > Finish Cancel

New JDBC Connection Profile

**Specify a Driver and Connection Details**

Select a driver from the drop-down and provide login details for the connection.

Drivers: PostgreSQL JDBC Driver

Properties

General Optional

Database: database

URL: jdbc:postgresql://10.10.0.17:5432/postgres

User name: scrum

Password: •••••

☒ Save password

☒ Connect when the wizard completes

☐ Connect every time the workbench is started

Test Connection

< Back Next > Finish Cancel

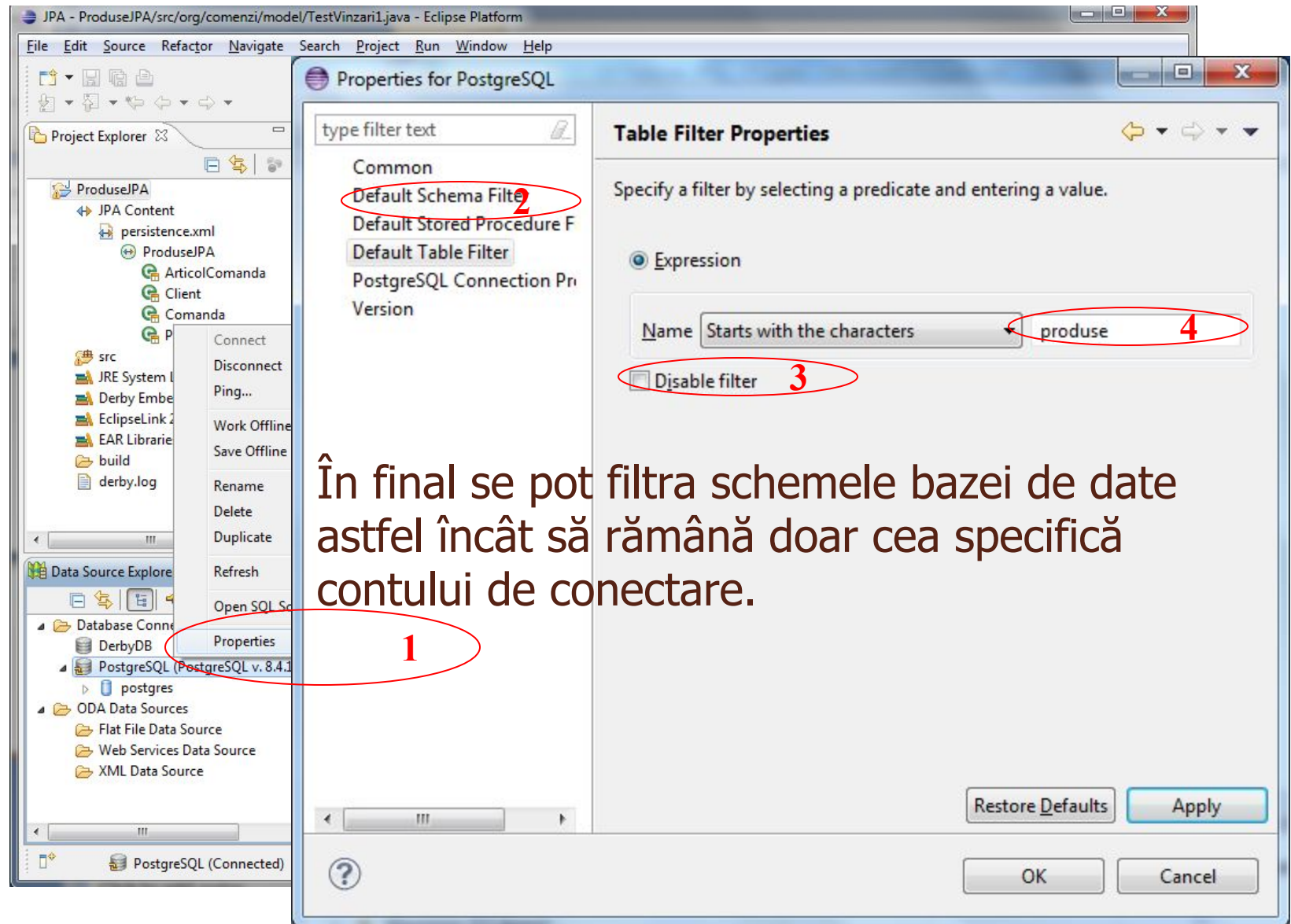
# Pentru conectarea la serverul PostgreSQL local

---

- URL:
    - jdbc:postgresql://localhost:5432/postgres
  - Username:
    - postgres
  - Password (de la instalarea serverului BD):
    - postgres
-



# Verificare conexiune în Data Source Explorer



## 1.2.3 Pregătire proiect JPA

---

- 1.2.3.1 Preluare definiție conexiune JDBC în configurația proiectului
  - 1.2.3.2 Crearea claselor entități
-

## 1.2.3.1 Configurare JPA – preluare conexiune PostgreSQL

---

- Din [perspectiva JPA] Project Explorer – Proiect JPA - **JPA Content** – **persistence.xml** – meniu contextual - **Open**
- Din [fereastra de lucru principală] Editor persistence.xml
  - Pagina **Connection** – secțiunea Persistence Unit Connection
  - Transaction Type: **Resource Local**
  - Database – EclipseLink Connection Pool
    - **Populate from connection – Connection Selection**
      - selectare nume conexiune JDBC definită anterior
      - Definire automată
        - **Driver**: [org.postgresql.Driver]
        - **URL** de forma:  
[jdbc:postgresql://85.122.22.12:5432/postgres]
        - **User & password**



# Configurare Proiect JPA:

## Preluare conexiune în definiția unității de persistență

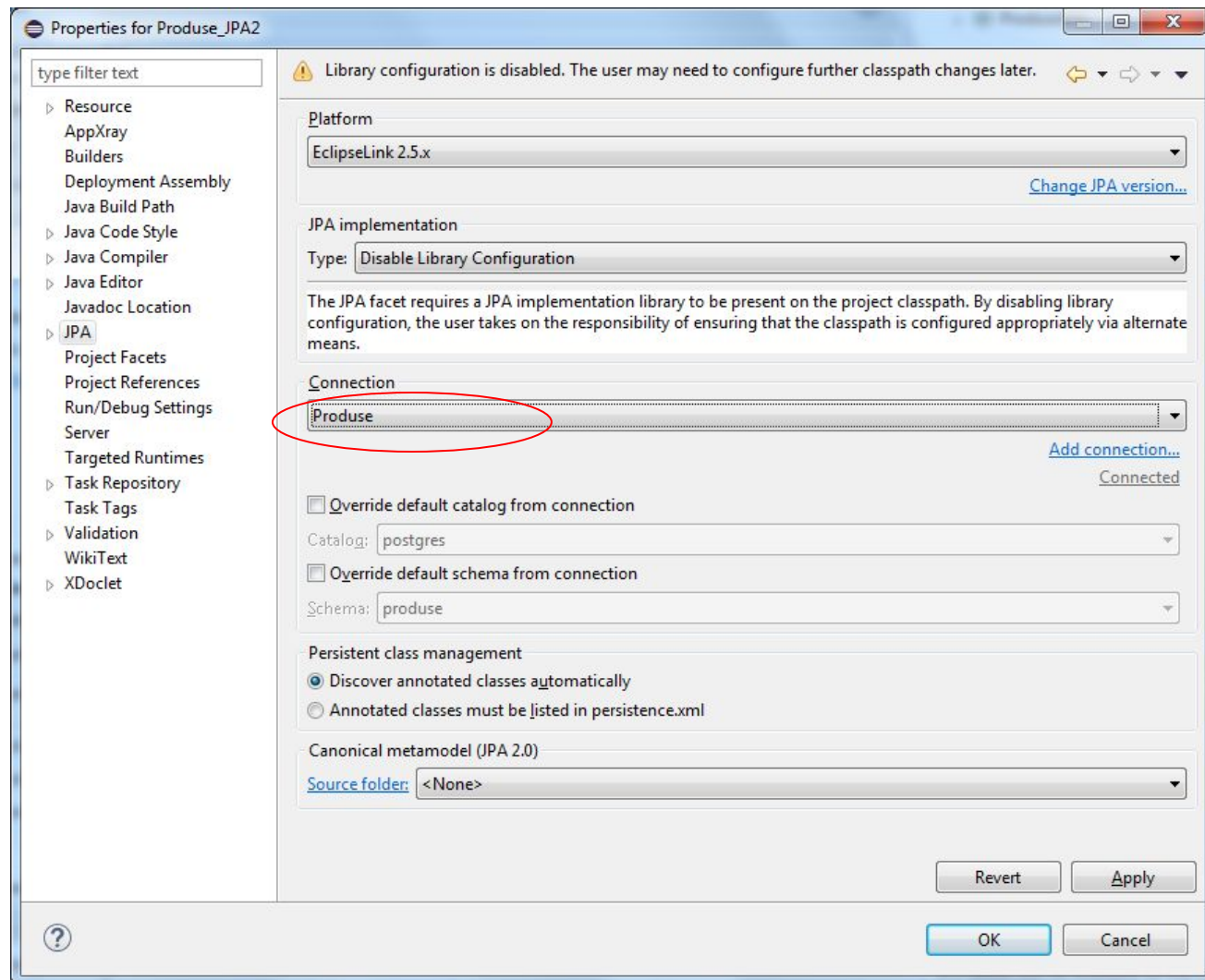
The screenshot illustrates the steps to configure a JPA persistence unit in Eclipse IDE. The interface is divided into several panes:

- Project Explorer:** Shows the project structure. The file `persistence.xml` is selected, and a context menu is open with the **Open** option highlighted. This step is labeled **1: Open**.
- persistance.xml Editor:** Displays the configuration for the **Connection** tab. The **Transaction type** is set to **Resource Local**, labeled **3**. The **EclipseLink connection pool** section is expanded, and the **Populate from connection...** link is highlighted, labeled **4**.
- Connection Selection Dialog:** A modal dialog box is open, showing the **Matching items** list. The item **Produce** is selected, labeled **5**. The dialog also shows the **Enter connection name or pattern** field and **OK** and **Cancel** buttons.
- Data Source Explorer:** Shows the available data sources, including **BIRT Classic Models Sample Database** and **Produce (PostgreSQL v. 9.3.5)**.
- Bottom Panel:** The **Customization** tab is selected, labeled **2**.
- Top Right:** The **5 Save** button is highlighted in a red oval.

The overall process involves opening the `persistence.xml` file, navigating to the **Connection** tab, selecting the **EclipseLink connection pool**, and choosing the **Produce** data source from the **Connection Selection** dialog. The **Customization** tab is also visible at the bottom.

# Preluare conexiune în proprietățile proiectului JPA

- Din Project Explorer – meniu contextual al proiectului – opțiunea properties – secțiunea Java Persistence



## 2. Declarare entități JPA

---

- 2.1 Creare clase POJO
  - 2.2 Annotare clase POJO
  - 2.3 Generare tabele (structuri relaționale persistente) în baza de date
  - 2.4 Testare structuri de persistență din aplicație
-

## 2.1 Creare clase POJO

---

- Fiecare clasă POJO trebuie să aibă:
  - **Attribute** încapsulate – private sau protected
    - Trebuie să existe în mod obligatoriu un atribut definitoriu pentru identitate – **atribut identificator** (cod, id, etc.).
  - **Proprietăți**
    - Source – Generate getters and setters.
  - **Constructor**
    - Cu parametri (opțional, dar recomandabil)
      - Source – Generate constructor using fields – selectare attribute
    - **Fără parametri (obligatoriu)**
      - Source – Generate constructor using fields – deselectare toate attributele
  - Metoda **equals** care să implementeze criteriul de egalitate (de ex. bazat pe atributul **id**)
    - Source – Generate hashCode and equals – selectare numai atribut definitoriu pentru identitate

## 2.1 Creare clase POJO

---

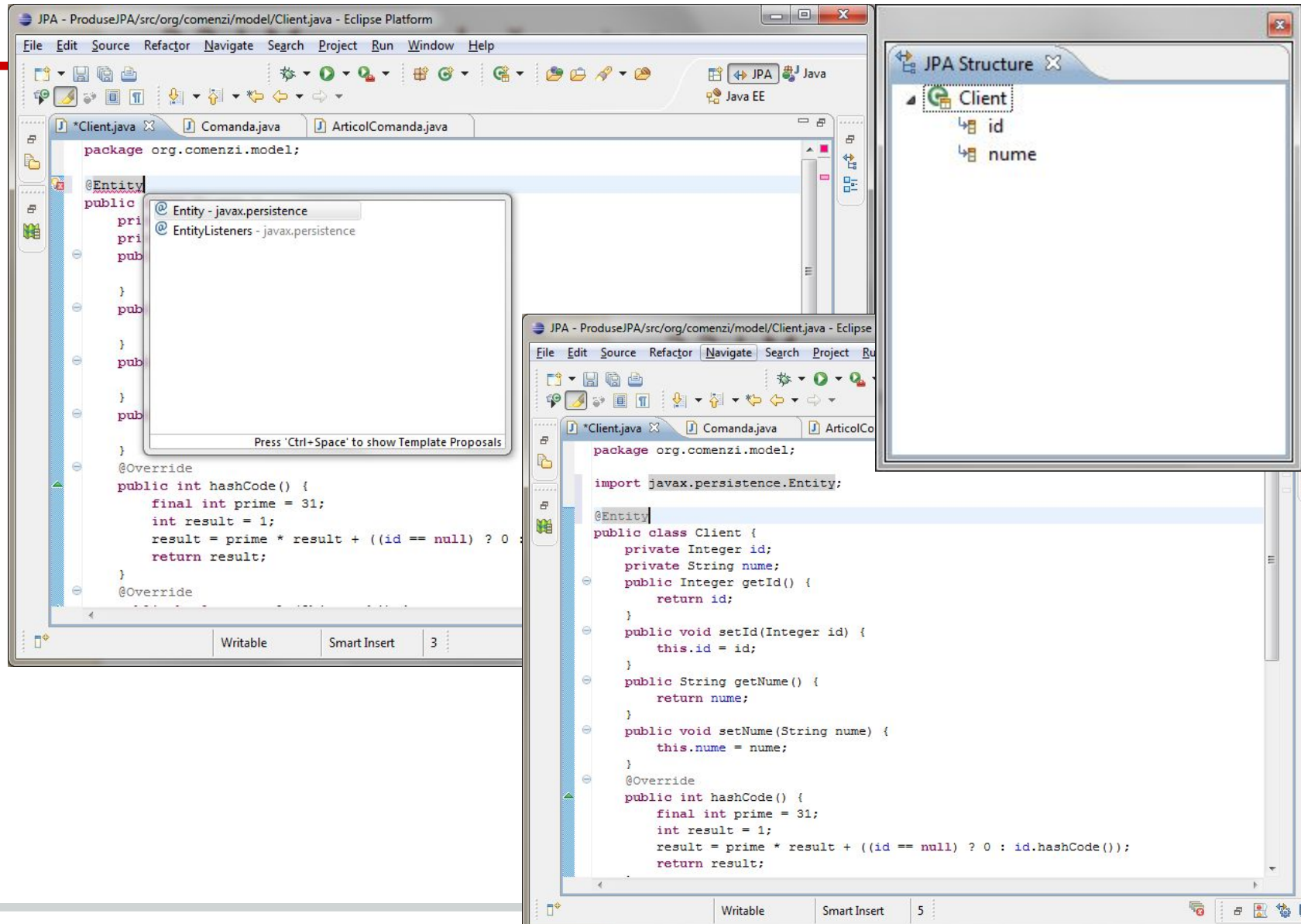
- Descărcați de pe portal din folderul L7.POO.JPA fișierele \*.java și copiați clasele din directorul **org.app.model** într-un subdirector cu același nume al directorului **src** al proiectului JPA.
  - Din mediul Eclipse invocați opțiunea **Refresh** din meniul contextual pe directorul **src**.
  - Verificați existența claselor entități: **Client, Probus, ArticolComanda, Probus**
-

## 2.2 Anotare clase POJO

---

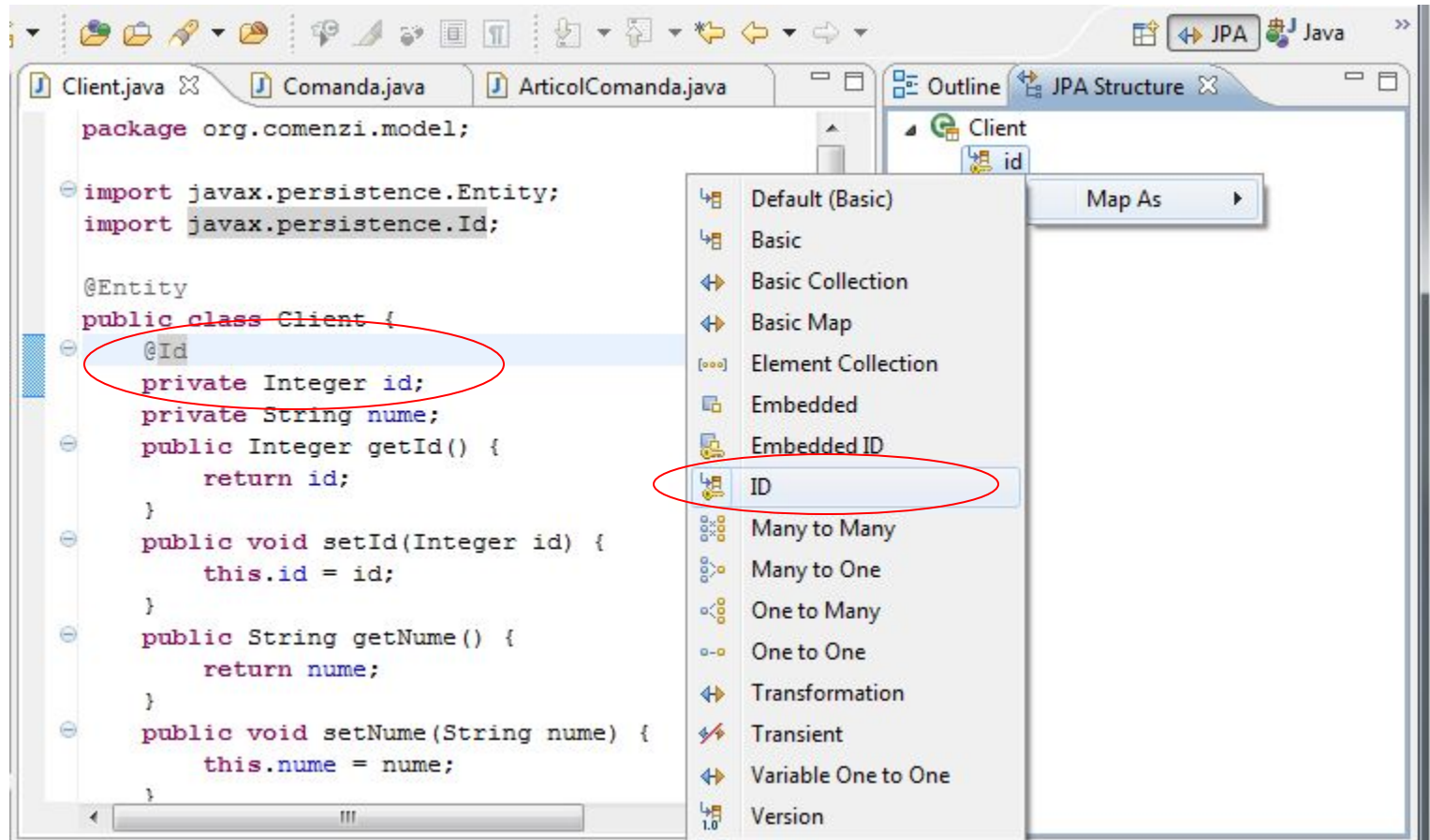
- Aplicați procesul de transformare a claselor POJO în entități persistente pentru *Produs*, *Client*, *ArticolComanda*, *Comanda* așa după cum vă este sugerat din paginile ce urmează.
  - Marcați mai întâi entitățile, identificatorii și eventualele tipuri de date (date calendaristice în special) pentru toate clasele.
  - Reveniți la final pentru marcarea relațiilor.
-

## 2.2.1 Marcare clasă entitate





## 2.2.2 Marcare attribut **identificator**





# Strategie generare *valori identificator*

## @Id @GeneratedValue

```
*Client.java  Comanda.java  ArticoloComanda.java

package org.comenzi.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import static javax.persistence.GenerationType.AUTO;

@Entity
public class Client {

    @Id
    @GeneratedValue(strategy = AUTO)
    private Integer id;
    private String nome;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }
}
```

Problems JPA Details Console

Attribute 'id' is mapped as ID.

▼ ID

Column

Name: Default (id)

Table: Default (Client)

► Details

☒ Mutable (True)

► Type

▼ Primary Key Generation

☒ Primary key generation

Strategy: Auto

Generator name:

► Table Generator

► Sequence Generator

## 2.2.3 Marcare **tip** attribut java.util.Date

The image shows an IDE window with three tabs: `Client.java`, `*Comanda.java`, and `ArticolComanda.java`. The `*Comanda.java` tab is active, displaying the following code:

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Temporal;
import static javax.persistence.TemporalType.*;

@Entity
public class Comanda {
    @Id
    private Integer id;
    @Temporal(DATE)
    private Date dataComanda;

    private List<ArticolComanda> articole = ...;
    private Client client;
```

The `@Temporal(DATE)` annotation and the `dataComanda` attribute are circled in red.

Below the code editor, the `JPA Details` tab is active, showing the configuration for the attribute `'dataComanda'`. The message at the top states: "Attribute 'dataComanda' is mapped as [default \(basic\)](#)." The configuration is as follows:

- Basic**
  - Column:
  - Table:
- Details**
  - Fetch:
  - ☐ Optional (True)
  - ☐ Mutable (False)
- Type**
  - ☐ Default
  - ☐ Lob
  - ☒ Temporal:
  - ☐ Enumerated:
  - ☐ Converted
    - Converter name:
    - [Define Converter](#)

## 2.2.4 Marcare relații entități

The screenshot displays an IDE with the following components:

- Code Editor:** Shows the `Comanda` entity class. The `@ManyToOne` annotation and the `private Client client;` field are highlighted with a red circle.
- Outline:** Shows the `Comanda` entity structure with attributes: `id`, `dataComanda`, `articole`, and `client`.
- Context Menu:** A menu is open over the `client` field, listing various JPA relationship types. The `Many to One` option is selected and circled in red.
- JPA Details Panel:** Located at the bottom, it shows the configuration for the `Many to One` relationship. The `Target entity` is set to `Default (org.comenzi.model.Client)`, which is also circled in red. Other settings include `Fetch: Default (Eager)` and `Join fetch: <None>`.

# Marcare relații entități M-1, 1-M

## @ManyToOne ... @OneToMany

The image shows an IDE with two Java files: `ArticolComanda.java` and `Comanda.java`. The `ArticolComanda` class has a `@ManyToOne` relationship with `Comanda`. The `Comanda` class has a `@OneToMany` relationship with `ArticolComanda`. The JPA Details panel is open, showing the configuration for the `@OneToMany` relationship.

**ArticolComanda.java:**

```
package org.comenzi.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class ArticolComanda {
    @Id
    private Integer id;

    @ManyToOne
    private Comanda comanda;

    private Double cantitate;
}
```

**Comanda.java:**

```
@Entity
public class Comanda {
    @Id
    private Integer id;
    @Temporal (DATE)
    private Date dataComanda;

    @OneToMany(mappedBy = "comanda")
    private List<ArticolComanda> articole
        = new ArrayList<ArticolComanda>();
}
```

**JPA Details Panel:**

- Relationship:** One to Many
- Target entity:** Default (org.comenzi.model.ArticolComanda)
- Fetch:** Default (Lazy)
- Join fetch:** <None>
- Private owned:** ☐
- Orphan removal (False):** ☒
- Cascade:** ☐ All ☐ Persist ☐ Merge ☐ Remove ☐ Refresh ☐ Detach
- Joining Strategy:** ☒ Mapped by
- Attribute:** comanda

## 2.3 Generare tabele în baza de date

---

- 2.3.1 Verificare configurare aspecte JPA management proiect
  - 2.3.2 Generare tabele din opțiunile meniului contextual ale plug-in-ului Eclipse *JPA Tools*
-

## 2.31. Verificare configurare aspecte JPA

---

- Verificare Proiect – meniu contextual - Properties
  - Project Facets – JPA Version 2.1
  - Java Persistence
    - Platforma: Eclipse Link 2.5.x
    - **Connection: (conexiunea postgresql)**
  - *(Clasele **entități** vor fi marcate cu **erori** datorită inexistenței structurilor de mapare-tabelelor- în baza de date)*
- Sincronizare JPA Content
  - Meniu contextual al **persistence.xml** – **Synchronize class list**



# Verificare și sincronizare Entități proiect și fișier persistence.xml

The screenshot displays the Eclipse IDE interface. On the left, the 'Project Explorer' shows the 'Produce\_JPA' project with 'JPA Content' expanded. The 'persistence.xml' file is selected, and a context menu is open with 'Synchronize Class List' highlighted. Below this, the 'src' folder contains the 'org.comenzi.model' package with several Java files. On the right, the 'Properties for Produce\_JPA' dialog is open, showing the 'JPA' tab. The 'Platform' is 'EclipseLink 2.5.x'. The 'JPA implementation' section shows 'Type: Disable Library Configuration'. The 'Connection' section shows 'Produce' as the selected connection. The 'Catalog' is 'postgres' and the 'Schema' is 'produce'. At the bottom, the 'persistence.xml' file is open in the editor, showing the XML content. The XML defines a persistence unit named 'ProduceJPA' with a transaction type of 'RESOURCE\_LOCAL'. It lists several classes: 'org.comenzi.model.ArticolComanda', 'org.comenzi.model.Client', 'org.comenzi.model.ClientPF', 'org.comenzi.model.Comanda', and 'org.comenzi.model.Produs'. It also includes properties for the JDBC connection: 'javax.persistence.jdbc.url' (jdbc:postgresql://85.122.22.12:5432/postgres), 'javax.persistence.jdbc.user' (produse), 'javax.persistence.jdbc.password' (produse), and 'javax.persistence.jdbc.driver' (org.postgresql.Driver).

Project Explorer

- Produce\_JPA
  - JPA Content
    - persistence.xml
      - Go Into
      - Open
      - Synchronize Class List
    - Produce/JPA
      - ArticolC
      - Client
      - ClientPF
      - Comanda
      - Produs
  - src
    - org.comenzi.model
      - ArticolComanda.java
      - Client.java
      - ClientPF.java
      - Comanda.java
      - Produs.java
      - TestClient0.java
      - TestClient1.java
    - META-INF
    - JRE System Library [JavaSE-1.6]
    - Apache Tomcat v7.0 [Apache Tomcat v7.0]
    - EAR Libraries
    - build

Properties for Produce\_JPA

Library configuration is disabled. The user may need to configure further classpath changes later.

Platform: EclipseLink 2.5.x

JPA implementation

Type: Disable Library Configuration

The JPA facet requires a JPA implementation library to be present on the project classpath. By disabling library configuration, the user takes on the responsibility of ensuring that the classpath is configured appropriately via alternate means.

Connection: Produce

☐ Override default catalog from connection

Catalog: postgres

☐ Override default schema from connection

Schema: produce

Revert Apply OK Cancel

persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="ProduceJPA" transaction-type="RESOURCE_LOCAL">
    <class>org.comenzi.model.ArticolComanda</class>
    <class>org.comenzi.model.Client</class>
    <class>org.comenzi.model.ClientPF</class>
    <class>org.comenzi.model.Comanda</class>
    <class>org.comenzi.model.Produs</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://85.122.22.12:5432/postgres"/>
      <property name="javax.persistence.jdbc.user" value="produse"/>
      <property name="javax.persistence.jdbc.password" value="produse"/>
      <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
    </properties>
  </persistence-unit>
</persistence>
```

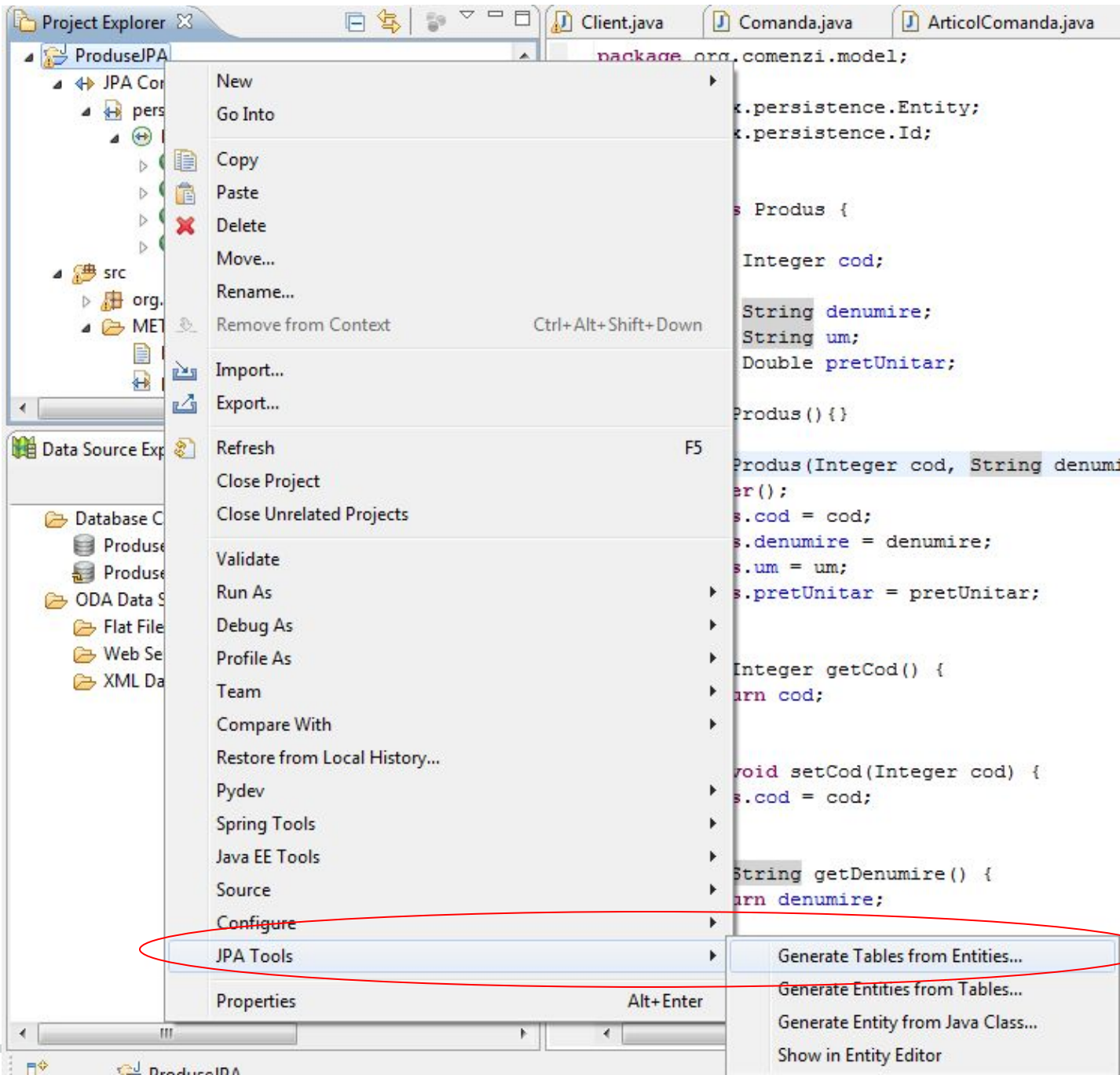
## 2.3.2. Generare tabele

---

- Proiect – meniu contextual – **JPA Tools - Generate tables from entities.**
  - Urmăriți:
    - Output-ul în fereastra Console
    - Existența tabelelor generate în Data Source Explorer.
-



# Acțiunea generare tabele



# Output generate tabelle

```
Console X
<terminated> C:\Program Files\Java\jre6\bin\javaw.exe (Nov 13, 2010 4:44:55 PM)

[EL Config]: The access type for the persistent class [class org.comenzi.model.Client] is set to [FIELD].
[EL Config]: The access type for the persistent class [class org.comenzi.model.ArticolComanda] is set to [FIELD].
[EL Config]: The target entity (reference) class for the many to one mapping element [field comanda] is being default
[EL Config]: The target entity (reference) class for the many to one mapping element [field produs] is being default
[EL Config]: The access type for the persistent class [class org.comenzi.model.Comanda] is set to [FIELD].
[EL Config]: The target entity (reference) class for the many to one mapping element [field client] is being default
[EL Config]: The target entity (reference) class for the one to many mapping element [field articole] is being default
[EL Config]: The access type for the persistent class [class org.comenzi.model.Produs] is set to [FIELD].
[EL Config]: The alias name for the entity class [class org.comenzi.model.Client] is being defaulted to: Client.
[EL Config]: The table name for entity [class org.comenzi.model.Client] is being defaulted to: CLIENT.
[EL Config]: The column name for element [field id] is being defaulted to: ID.
[EL Config]: The column name for element [field nume] is being defaulted to: NUME.
[EL Config]: The alias name for the entity class [class org.comenzi.model.ArticolComanda] is being defaulted to: Arti
[EL Config]: The table name for entity [class org.comenzi.model.ArticolComanda] is being defaulted to: ARTICOLCOMANDA
[EL Config]: The column name for element [field id] is being defaulted to: ID.
[EL Config]: The column name for element [field cantitate] is being defaulted to: CANTITATE.
[EL Config]: The alias name for the entity class [class org.comenzi.model.Comanda] is being defaulted to: Comanda.
[EL Config]: The table name for entity [class org.comenzi.model.Comanda] is being defaulted to: COMANDA.
[EL Config]: The column name for element [field id] is being defaulted to: ID.
[EL Config]: The column name for element [field dataComanda] is being defaulted to: DATACOMANDA.
[EL Config]: The alias name for the entity class [class org.comenzi.model.Produs] is being defaulted to: Produs.
[EL Config]: The table name for entity [class org.comenzi.model.Produs] is being defaulted to: PRODUS.
[EL Config]: The column name for element [field cod] is being defaulted to: COD.
[EL Config]: The column name for element [field pretUnitar] is being defaulted to: PRETUNITAR.
[EL Config]: The column name for element [field denumire] is being defaulted to: DENUMIRE.
[EL Config]: The column name for element [field um] is being defaulted to: UM.
[EL Config]: The primary key column name for the mapping element [field produs] is being defaulted to: COD.
[EL Config]: The foreign key column name for the mapping element [field produs] is being defaulted to: PRODUS_COD.
[EL Config]: The primary key column name for the mapping element [field comanda] is being defaulted to: ID.
[EL Config]: The foreign key column name for the mapping element [field comanda] is being defaulted to: COMANDA_ID.
[EL Config]: The primary key column name for the mapping element [field client] is being defaulted to: ID.
[EL Config]: The foreign key column name for the mapping element [field client] is being defaulted to: CLIENT_ID.
[EL Info]: EclipseLink, version: Eclipse Persistence Services - 2.1.0.v20100614-r7608
[EL Fine]: Detected Vendor platform: org.eclipse.persistence.platform.database.JavaDBPlatform
[EL Config]: Connection(15352895)--connecting(DatabaseLogin(
platform=>JavaDBPlatform
```



# Verificare tabele generate în baza de date

The screenshot displays an IDE interface with three main panels:

- Project Explorer:** Shows the project structure. Under "JPA Content", there is a "persistence.xml" file and several entities: "ProduceJPA", "ArticolComanda", "Client", "Comanda", and "Produce".
- Data Source Explorer:** Shows the database connections. Under "ProduseDB (Apache Derby v. 10.5.1.1 - (764942))", there is a "ProduseDB" connection. Under "Schemas", there is a "PRODUSE" schema. Under "Tables", the "CLIENT" table is highlighted with a red oval. A context menu is open over the "CLIENT" table, showing options: "Data", "Edit", "Load...", "Extract...", and "Sample Contents".
- ArticolComanda.java:** Shows the table structure for "ArticolComanda". It has two columns: "ID [INTEGER]" and "NUME [VARCHAR(255)]". There is a "<new row>" button.

## 2.4 Testare structuri persistente

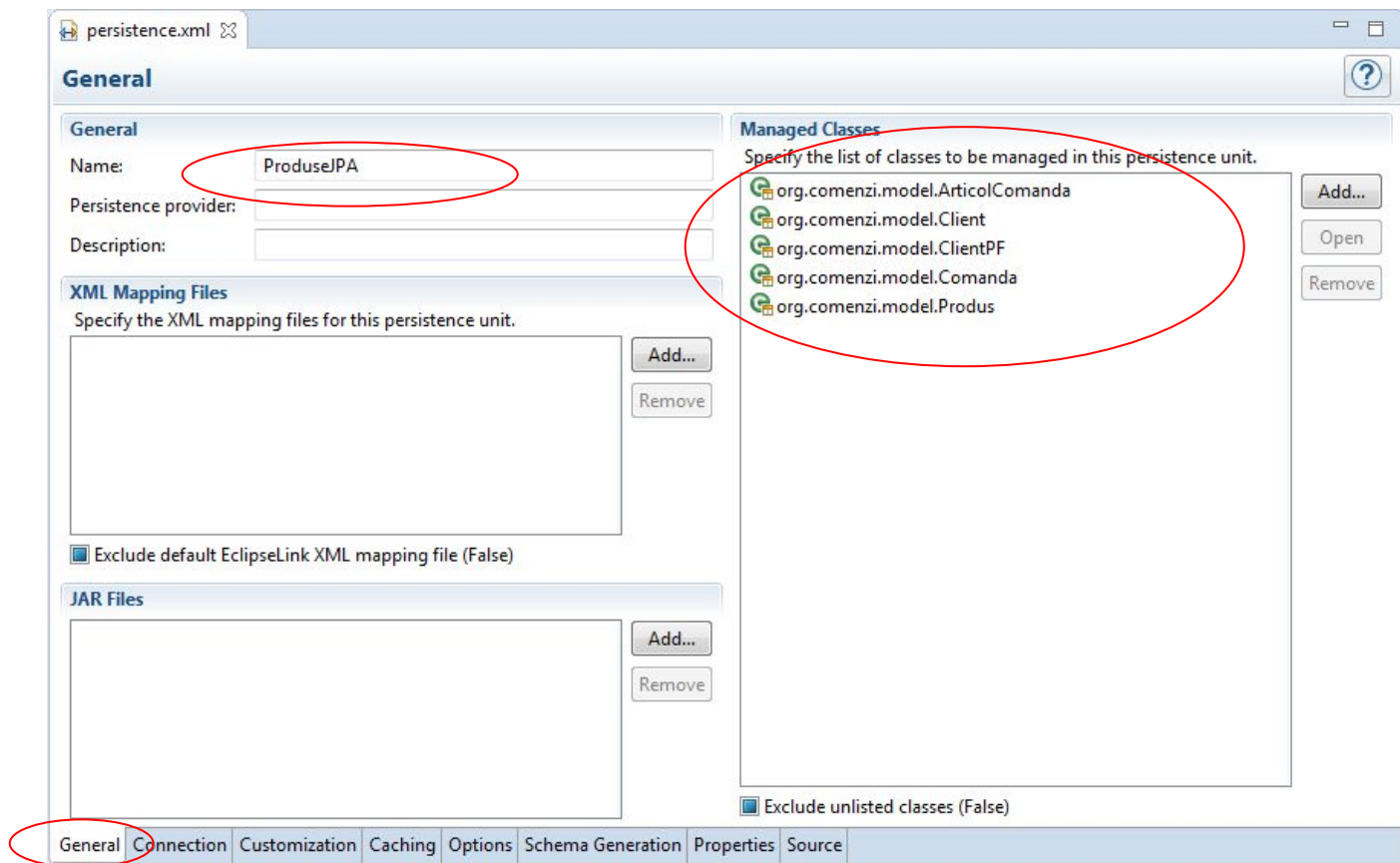
---

- 2.4.1 Verificare persistence.xml
  - Secțiunea *General*: Nume (persistence unit *name*): ProduseJPA
    - Secțiunea *Conexiune*:  
`jdbc:postgresql://serverhost:port/databasename`
  - Secțiunea *Schema Generation*: Mod sincronizare meta-date entități – schemă bază de date,
    - *Database action\**:
      - Resetare pe *None* (după ce tabelele au fost generate);
- 2.4.2 Crearea testului JPA – Etape:
  - Creare context persistență
  - Inițializare configurație - *EntityManagerFactory*
  - Inițializare sesiune de lucru cu baza de date - *EntityManager*
  - Creare/modificare/restaurare/ștergere obiecte entități
    - `entityManager.persist()/merge()/find()/remove()`
  - Invocare context persistență: realizare tranzacție cu baza de date
    - `entityManager.getTransaction().begin()-commit()`

*\*Drop and Create doar dacă se dorește regenerarea tabelelor).*

# 2.4.1 Verificare persistence XML

- Verificare nume persistence unit



# 2.4.1 Verificare persistence XML

- Verificare conexiune

The screenshot shows the 'Connection' tab of the 'Persistence Unit Connection' configuration window. The window title is 'persistence.xml'. The 'Transaction type' is set to 'Resource Local'. The 'Batch writing' is set to 'Default (None)'. The 'Statement caching' is set to 'Default (50)'. The 'Native SQL' checkbox is unchecked. The 'Database' section has 'JTA data source' and 'Non-JTA data source' fields. The 'EclipseLink connection pool' section has a 'Populate from connection...' link. The 'Driver' is set to 'org.postgresql.Driver'. The 'URL' is set to 'jdbc:postgresql://85.122.22.12:5432/postgres'. The 'User' is set to 'produse'. The 'Password' is masked with dots. The 'Bind parameters' checkbox is checked. The 'Read Connection' section has 'Shared' unchecked, 'Minimum' set to 'Default (2)', and 'Maximum' set to 'Default (2)'. The 'Write Connection' section has 'Minimum' set to 'Default (5)' and 'Maximum' set to 'Default (10)'. The 'General' tab is selected in the bottom tab bar.

**Connection**

**Persistence Unit Connection**  
Configure the data source or JDBC connection properties.

Transaction type: **Resource Local**

Batch writing: **Default (None)**

☒ Statement caching: **Default (50)**

☒ Native SQL (False)

Database

JTA data source:

Non-JTA data source:

EclipseLink connection pool

[Populate from connection...](#)

Driver: **org.postgresql.Driver** **Browse...**

URL: **jdbc:postgresql://85.122.22.12:5432/postgres**

User: **produse**

Password: **.....**

☒ Bind parameters (True)

**Read Connection**

☒ Shared (False)

Minimum: **Default (2)**

Maximum: **Default (2)**

**Write Connection**

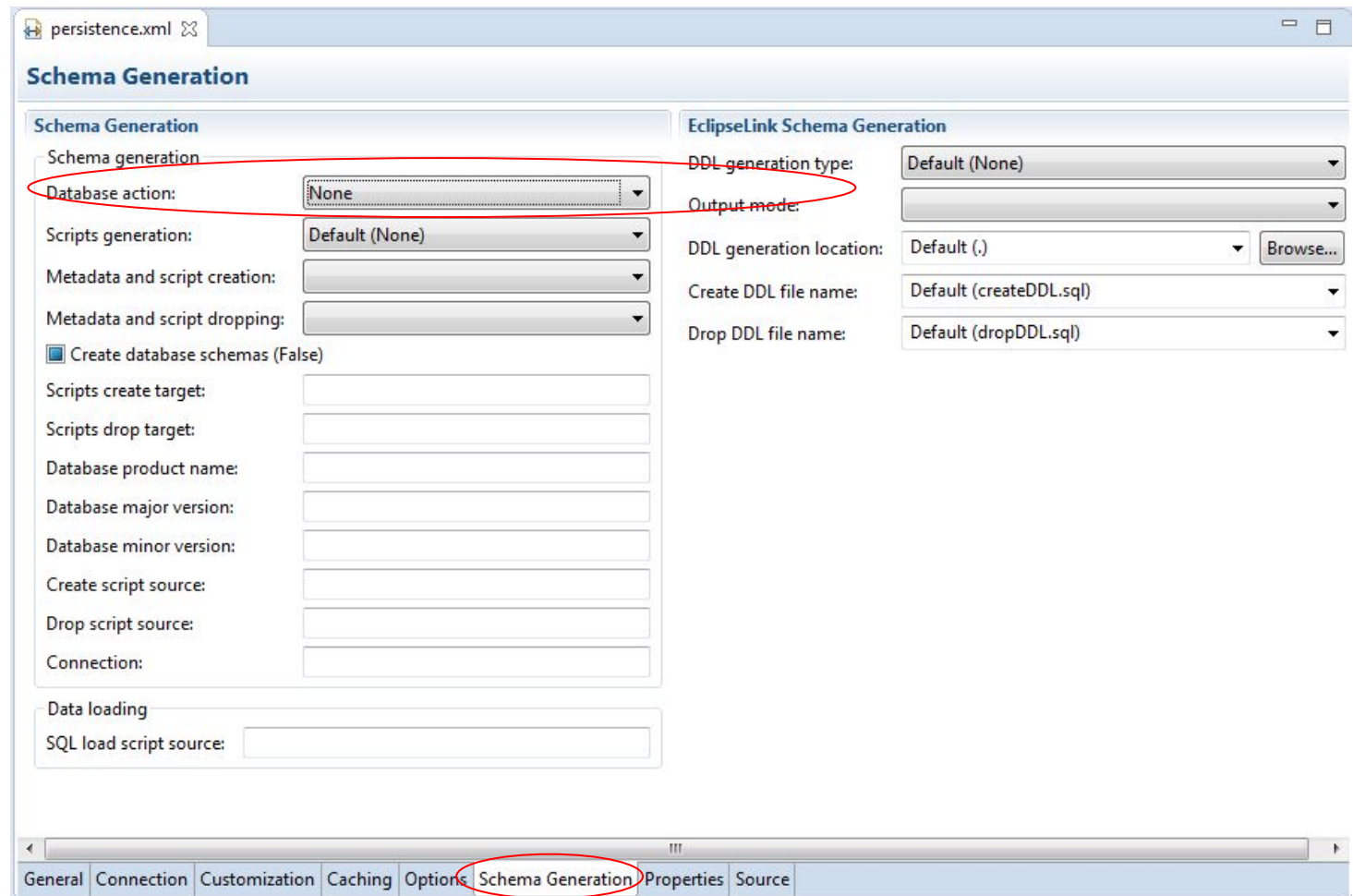
Minimum: **Default (5)**

Maximum: **Default (10)**

General **Connection** Customization Caching Options Schema Generation Properties Source

# 2.4.1 Verificare persistence XML

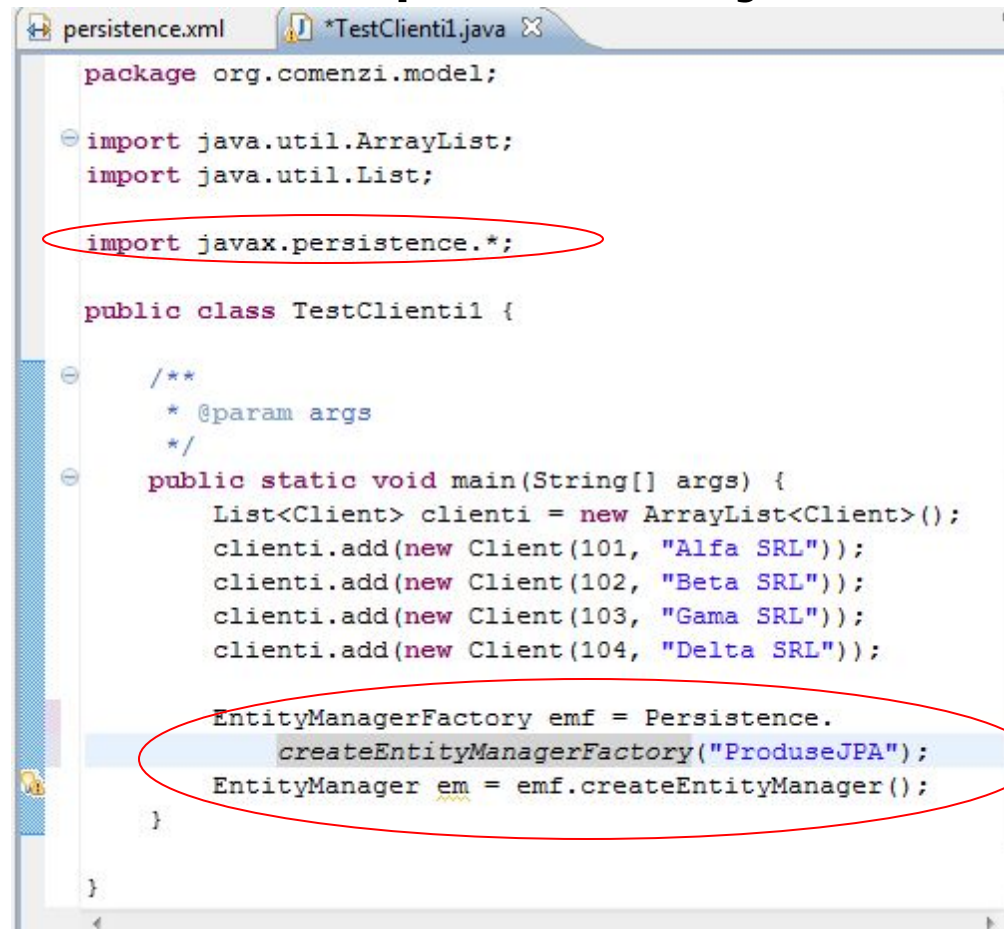
- Verificare mod sincronizare meta-date/ generare schemă bază de date





## 2.4.2 Creare test

- Inițializare context persistență



```
package org.comenzi.model;

import java.util.ArrayList;
import java.util.List;
import javax.persistence.*;

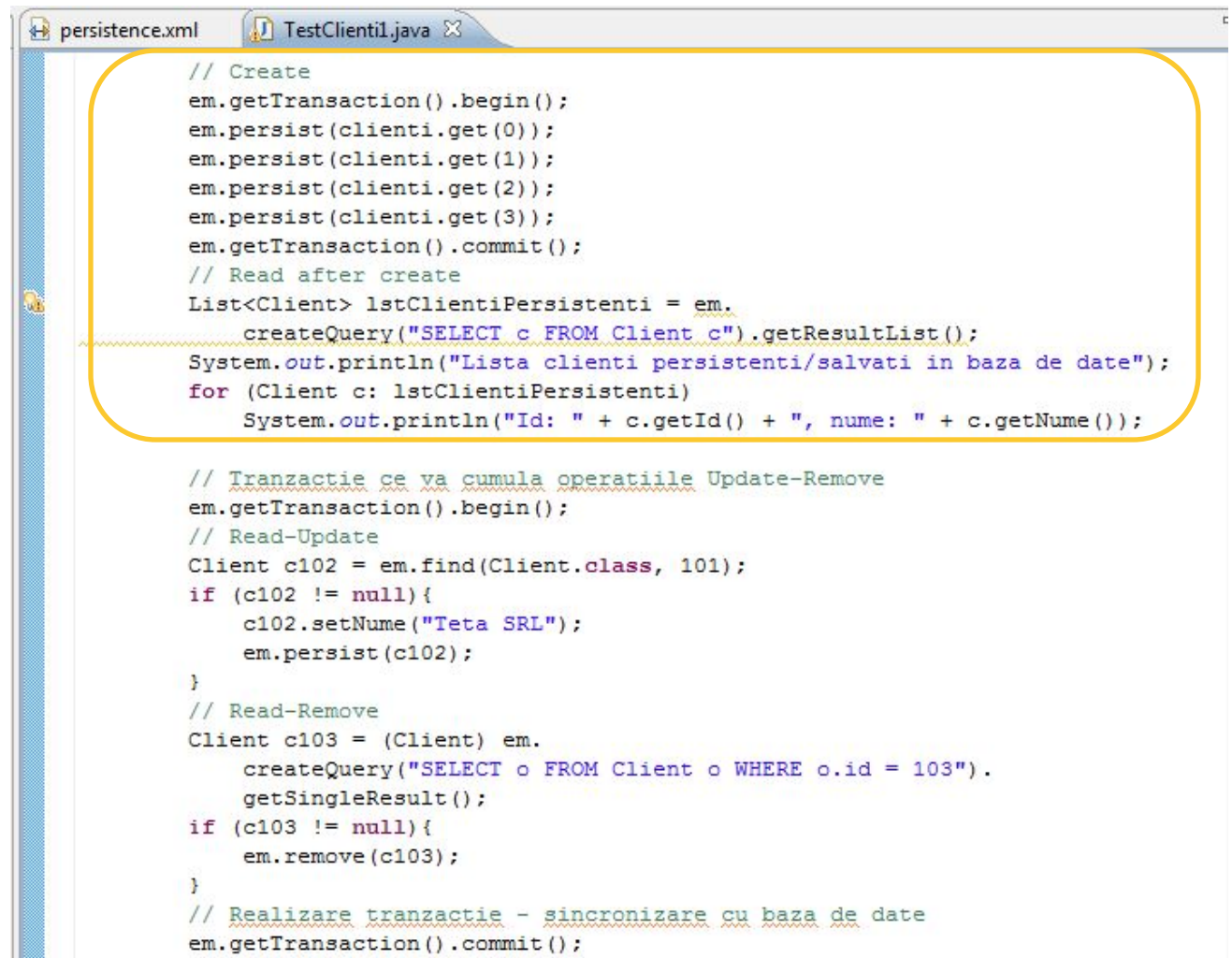
public class TestClienti1 {

    /**
     * @param args
     */
    public static void main(String[] args) {
        List<Client> clienti = new ArrayList<Client>();
        clienti.add(new Client(101, "Alfa SRL"));
        clienti.add(new Client(102, "Beta SRL"));
        clienti.add(new Client(103, "Gama SRL"));
        clienti.add(new Client(104, "Delta SRL"));

        EntityManagerFactory emf = Persistence.
            createEntityManagerFactory("ProduseJPA");
        EntityManager em = emf.createEntityManager();
    }
}
```

## 2.4.2 Creare test

- Create-Read



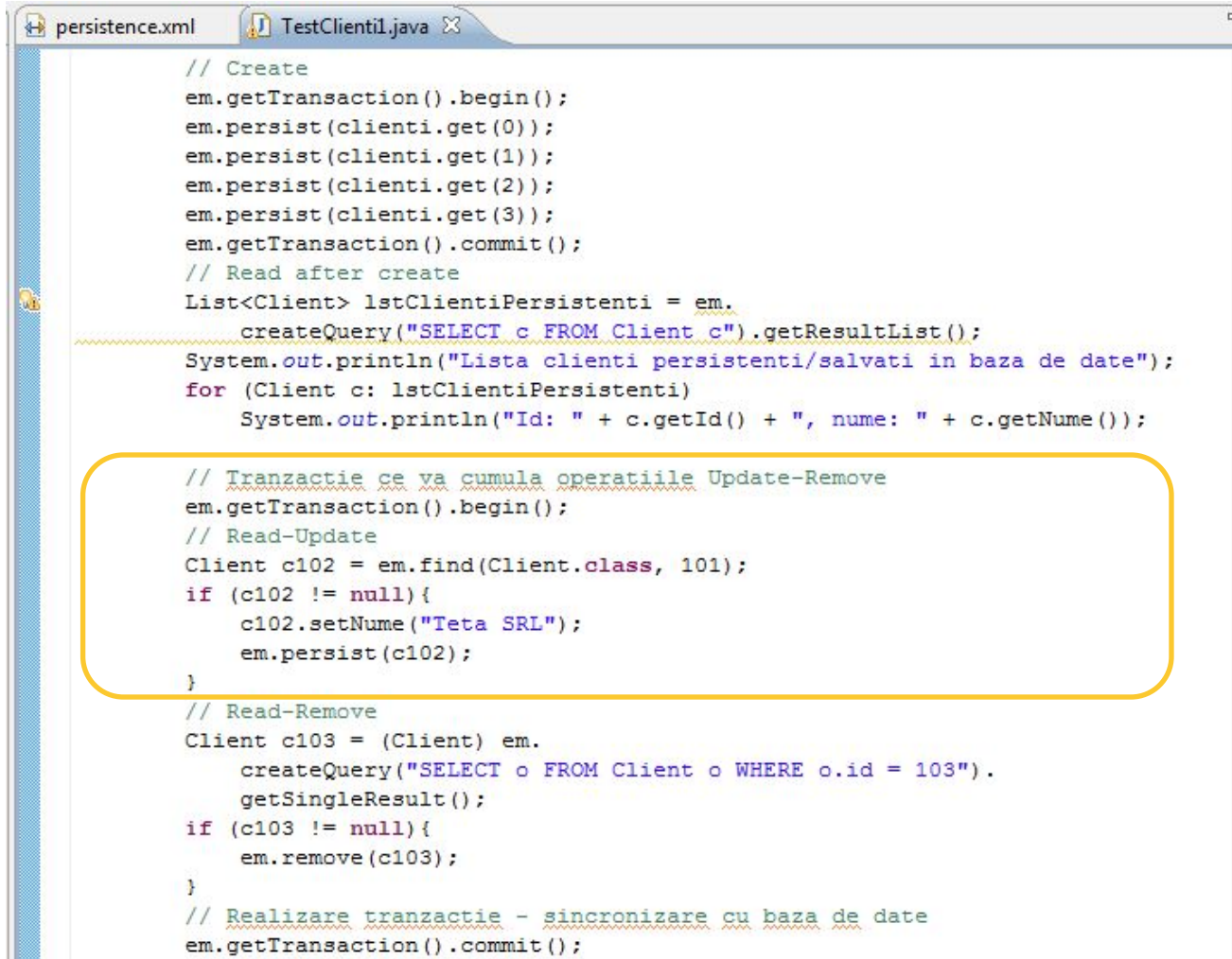
```
persistence.xml TestClientil.java X

// Create
em.getTransaction().begin();
em.persist(clienti.get(0));
em.persist(clienti.get(1));
em.persist(clienti.get(2));
em.persist(clienti.get(3));
em.getTransaction().commit();
// Read after create
List<Client> lstClientiPersistenti = em.
    createQuery("SELECT c FROM Client c").getResultList();
System.out.println("Lista clienti persistenti/salvati in baza de date");
for (Client c: lstClientiPersistenti)
    System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());

// Tranzactie ce va cumula operatiile Update-Remove
em.getTransaction().begin();
// Read-Update
Client c102 = em.find(Client.class, 101);
if (c102 != null){
    c102.setNume("Teta SRL");
    em.persist(c102);
}
// Read-Remove
Client c103 = (Client) em.
    createQuery("SELECT o FROM Client o WHERE o.id = 103").
    getSingleResult();
if (c103 != null){
    em.remove(c103);
}
// Realizare tranzactie - sincronizare cu baza de date
em.getTransaction().commit();
```

## 2.4.2 Creare test

- Read-Update

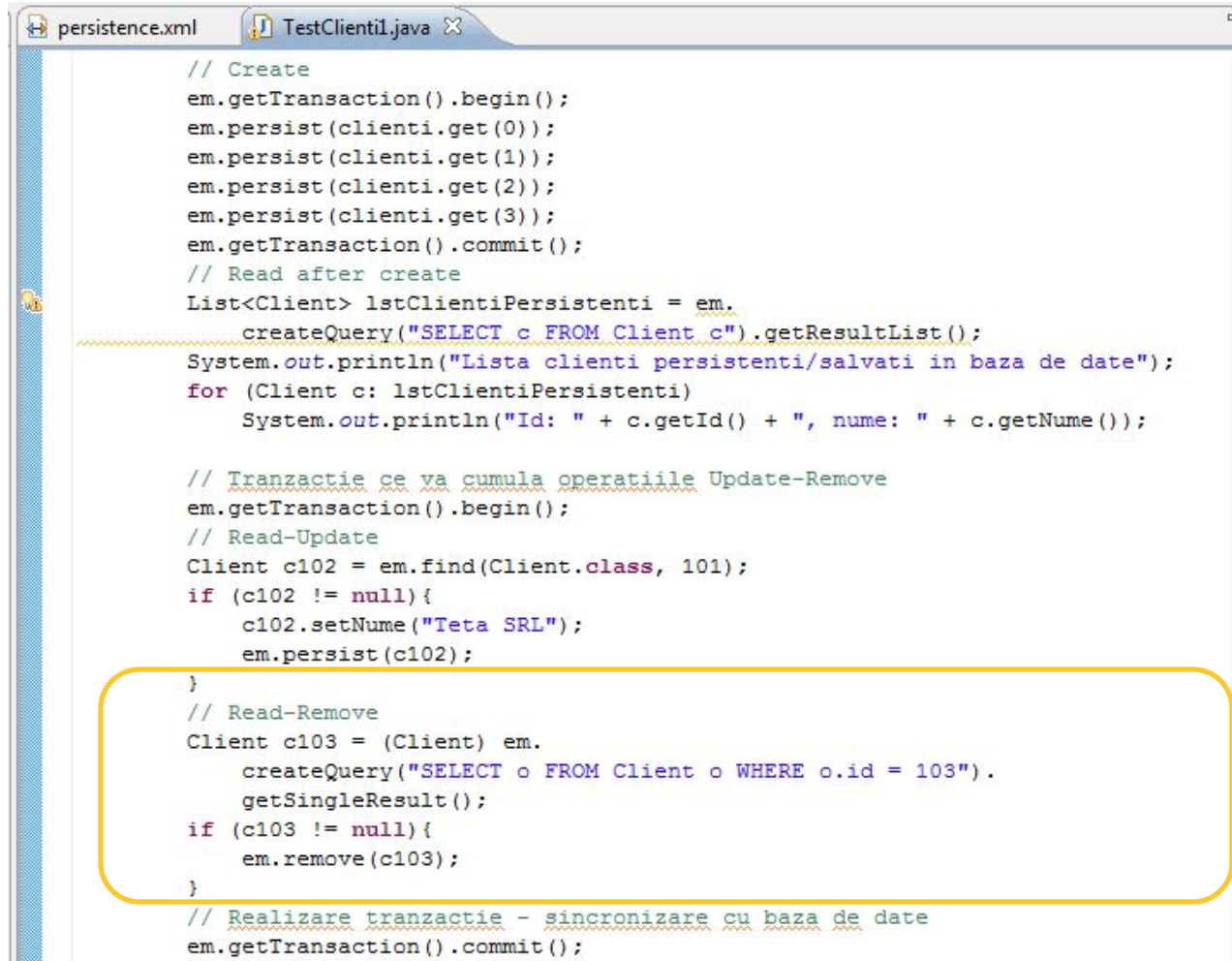


```
// Create
em.getTransaction().begin();
em.persist(clienti.get(0));
em.persist(clienti.get(1));
em.persist(clienti.get(2));
em.persist(clienti.get(3));
em.getTransaction().commit();
// Read after create
List<Client> lstClientiPersistenti = em.
    createQuery("SELECT c FROM Client c").getResultList();
System.out.println("Lista clienti persistenti/salvati in baza de date");
for (Client c: lstClientiPersistenti)
    System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());

// Tranzactie ce va cumula operatiile Update-Remove
em.getTransaction().begin();
// Read-Update
Client c102 = em.find(Client.class, 101);
if (c102 != null) {
    c102.setNume("Teta SRL");
    em.persist(c102);
}
// Read-Remove
Client c103 = (Client) em.
    createQuery("SELECT o FROM Client o WHERE o.id = 103").
    getSingleResult();
if (c103 != null) {
    em.remove(c103);
}
// Realizare tranzactie - sincronizare cu baza de date
em.getTransaction().commit();
```

## 2.4.2 Creare test

- Read-Remove



```
// Create
em.getTransaction().begin();
em.persist(clienti.get(0));
em.persist(clienti.get(1));
em.persist(clienti.get(2));
em.persist(clienti.get(3));
em.getTransaction().commit();
// Read after create
List<Client> lstClientiPersistenti = em.
    createQuery("SELECT c FROM Client c").getResultList();
System.out.println("Lista clienti persistenti/salvati in baza de date");
for (Client c: lstClientiPersistenti)
    System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());

// Tranzactie ce va cumula operatiile Update-Remove
em.getTransaction().begin();
// Read-Update
Client c102 = em.find(Client.class, 101);
if (c102 != null){
    c102.setNume("Teta SRL");
    em.persist(c102);
}
// Read-Remove
Client c103 = (Client) em.
    createQuery("SELECT o FROM Client o WHERE o.id = 103").
    getSingleResult();
if (c103 != null){
    em.remove(c103);
}
// Realizare tranzactie - sincronizare cu baza de date
em.getTransaction().commit();
```



## Rezultat test

```
persistence.xml  TestClient1.java

List<Client> lstClientiPersistenti = em.
    createQuery("SELECT c FROM Client c").getResultList();
System.out.println("Lista clienti persistenti/salvati in baza de date");
for (Client c: lstClientiPersistenti)
    System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());

// Tranzactie ce va cumula operatiile Update-Remove
em.getTransaction().begin();
// Read-Update
Client c102 = em.find(Client.class, 101);
if (c102 != null){
    c102.setNume("Teta SRL");
    em.persist(c102);
}
// Read-Remove
Client c103 = (Client) em.
    createQuery("SELECT o FROM Client o WHERE o.id = 103").
    getSingleResult();
if (c103 != null){
    em.remove(c103);
}
// Realizare tranzactie - sincronizare cu baza de date
em.getTransaction().commit();

lstClientiPersistenti = em.
    createQuery("SELECT c FROM Client c").getResultList();
System.out.println("Lista finala clienti persistenti/salvati in baza de date");
for (Client c: lstClientiPersistenti)
    System.out.println("Id: " + c.getId() + ", nume: " + c.getNume());

Problems  JPA Details  Console  Progress
<terminated> TestClient1 [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Nov 23, 2010 10:27:22 AM)

Lista clienti persistenti/salvati in baza de date
Id: 104, nume: Delta SRL
Id: 102, nume: Beta SRL
Id: 101, nume: Alfa SRL
Id: 103, nume: Gama SRL

Lista finala clienti persistenti/salvati in baza de date
Id: 104, nume: Delta SRL
Id: 102, nume: Teta SRL
Id: 101, nume: Teta SRL
```