# BAB IV Aplikasi CRUD Menggunakan JPA, EJB, dan MySQL

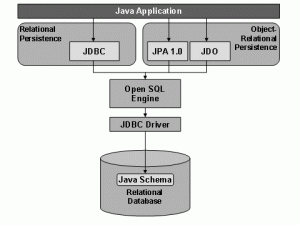
## Tujuan

1. Praktikan dapat menggunakan operasi CRUD dengan database MySQL
2. Praktikan dapat memahami fungsi dari EJB dan JPA dalam pembuatan aplikasi berbasis komponen
3. Praktikan dapat mengimplementasikan EJB
4. Praktikan dapat mengimplementasikan JPA

## Dasar teori

### Java Persistance API

Java Persistence API merupakan tool untuk mengolah ataupun pengatur data relational dalam platform Java Standard Edition dan Java Enterprise Edition. JPA sendiri merupakan alat dalam pembuatan aplikasi berbentuk framework dalam pemrograman java dengan pendekatan Object Relational Maping (ORM). ORM sendiri merupakan sebuah konsep yang berdiri sendiri, tidak terkait dengan Java. Namun hubungan ORM dengan JPA sangat dekat karena JPA merupakan standart ORM dalam Java, dan harus diikuti oleh pengguna ORM di Java agar ada standart yang sama antara ORM di Java dengan yang diluar Java.

[](https://dartoblog.files.wordpress.com/2012/08/jpa.gif)

Gambar 4. 1 Bagan JPA

Dengan menggunakan JPA, memungkinkan manipulasi data tanpa menggunakan query, namun bukan berarti tanpa menggunakan query sama sekali, tetap ada penggunaan query disana. Cara JPA ini dinilai lebih baik dari teknik manipulasi data dengan jdbc. Jika kita menggunakan JPA, maka cara kita terhubung ke database sama semua, baik pakai MySQl, SQL Server ataupun PostgreSQL.

API JPA terdapat dalam package javax.persistence. Di dalamnya mengandung Query khusus yang disebut (JPQL)Java Persistence Query Language. Beberapa Library yang mengimplementasikan JPA antara lain adalah Hibernate dan EclipseLink.

Kelebihan JPA yang cukup bermanfaat adalah tidak perlu membuat query untuk manipulasi data. Selain itu kita dapat dengan mudah mengelola transaksi dengan API.

Kita bisa menghindari pembuatan Data Access Object yang rumit dan komplek sekali jika menggunakan JPA ini. Dan yang cukup bagus, kita juga dapat mengelola Plain Old Java Object disini.

*(Sumber: https://dartoblog.wordpress.com/2012/08/01/pengenalan-jpa-java-persistence-api/)*

### Session Bean

Session Bean digunakan untuk mereprsentasikan proses, kendali dan alur kerja. Contohnya, session bean digunakan untuk melakukan validasi credit card, mencariketersediaan jadwal penerbangan, dan menyimpan shopping cart. Session Bean bisa Stateless atau Stateful. SessionBean tidak disimpan di dalam database, dan tidak akan survive jika server mengalami crash. Session bean hanya diakses oleh satu client, dan transactional.

*(Sumber: https://id.wikipedia.org/wiki/Enterprise\_JavaBeans)*

### Entity Unit

Entity Unit atau Entity Bean digunakan untuk merepresentasikan business objek. Contohnya, entity bean digunakan untuk merepresentasikan Product, Order, Student, Course, Employee, dan To Do. Entity Bean tersimpan di database, jika server mengalami crash dia akan survive. Terdapat dua strategi penyimpanan yaitu Bean Managed Persistence dan Container Managed Persistence. Entity Bean adalah di-share oleh banyak client dan transactional.

*(Sumber: https://id.wikipedia.org/wiki/Enterprise\_JavaBeans)*

### Java Servlet

Servlet adalah teknologi Java untuk aplikasi web berupa class yang digunakan untuk menerima request dan memberi respon melalui protokol http (html, xml, file dan sebagainya). Pada dasarnya Servlet merupakan file java class yang telah dikompilasi dan dijalankan oleh servlet container atau application server. Istilah application server digunakan apabila software server dapat menjalankan servlet, JSP serta teknologi J2EE utama seperti EJB (Enterprise Java Bean). Contoh Application Servlet adalah BEA Web Logic, IBM Websphere, Jboss, dsb. Servlet container biasanya juga merupakan JSP container, seperti Apache Tomcat, Macromedia Jrun, Resin.

*(Sumber: https://ptwfighter.wordpress.com/2011/09/28/servlet-jsp-dan-web-browser/)*

### MySQL

MySQL adalah sebuah perangkat lunak system manajemen basis data SQL (DBMS) yang multithread, dan multi-user. MySQL adalah implementasi dari system manajemen basisdata relasional (RDBMS). MySQL dibuah oleh TcX dan telah dipercaya mengelola system dengan 40 buah database berisi 10.000 tabel dan 500 di antaranya memiliki 7 juta baris.

Pada saat ini MySQL merupakan database server yang sangat terkenal di dunia, semua itu tak lain karena bahasa dasar yang digunakan untuk mengakses database yaitu SQL. SQL (Structured Query Language) pertama kali diterapkan pada sebuah proyek riset pada laboratorium riset San Jose, IBM yang bernama system R. Kemudian SQL juga dikembangan oleh Oracle, Informix dan Sybase. Dengan menggunakan SQL, proses pengaksesan database lebih user-friendly dibandingan dengan yang lain, misalnya dBase atau Clipper karena mereka masih menggunakan perintah-perintah pemrograman murni. SQL dapat digunakan secara berdiri sendiri maupun di lekatkan pada bahasa pemograman seperti C, dan Delphi.

*(Sumber: https://upyes.wordpress.com/2013/02/06/pengertian-dan-sejarah-mysql/)*

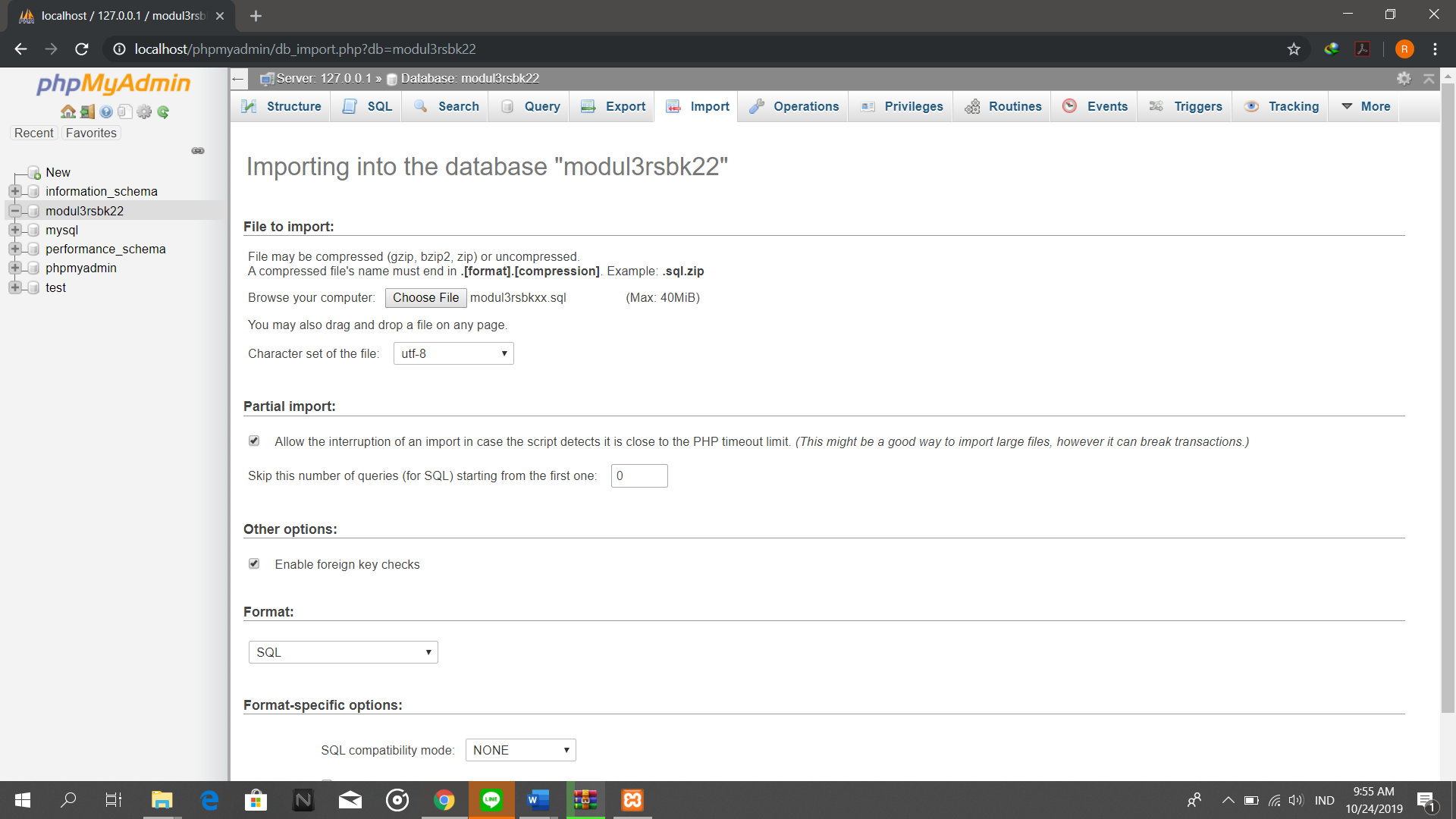
## Langkah Percobaan

1. Jalankan XAMPP, jika ada bentrok dengan salah satu PORT, matikan PORT tersebut lalu nyalakan kembali (biasanya bentrok sama vmware/oracle).
2. Buat database di MySQL. Beri nama “modul3rsbk22”.



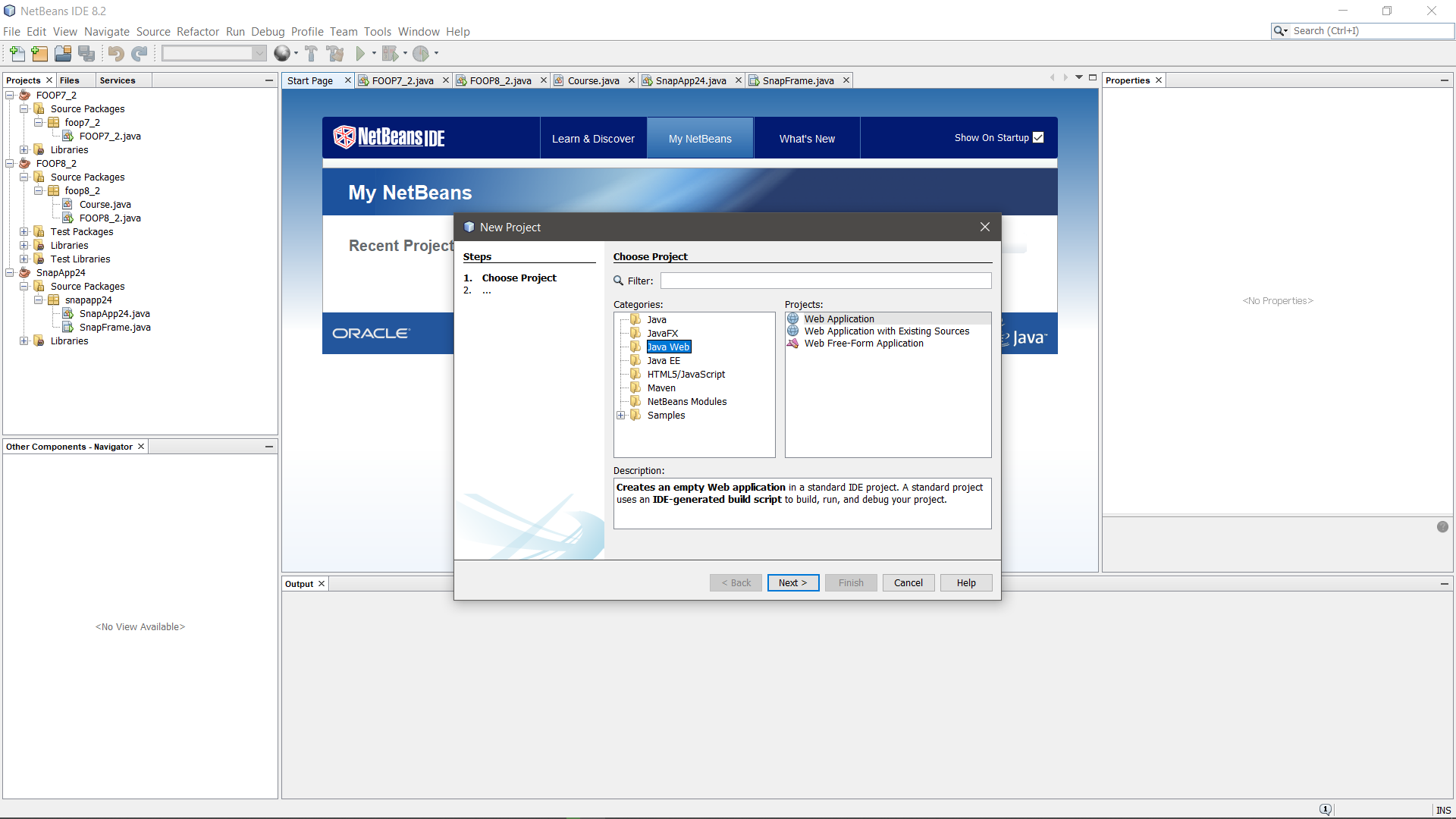
Gambar 4. 2 Membuat database baru

1. Import file SQL ke MySQL anda, pilih file .sql nya dan pilih go.



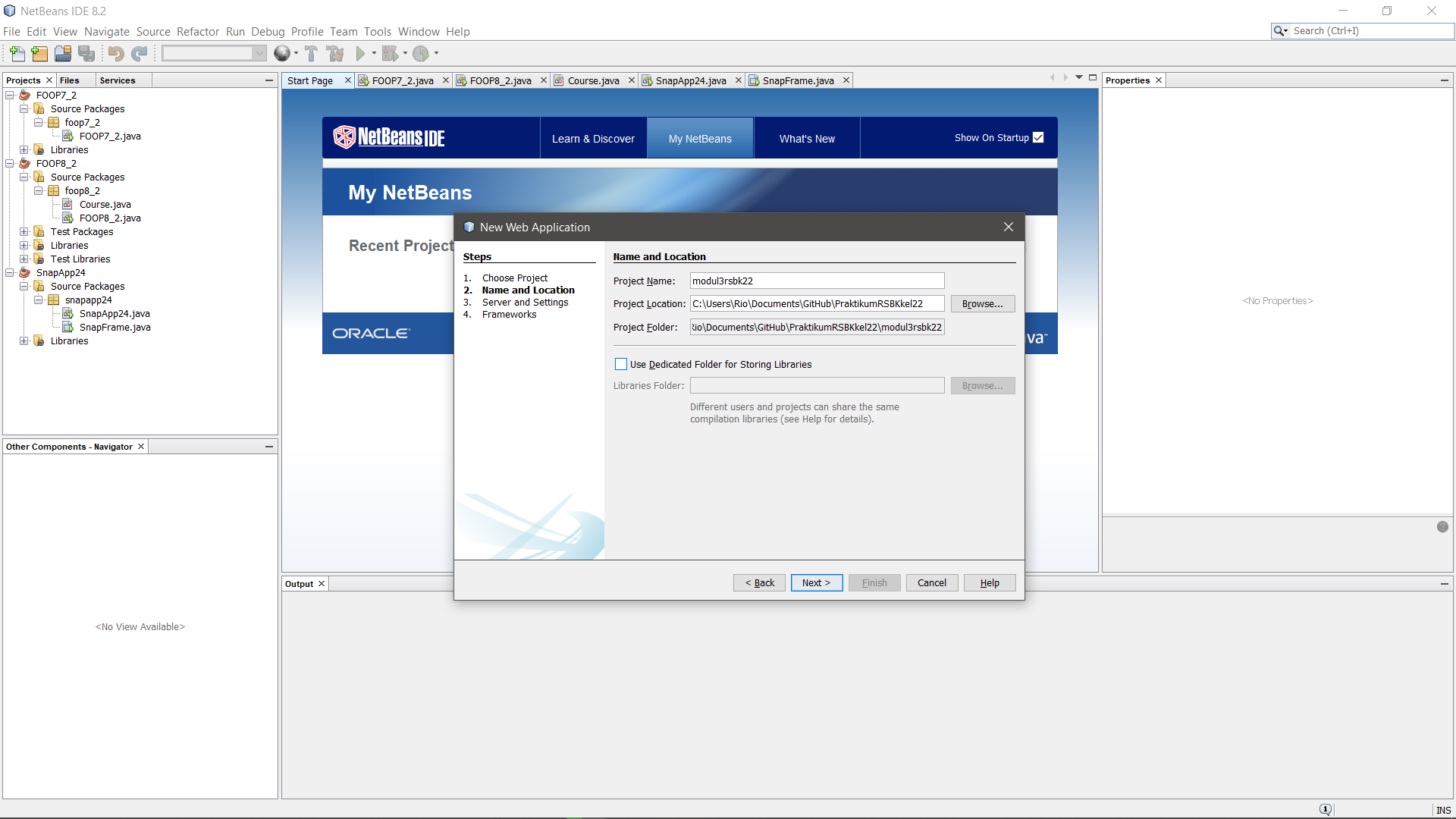
Gambar 4. 3 Mengimport file SQL ke MySQL

1. Buka Netbeans lalu buat project baru pada Netbeans, pilih Java Web → Web Application

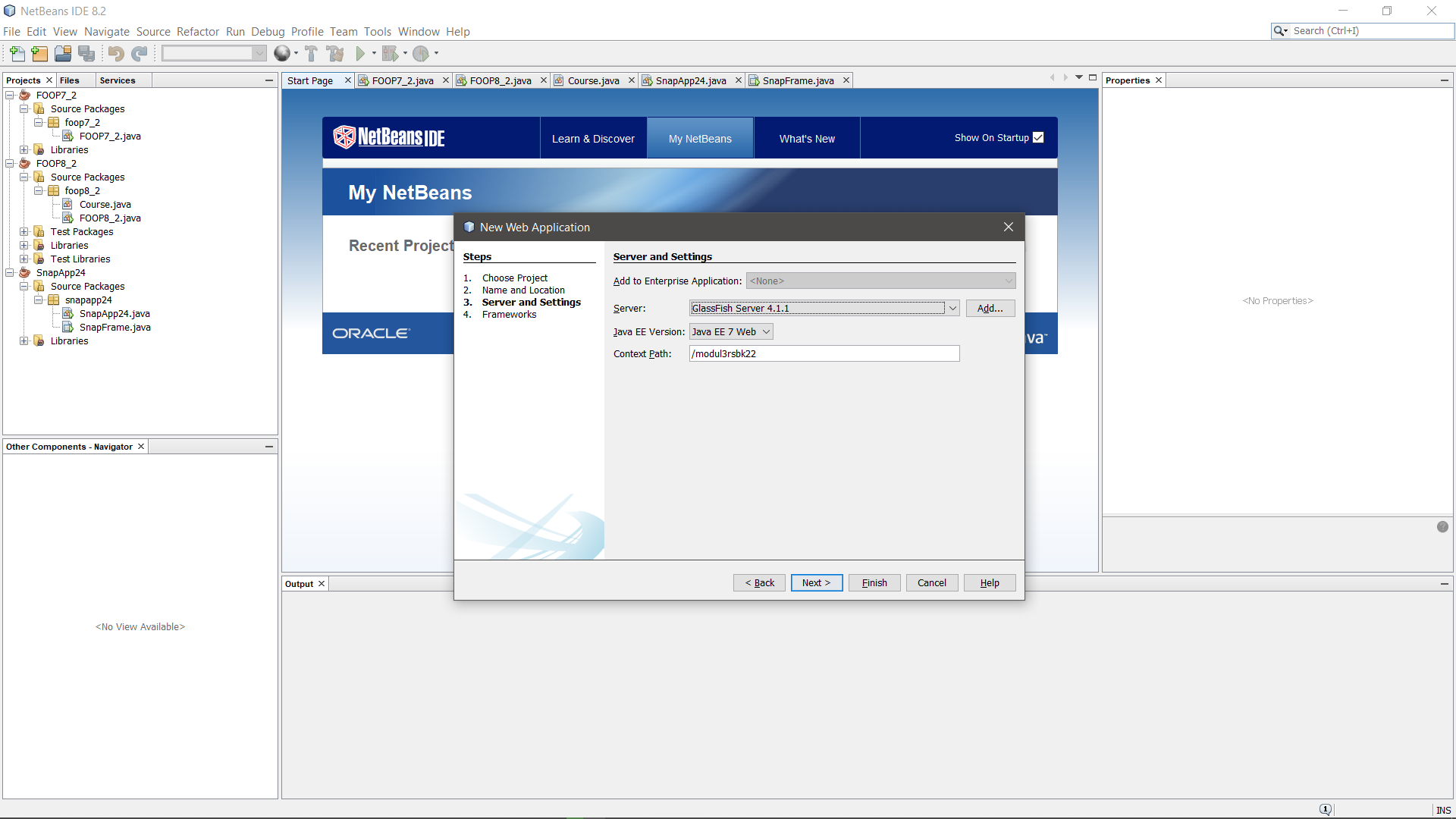


Gambar 4. 4 Membuat project baru berupa Java Web Application

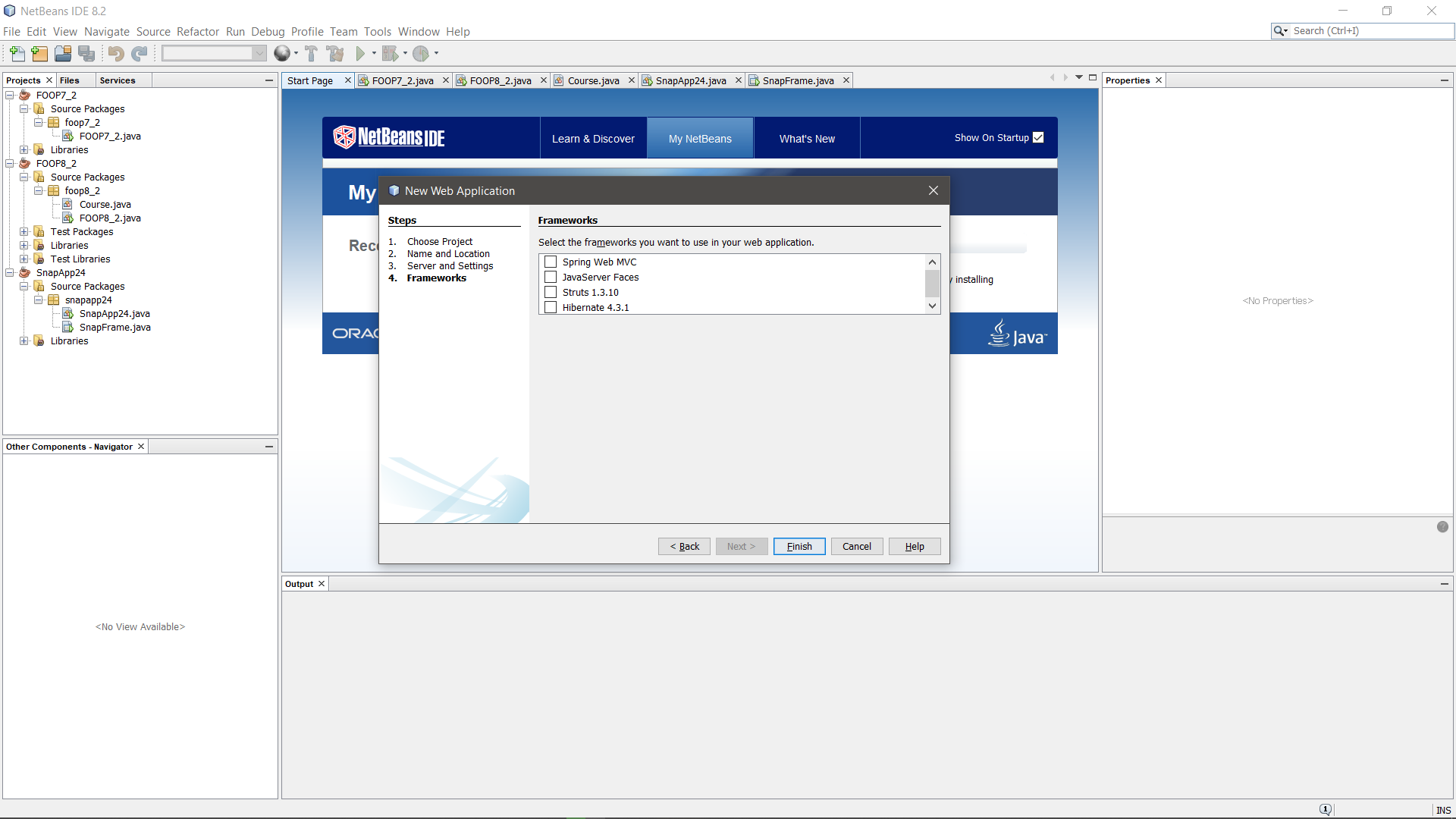
1. Beri nama project “**modul3rsbk22**” dan pilih Glassfish Server.



Gambar 4. 5 Pengaturan dalam membuat web application (1)

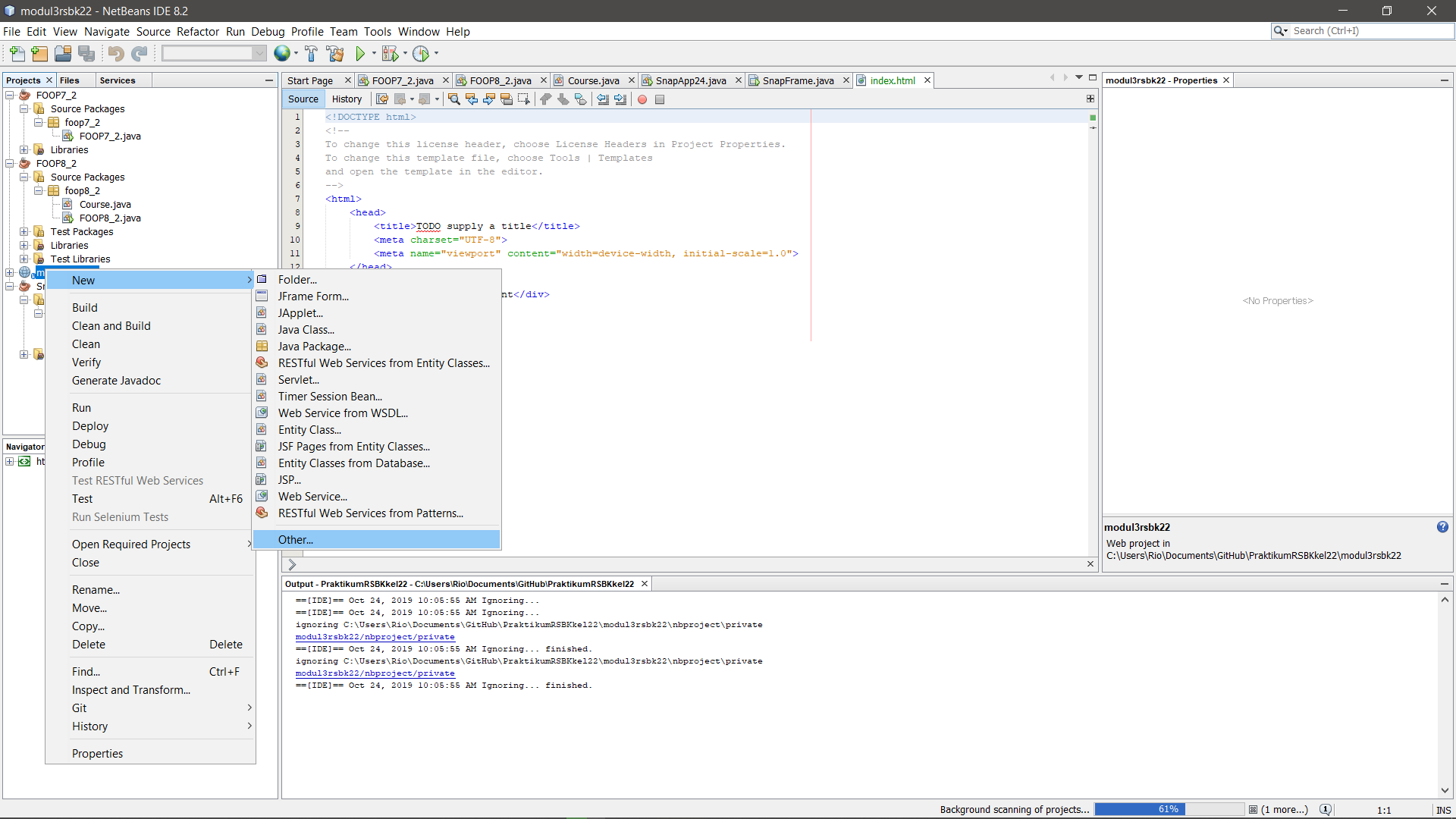


Gambar 4. 6 Pengaturan dalam membuat web application (2)

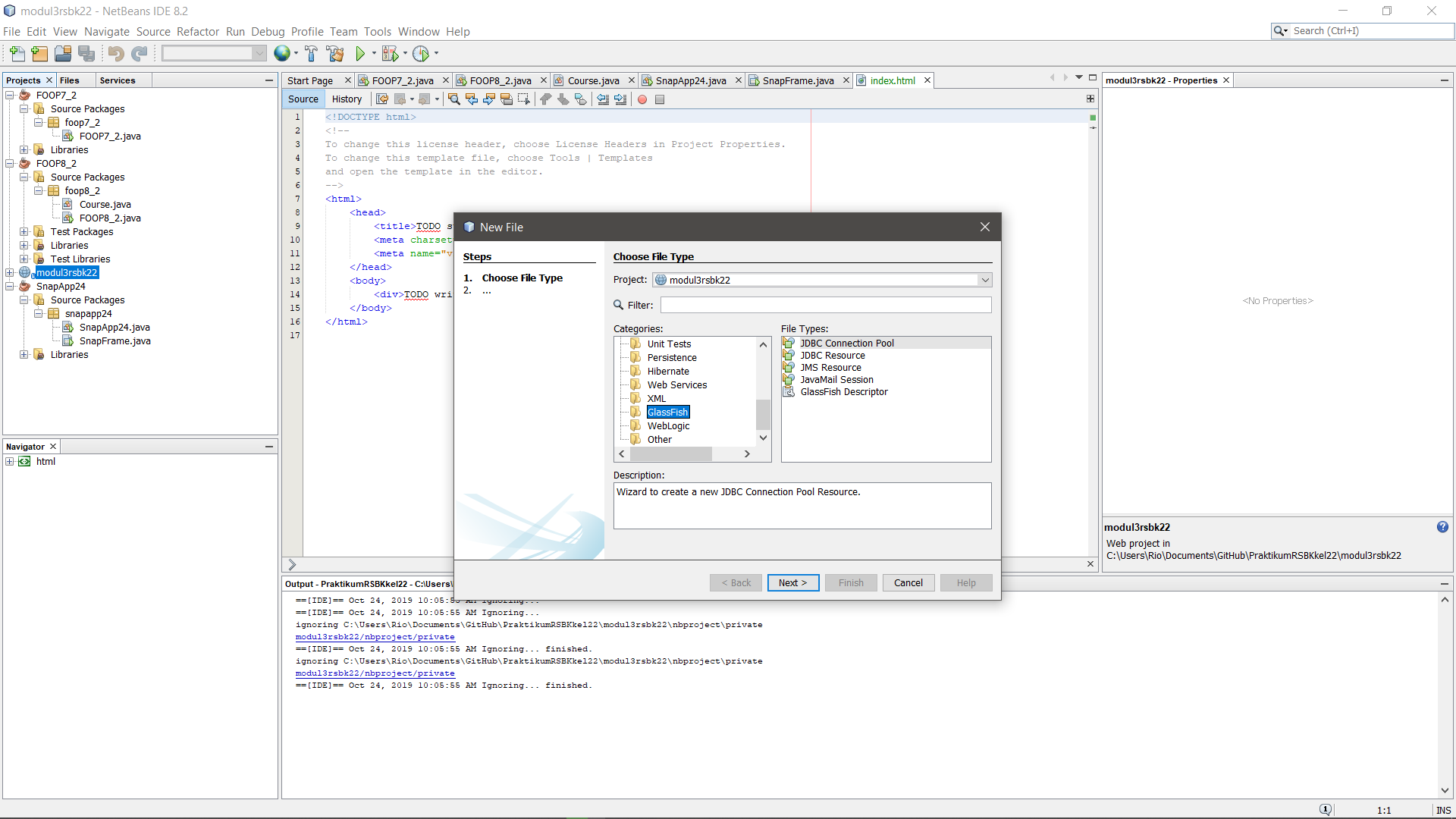


Gambar 4. 7 Pengaturan dalam membuat web application (3)

1. Buat JDBC Connection Pool, klik kanan pada project, new file, other. Lalu masuk ke kategori GlassFish, pilih tipe file JDBC Connection Pool.

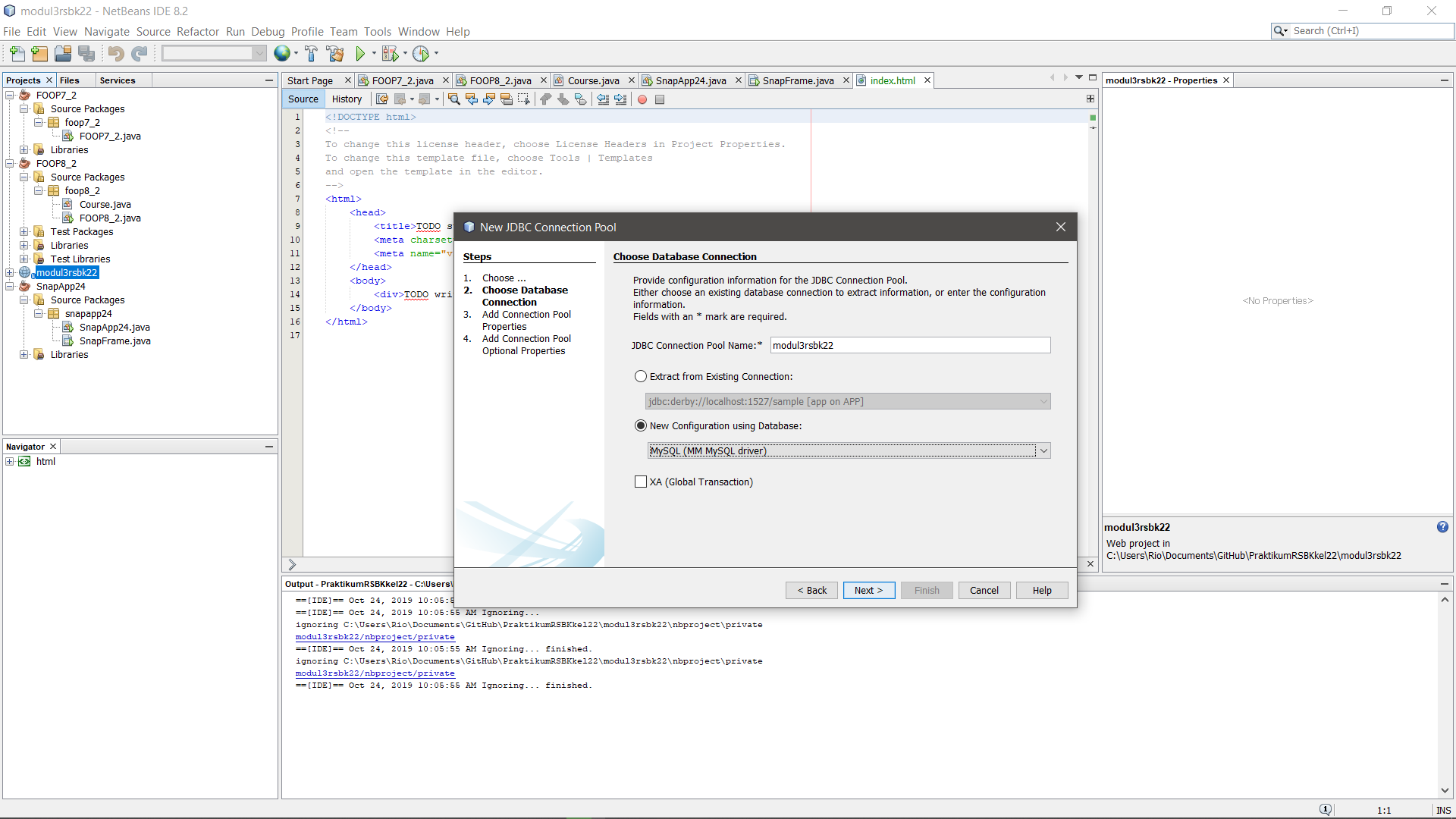


Gambar 4. 8 Memilih JDBC Connection Pool (1)



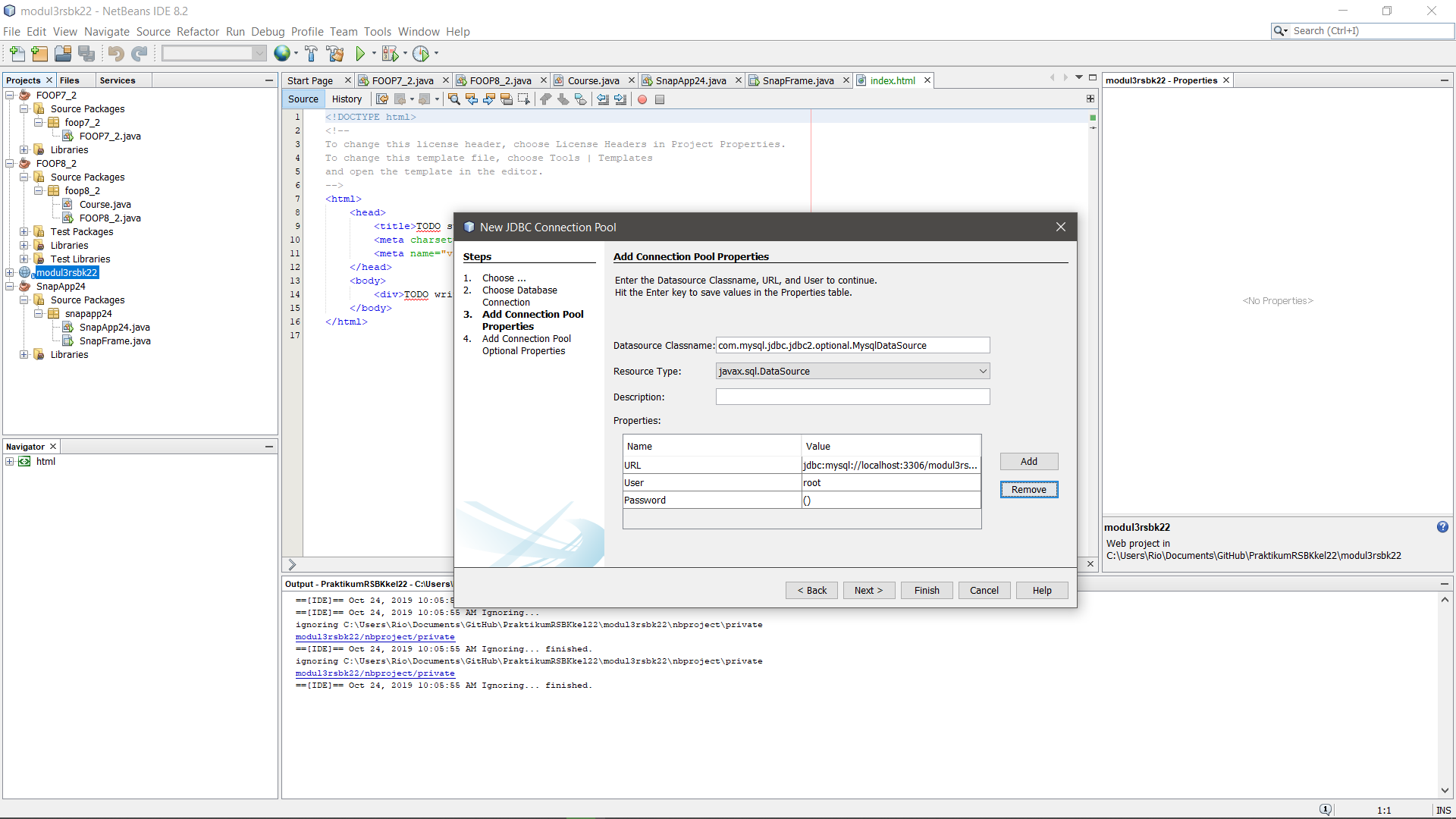
Gambar 4. 9 Memilih JDBC Connection Pool (2)

1. Beri nama connection pool modul3rsbk22 dan pilih ‘New Configuration using Database’ → ‘MySQL (MM MySQL driver).



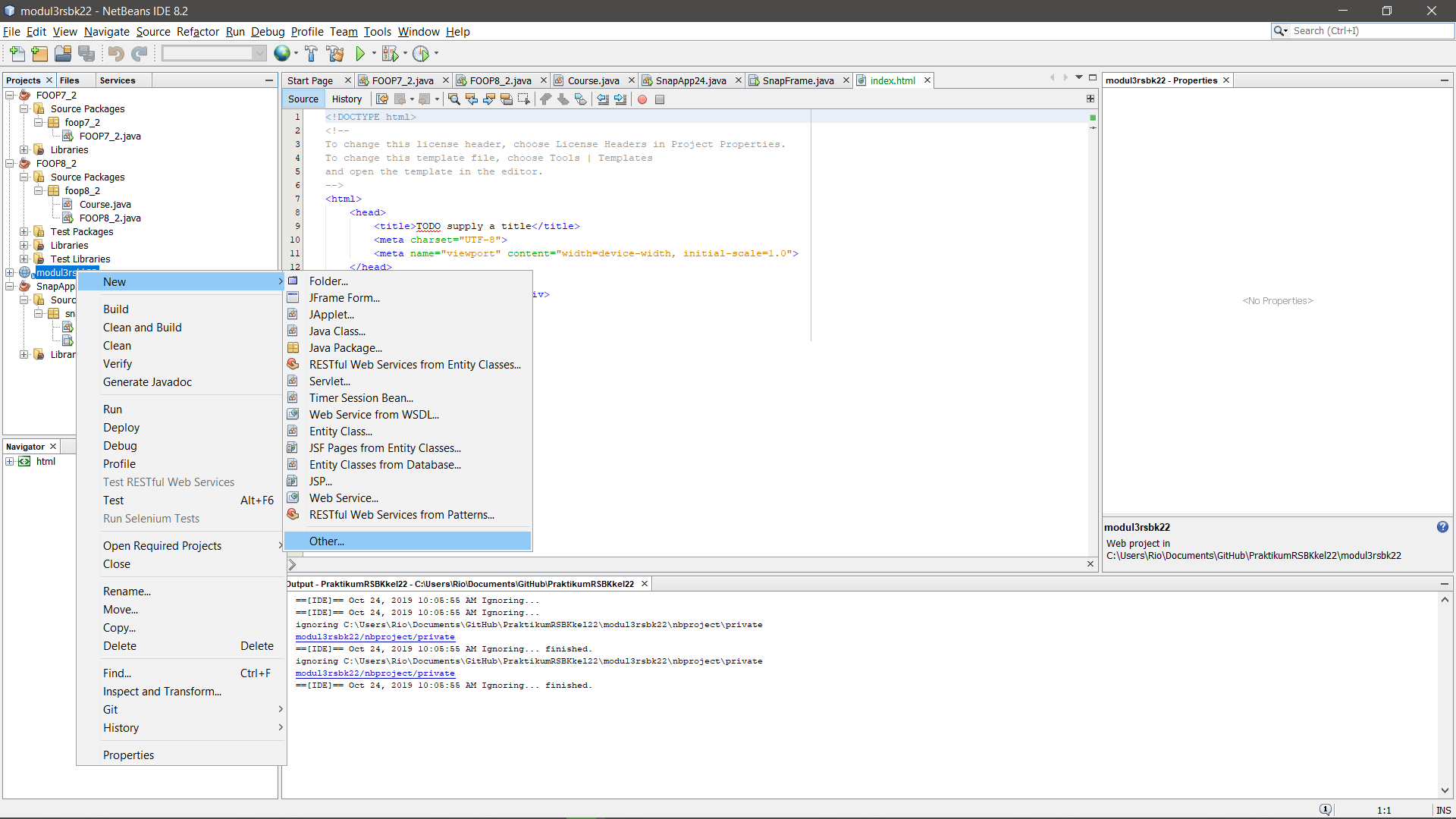
Gambar 4. 10 Membuat JDBC Connection Pool (1)

1. Masukkan URL, User, dan Password MySQL yang sudah dibuat. URL ke ***localhost*** dengan port default MySQL **3306**, lalu nama database yang akan kita gunakan yaitu **modul3rsbk22**. Untuk user pakai **root** dan passwordnya **()**. Kemudian pilih Finish.

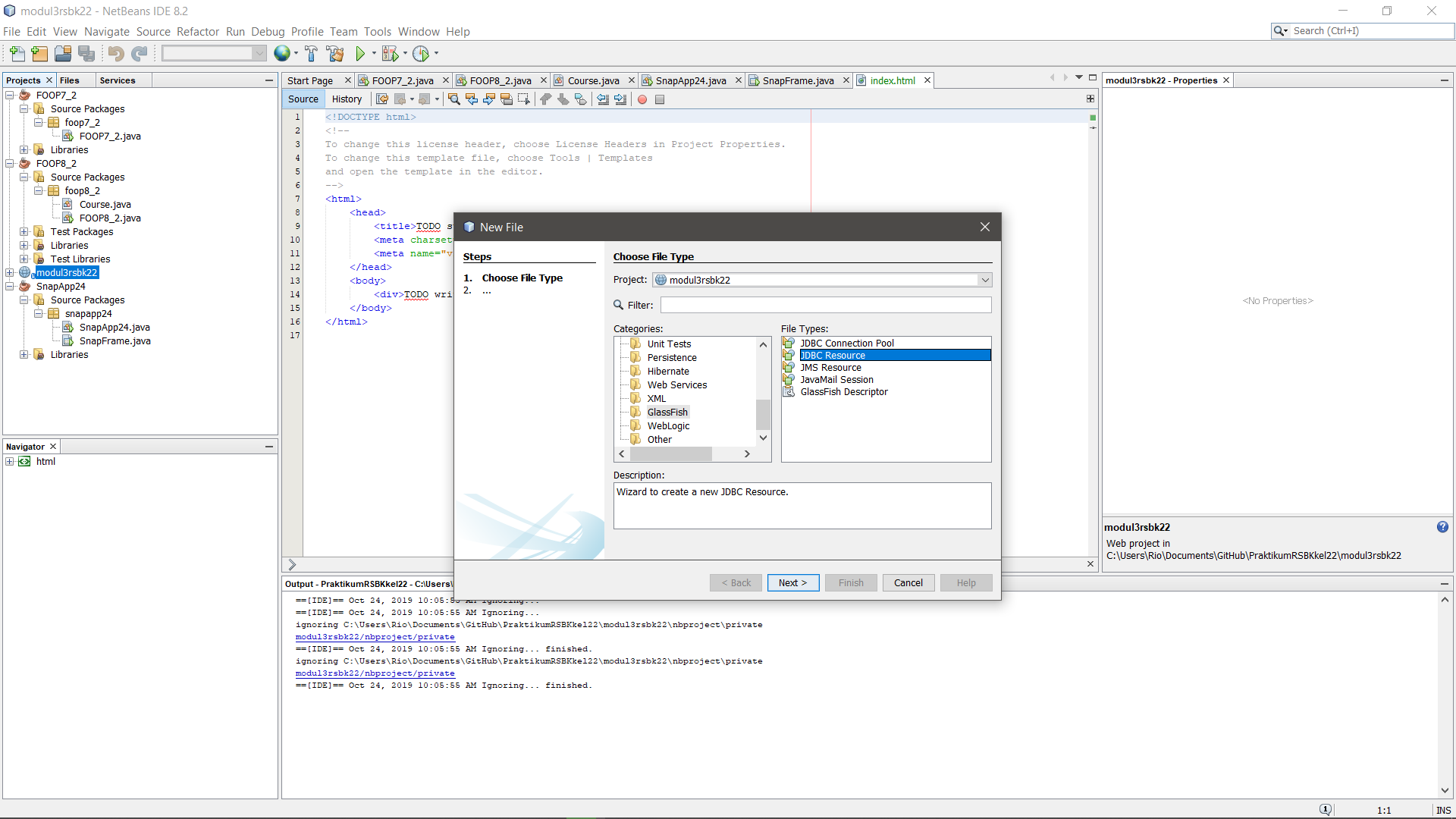


Gambar 4. 11 Membuat JDBC Connection Pool (2)

1. Buat JDBC Resource. Dengan klik kanan pada Project, New File, Other. Lalu pada kategori pilih GlassFish, kemudian pilih JDBC Resource.

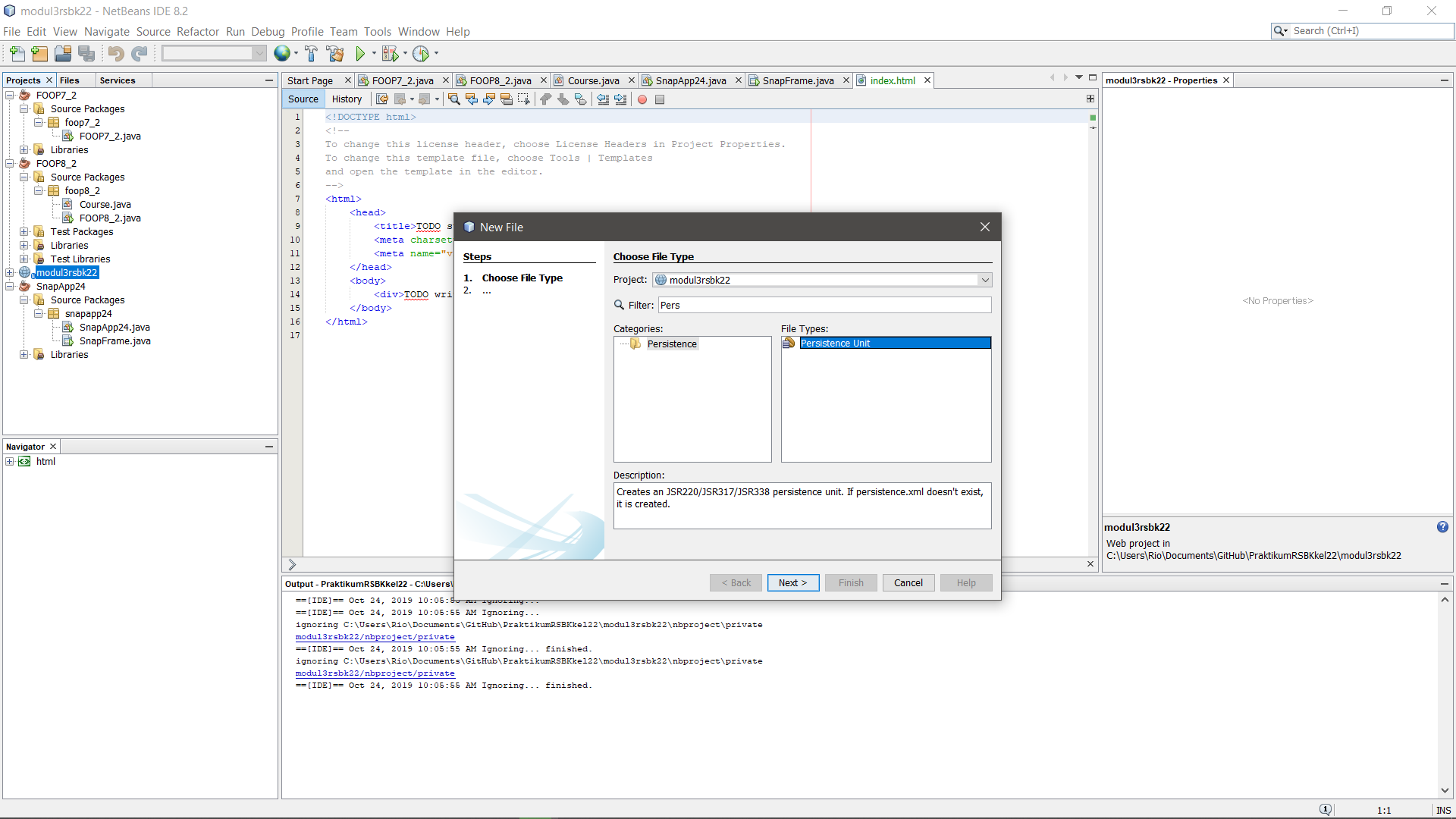


Gambar 4. 12 Menambahkan JDBC Resource (1)

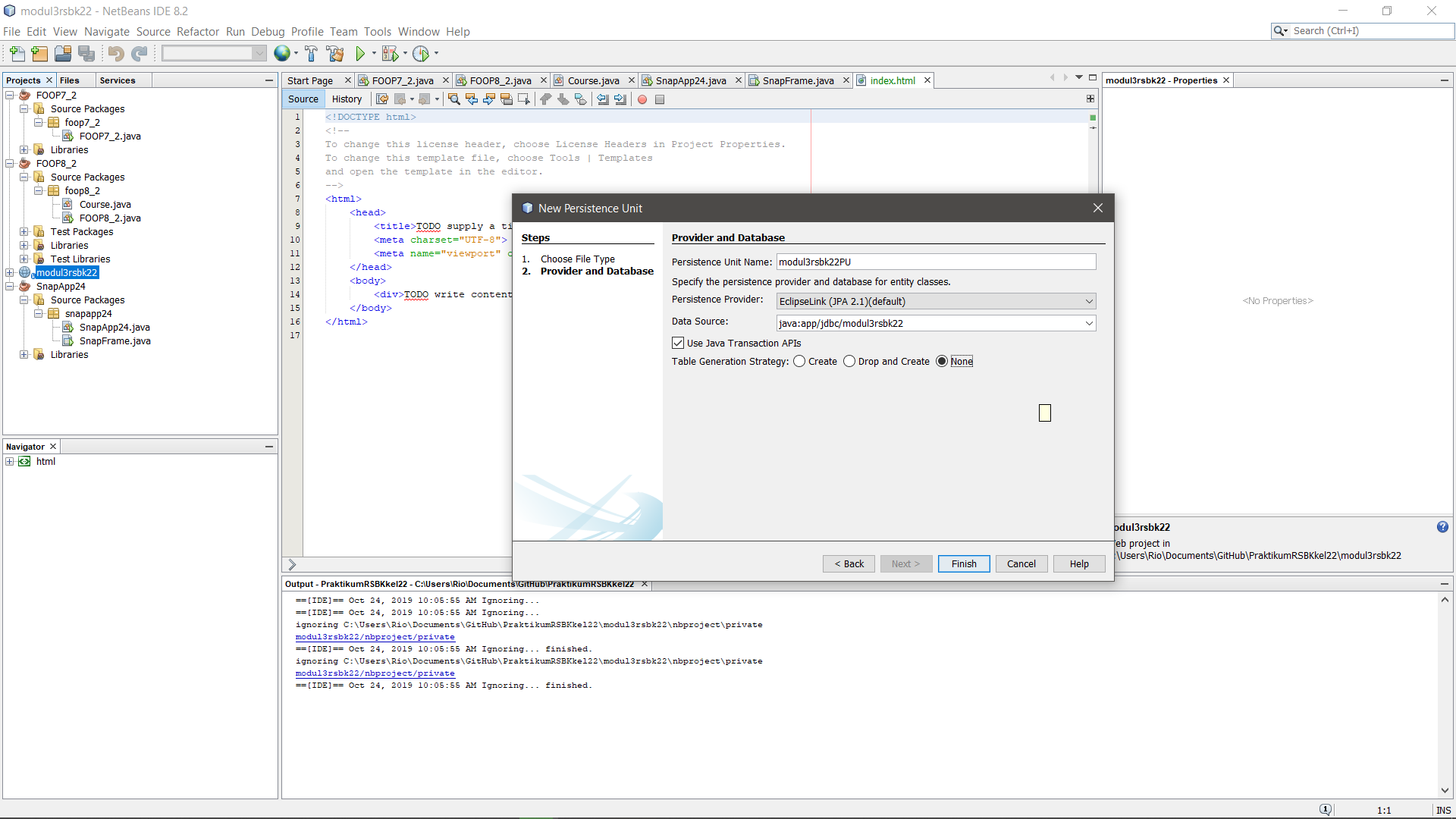


Gambar 4. 13 Menambahkan JDBC Resource (2)

1. Pilih ‘Use Existing JDBC Connection Pool’, pilih List JDBC yang baru kita buat tadi. Kemudian isikan JNDI Name seperti yang tadi, yaitu modul3rsbk22. Lalu pilih Finish.
2. Buat Persistence Unit dengan klik kanan pada project, New File, pilih Persistence Unit (Jika tidak ketemu, pilih other, lalu pada kolom filter ketik persistence unit). Nama Persistence Unit akan otomatis sesuai dengan Project, lalu pilih data resource yang tadi sudah dibuat. Table Generation Strategy pilih ‘None’

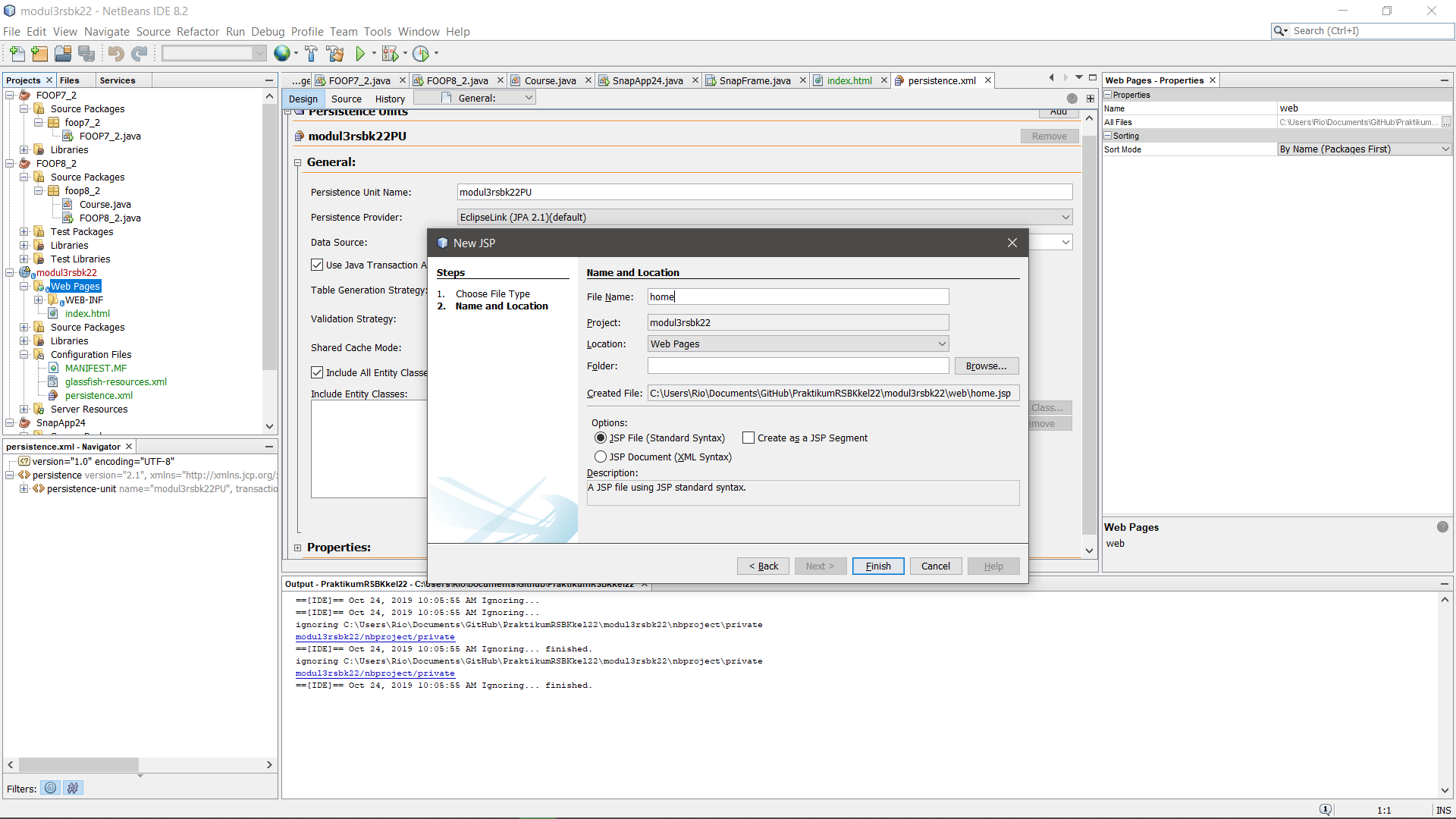


Gambar 4. 14 Menambahkan Persistance Unit (1)

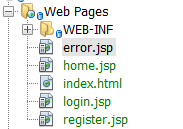


Gambar 4. 15 Menambahkan Persistance Unit (2)

1. Buat 4 JSP page pada folder “Web Pages” dengan klik kanan folder lalu new file. JSP. Beri nama: home.jsp, register.jsp, login.jsp, dan error.jsp. Masukkan source code yang tersedia.



Gambar 4. 16 Membuat file home.jsp



Gambar 4. 17 Tambahan file yang telah dibuat

**register.jsp**

|  |
| --- |
| <%--  Document : register  Created on : Sep 22, 2019, 2:48:05 PM  Author : WIN 10  --%>  <%@page contentType="text/html" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>Register Data</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <script src="js/bootstrap.min.js"></script>  <style>  .menu {  margin-left: -15px;  margin-right: 15px;  }  .daftar{  border: 2px solid #e5e5e5;  border-radius: 10px;  padding: 20px;  }  .daftar a{  margin-top: 2%;  }  .detail{  padding: 10px 0px;  }  .nav{  padding: 0px;  border: 1px solid #e5e5e5;  border-radius: 5px;  }  .nav li{  border-bottom: 1px solid #e5e5e5;  border-radius: 5px;  }  </style>  </head>  <body>  <div class="container">  <div class="jumbotron row">  <a href="./login.jsp" class="btn btn-md btn-success" style="float:right" />Login</a><br>  <center><h2><b>Data Mahasiswa</b></h2>  <h4>Modul RSBK - Kelompok22</h4></center>  </div>  <div class="row content">  <div class="col-md-12">  <div class="col-md-4 col-md-offset-4 daftar">  <p class="form-title">Sign Up</p>  <form method="POST" action="./RegisterServlet">  <div class="form-group">  <label>Username</label>  <input type="text" class="form-control" placeholder="Username" name="userName" type="text" autofocus />  </div>  <div class="form-group">  <label>Password</label>  <input type="password" class="form-control" placeholder="Password" name="password" value="" required />  </div>  <input type="submit" name="register" value="Register" class="btn btn-success" />  </form>  </div>  </div>  </div>  </div>  </body>  </html> |

**login.jsp**

|  |
| --- |
| <%--  Document : login  Created on : Sep 22, 2019, 12:45:49 PM  Author : WIN 10  --%>  <%@page contentType="text/html" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>Login Data</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <script src="js/bootstrap.min.js"></script>  <style>  .menu {  margin-left: -15px;  margin-right: 15px;  }  .daftar{  border: 2px solid #e5e5e5;  border-radius: 10px;  padding: 20px;  }  .daftar a{  margin-top: 2%;  }  .detail{  padding: 10px 0px;  }  .nav{  padding: 0px;  border: 1px solid #e5e5e5;  border-radius: 5px;  }  .nav li{  border-bottom: 1px solid #e5e5e5;  border-radius: 5px;  }  </style>  </head>  <body>  <div class="container">  <div class="jumbotron row">  <a href="./register.jsp" class="btn btn-md btn-success" style="float:right" />Register</a><br>  <center><h2><b>Data Mahasiswa</b></h2>  <h4>Modul RSBK - Kelompok22</h4></center>  </div>  <div class="row content">  <div class="col-md-12">  <div class="col-md-4 col-md-offset-4 daftar">  <p class="form-title">Sign In</p>  <form method="POST" action="./LoginServlet">  <div class="form-group">  <label>Username</label>  <input type="text" class="form-control" placeholder="Username" name="userName" type="text" autofocus />  </div>  <div class="form-group">  <label>Password</label>  <input type="password" class="form-control" placeholder="Password" name="password" value="" required />  </div>  <input type="submit" name="login" value="Login" class="btn btn-md btn-success" />  </form>  </div>  </div>  </div>  </div>  </body>  </html> |

**home.jsp**

|  |
| --- |
| <%--  Document : home  Created on : Sep 22, 2019, 12:45:58 PM  Author : WIN 10  --%>  <%@page contentType="text/html" pageEncoding="UTF-8"%>  <%@taglib prefix="s" uri="http://java.sun.com/jsp/jstl/core" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Home Page</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <script src="js/bootstrap.min.js"></script>  <style>  .menu {  margin-left: -15px;  margin-right: 15px;  }  .daftar{  border: 2px solid #e5e5e5;  border-radius: 5px;  padding: 5px;  }  .table th, .table td{  text-align: center;  }  .nav{  padding: 5px;  border: 2px solid #e5e5e5;  border-radius: 5px;  }  .nav li{  border-bottom: 2px solid #e5e5e5;  border-radius: 5px;  }  .daftar h3{  margin-top: 50px;  margin-bottom: 25px;  }  </style>  </head>  <div class="container">  <div class="jumbotron row">  <center><h2><b>Data Mahasiswa</b></h2>  <p>Modul RSBK - Kelompok22</p>  <h5>Selamat Datang, <%=session.getAttribute("loginName")%></h6></center>  </div>  <div class="row content col-md-8 col-md-offset-2">  <div class="col-md-3 menu">  <ul class="nav nav-pills nav-stacked" style="">  <li><a href="#">Home</a></li>  <li><a href="./login.jsp">Logout</a></li>  </ul>  </div>  <div class="col-md-9 daftar">  <form action="./StudentServlet" method="POST">  <table class="table table-bordered">  <tr>  <td>Student ID</td>  <td><input class="form-control" type="text" name="studentId" value="${student.studentId}" /></td>  </tr>  <tr>  <td>First Name</td>  <td><input class="form-control" type="text" name="firstname" value="${student.firstName}" /></td>  </tr>  <tr>  <td>Last Name</td>  <td><input class="form-control" type="text" name="lastname" value="${student.lastName}" /></td>  </tr>  <tr>  <td colspan="2">  <input type="submit" class="btn btn-primary btn-sm" name="action" value="Add" />  <input type="submit" class="btn btn-default btn-sm" name="action" value="Edit" />  <input type="submit" class="btn btn-danger btn-sm" name="action" value="Delete" />  <input type="submit" class="btn btn-warning btn-sm" name="action" value="Search" />  </td>  </tr>  </table>  </form>  <h3 align="center">Informasi Data</h3>  <table class="table table-bordered table-hover">  <thead>  <tr>  <th>No. ID</th>  <th>First Name</th>  <th>Last Name</th>  </tr>  </thead>  <tbody>  <s:forEach items="${allStudents}" var="stud">  <tr>  <td>${stud.studentId}</td>  <td>${stud.firstName}</td>  <td>${stud.lastName}</td>  </tr>  </s:forEach>  </tbody>  </table>  </div>  </div>  </div>  </html> |

**error.jsp**

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Error Page</title>

</head>

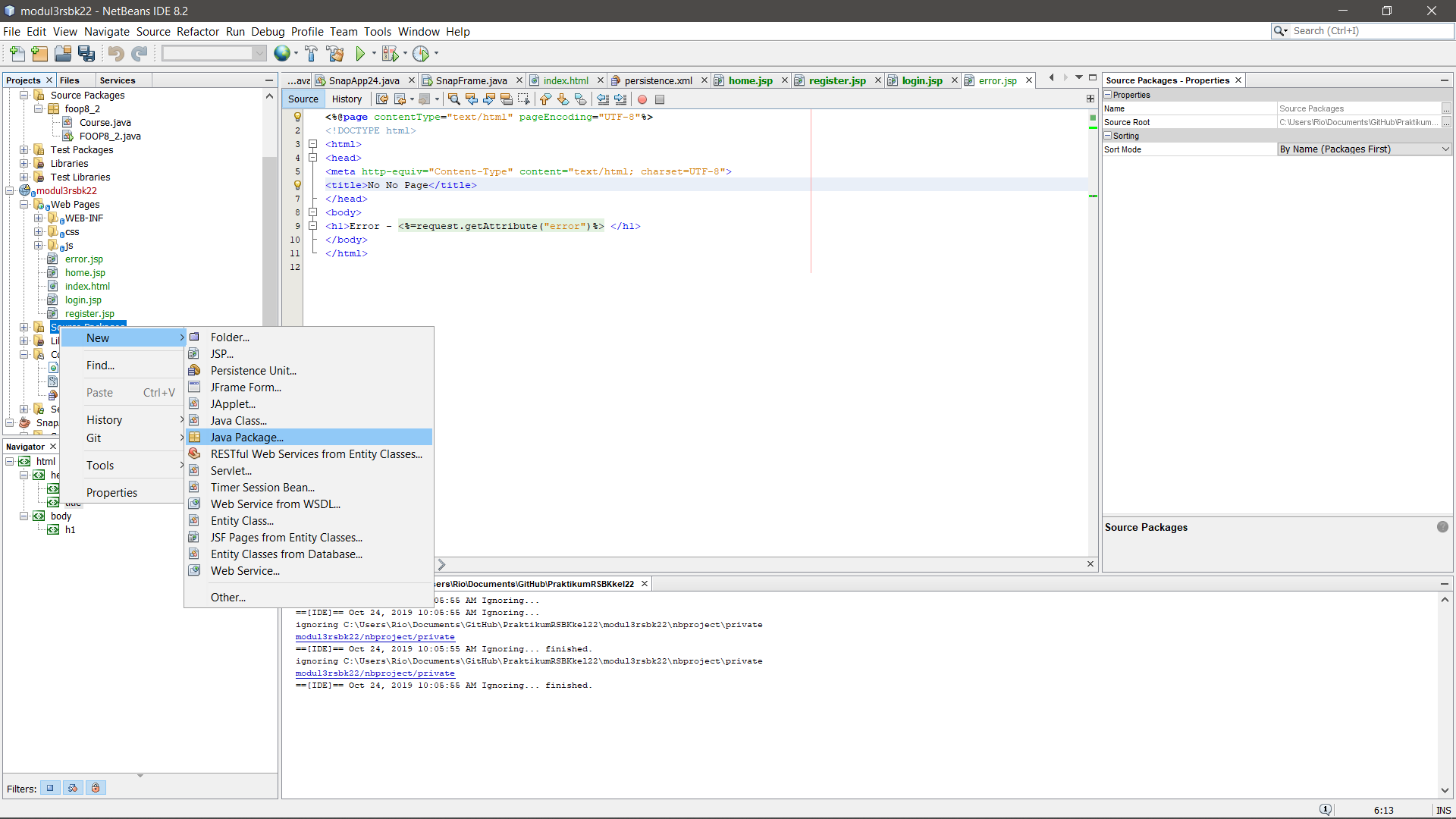
<body>

<h1>Error - <%=request.getAttribute("error")%> </h1>

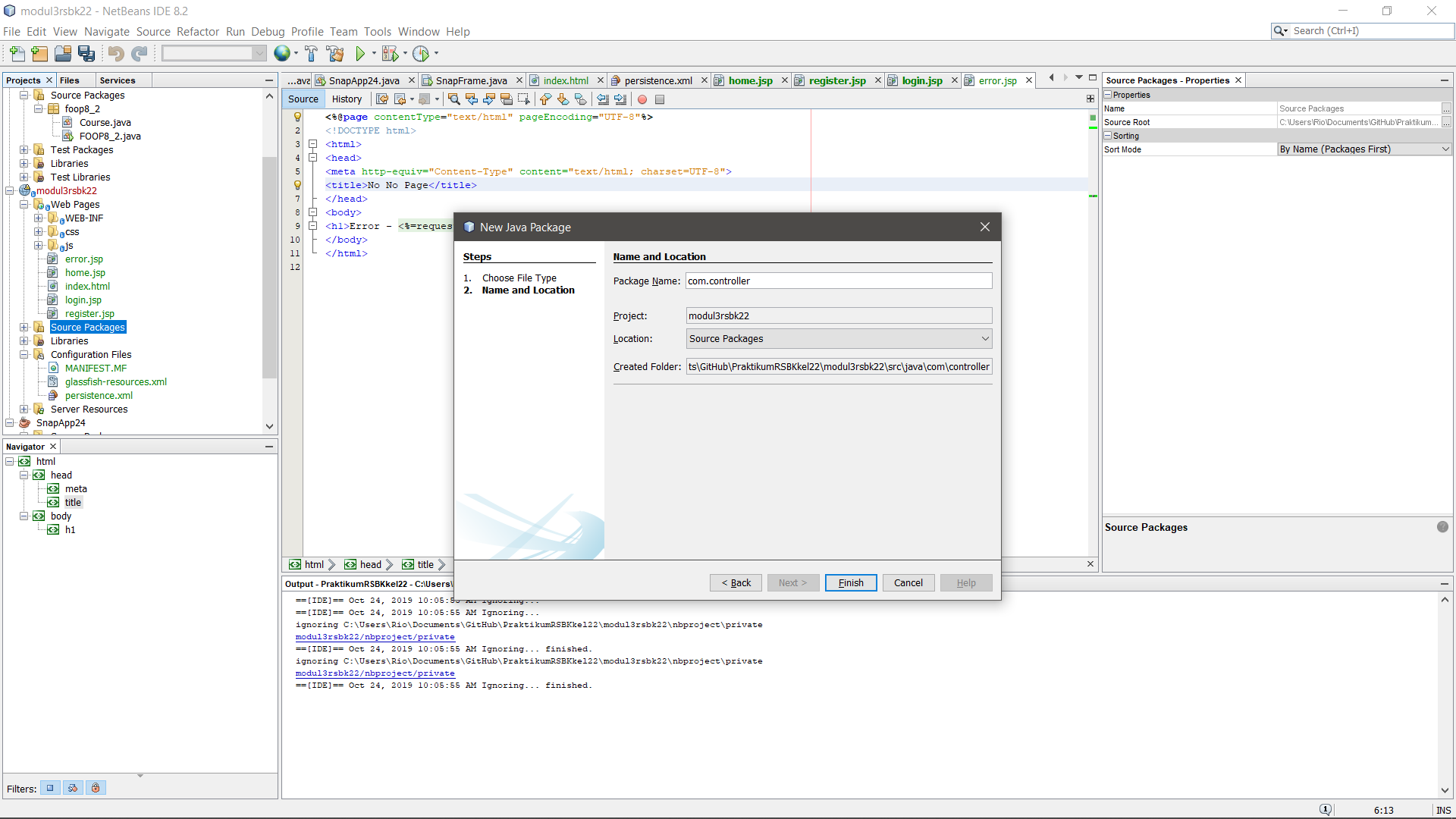
</body>

</html>

1. Masukkan asset dengan copy dan paste folder css dan js ke web pages
2. Buat 3 java package pada folder “Source Packages”, beri nama: “com.controller”, “com.dao”, dan “com.model”.



Gambar 4. 18 Membuat package baru

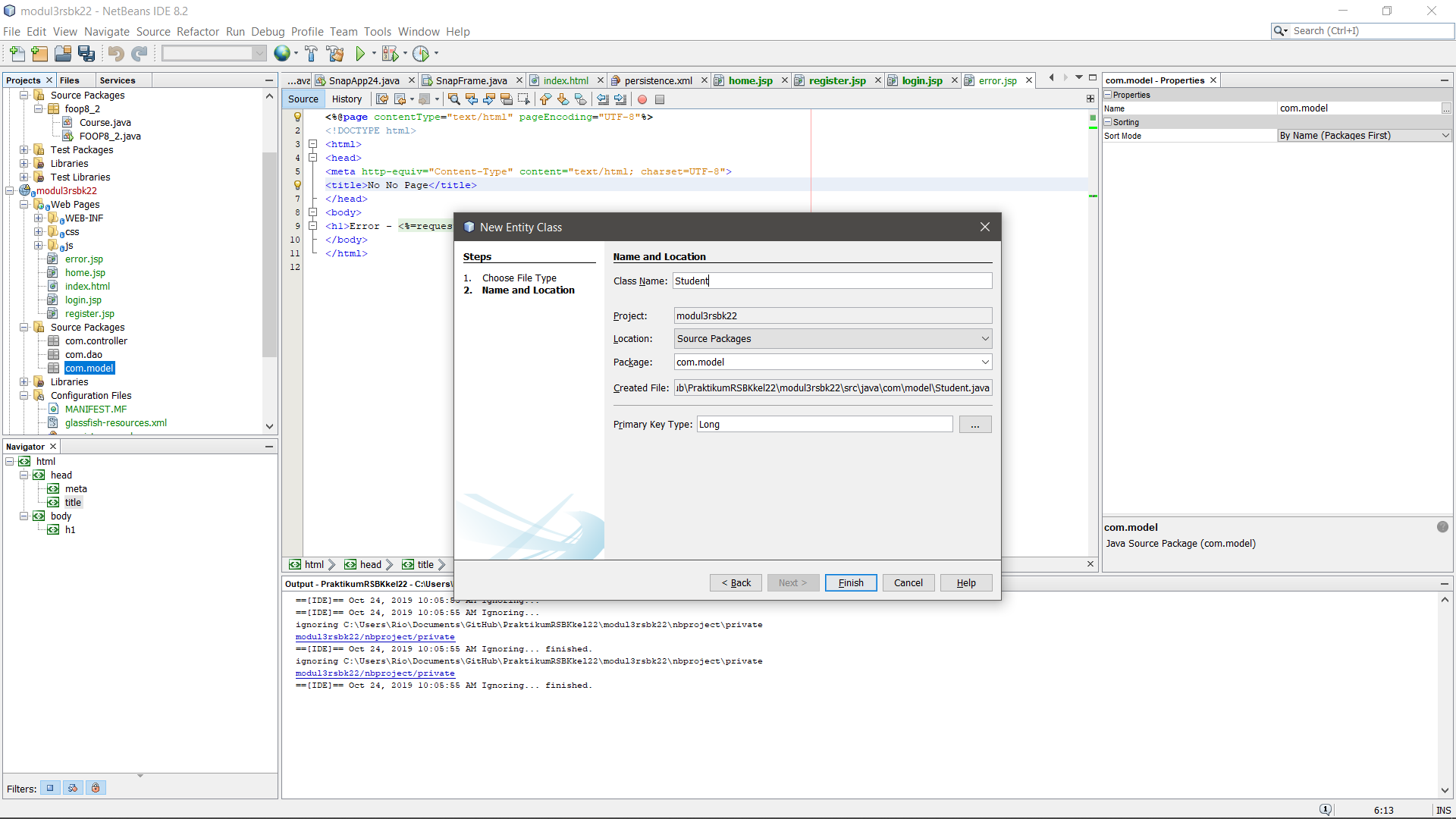


Gambar 4. 19 Menambahkan package com.controller

1. Buat 2 Entity Class pada package “com.model”, beri nama : “User” dan “Student”. Masukkan source code yang tersedia.



Gambar 4. 20 Membuat class baru



Gambar 4. 21 Menambahkan class baru dalam package com.model

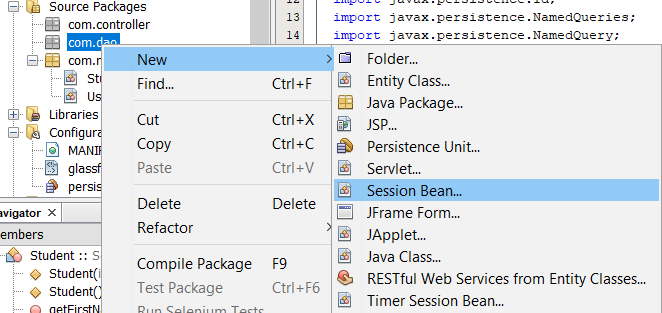
**User.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.model;  import java.io.Serializable;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.NamedQueries;  import javax.persistence.NamedQuery;  import javax.persistence.Table;  import javax.persistence.Column;  /\*\*  \*  \* @author USER  \*/  @Entity  @Table  @NamedQueries({@NamedQuery(name="User.getAll",query="SELECT e FROM User e")})  public class User implements Serializable {  private static final long serialVersionUID = 1L;  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int userId;  @Column  private String userName;  @Column  private String password;  public User() {  }  public User(String userName, String password) {  this.userName = userName;  this.password = password;  }  public int getUserId() {  return userId;  }  public void setUserId(int userId) {  this.userId = userId;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  } |

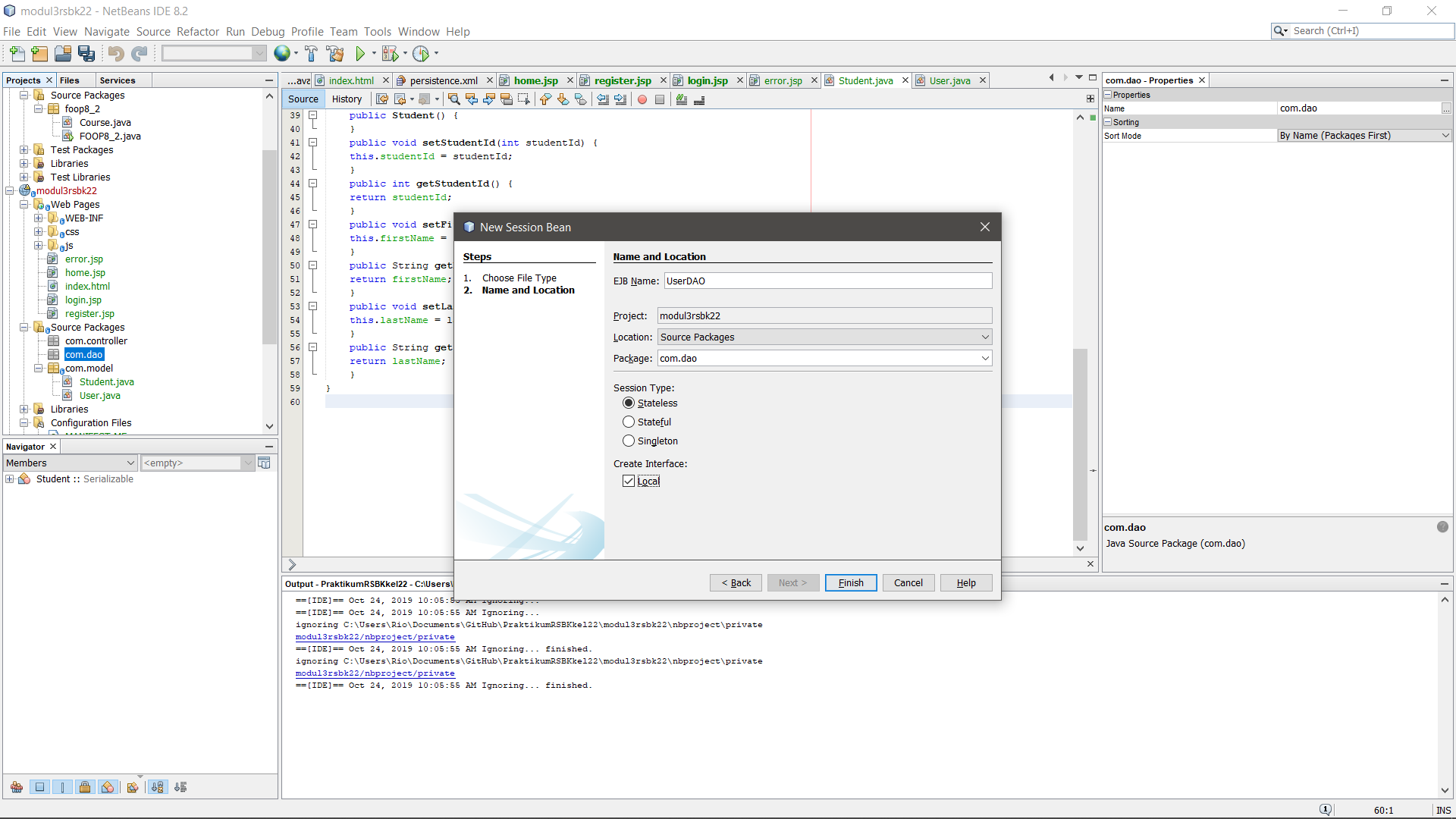
**Student.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.model;  import java.io.Serializable;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.NamedQueries;  import javax.persistence.NamedQuery;  import javax.persistence.Table;  import javax.persistence.Column;  /\*\*  \*  \* @author USER  \*/  @Entity  @Table  @NamedQueries({@NamedQuery(name="Student.getAll",query="SELECT e FROM Student e order by e.studentId")})  public class Student implements Serializable {  @Id  @GeneratedValue(strategy=GenerationType.AUTO)  private int studentId;  @Column  private String firstName;  @Column  private String lastName;  public Student(int studentId, String firstName, String lastName) {  this.studentId = studentId;  this.firstName = firstName;  this.lastName = lastName;  }  public Student() {  }  public void setStudentId(int studentId) {  this.studentId = studentId;  }  public int getStudentId() {  return studentId;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getFirstName() {  return firstName;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getLastName() {  return lastName;  }  } |

1. Buat 2 Session Beans pada package “com.dao”, beri nama : “UserDAO” dan “StudentDAO”. Ketika membuat session bean, pilih Session Type ‘Stateless’ dan Create Interface ‘Local’. Masukkan source code yang tersedia.



Gambar 4. 22 Membuat session bean baru



Gambar 4. 23 Membuat session bean UserDAO

**UserDAO.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import javax.ejb.Stateless;  import com.model.User;  import java.util.List;  import javax.persistence.EntityManager;  import javax.persistence.PersistenceContext;  /\*\*  \*  \* @author WIN 10  \*/  @Stateless  public class UserDAO implements UserDAOLocal {  @PersistenceContext  private EntityManager em;  @Override  public boolean credential(String userName, String password) {  List<User> s = (List<User>)em.createQuery("select e from User e where e.userName='"+userName+"' and e.password='"+password+"'").getResultList();  System.out.println("is list empty ?"+s.isEmpty()+" for the"+userName+" and "+password);  if(!s.isEmpty())  return true;  else  return false;  }  @Override  public boolean checkUser(String userName) {  List<User> s = (List<User>)em.createQuery("select e from User e where e.userName='"+userName+"'").getResultList();  if(s.isEmpty())  return true;  else  return false;  }  @Override  public void addUser(User user){  em.merge(user);  em.flush();  }  // Add business logic below. (Right-click in editor and choose  // "Insert Code > Add Business Method")  } |

**UserDAOLocal.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.User;  import javax.ejb.Local;  /\*\*  \*  \* @author WIN 10  \*/  @Local  public interface UserDAOLocal {  public boolean credential(String userName, String password);  public boolean checkUser (String userName);  void addUser (User user);  } |

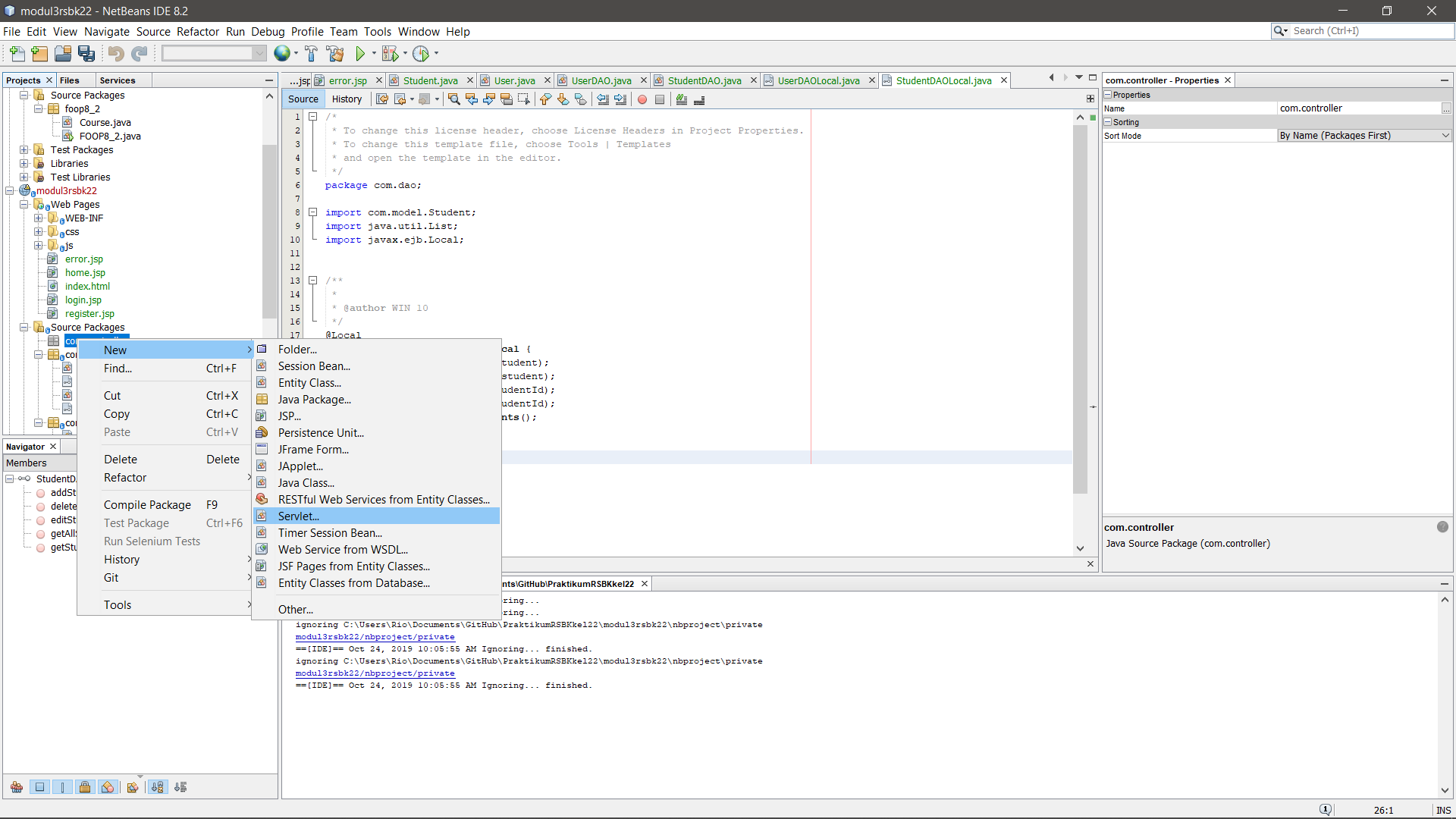
**StudentDAO.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.Student;  import java.util.List;  import javax.ejb.Stateless;  import javax.persistence.EntityManager;  import javax.persistence.PersistenceContext;  /\*\*  \*  \* @author WIN 10  \*/  @Stateless  public class StudentDAO implements StudentDAOLocal {  @PersistenceContext  private EntityManager em;  @Override  public void addStudent(Student student) {  em.merge(student);  em.flush();  }  @Override  public void editStudent(Student student) {  em.merge(student);  em.flush();  }  @Override  public void deleteStudent(int studentId) {  em.remove(getStudent(studentId));  em.flush();  }  @Override  public Student getStudent(int studentId) {  em.flush();  return em.find(Student.class, studentId);  }  @Override  public List<Student> getAllStudents() {  em.flush();  return em.createNamedQuery("Student.getAll").getResultList();  }  } |

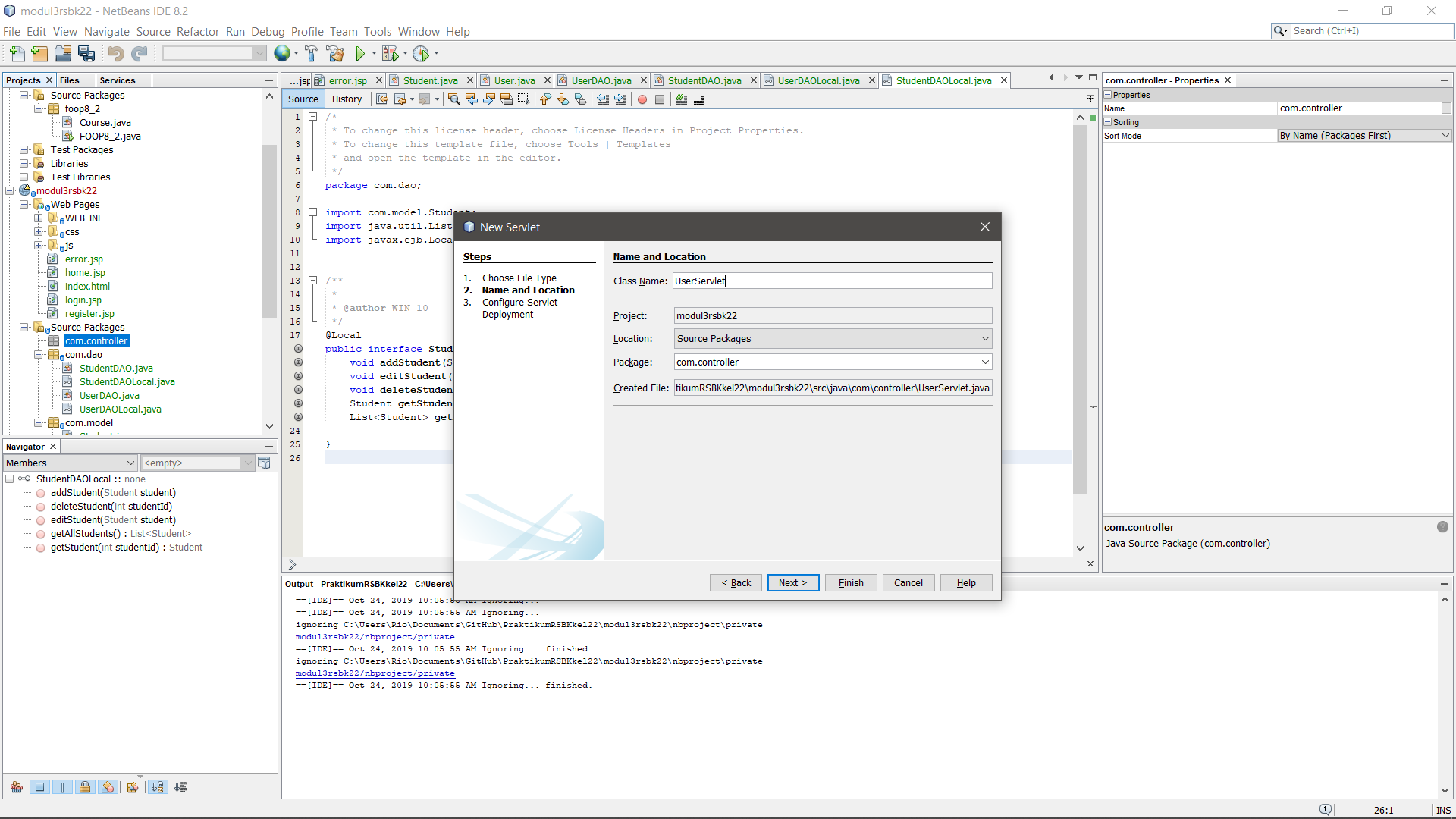
**StudentDAOLocal.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.Student;  import java.util.List;  import javax.ejb.Local;  /\*\*  \*  \* @author WIN 10  \*/  @Local  public interface StudentDAOLocal {  void addStudent(Student student);  void editStudent(Student student);  void deleteStudent(int studentId);  Student getStudent(int studentId);  List<Student> getAllStudents();  } |

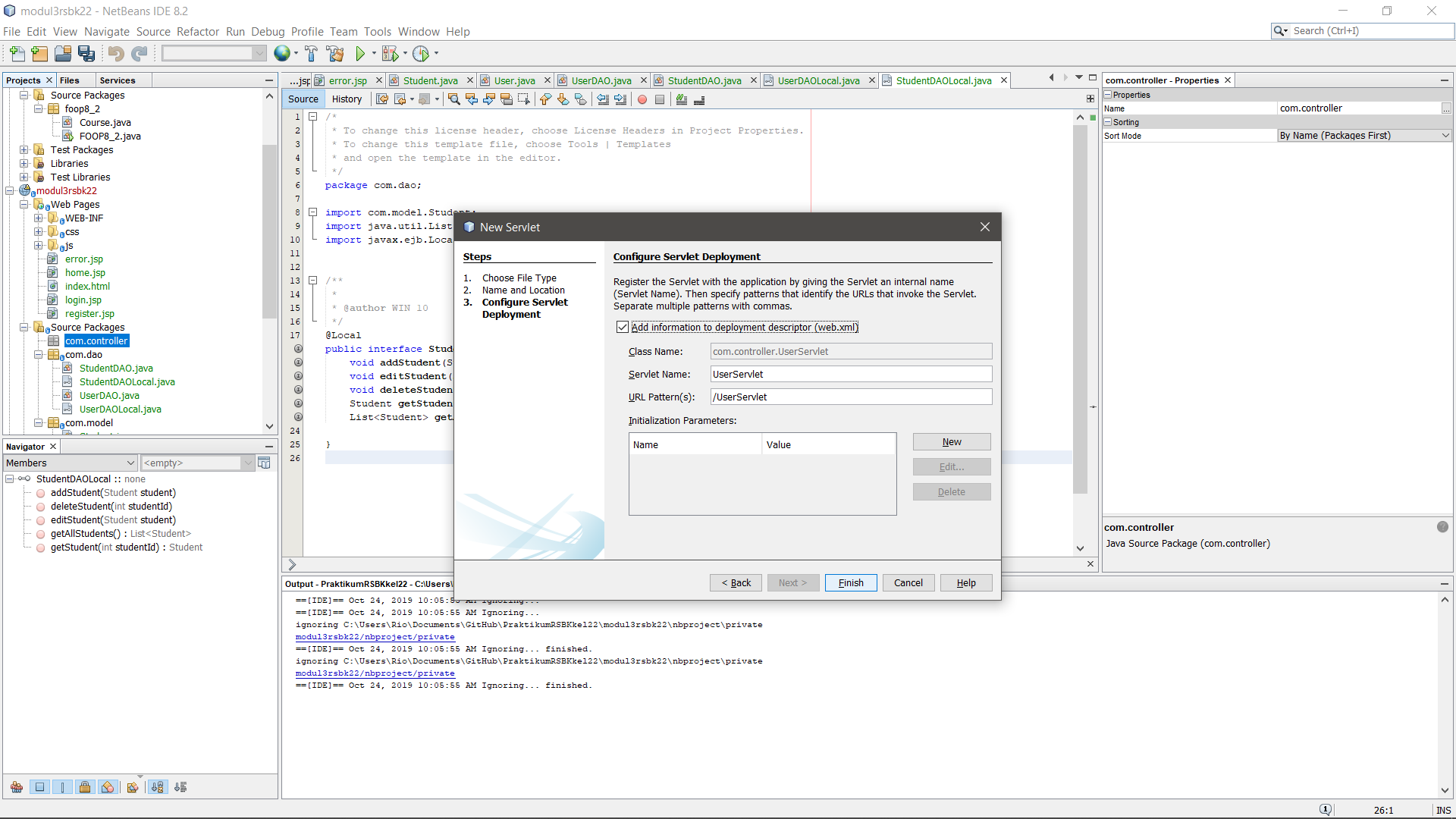
1. Buat 3 Servlet pada package “com.controller”, beri nama : “LoginServlet”, “RegisterServlet”, “StudentServlet”. Ketika membuat servlet, centang pada ‘Add information to deployment descriptor (web.xml)’. Kemudian masukkan source code yang tersedia.



Gambar 4. 24 Menambahkan servlet baru



Gambar 4. 25 Membuat UserServlet (1)



Gambar 4. 26 Membuat UserServlet (2)

**LoginServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import com.dao.UserDAOLocal;  /\*\*  \*  \* @author WIN 10  \*/  @WebServlet(name = "LoginServlet", urlPatterns = {"/LoginServlet"})  public class LoginServlet extends HttpServlet {  @EJB  private UserDAOLocal userDAO;  boolean check = false;    /\*\*  \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>  \* methods.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  String userName = request.getParameter("userName");  String password = request.getParameter("password");  HttpSession session = request.getSession();  check = userDAO.credential(userName,password);  System.out.println("check is"+check+" "+userName);  if(check)  {  session.setAttribute("userName", userName);  request.getRequestDispatcher("./StudentServlet").forward(request, response);  } else {  request.setAttribute("error", "Wrong Username or Password");  request.getRequestDispatcher("error.jsp").forward(request, response);  }  }  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">  /\*\*  \* Handles the HTTP <code>GET</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Handles the HTTP <code>POST</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Returns a short description of the servlet.  \*  \* @return a String containing servlet description  \*/  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

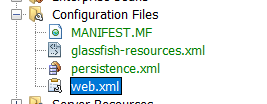
**RegisterServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import com.model.User;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.dao.UserDAOLocal;  /\*\*  \*  \* @author WIN 10  \*/  public class RegisterServlet extends HttpServlet {  @EJB  private UserDAOLocal UserDAO;  boolean check = true;  /\*\*  \* Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  String userName = request.getParameter("userName");  String password = request.getParameter("password");  check = UserDAO.checkUser(userName);  System.out.println("check is"+check+" "+userName);  if(check){  User user = new User(userName, password);  UserDAO.addUser(user);  request.setAttribute("user", user);  request.getRequestDispatcher("login.jsp").forward(request, response);  }else{  request.setAttribute("error", "Username already taken");  request.getRequestDispatcher("error.jsp").forward(request, response);  }    }  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">  /\*\*  \* Handles the HTTP <code>GET</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Handles the HTTP <code>POST</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Returns a short description of the servlet.  \*  \* @return a String containing servlet description  \*/  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

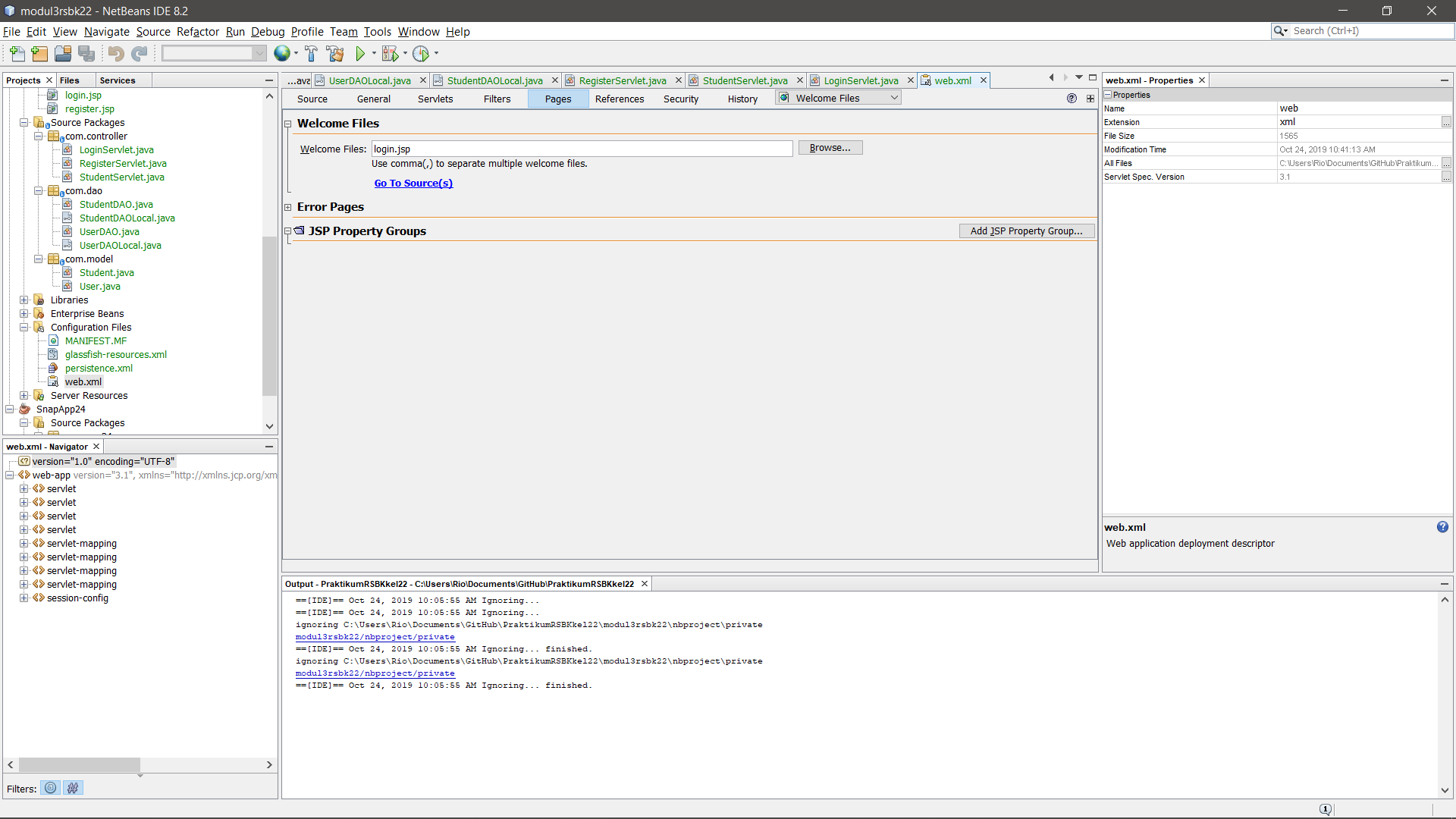
**StudentServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import com.dao.StudentDAOLocal;  import com.model.Student;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \*  \* @author WIN 10  \*/  @WebServlet(name = "StudentServlet")  public class StudentServlet extends HttpServlet {  @EJB  private StudentDAOLocal studentDao;  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  String studentIdStr = request.getParameter("studentId");  int studentId=0;  if(studentIdStr!=null && !studentIdStr.equals("")){  studentId=Integer.parseInt(studentIdStr);  }  String firstname = request.getParameter("firstname");  String lastname = request.getParameter("lastname");  Student student = new Student(studentId, firstname, lastname);  if("Add".equalsIgnoreCase(action)){  studentDao.addStudent(student);  }else if("Edit".equalsIgnoreCase(action)){  studentDao.editStudent(student);  }else if("Delete".equalsIgnoreCase(action)){  studentDao.deleteStudent(studentId);  }else if("Search".equalsIgnoreCase(action)){  student = studentDao.getStudent(studentId);  }  request.setAttribute("student", student);  request.setAttribute("allStudents", studentDao.getAllStudents());  request.getRequestDispatcher("home.jsp").forward(request, response);  }  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

1. Buka “web.xml” di folder ‘Configuration Files’. Pada tab ‘Pages’, isikan Welcome Files dengan “login.jsp”.



Gambar 4. 27 Membuka file web.xml



Gambar 4. 28 Mengisikan welcome file dengan login.jsp

1. Lakukan ‘Clean and Build’ (Shift + F11), kemudian ‘Run’ (F6)

## Analisa Percobaan

Dalam percobaan tersebut, terdapat tiga buah package utama yang digunakan, yaitu package “com.controller”, package “com.dao”, dan package “com.model”.

Package “com.model” memiliki 2 buah entity class yaitu User dan Student. Berikut adalah *source code* yang digunakan pada Class User dan Student:

**User.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.model;  import java.io.Serializable;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.NamedQueries;  import javax.persistence.NamedQuery;  import javax.persistence.Table;  import javax.persistence.Column;  /\*\*  \*  \* @author USER  \*/  @Entity  @Table  @NamedQueries({@NamedQuery(name="User.getAll",query="SELECT e FROM User e")})  public class User implements Serializable {  private static final long serialVersionUID = 1L;  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private int userId;  @Column  private String userName;  @Column  private String password;  public User() {  }  public User(String userName, String password) {  this.userName = userName;  this.password = password;  }  public int getUserId() {  return userId;  }  public void setUserId(int userId) {  this.userId = userId;  }  public String getUserName() {  return userName;  }  public void setUserName(String userName) {  this.userName = userName;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  } |

**Student.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.model;  import java.io.Serializable;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.NamedQueries;  import javax.persistence.NamedQuery;  import javax.persistence.Table;  import javax.persistence.Column;  /\*\*  \*  \* @author USER  \*/  @Entity  @Table  @NamedQueries({@NamedQuery(name="Student.getAll",query="SELECT e FROM Student e order by e.studentId")})  public class Student implements Serializable {  @Id  @GeneratedValue(strategy=GenerationType.AUTO)  private int studentId;  @Column  private String firstName;  @Column  private String lastName;  public Student(int studentId, String firstName, String lastName) {  this.studentId = studentId;  this.firstName = firstName;  this.lastName = lastName;  }  public Student() {  }  public void setStudentId(int studentId) {  this.studentId = studentId;  }  public int getStudentId() {  return studentId;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getFirstName() {  return firstName;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }  public String getLastName() {  return lastName;  }  } |

Dapat dilihat pada *source code* tersebut bahwa Class User digunakan untuk men-set dan get User ID, UserName, dan Password, sedangkan class Student digunakan untuk men-set dan get Student ID, FirstName, dan LastName. Kesimpulannya, package “com.model” ini digunakan untuk mengatur set dan get yang akan digunakan pada aplikasi untuk pengisian *text field*.

Kemudian package “com.dao”, ia memiliki 2 buah entity class juga yaitu “UserDAO” dan “StudentDAO”. Kedua class ini memiliki Session Type Stateless dan interface local. Beriku adalah *source code* pada UserDAO dan StudentDAO.

**UserDAO.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import javax.ejb.Stateless;  import com.model.User;  import java.util.List;  import javax.persistence.EntityManager;  import javax.persistence.PersistenceContext;  /\*\*  \*  \* @author WIN 10  \*/  @Stateless  public class UserDAO implements UserDAOLocal {  @PersistenceContext  private EntityManager em;  @Override  public boolean credential(String userName, String password) {  List<User> s = (List<User>)em.createQuery("select e from User e where e.userName='"+userName+"' and e.password='"+password+"'").getResultList();  System.out.println("is list empty ?"+s.isEmpty()+" for the"+userName+" and "+password);  if(!s.isEmpty())  return true;  else  return false;  }  @Override  public boolean checkUser(String userName) {  List<User> s = (List<User>)em.createQuery("select e from User e where e.userName='"+userName+"'").getResultList();  if(s.isEmpty())  return true;  else  return false;  }  @Override  public void addUser(User user){  em.merge(user);  em.flush();  }  // Add business logic below. (Right-click in editor and choose  // "Insert Code > Add Business Method")  } |

**UserDAOLocal.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.User;  import javax.ejb.Local;  /\*\*  \*  \* @author WIN 10  \*/  @Local  public interface UserDAOLocal {  public boolean credential(String userName, String password);  public boolean checkUser (String userName);  void addUser (User user);  } |

**StudentDAO.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.Student;  import java.util.List;  import javax.ejb.Stateless;  import javax.persistence.EntityManager;  import javax.persistence.PersistenceContext;  /\*\*  \*  \* @author WIN 10  \*/  @Stateless  public class StudentDAO implements StudentDAOLocal {  @PersistenceContext  private EntityManager em;  @Override  public void addStudent(Student student) {  em.merge(student);  em.flush();  }  @Override  public void editStudent(Student student) {  em.merge(student);  em.flush();  }  @Override  public void deleteStudent(int studentId) {  em.remove(getStudent(studentId));  em.flush();  }  @Override  public Student getStudent(int studentId) {  em.flush();  return em.find(Student.class, studentId);  }  @Override  public List<Student> getAllStudents() {  em.flush();  return em.createNamedQuery("Student.getAll").getResultList();  }  } |

**StudentDAOLocal.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.dao;  import com.model.Student;  import java.util.List;  import javax.ejb.Local;  /\*\*  \*  \* @author WIN 10  \*/  @Local  public interface StudentDAOLocal {  void addStudent(Student student);  void editStudent(Student student);  void deleteStudent(int studentId);  Student getStudent(int studentId);  List<Student> getAllStudents();  } |

Dapat dilihat pada *source code* di atas bahwa UserDAO digunakan untuk mengkonfirmasi username dan password yang akan digunakan untuk terhubung dengan database tanpa memaparkan rincian database itu sendiri, dengan kata lain aplikasi yang memberikan pemetaan kepada database. Pada bagian UserDAOLocal, ia digunakan oleh class UserDAO-nya untuk diimplementasikan apa yang telah dioperasikan oleh class, dengan kata lain interface local pada UserDAO hanya sebagai deklarasi method abstract saja.

Sementara itu StudentDAO digunakan untuk menghubungkan ke database untuk bagian addStudent, editStudent, dan deleteStudent tanpa memaparkan rincian dalam database itu sendiri. Sama seperti UserDAO, StudentDAOLocal digunakan oleh class StudentDAO-nya untuk diimplementasikan apa yang telah dioperasikan oleh class, dengan kata lain interface local pada StudentDAO hanya sebagai deklarasi method abstract saja.

Package “com.controller” memiliki 3 buah Servlet, yaitu “LoginServlet”, “RegisterServlet”, dan “StudentServlet”. Berikut adalah *source code* dari ketiganya.

**LoginServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import javax.servlet.http.HttpSession;  import com.dao.UserDAOLocal;  /\*\*  \*  \* @author WIN 10  \*/  @WebServlet(name = "LoginServlet", urlPatterns = {"/LoginServlet"})  public class LoginServlet extends HttpServlet {  @EJB  private UserDAOLocal userDAO;  boolean check = false;    /\*\*  \* Processes requests for both HTTP <code>GET</code> and <code>POST</code>  \* methods.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  String userName = request.getParameter("userName");  String password = request.getParameter("password");  HttpSession session = request.getSession();  check = userDAO.credential(userName,password);  System.out.println("check is"+check+" "+userName);  if(check)  {  session.setAttribute("userName", userName);  request.getRequestDispatcher("./StudentServlet").forward(request, response);  } else {  request.setAttribute("error", "Wrong Username or Password");  request.getRequestDispatcher("error.jsp").forward(request, response);  }  }  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">  /\*\*  \* Handles the HTTP <code>GET</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Handles the HTTP <code>POST</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Returns a short description of the servlet.  \*  \* @return a String containing servlet description  \*/  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

**RegisterServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import com.model.User;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import com.dao.UserDAOLocal;  /\*\*  \*  \* @author WIN 10  \*/  public class RegisterServlet extends HttpServlet {  @EJB  private UserDAOLocal UserDAO;  boolean check = true;  /\*\*  \* Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  response.setContentType("text/html;charset=UTF-8");  String userName = request.getParameter("userName");  String password = request.getParameter("password");  check = UserDAO.checkUser(userName);  System.out.println("check is"+check+" "+userName);  if(check){  User user = new User(userName, password);  UserDAO.addUser(user);  request.setAttribute("user", user);  request.getRequestDispatcher("login.jsp").forward(request, response);  }else{  request.setAttribute("error", "Username already taken");  request.getRequestDispatcher("error.jsp").forward(request, response);  }    }  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">  /\*\*  \* Handles the HTTP <code>GET</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Handles the HTTP <code>POST</code> method.  \*  \* @param request servlet request  \* @param response servlet response  \* @throws ServletException if a servlet-specific error occurs  \* @throws IOException if an I/O error occurs  \*/  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  /\*\*  \* Returns a short description of the servlet.  \*  \* @return a String containing servlet description  \*/  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

**StudentServlet.java**

|  |
| --- |
| /\*  \* To change this license header, choose License Headers in Project Properties.  \* To change this template file, choose Tools | Templates  \* and open the template in the editor.  \*/  package com.controller;  import com.dao.StudentDAOLocal;  import com.model.Student;  import java.io.IOException;  import javax.ejb.EJB;  import javax.servlet.ServletException;  import javax.servlet.annotation.WebServlet;  import javax.servlet.http.HttpServlet;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  /\*\*  \*  \* @author WIN 10  \*/  @WebServlet(name = "StudentServlet")  public class StudentServlet extends HttpServlet {  @EJB  private StudentDAOLocal studentDao;  protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String action = request.getParameter("action");  String studentIdStr = request.getParameter("studentId");  int studentId=0;  if(studentIdStr!=null && !studentIdStr.equals("")){  studentId=Integer.parseInt(studentIdStr);  }  String firstname = request.getParameter("firstname");  String lastname = request.getParameter("lastname");  Student student = new Student(studentId, firstname, lastname);  if("Add".equalsIgnoreCase(action)){  studentDao.addStudent(student);  }else if("Edit".equalsIgnoreCase(action)){  studentDao.editStudent(student);  }else if("Delete".equalsIgnoreCase(action)){  studentDao.deleteStudent(studentId);  }else if("Search".equalsIgnoreCase(action)){  student = studentDao.getStudent(studentId);  }  request.setAttribute("student", student);  request.setAttribute("allStudents", studentDao.getAllStudents());  request.getRequestDispatcher("home.jsp").forward(request, response);  }  @Override  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  @Override  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  processRequest(request, response);  }  @Override  public String getServletInfo() {  return "Short description";  }// </editor-fold>  } |

Dalam *source code* tersebut, LoginServlet digunakan sebagai penghubung ke *web-server* untuk menanggapi permintaan dari *client* untuk user login ke dalam aplikasi. Kemudian RegisterServlet digunakan sebagai penghubung ke *web-server* untuk menanggapi permintaan dari *client* untuk user register ke dalam aplikasi. Terakhir, StudentServlet digunakan sebagai penghubung ke *web-server* untuk menanggapi permintaan dari *client* untuk user terhubung dengan StudentDAO ke dalam aplikasi sehingga user dapat menginput, mengedit, dan menghapus data pada Student.

Berikan analisa dari package dao, controller dan model serta jsp.

Source code diketik dengan font Courier New 10.

Sertakan screenshot dibawah analisa (singkat aja)

Screenshot wajib :

Halaman Login

Halaman Register

Halaman home

Fungsi add

Delete

Edit

Search

## TUGAS

* Memperbaiki yang error
* Mempercantik tampilan
* Menambah kolom “alamat” pada table student / menambahkan halaman profil (optional, jika dilakukan nilai tugas 100)
* Upload tugas ke github dan sertakan link

## KESIMPULAN

1. JPA digunakan untuk mengaloah dan mengatur data yang dibutuhkan untuk Java.
2. Entity class pada package “com.model” dalam percobaan digunakan untuk mengatur hubungan dengan textfield pada web serta memberikan tampilan.
3. Entity class pada package “com.dao” dalam percobaan digunakan untuk dihubungkan dengan database untuk mengatur set dan get sesuai isi yang dibutuhkan.
4. Servlet pada package “com.controller” dalam percobaan digunakan untuk dihubungkan dengan database dan mengatur koneksi pada aplikasi dan database.
5. Session bean pada percobaan menggunakan kondisi stateless sehingga ia tidak menyimpan state pada setiap kali eksekusi dilakukan.

KETENTUAN :

* ACC maksimal Jum’at, 12 November 2019
* Kelompok Genap ACC ke Agyan Atma (bitblast77@gmail.com) Line : agyanatma
* Kelompok Ganjil ACC ke Yogie Meysa Tama (yogiemt@student.ce.undip.ac.id) Line : yogie250598
* Laporan\_RSBK\_BAB IV\_Kelompok22
* Demo Tugas dilakukan sebelum AC