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

## Tujuan

1. Praktikan dapat menggunakan operasi CRUD dengan *database* MySQL
2. Praktikan dapat mengimplementasikan EJB
3. Praktikan dapat mengimplementasikan JPA
4. Praktikan dapat memahami penggunaan *session bean*

## Dasar Teori

## Java Persistence 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.

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.

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

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

## Session Bean

Merupakan pemodelan dari proses bisnis. *Enterprise* *bean* jenis ini mengerjakan sebuah aksi di dalam sistem, seperti mengaskses basis data, menghitung angka-angka, atau memanggil *enterprise* beans lain. *Session* *bean* digunakan untuk membuat *bean* yang memiliki bisnis proses per *session* saja. Dalam pengaksesannya tidak membutuhkan data yang tetap ada. *Session bean* hanya dapat memiliki satu klien pada saat yang bersamaan. Cara kerja dari *bean* jenis ini adalah per *session*, dimana hanya dapat memiliki satu klien per *session*. Saat klien mengalami terminasi, maka *session* *bean* tidak lagi terasosiasi dengan klien. *Session bean* memiliki dua tipe dalam pengelolaan hubungan dengan klien, yaitu *statefull* dan *stateless session bean*.

(Sumber: https://xb4mzx.wordpress.com/2011/02/12/enterprise-java-beans-ejb)

## Entity Unit

*Entity* *Bean* merepresentasikan suatu objek bisnis di dalam basisdata. *Entity bean* ini biasanya menjadi obyek yang dimanipulasi oleh *session beans*. *Entity bean* tidak ditujukan untuk dipanggil langsung dari aplikasi *client*. *Entity beans* adalah solusi EJB untuk menyimpan data ke *database*. Setiap *entity bean* berkorespondensi dengan sebuah tabel di database, dan setiap instance dari entity bean berkorespondensi dengan sebuah *record* pada tabel tersebut. *Entity bean* bersifat *persistent* (menetap /konstan /disimpan), dapat diakses oleh banyak klien, memiliki *primary* *keys*, dan dapat memiliki relasi dengan entity bean lainnya.

Entity bean digunakan jika memenuhi kondisi sebagai berikut:

1. Bean merupakan pengaksesan entitas basis data bukan sebagai prosedur proses bisnis.
2. Bean membutuhkan data yang bersifat persistent dan tersimpan dalam basis data.

(Sumber: https://xb4mzx.wordpress.com/2011/02/12/enterprise-java-beans-ejb)

## 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.

Proses kerja servlet :

1. Servlet diload ke JVM oleh Servlet *container* apabila terjadi *request* pertama kali oleh client.
2. Proses penanganan request dijalankan sebagai thread dari *web server* atau servlet *container*. Setelah diload maka servlet tetap ada di memori untuk menangani request berikutnya.
3. Tiap kali menangani request, servlet *container* membandingkan timestamp dari servlet dalam memori dengan *file* *class* java servlet. Apabila timestamp *file* java servlet ada yang lebih baru maka secara otomatis servlet *container* akan meload servlet yang baru dari *class* servlet.

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

## MySQL

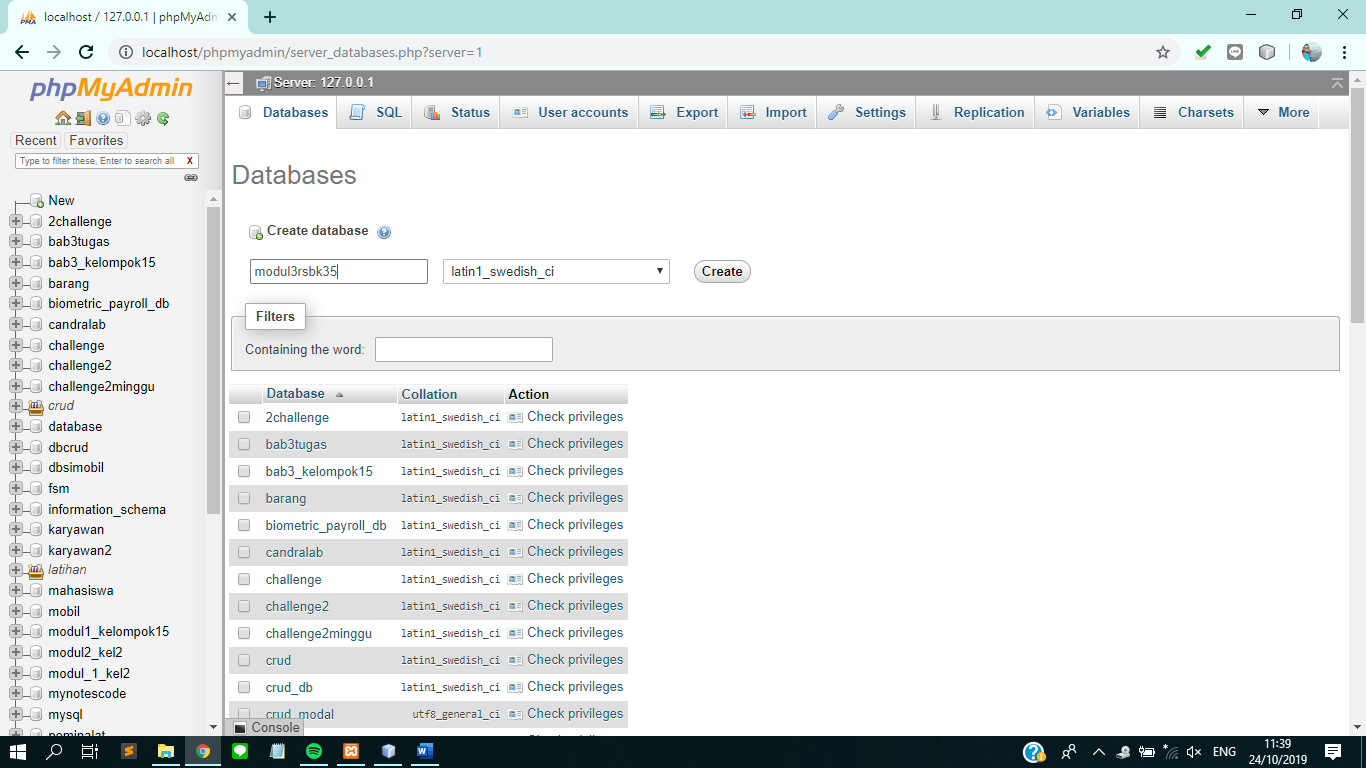
MySQL adalah sebuah perangkat lunak *system* manajemen basis data SQL (DBMS) yang *multithread*, dan multi-user. MySQL adalah implementasi dari system manajemen basisdata relasional (RDBMS). MySQL dibuah oleh TcX dan telah dipercaya mengelola system dengan 40 buah *database* berisi 10.000 tabel dan 500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

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

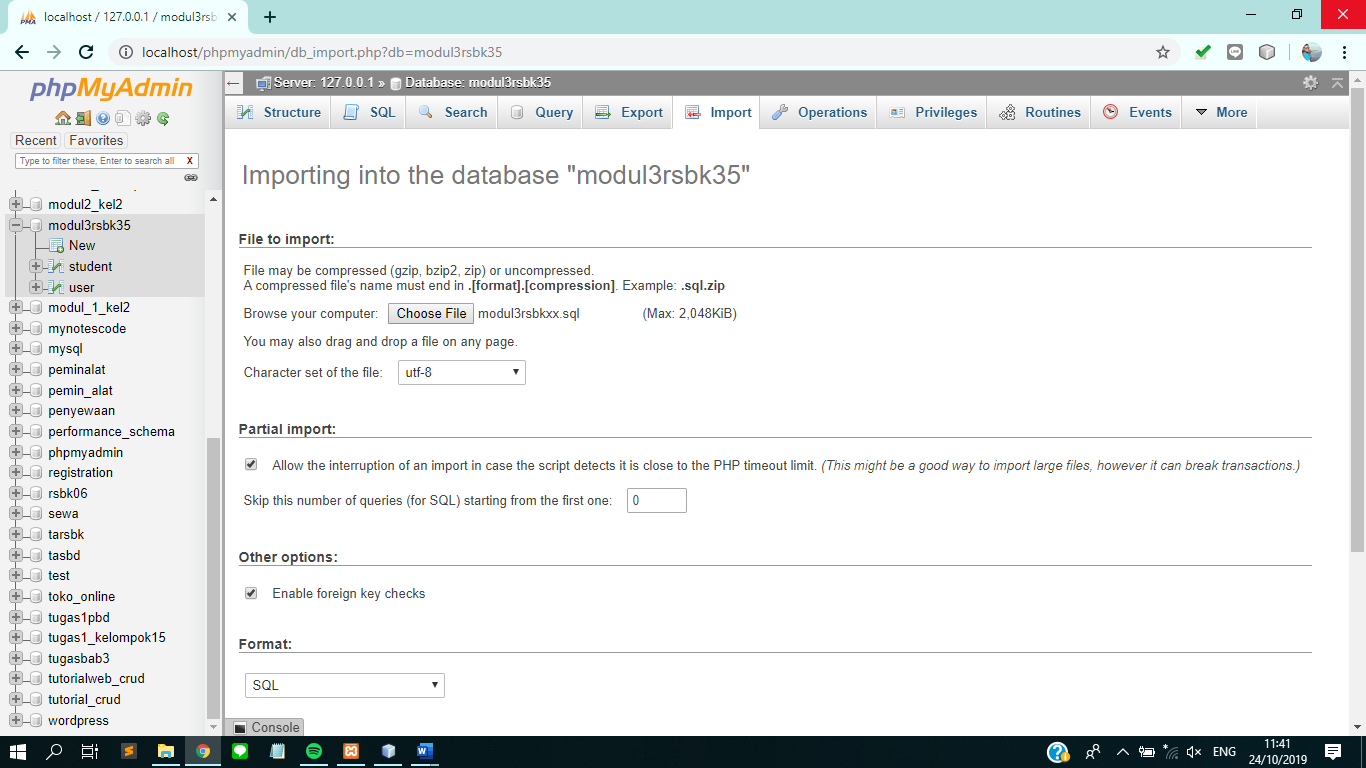
## Langkah Percobaan

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 “modul3rsbkxx” ganti xx dengan kelompok.



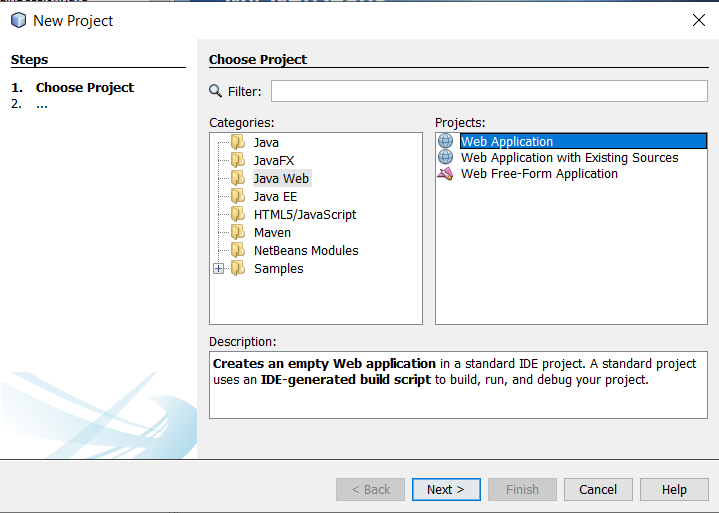
Gambar 4.1 Membuat database baru

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



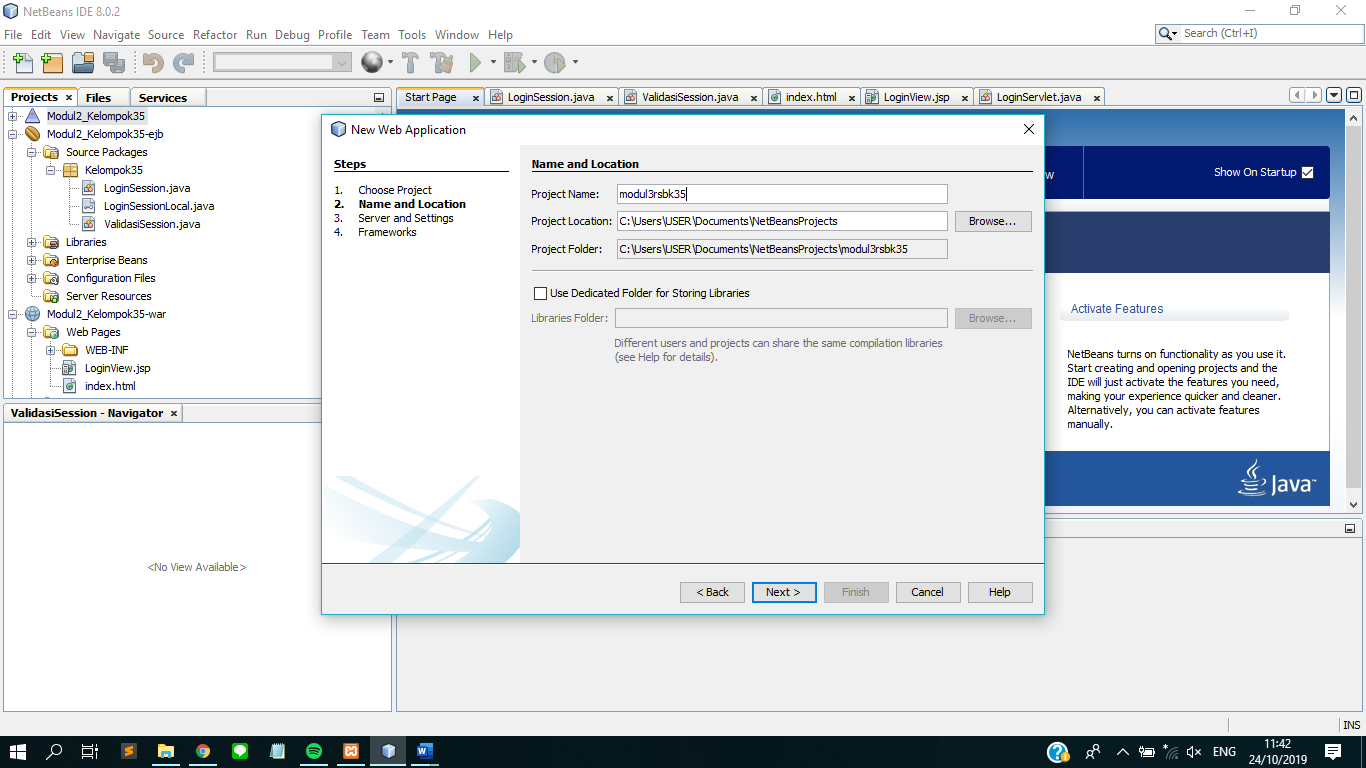
Gambar 4.2 Import database

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

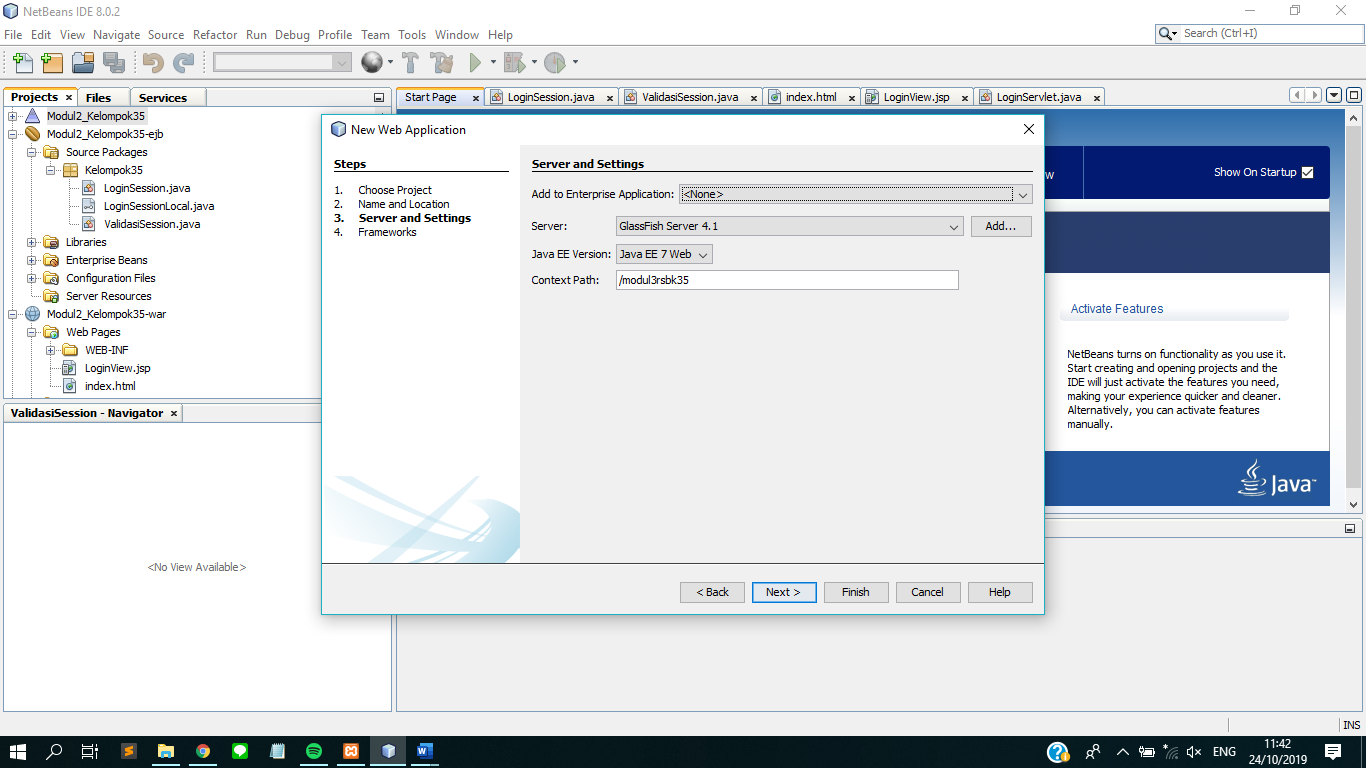


Gambar 4.3 New Project pada NetBeans

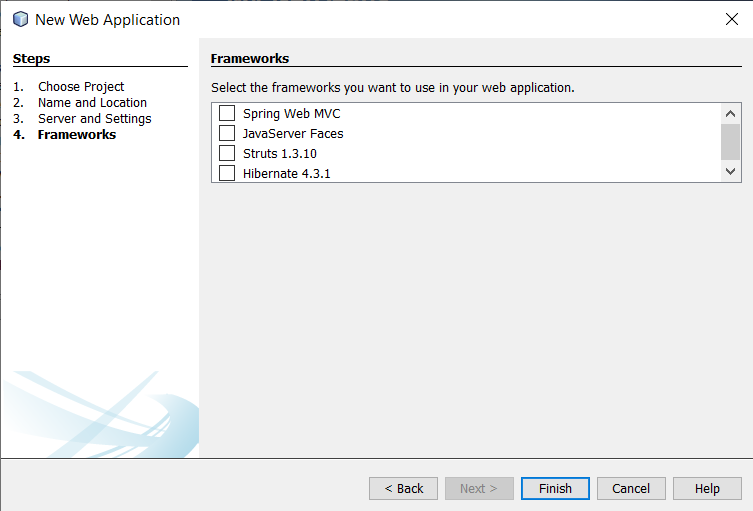
1. Beri nama project “**modul3rsbkxx**” dan pilih Glassfish Server (xx nomor kelompok).



Gambar 4.4 New Web Application pada NetBeans

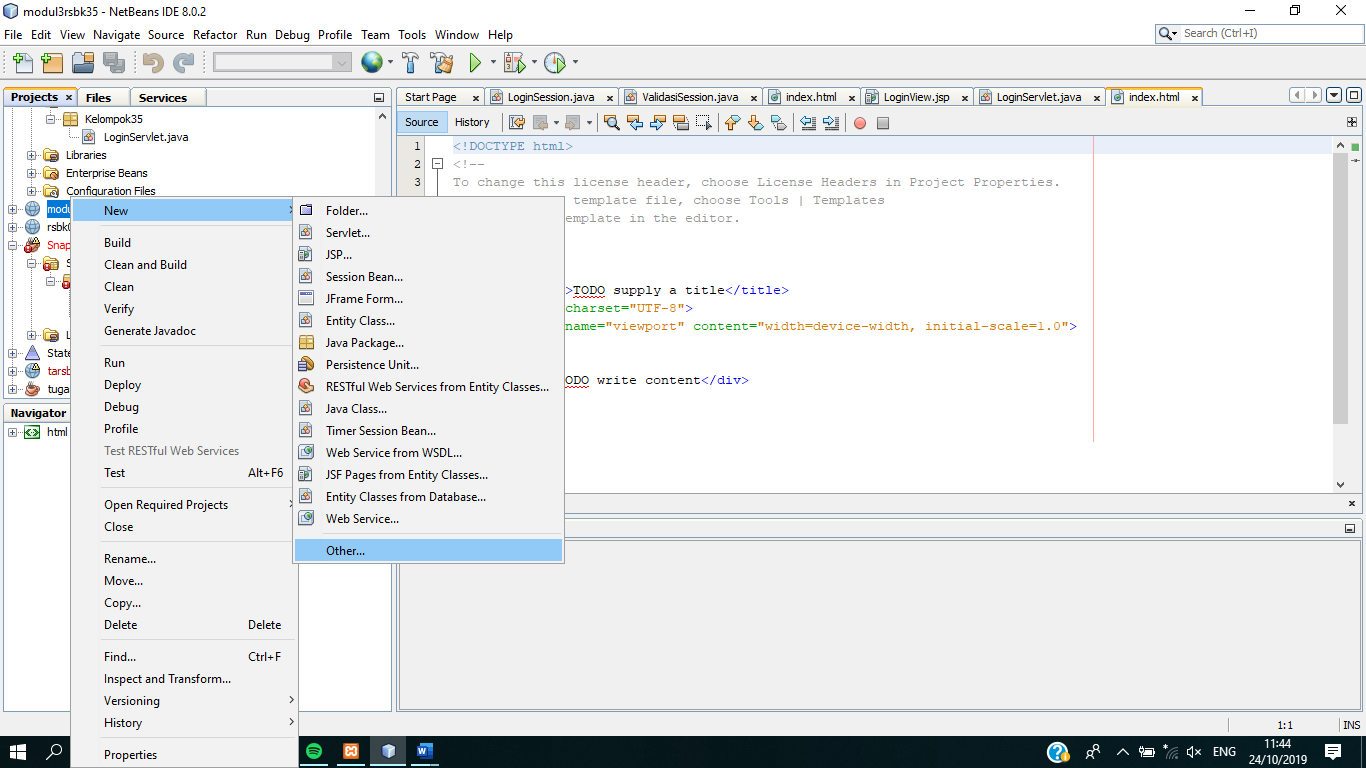


Gambar 4.5 Memilih GlassFish Server 4.1

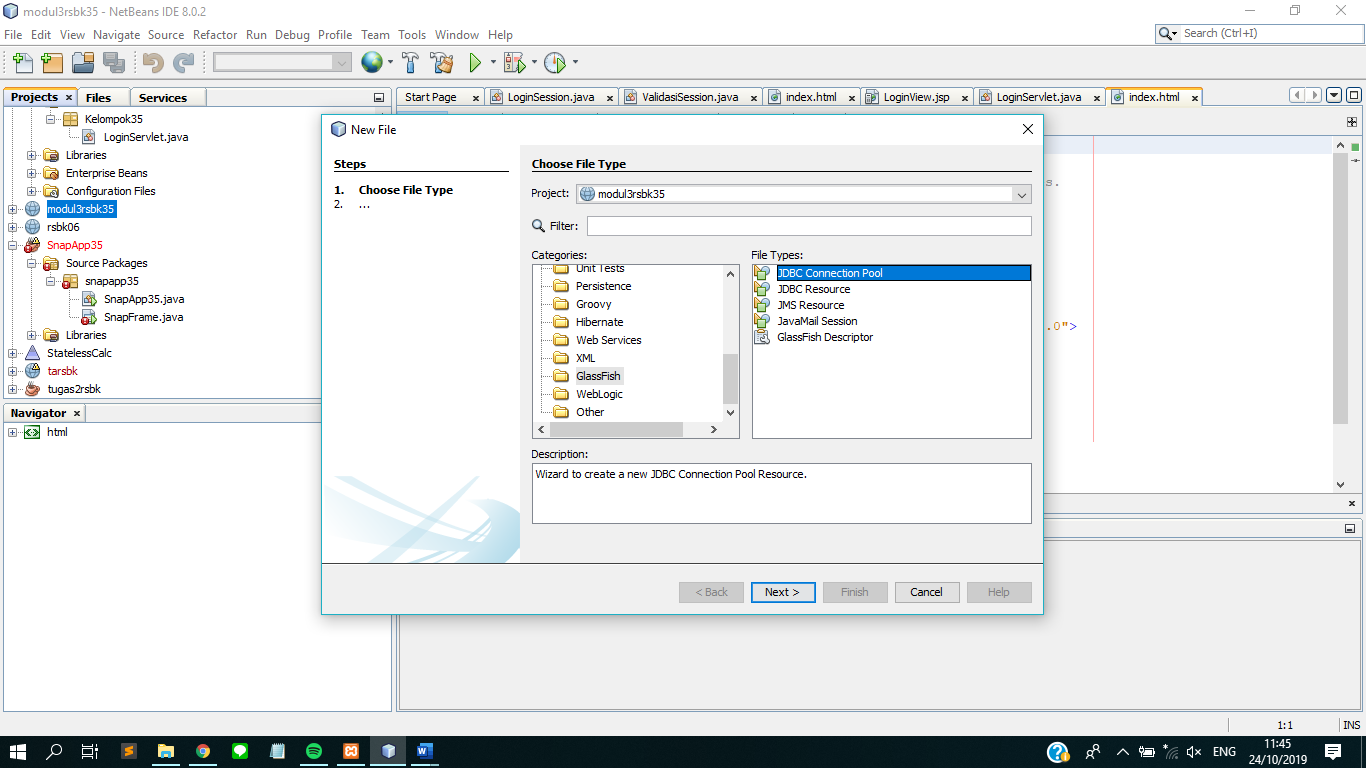
****

Gambar 4.6 Selesai membuat New Project

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

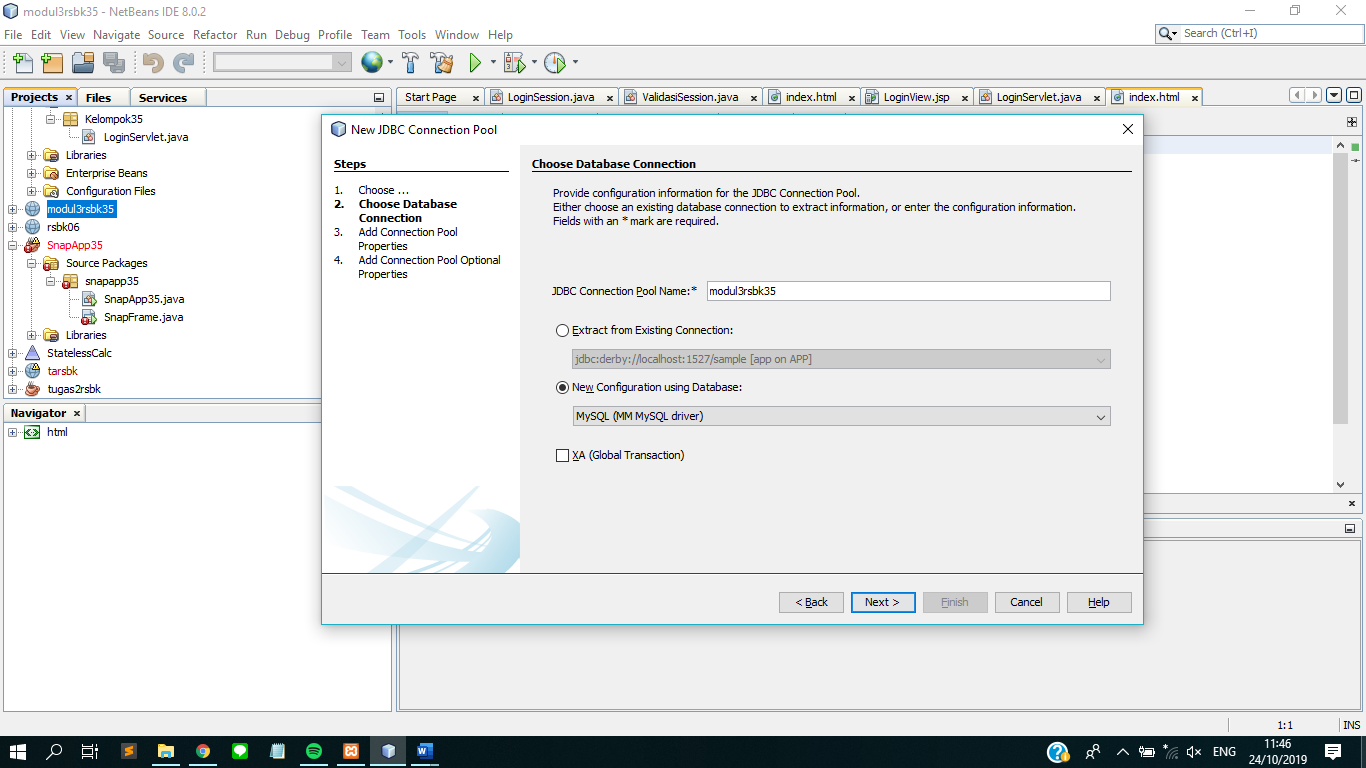


Gambar 4.7 Membuat JDBC Connection Pools



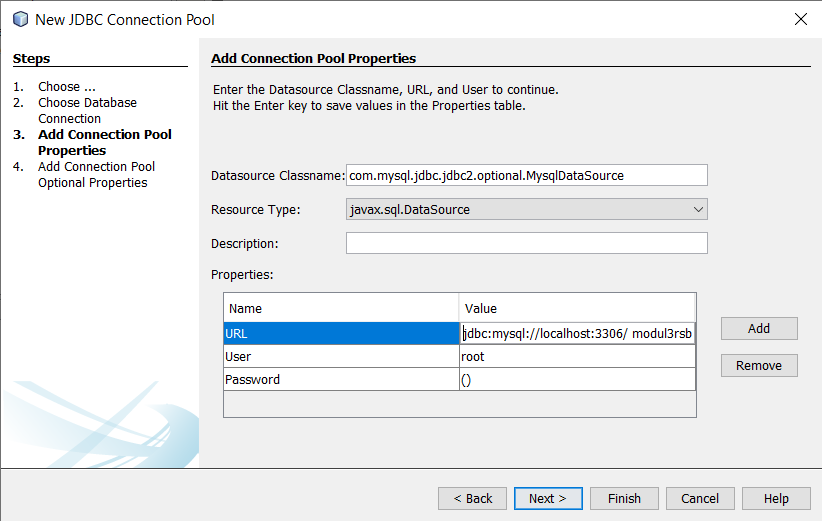
Gambar 4.8 Memilih GlassFish → JDBC Connection Pool

1. Beri nama connection pool (sama seperti nama projectnya, misal modul3rsbkxx) dan pilih ‘New Configuration using Database’ → ‘MySQL (MM MySQL driver).



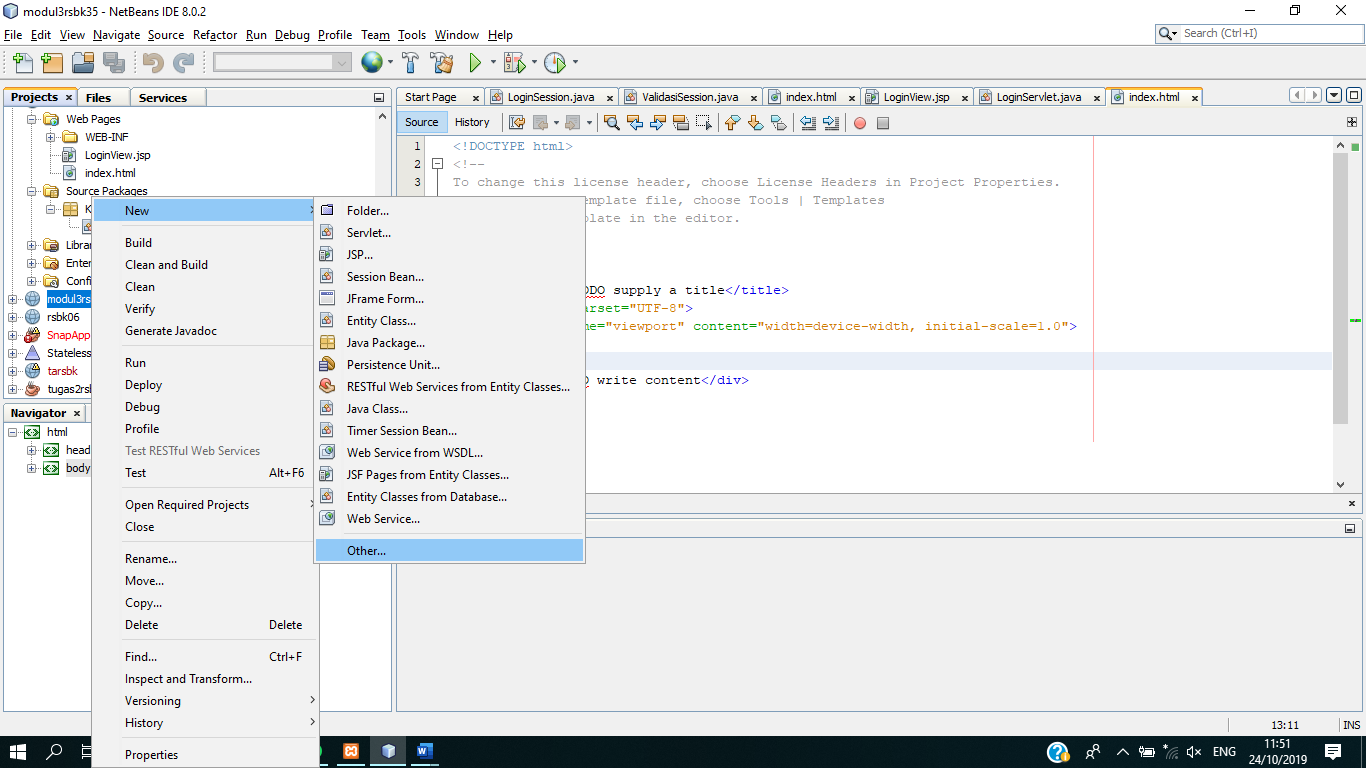
Gambar 4.9 Memberi nama pada JDBC Connection Pool

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 (missal **modul3rsbkxx**). Untuk user pakai **root** dan passwordnya **()**. Kemudian pilih Finish.

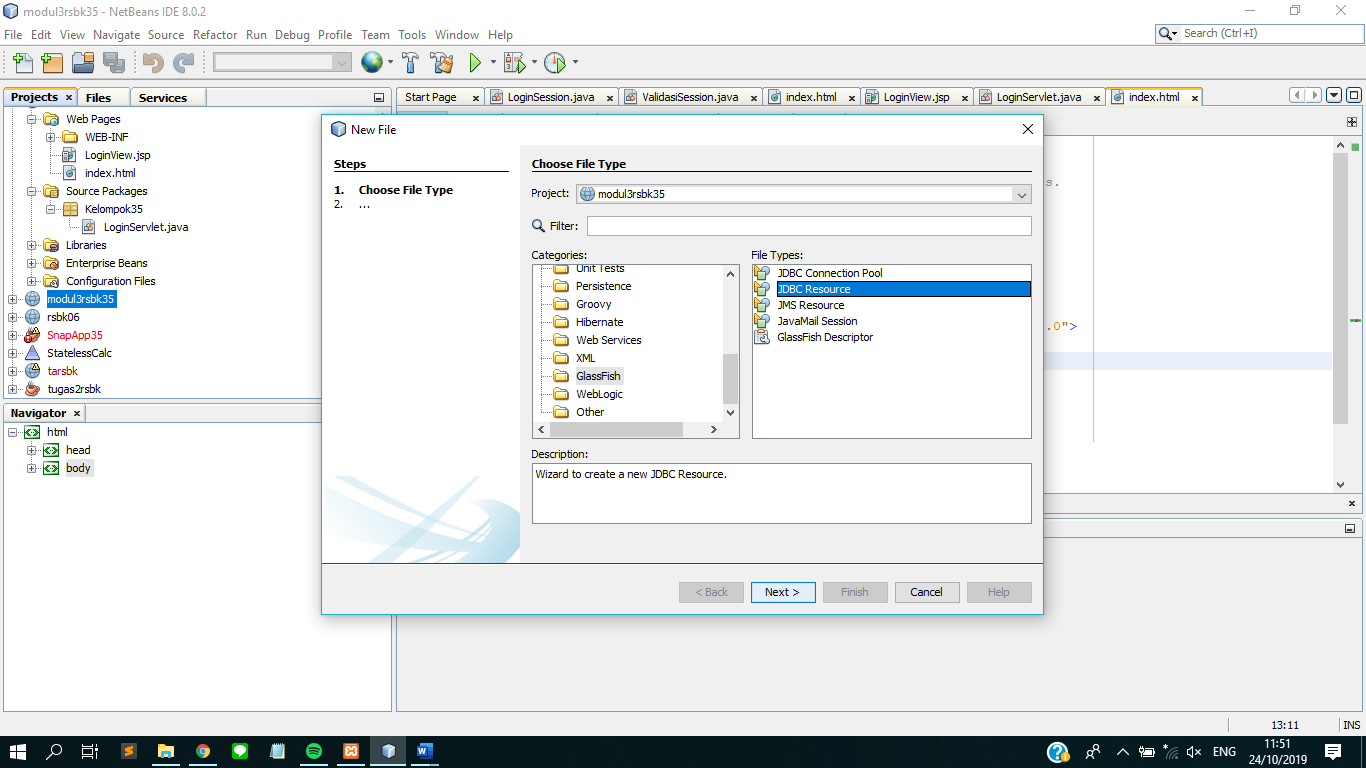


Gambar 4.10 Menambahkan properties JDBC Connection Pool

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

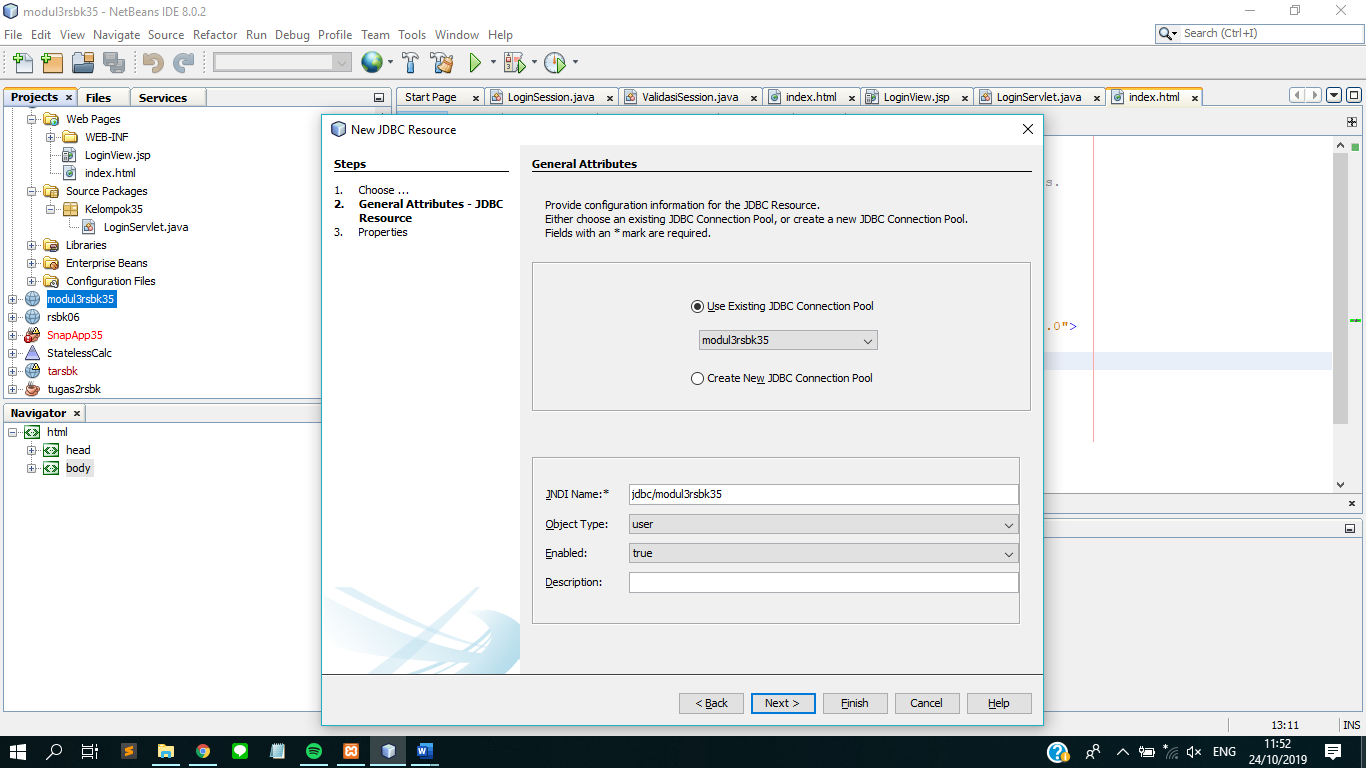


Gambar 4.11 Membuat JDBC Resource



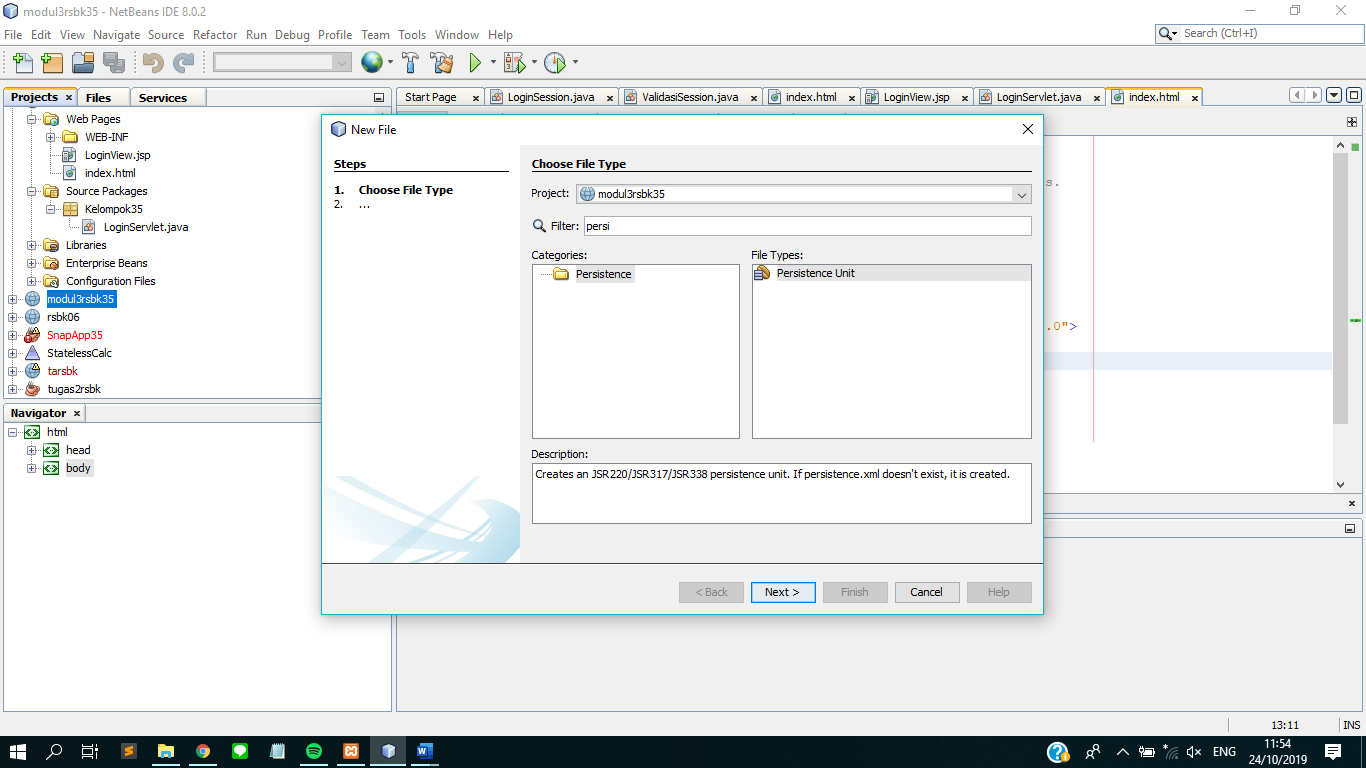
Gambar 4.12 Memilih GlassFish → JDBC Resource

1. Pilih ‘Use Existing JDBC Connection Pool’, pilih List JDBC yang baru kita buat tadi. Kemudian isikan JNDI Name seperti yang tadi. Misal modul3rsbkxx. Lalu pilih Finish.

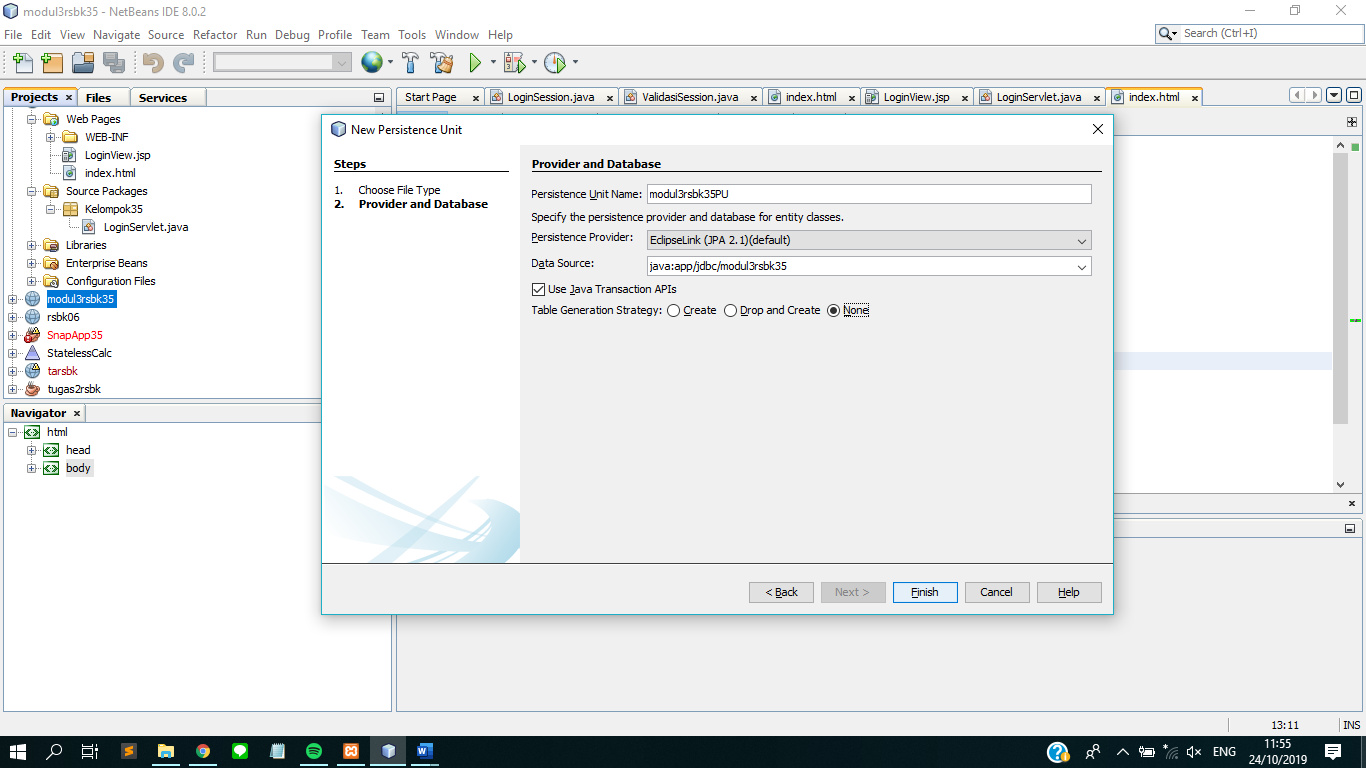


Gambar 4.13 General Attributes – JDBC Resource

1. 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’

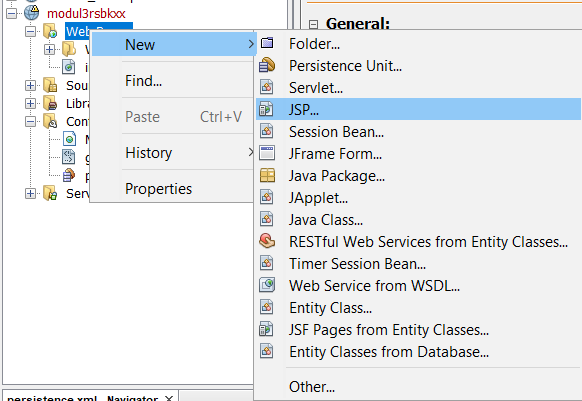


Gambar 4.14 Membuat Persistence Unit

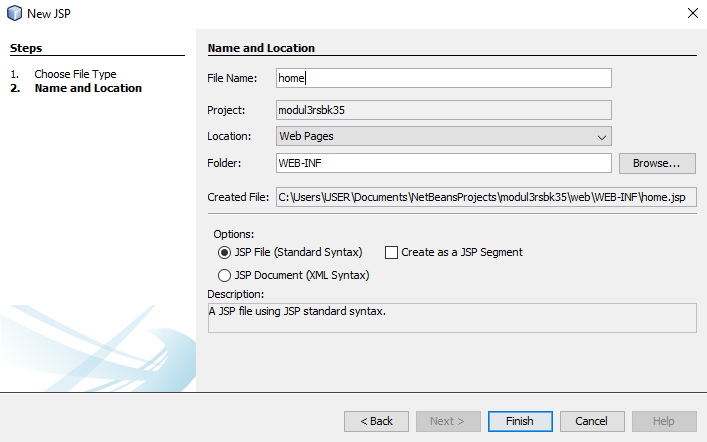


Gambar 4.15 Provider dan Database pada Persistence Unit

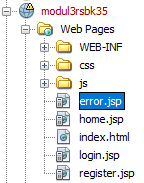
1. Buat 3 JSP page pada folder “Web Pages” dengan klik kanan folder lalu new file. JSP. Beri nama: home.jsp, register.jsp, login.jsp, dan error.jsp. Masukkan source code yang tersedia.



Gambar 4.16 Membuat JSP



Gambar 4.17 Memberi nama pada JSP



Gambar 4.18 File JSP 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 - Kelompokxx**</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 - Kelompokxx**</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> |

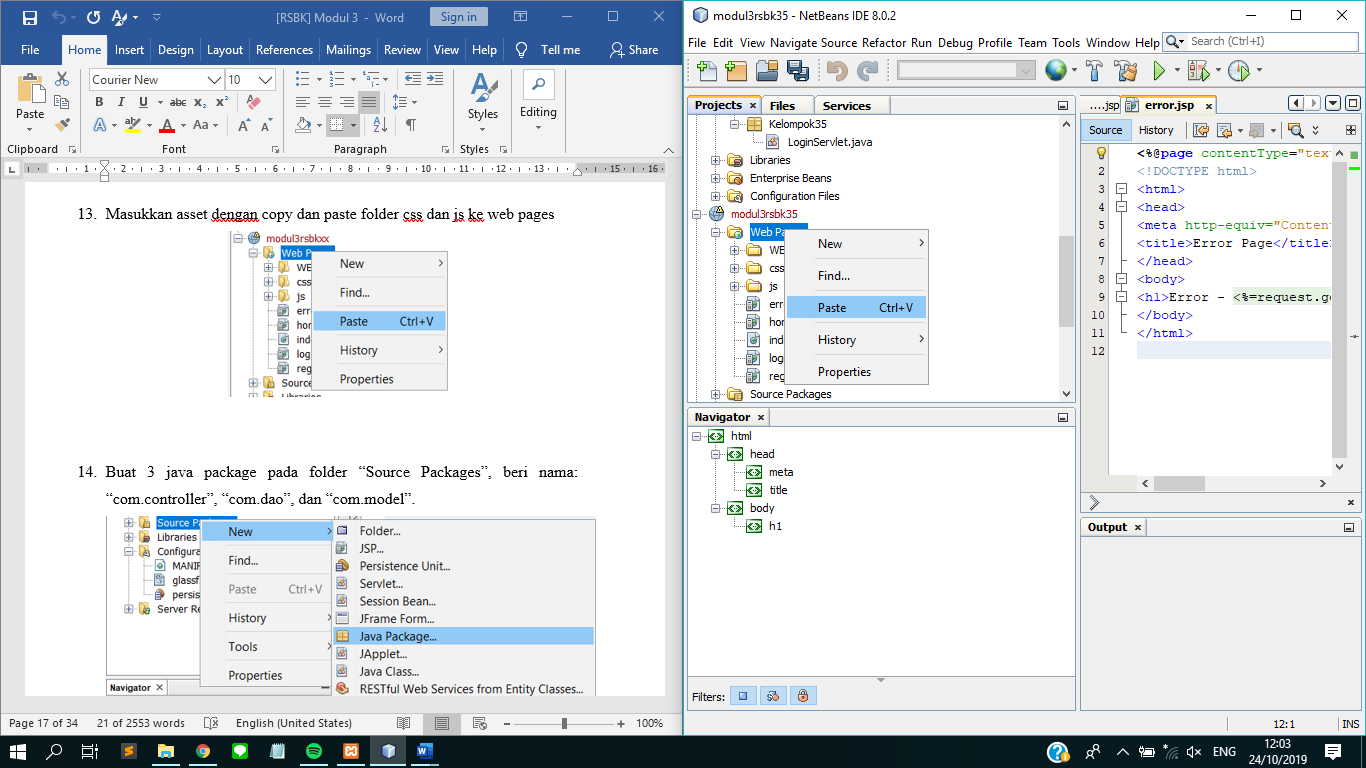
**home.jsp**

|  |
| --- |
| <%--  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 - Kelompokxx**</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**

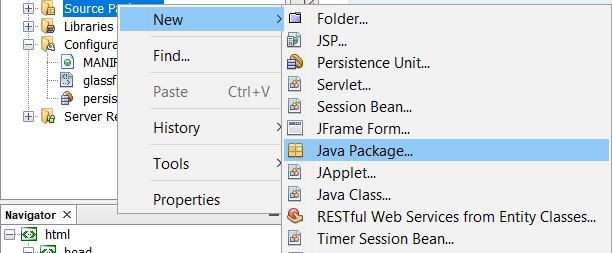
|  |
| --- |
| <%@page contentType="text/html" pageEncoding="UTF-8"%> |
| <!DOCTYPE html> |
| <html> |
| <head> |
| <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> |
| <title>Error Page</title> |
| </head> |
| <body> |
| <h1>Error - <%=request.getAttribute("error")%> </h1> |
| </body> |
| </html> |

1. Masukkan asset dengan copy dan paste folder css dan js ke web pages

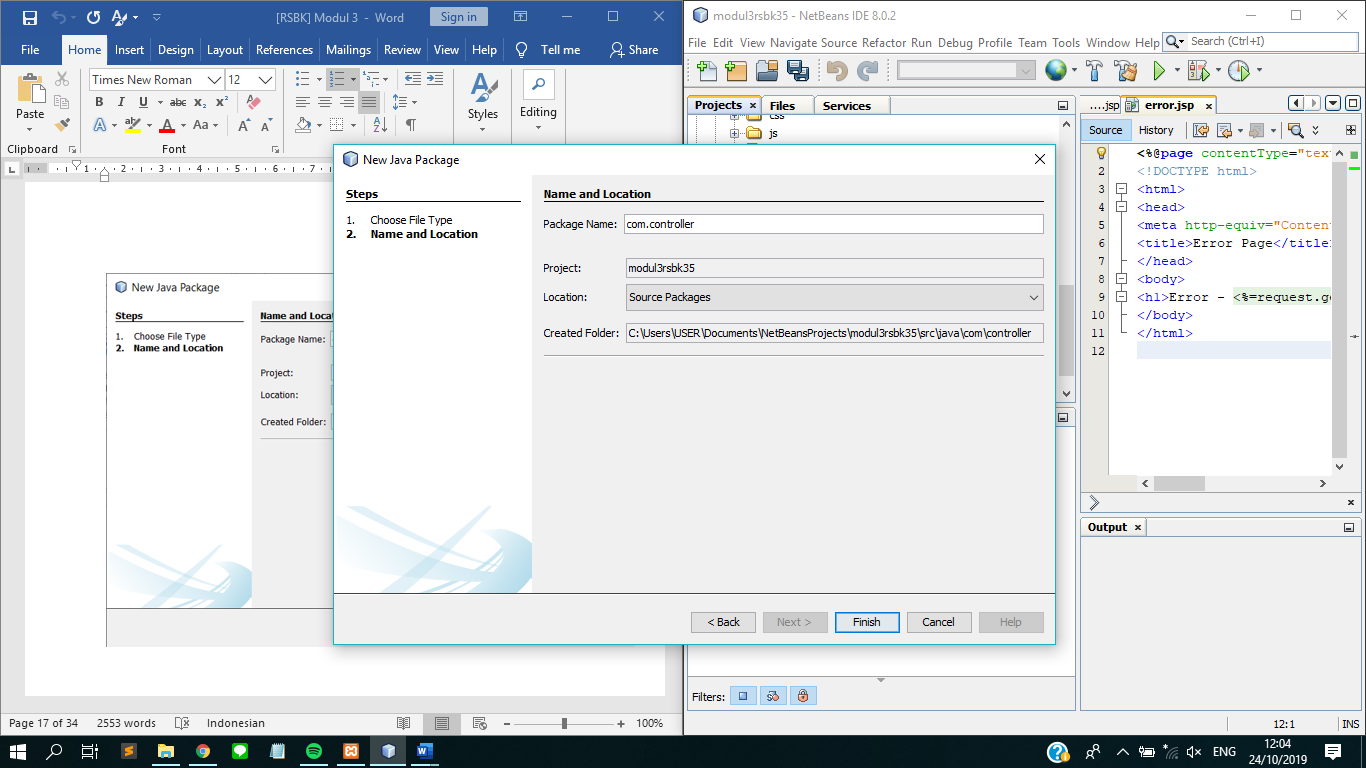


Gambar 4.19 Mem-paste folder css dan js

1. Buat 3 java package pada folder “Source Packages”, beri nama: “com.controller”, “com.dao”, dan “com.model”.

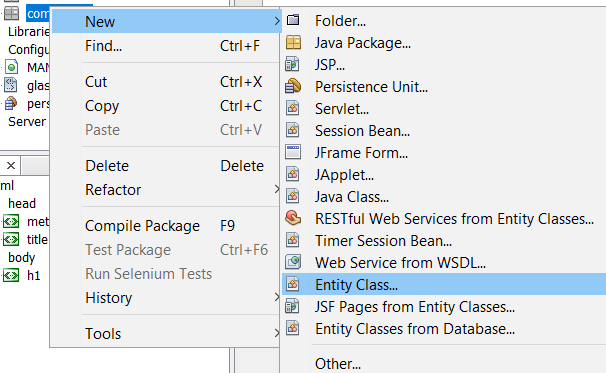


Gambar 4.20 Membuat Java Package

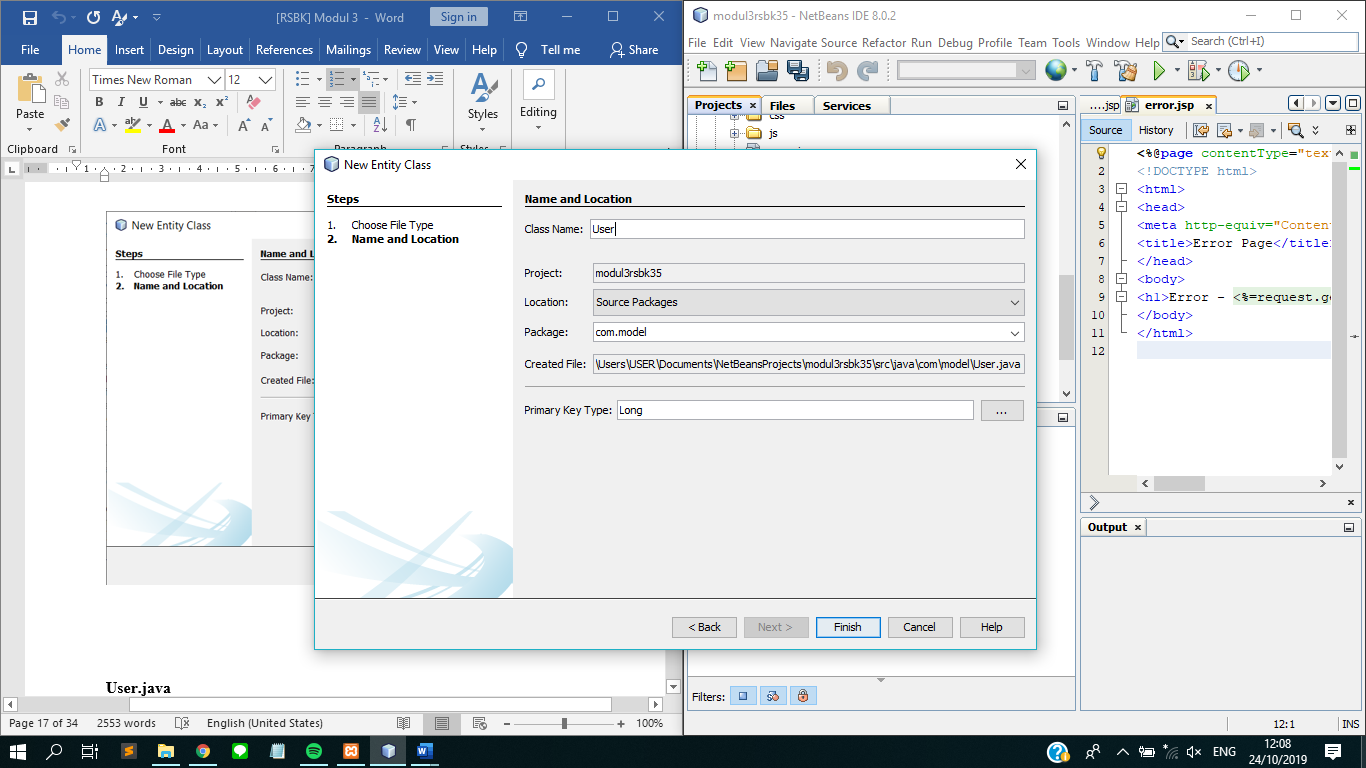


Gambar 4.21 Memberi nama Java Package

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



Gambar 4.22 Membuat Entity Class



Gambar 4.23 Memberi nama Entity Class

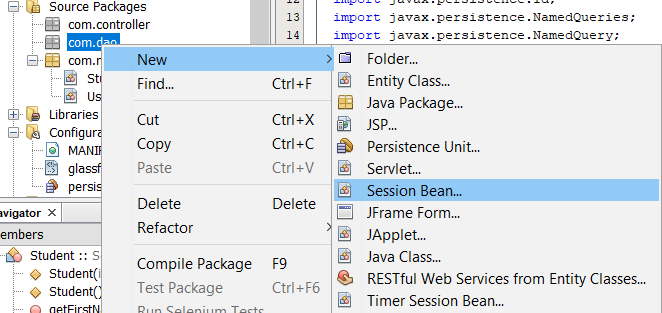
**User.java**

|  |
| --- |
| /\*  \* 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;  }  } |

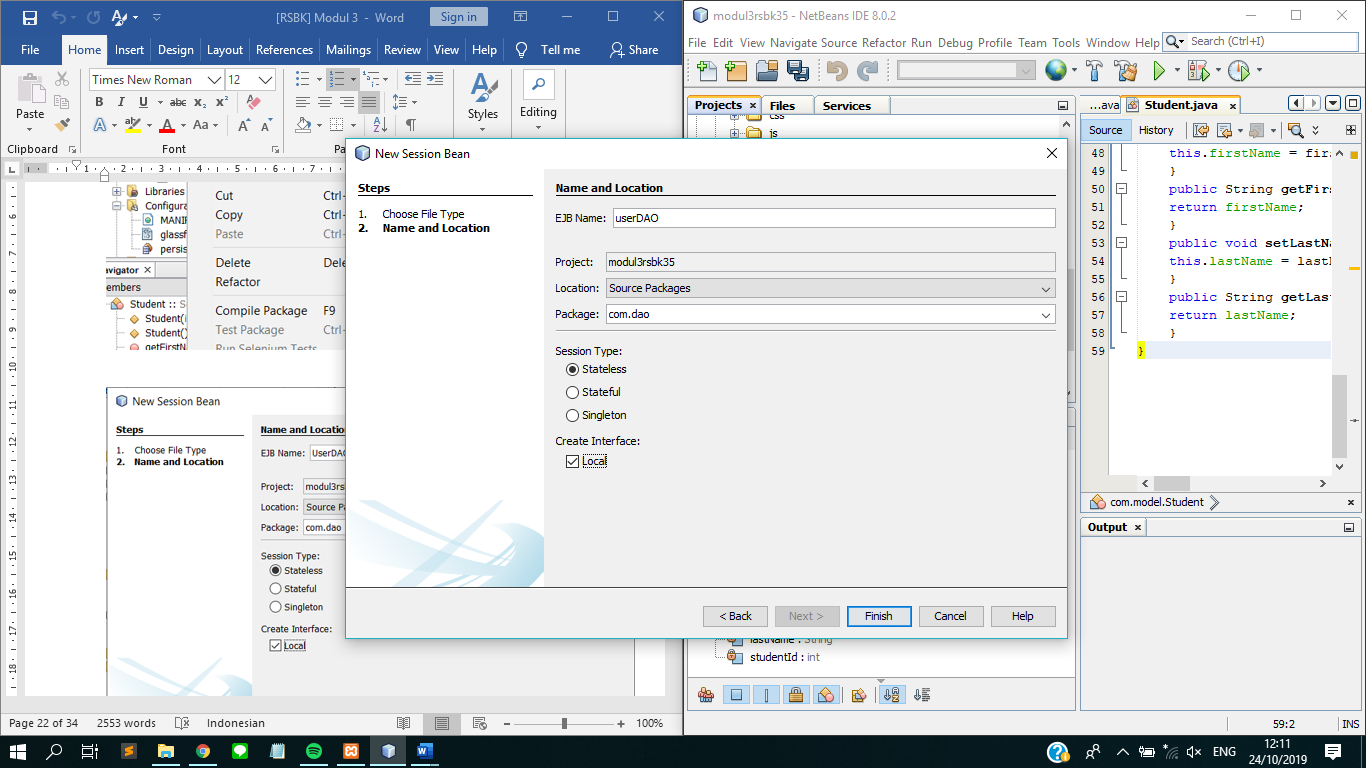
**Student.java**

|  |
| --- |
| /\*  \* 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 4.24 New →Session Beans



Gambar 4.25 Membuat Session Bean

**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);  } |

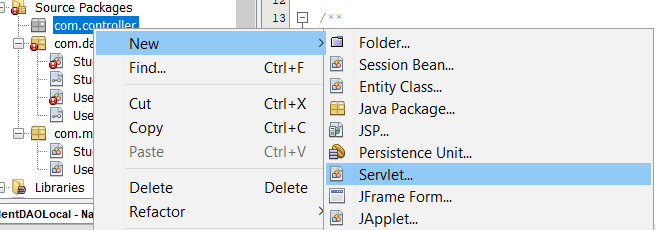
**StudentDAO.java**

|  |
| --- |
| /\*  \* 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();  }  } |

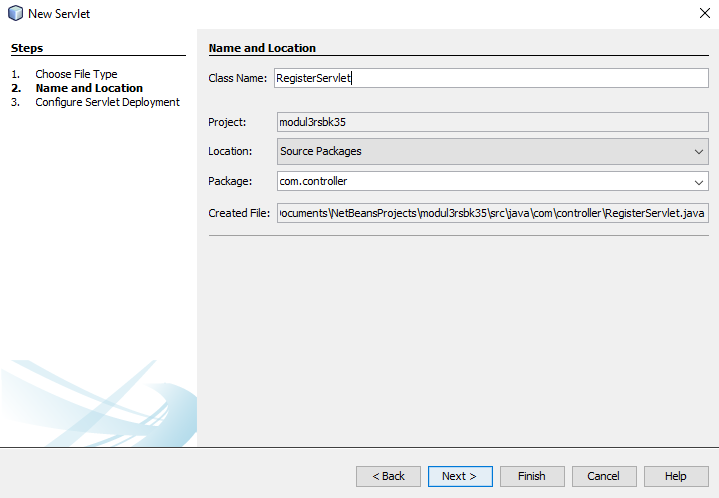
**StudentDAOLocal.java**

|  |
| --- |
| /\*  \* 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();  } |

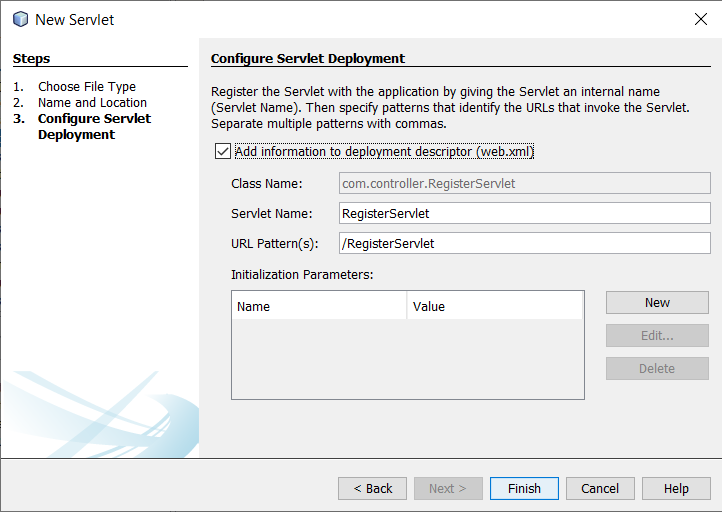
1. Buat 3 Servlet pada package “com.controller”, beri nama : “UserServlet”, “RegisterServlet”, “StudentServlet”. Ketika membuat servlet, centang pada ‘Add information to deployment descriptor (web.xml)’. Kemudian masukkan source code yang tersedia.



Gambar 4.26 New →Servlet



Gambar 4.27 Memberi nama Servlet



Gambar 4.28 Configure Servlet Deployment

**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>  } |

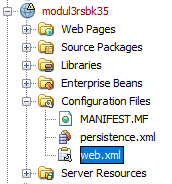
**RegisterServlet.java**

|  |
| --- |
| /\*  \* 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>  } |

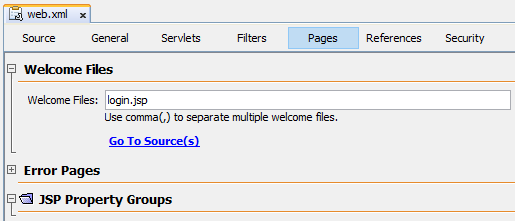
**StudentServlet.java**

|  |
| --- |
| /\*  \* 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 4.29 Configuration Files → web.xml

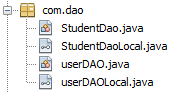


Gambar 4.30 Mengisi Welcome Pages dengan login.jsp

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

## Analisa Percobaan

1. **Dao**



Gambar 4.31 Package com.dao

*Folder* *package* com.dao terdiri dari 4 *file session bean*. Empat buah *file* tersebut yaitu StudentDAO.java, StudentDAOLocal.java, UserDAO.java, dan UserDAOLocal.java. *File-file* tersebut digunakan untuk proses dalam melakukan *login, register* dan juga berisi *query* CRUD yang akan digunakan untuk memanipulasi data.

**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")  } |

Pada file UserDAO terdapat beberapa *listing* *code* yang akan menjalankan fungsi untuk melalukan *log in* yang mana *username* dan *password* akan dicocokan dengan data yang ada pada *database* modul3kel35. Terdapat pula *method* addUser untuk menambahkan *user* baru *database* modul3kel35.

**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);  } |

Pada *file* UserDAOLocal.java terdapat 2 *method* dengan nama addUser dan juga checkUser. *Method* checkUser memiliki parameter *username* dan *password*. Parameter tersebut akan divalidasi oleh UserDAO.java.

**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();  }  } |

Pada file StudentDAO.java terdapat beberapa *code* yang digunakan untuk memanggil data pada tabel *student* yang kemudian akan ditampilkan pada halaman *home*. Data tersebut dapat dimanipulasi sesuai dengan keinginan *user*.

**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();  } |

StudentDAOLocal.java mendeklarasikan variabel yang berfungsi untuk menyimpan data atau nilai yang akan diproses oleh StudentDAO.java.

1. **Controller**



Gambar 4.32 Package com.controller

*Controller* merupakan bagian yang diimplementasikan menggunakan servlet yang berfungsi untuk mengatur aliran *request-response* dan penghubung antara *model* dan *view*. Terdapat 3 controller yaitu LoginServlet.java, RegistersServlet.java dan StudentServlet.java.

**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>  } |

LoginServlet.java berfungsi untuk menghubungkan login.jsp dan DAO-nya yang digunakan untuk *login* yaitu UserDAO.java. LoginServlet akan menerima dan merespon jika ada *request* baik dari login.jsp maupun DAO-nya.

**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>  } |

RegistersServlet.java berfungsi untuk menghubungkan register.jsp dan DAO-nya yang digunakan untuk *register* yaitu UserDAO.java. RegistersServlet akan menerima dan merespon jika ada *request* baik dari register.jsp maupun DAO-nya.

**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>  } |

StudentServlet.java berfungsi untuk menghubungkan student.jsp dan DAOnya yang digunakan untuk *login* yaitu StudentDAO.java. StudentServlet akan menerima dan merespon jika ada request baik dari Student.jsp maupun DAOnya

1. **Model**



Gambar 4.33 Package com.model

File Model yang berisi method-method untuk mengambil data serta memberi nilai dari data. Terdapat 2 model yaitu User.java dan Student.java.

**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;  }  } |

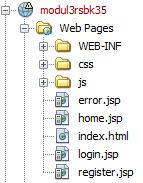
User.java berfungsi untuk memberi nilai pada data yang nantinya akan dihubungkan ke database tabel *login* dengan *view* login.jsp.

**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;  }  } |

Student.java berfungsi untuk memberi nilai pada data yang nantinya akan dihubungkan ke database tabel *student* dengan *view* student.jsp

1. JSP



Gambar 4.34 File-file JSP

JSP merupakan file yang digunakan untuk membuat tampilan halaman web. Terdapat 4 file JSP yaitu login.jsp, home.jsp, register.jsp dan error.jsp. Login adalah tampilan halaman ketika *login* menggunakan *username* dan *password*. *Home* adalah tampilan halaman setelah *login* yang digunakan untuk memanipulasi data. *Register* adalah halaman yang digunakan pengguna baru untuk mendaftar menggunakan *username* dan *password*. *Error* sebagai tampilan halaman *error* misal ketika salah memasukkan data.

**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 – Kelompok 35**</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> |

Register.jsp adalah tampilan halaman *register* untuk menambah user baru dengan cara mengisikan *username* dan *password* yang nantinya data atau nilai yang dimasukkan akan diambil dan dikoreksi oleh DAO dan model yang dihubungkan oleh *controller*.

**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 – Kelompok 35**</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> |

Login.jsp adalah tampilan atau *view* halaman *login* dengan fungsi mengisikan *username* dan *password* yang nantinya data atau nilai yang dimasukkan akan diambil dan dikoreksi oleh DAO dan model yang dihubungkan oleh *controller*.

**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 – Kelompok 35**</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> |

Home.jsp adalah tampilan atau *view* halaman *home* dengan fungsi menampilkan tabel *student*, *texfield* dan *button*-*button* untuk memanipulasi data pada tabel *student* yang nantinya data atau nilai yang dimasukkan akan diambil dan dikoreksi oleh DAO dan model yang dihubungkan oleh *controller*.

**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>

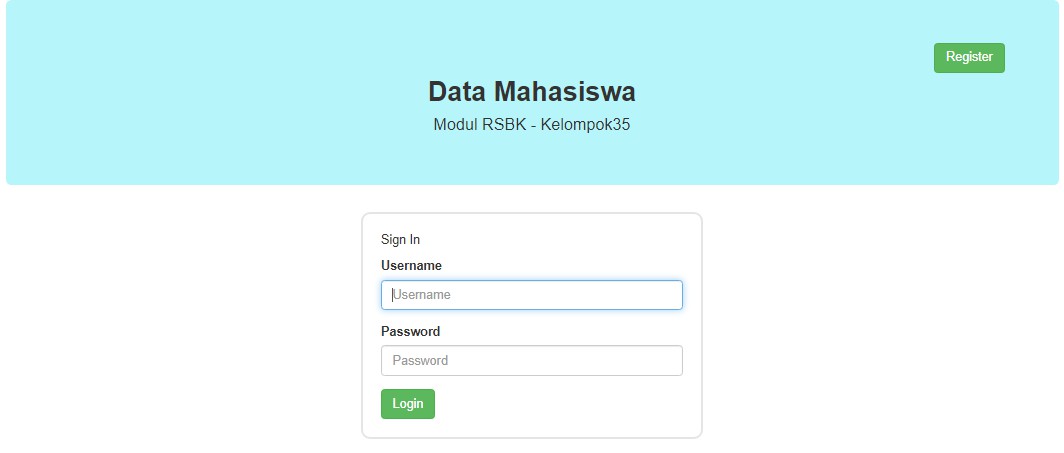
</body>

</html>

Error.jsp adalah tampilan halaman yang menampilkan pesan *error* apabila ada kesalahan dalam transaksi.

1. **Tampilan Halaman Aplikasi**

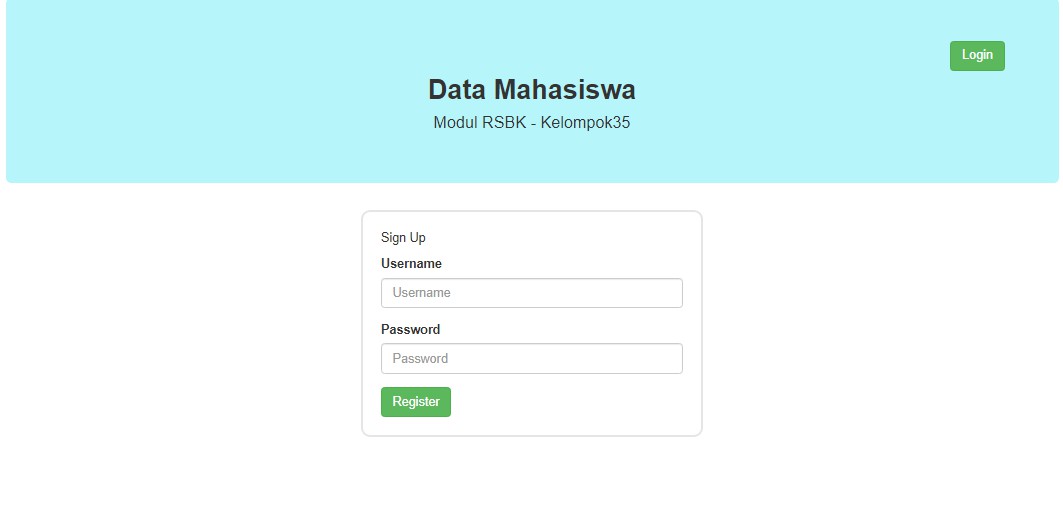
* Halaman Log in



Gambar 4.35 Halaman Log In

Gambar di atas adalah halaman log in yang telah dibuat. Pada halaman login.jsp tersebut terdapat 2 textfield yang digunakan untuk memasukan username dan password. Terdapat juga button login yang ketika ditekan akan melakukan fungsi login dan beralih ke halaman berikutnya. Di pojok kanan atas terdapat button register yang jika ditekan akan masuk ke halaman register.jsp.

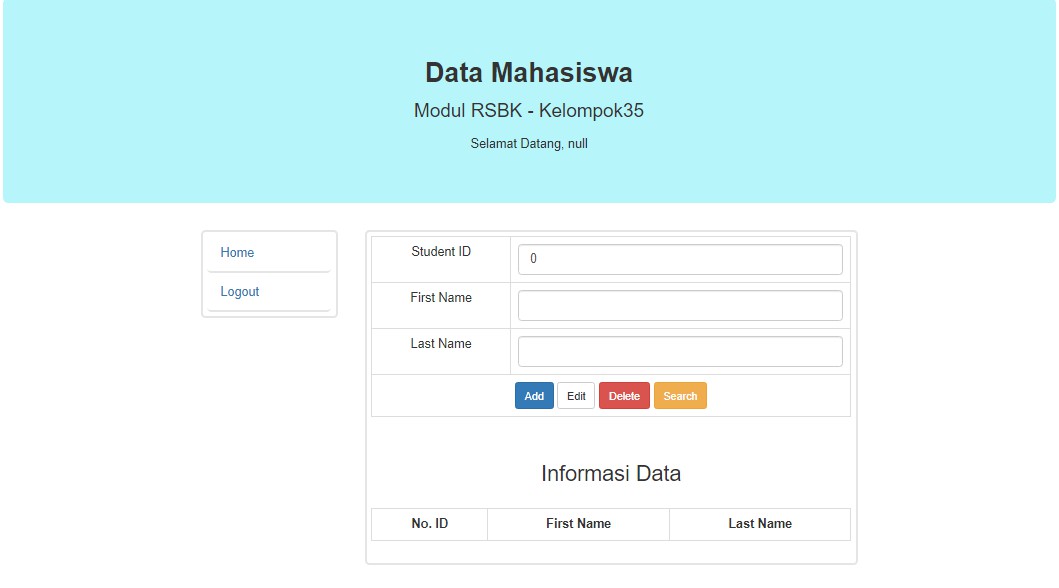
* Halaman Register



Gambar 4.36 Halaman Register

Gambar di atas adalah halaman register yang telah dibuat. Pada halaman register.jsp tersebut terdapat 2 textfield yang digunakan untuk memasukan username dan password pengguna yang akan mendaftar kemudian akan dimasukan ke database. Terdapat juga button register yang ketika ditekan akan melakukan fungsi register dan beralih ke halaman berikutnya. Di pojok kanan atas terdapat button login yang jika ditekan akan masuk ke halaman login.jsp

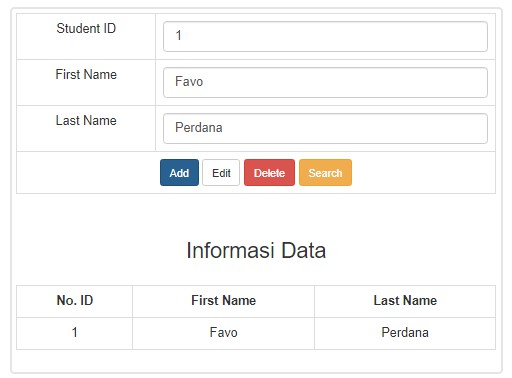
* Halaman home



Gambar 4.37 Halaman Home

Berisi informasi mengenai entitas student dengan atribut StudentID, firstName, lastName, dan alamat (tambahan tugas). Fungsi yang disediakan yaitu Add, Edit, Delete, dan Search. Method dari fungsi tersebut ada di kelas StudentDAO, dengan menggunakan em (*method* dari *library* EntityManager) yang menghubungkan ke database. Misal, pada *method* deleteStudent akan menjalankan em.remove(getStudent(studentId));.

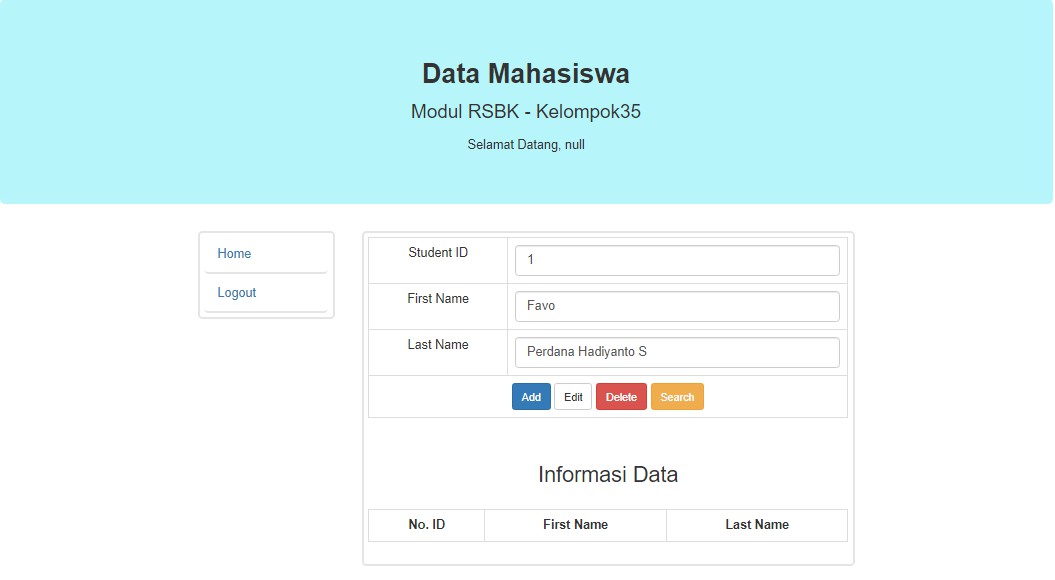
* Add



Gambar 4.38 Fungsi Add

Menggunakan method addStudent(), yang menjalankan operasi em.merge(student); dan em.flush();. Operasi em.merge akan menyatukan perubahan yang terjadi pada entitas student, dan em.flush akan menyimpan semua perubahan yang terjadi (seperti melakukan *commit*).

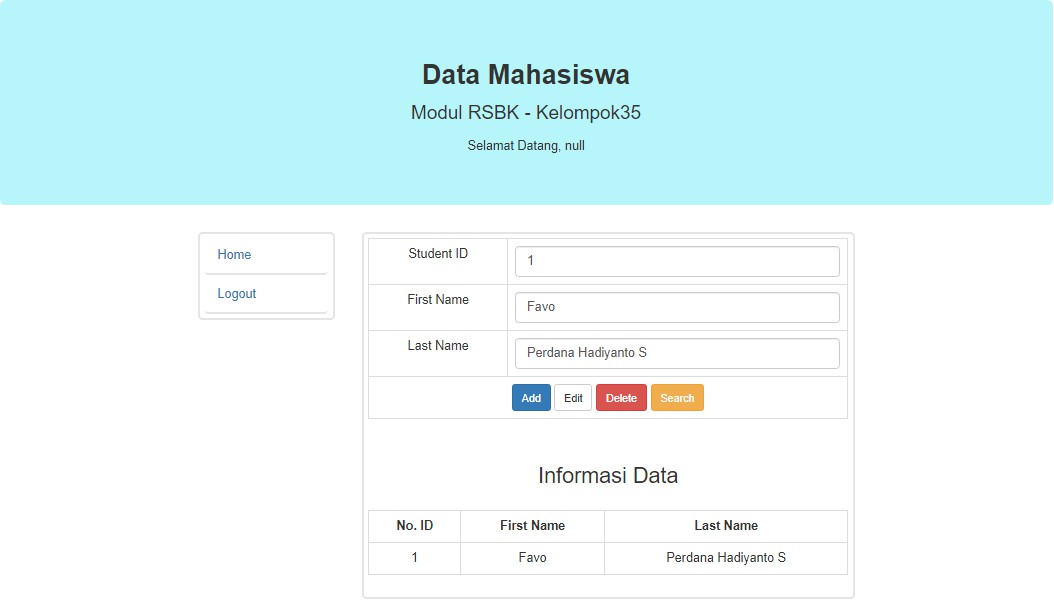
* Delete



Gambar 4.39 Fungsi Delete

Menggunakan method deleteStudent() dengan parameter studentId. Di dalam *method* tersebut akan menjalankan operasi em.remove(getStudent(studentId)); dan em.flush();. Operasi em.remove digunkaan untuk menghapus objek dari database, dan em.flush akan menyimpan semua perubahan yang terjadi (seperti melakukan *commit*).

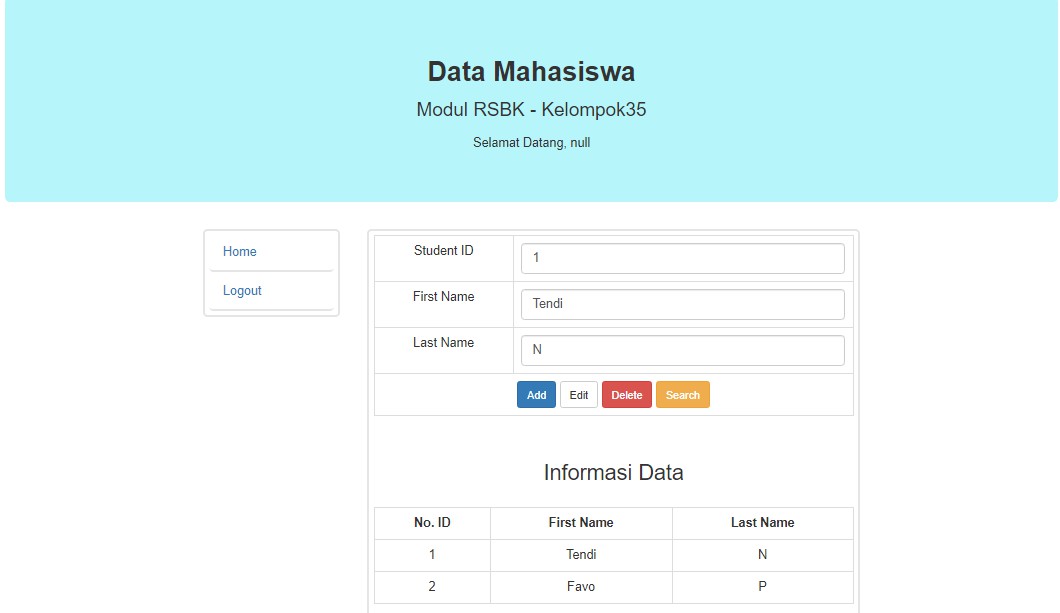
* Edit



Gambar 4.40 Fungsi Delete

Menggunakan *method* editStudent(), dengan operasi yang sama dengan fungsi Add yaitu merge dan flush, karena sama-sama menyimpan perubahan objek pada database.

* Search

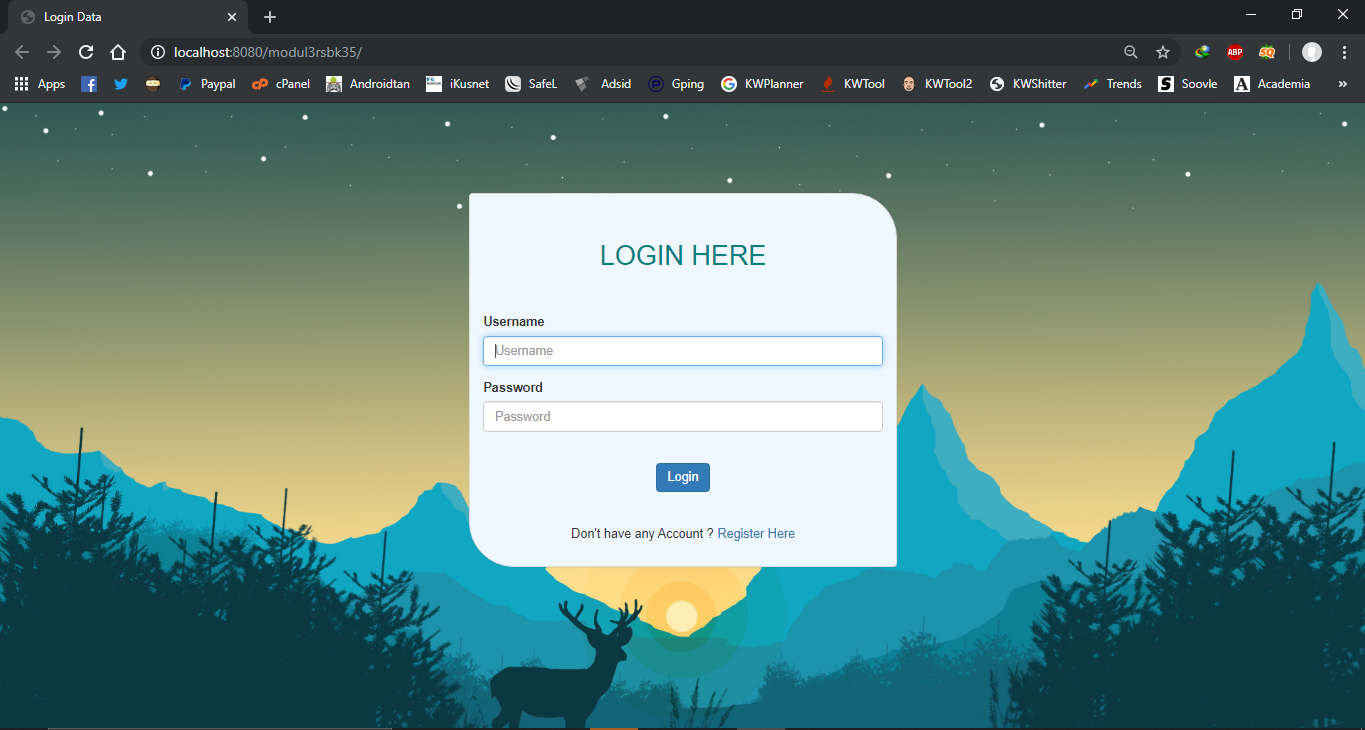


Gambar 4.41 Fungsi Search

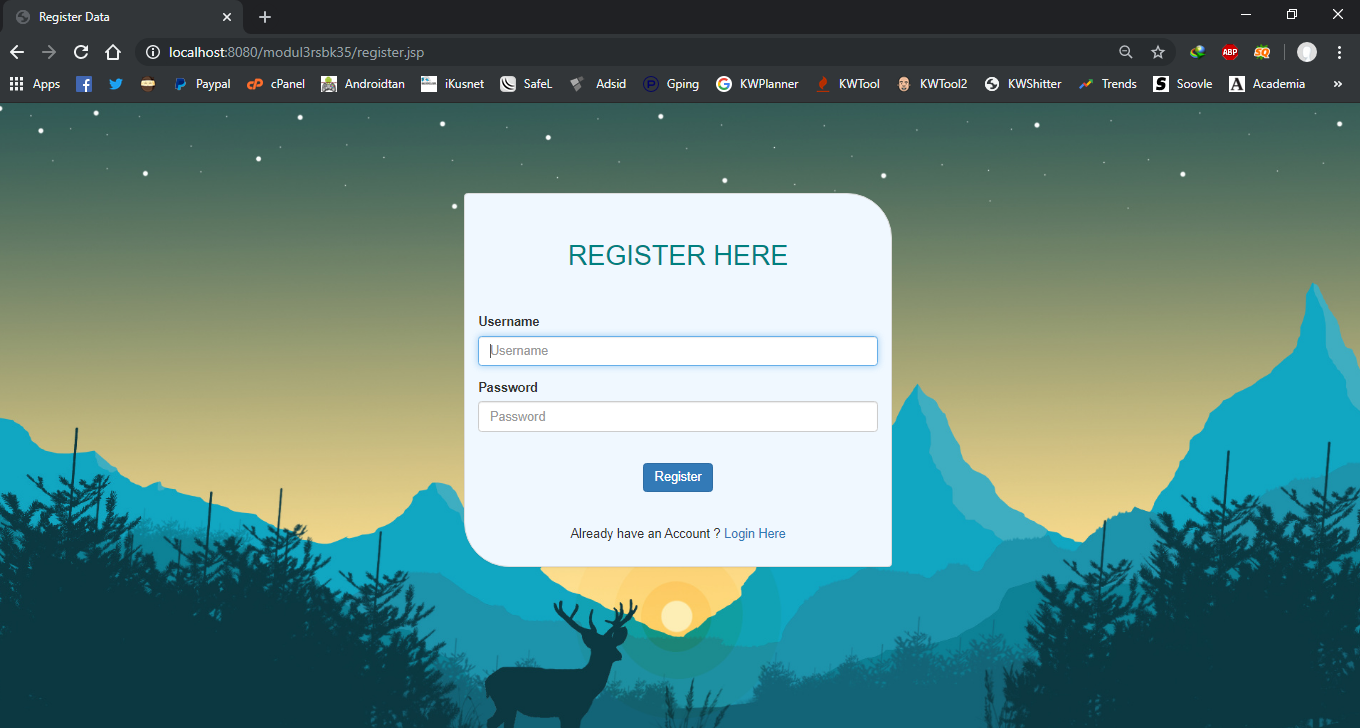
Menggunakan *method* getStudentd() dengan parameter studentId. Dalam method tersebut akan menjalankan operasi em.flush yang akan men-trigger database, kemudian menjalankan operasi return em.find(Student.class, studentId); yang akan mencari studentId yang sesuai pada kelas Student.java.

## Tugas

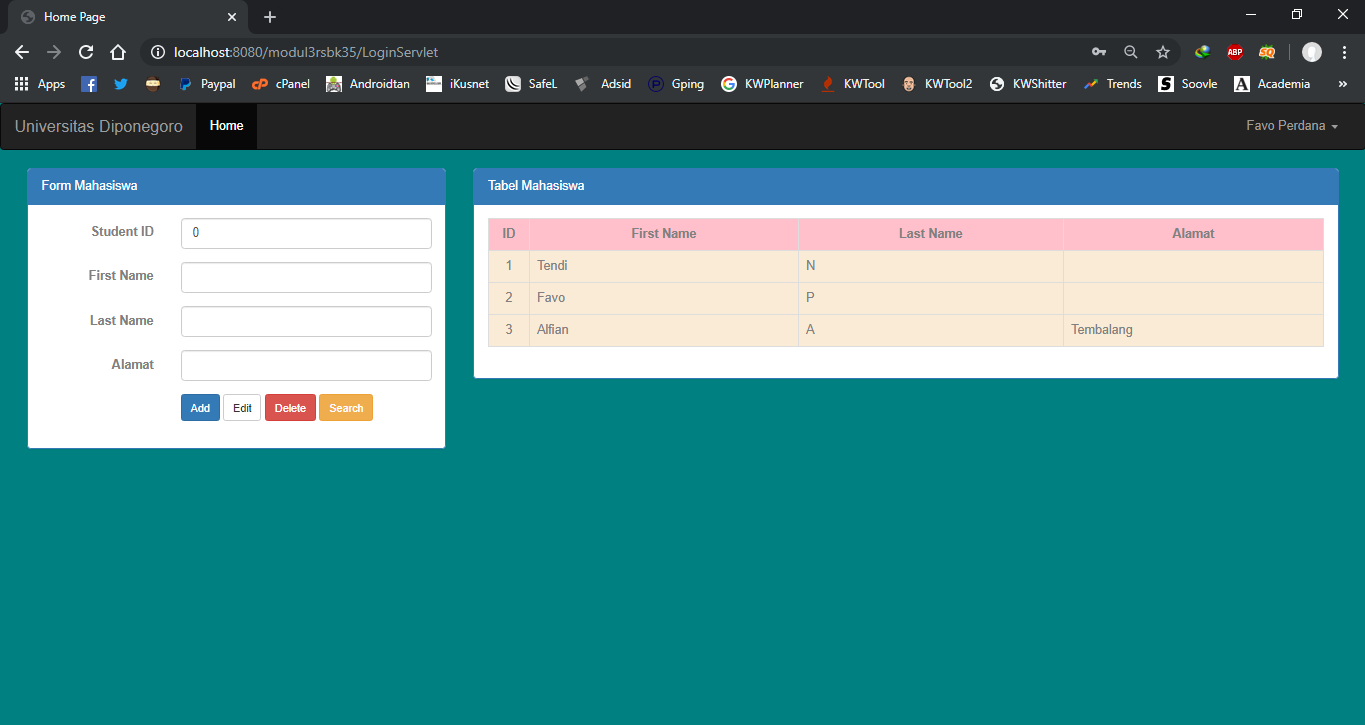
Pada Tugas Modul 3 ini diminta untuk memperbaiki yang error, mempercantik tampilan, menambah kolom “alamat”, serta meng-upload tugas ke github. Pada saat praktikum, project kami mengalami kegagalan, namun sudah diulangi lagi dan diperbaiki sehingga berhasil. Berikut adalah tampilan awal yang sudah di desain.



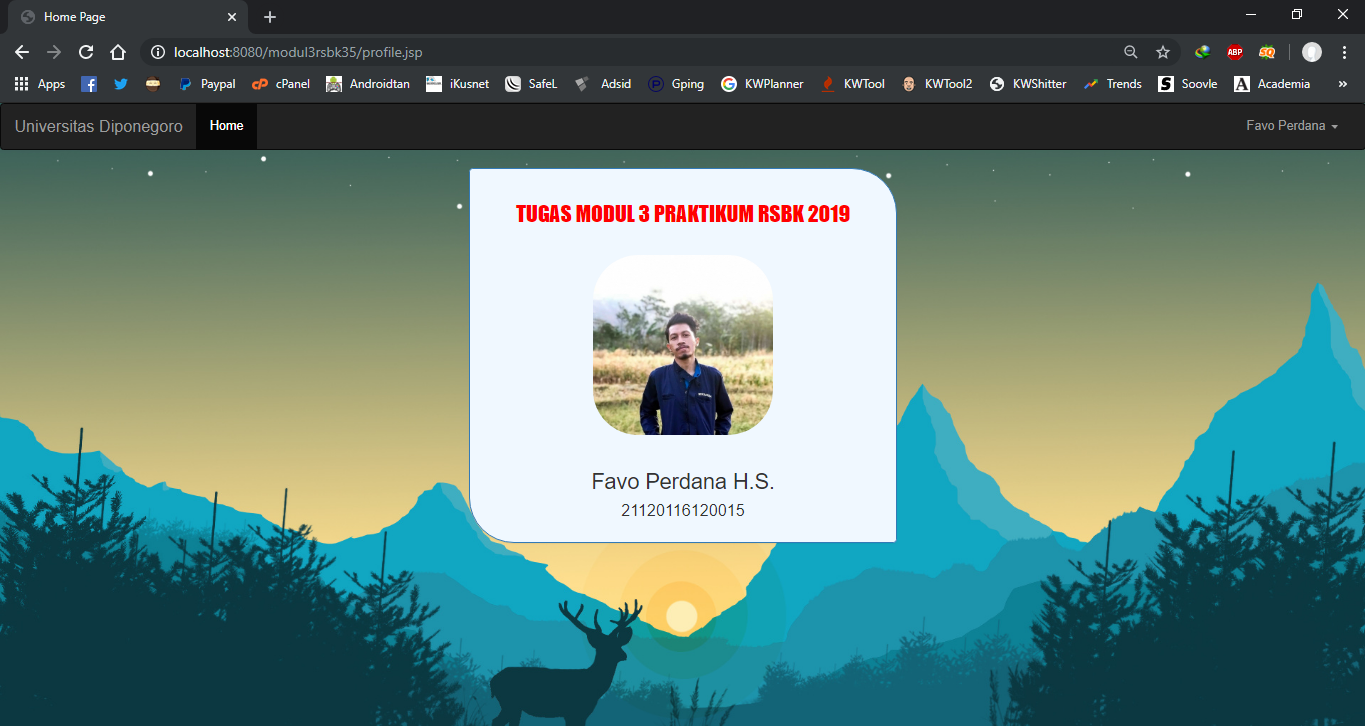
Gambar 4.42 Tampilan awal halaman Login



Gambar 4.43 Tampilan awal halaman Register



Gambar 4.44 Tampilan awal halaman Home



Gambar 4.45 Tampilan halaman profil

Penambahan kolom alamat tersebut dilakukan dengan menambahkan atribut alamat pada database mysql, kemudian melakukan perubahan pada kelas Student. Pada kelas *Student*, menambah konstruktor alamat this.alamat = alamat;, dan menambah *setter getter* alamat. Dan pada home.jsp ditambahkan kolom untuk alamat dengan value="${student.alamat}" yang memanggil isi dari atribut alamat database dari entitas *student*.

*https://github.com/perdanafavo/PraktikumRSBKkel35*

## Kesimpulan

1. Dengan menggunakan JPA, memungkinkan manipulasi data tanpa menggunakan *query*, namun bukan berarti tanpa menggunakan *query* sama sekali, tetap ada penggunaan *query* disana.
2. JPA dapat mempermudah kita untuk mengelola transaksi dengan API.
3. Fungsi *servlet* adalah untuk membuat web dinamis bagi *user*.
4. *Session bean* digunakan untuk membuat *bean* yang memiliki bisnis proses per *session* saja.
5. *Entity class* berisi tentang *entites* apa saja yang digunakan dalam sebuah *class*.