BAB IV
APLIKASI CRUD dengan JPA, EJB, dan MYSQL

4.1 Tujuan

1. Praktikan dapat menggunakan operasi CRUD dengan database MySQL
2. Praktikan dapat mengimplementasikan EJB
3. Praktikan dapat mengimplementasikan JPA
4. Praktikan dapat memahami dan menerapkan penggunaan *session bean.*
5. Praktikan dapat mengunakan fungsi jsp dan beberapa file lainnya seperti *Entity Class*, Model, Dao serta *Controller.*

4.2 Dasar Teori

4.2.1 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.

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/)

4.2.2 Session Bean

Session Bean merupakan pemodelan dari proses bisnis. *Enterprise bean* jenis ini mengerjakan sebuah aksi di dalam sistem, seperti mengakses 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 ter asosiasi 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)

4.2.3 Entity Unit

Entity Bean merepresentasikan suatu objek bisnis di dalam basisdata. *Entity bean* ini biasanya menjadi objek 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)

4.2.4 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.

Kelebihan servlet:

1. Performance Servlet baik karena tidak ada proses pembuatan berulang untuk tiap *request* dari client. Jadi tiap request ditangani oleh proses servlet container (apache tomcat), di mana servlet tidak dibuat dan dihapus berulang-ulang tetapi tetap tersimpan pada memori untuk menangani request selanjutnya.

2. Servlet memiliki kemampuan yang lengkap, antara lain penanganan request ke request, penanganan cookie dan session, akses database dengan JDBC, caching serta library yang lengkap untuk pembuatan aplikasi web.
3. Servlet memiliki fasilitas *security* yang baik dan merupakan bagian dari teknologi Java yang sudah dari asalnya didesain dengan *security* yang baik.
4. Teknologi Java Servlet portabel karena dapat dijalankan di berbagai servlet container, application server, maupun sistem operasi.
5. Proses development yang lebih cepat. Dengan menggunakan Servlet dapat menggunakan library java yang lengkap maupun menggunakan komponen yang sudah ada.
6. Karena servlet merupakan teknologi java yang memiliki penanganan memori yang baik serta memiliki garbage collection sehingga aplikasi web menjadi aplikasi yang tangguh dan stabil.

Proses Kerja Servlet

1. Servlet di load 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 di load 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 me load servlet yang baru dari class servlet.

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

4.2.5 MySQL

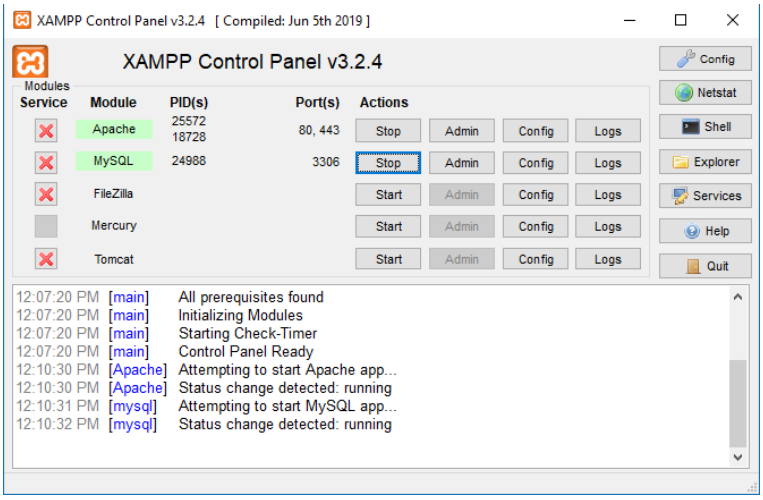
MySQL adalah sebuah perangkat lunak *system* manajemen basis data SQL (DBMS) yang *multithread*, dan multi-user. MySQL adalah implementasi dari system manajemen basisdata relasional (RDBMS). MySQL dibuah oleh TcX dan telah dipercaya mengelola system dengan 40 buah *database* berisi 10.000 tabel dan 500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

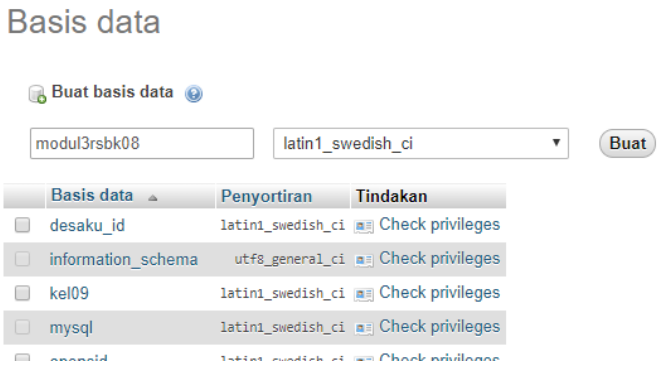
4.3 Langkah Percobaan

1. Jalankan XAMPP, jika ada bentrok dengan salah satu PORT, matikan PORT tersebut lalu nyalakan kembali (biasanya bentrok sama vmware/oracle).



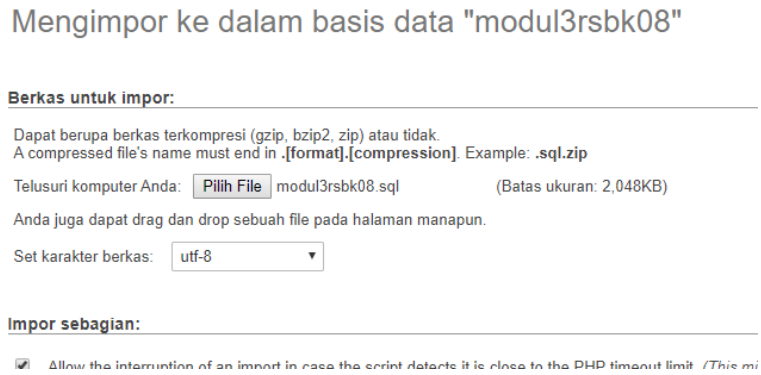
Gambar 4.1 Program Control Panel

2. Buat database di MySQL. Beri nama “modul3rsbk08”



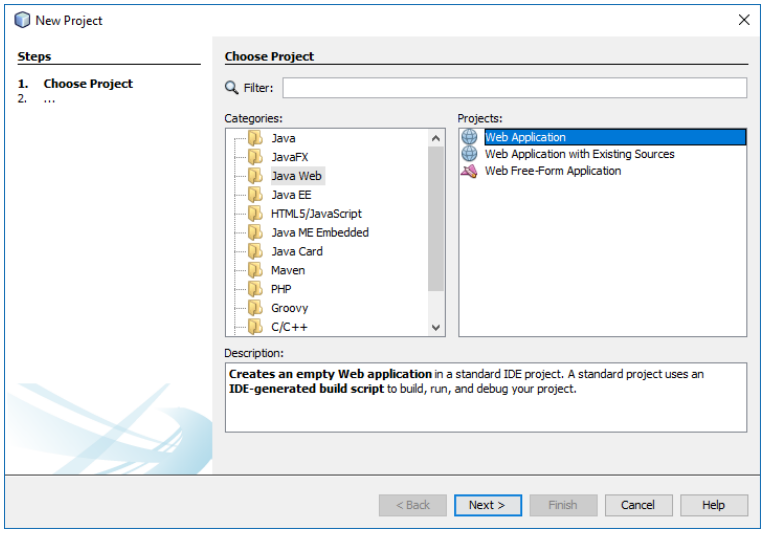
Gambar 4.2 Memuat Database modul3rsbk08 pada MySQL Database

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



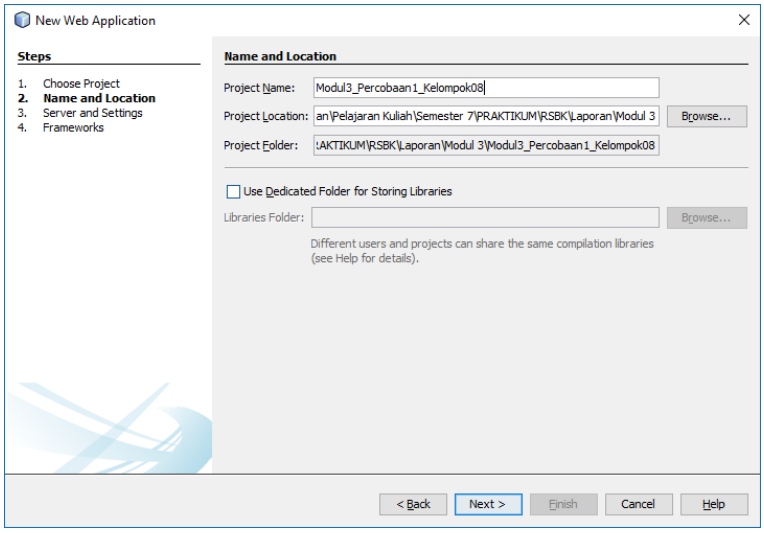
Gambar 4.3 Import file SQL ke MySQL Database

4. Buka NetBeans lalu buat project baru pada NetBeans, pilih Java Web → Web Application

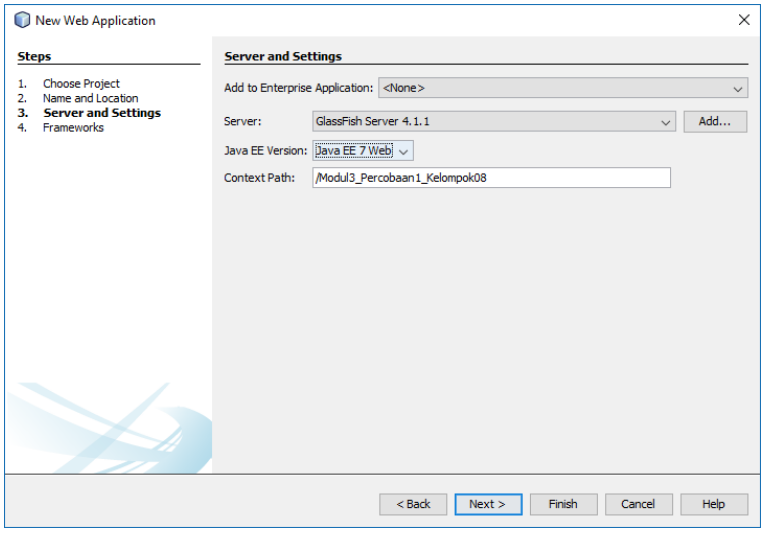


Gambar 4.4 Jendela New Project Web Application pada Java

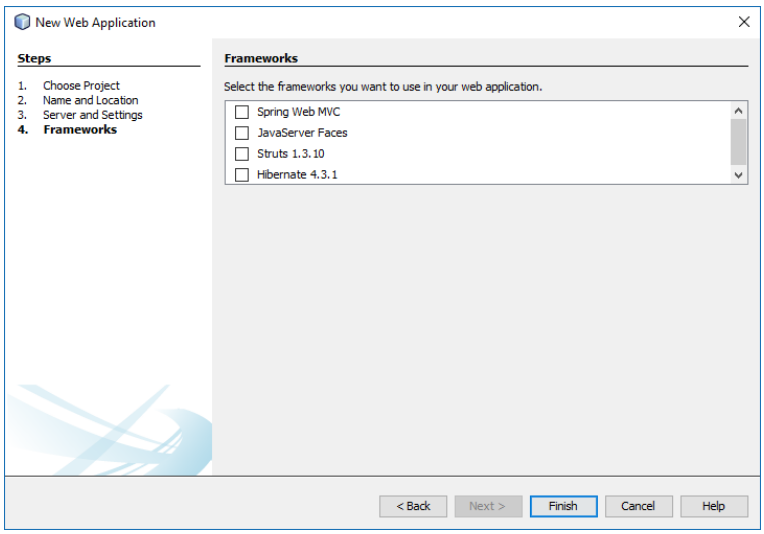
5. Beri nama project "**Modul3_Percobaan1_Kelompok08**" dan pilih Glassfish Server.



Gambar 4.5 Jendela *Project Name* dan *Locations*

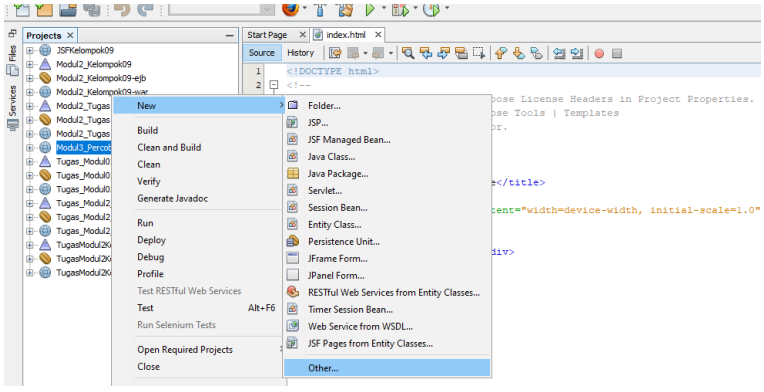


Gambar 4. 6 Jendela *New Project Server and Setting*

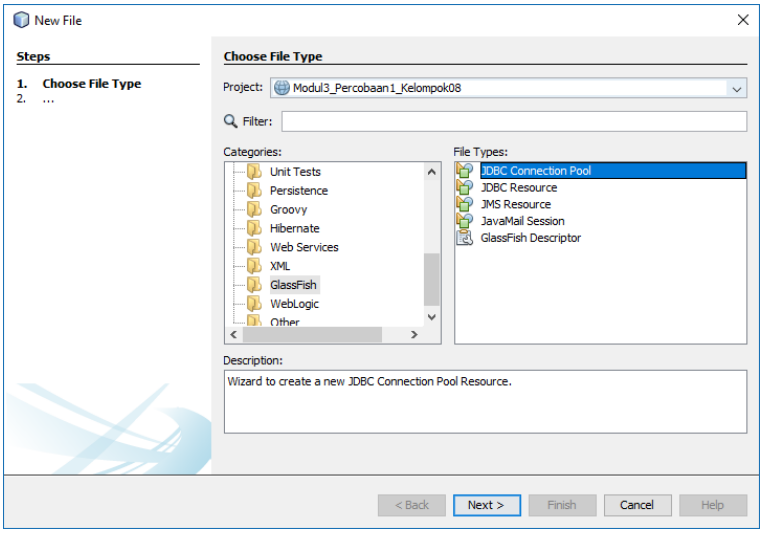


Gambar 4.7 Jendela *New Project Frameworks*

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

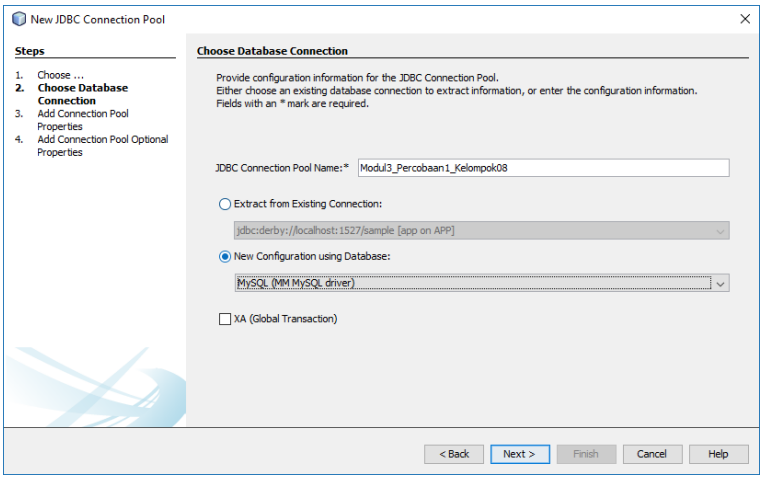


Gambar 4.8 Langkah Membuat JDBC Connection Pool



Gambar 4.9 Jendela New JDBC Connection Pool

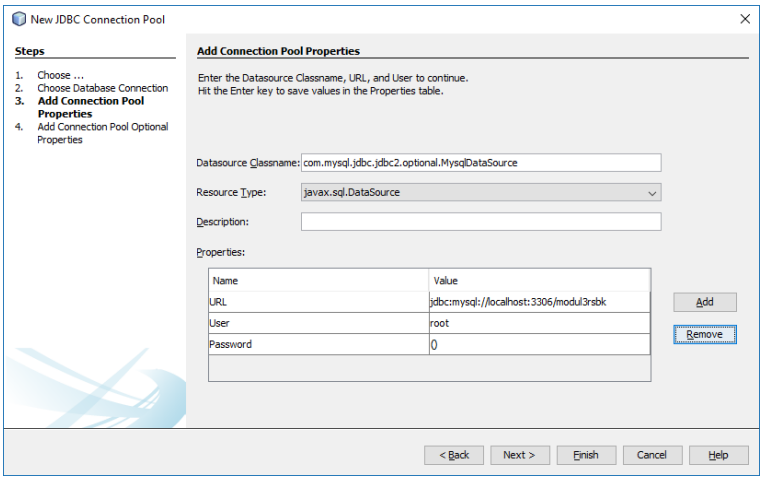
7. Beri nama connection pool (sama seperti nama projectnya, misal Modul3_Percobaan1_Kelompok08) dan pilih 'New Configuration using Database' → 'MySQL (MM MySQL driver).



Gambar 4. 10 Memberi nama Connection Pool dengan Modul3_Percobaan1_Kelompok08

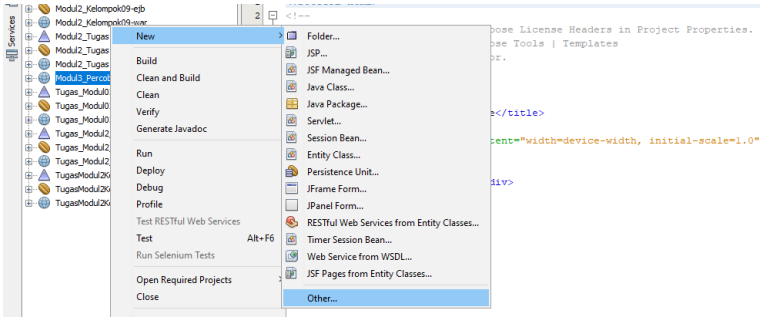
8. Masukkan URL, User, dan Password MySQL yang sudah dibuat. URL ke ***localhost*** dengan port default MySQL **3306**, lalu nama database yang akan kita gunakan (missal **Modul3_Percobaan1_Kelompok08**). Untuk user pakai **root** dan passwordnya **()**. Kemudian pilih Finish.

   jdbc:mysql://localhost:3306/ Modul3_Percobaan1_Kelompok08

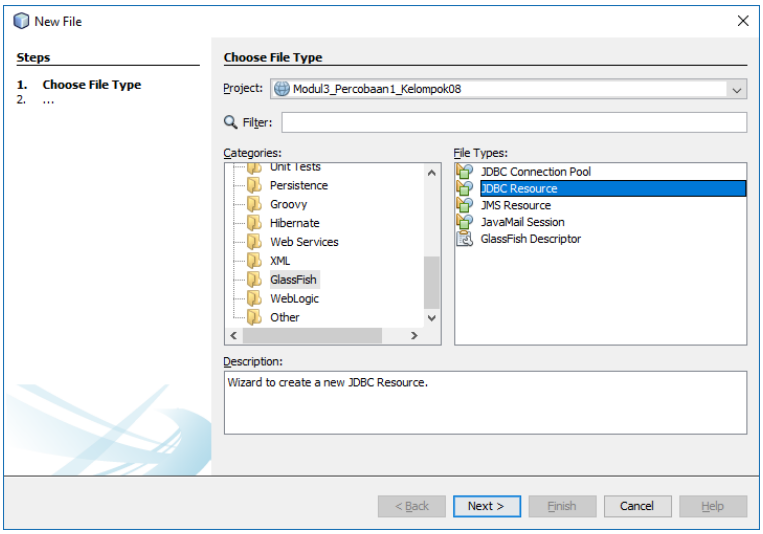


Gambar 4. 11 Jendela *Add Connection Pool Properties*

9. Buat JDBC Resource. Dengan klik kanan pada Project, New File, Other. Lalu pada kategori pilih Glass Fish, kemudian pilih JDBC Resource.

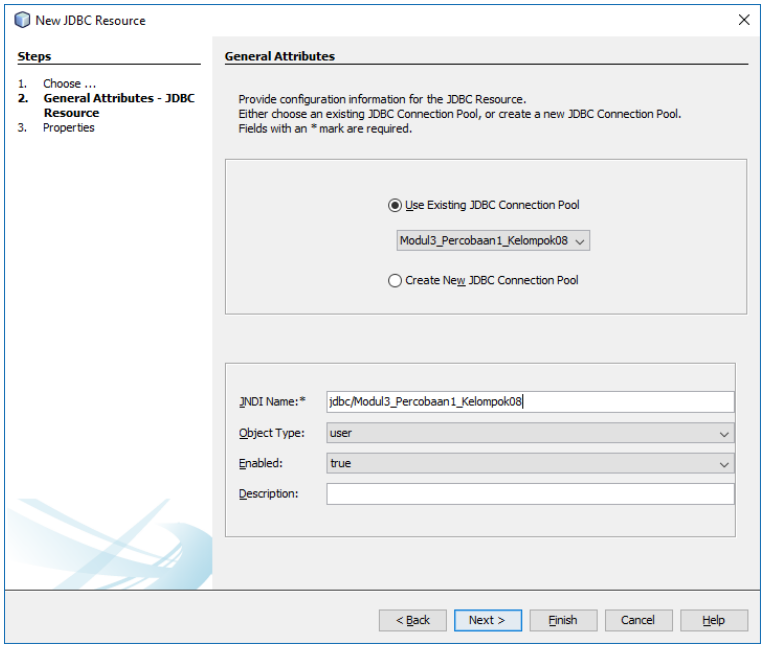


Gambar 4.12 Langkah Membuat JDBC Resource



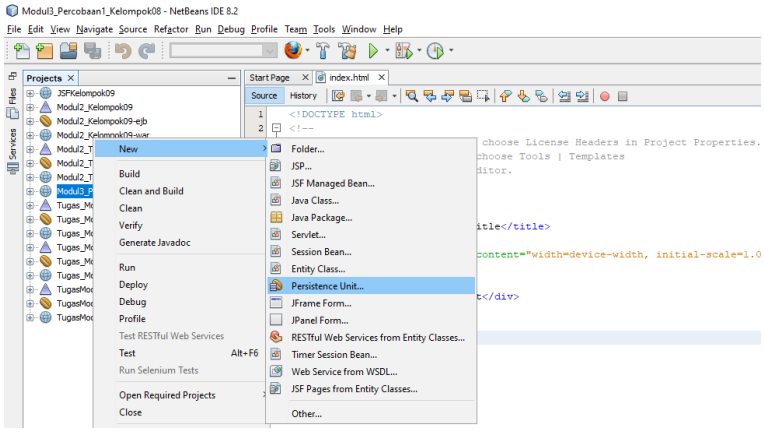
Gambar 4.13 *Create New File* JDBC Resource

10. Pilih ‘Use Existing JDBC Connection Pool’, pilih List JDBC yang baru kita buat tadi. Kemudian isikan JNDI Name seperti yang tadi. Misal Modul3_Percobaan1_Kelompok08. Lalu pilih Finish.

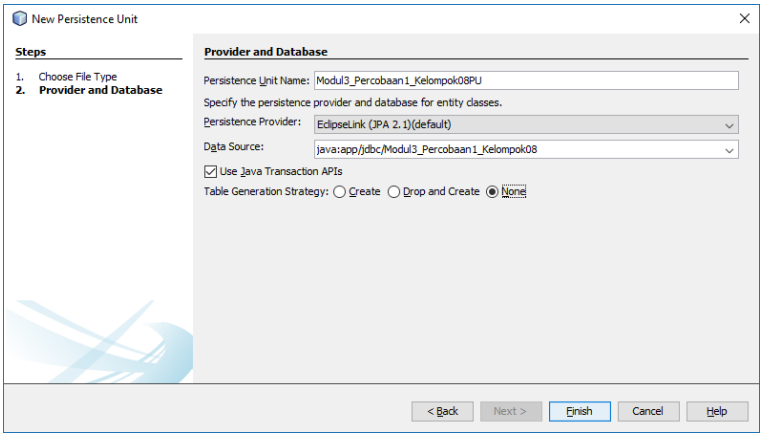


Gambar 4.14 Jendeal *New JDBC Resource Name*

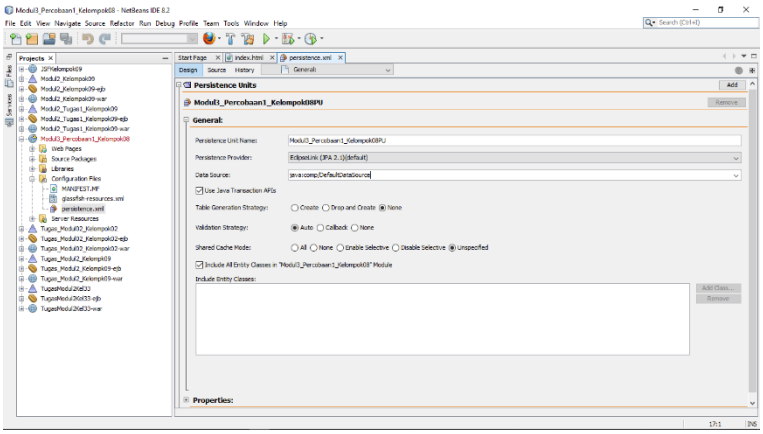
11. Buat Persistence Unit dengan klik kanan pada project, New File, pilih Persistence Unit (Jika tidak ketemu, pilih other, lalu pada kolom filter ketik persistence unit). Nama Persistence Unit akan otomatis sesuai dengan Project, lalu pilih data resource yang tadi sudah dibuat. Table Generation Strategy pilih ‘None’



Gambar 4.15 Langkah Membuat Persistence Unit

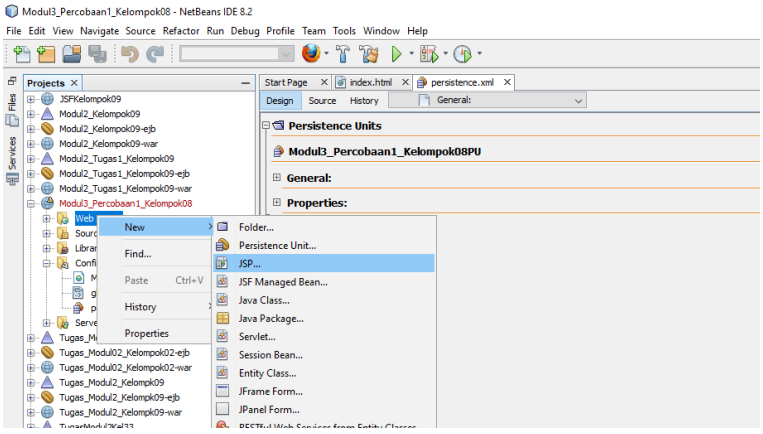


Gambar 4.16 Jendela New Persistence Unit Provider and Database

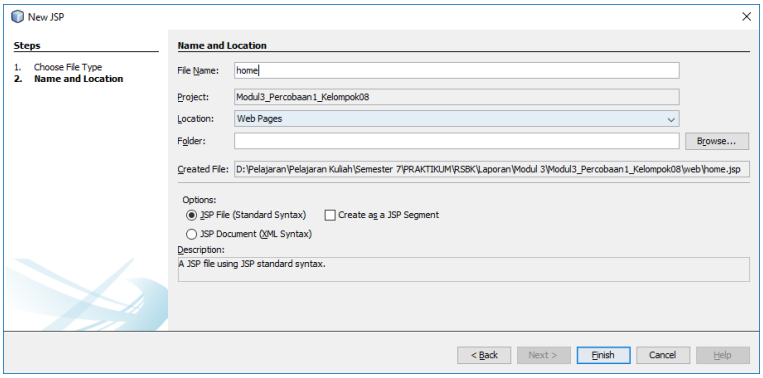


Gambar 4. 17 Add Data Source in Persintence.xml

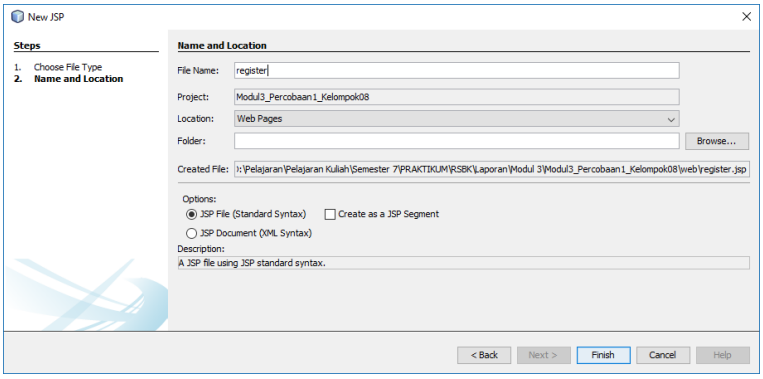
12. Buat 4 JSP page pada folder “Web Pages” dengan klik kanan folder lalu new file. JSP. Beri nama: home.jsp, register.jsp, login.jsp, dan error.jsp. Masukkan source code yang tersedia.



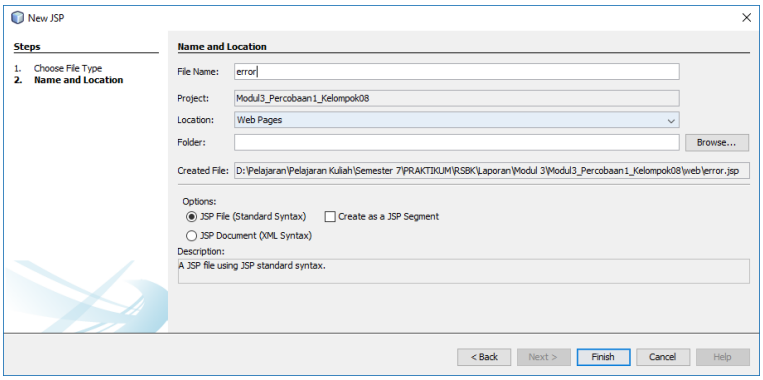
Gambar 4.18 Langkah membuat file JSP



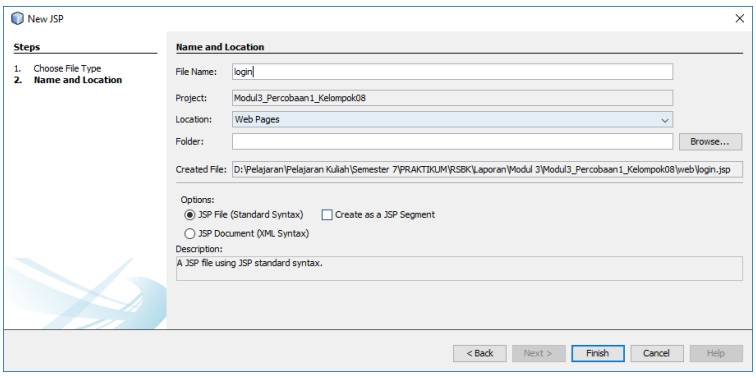
Gambar 4.19 Jendela New JSP file name home



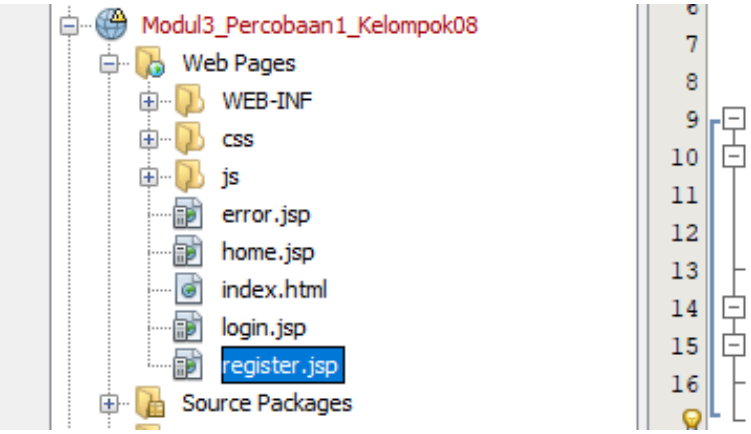
Gambar 4.20 Jendela New JSP file name register



Gambar 4.21 Jendela New JSP file name error



Gambar 4.22 Jendela New JSP file name login



Gambar 4.23 File jsp pada folder Web Pages

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 – Kelompok08</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 – Kelompok08</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;
```

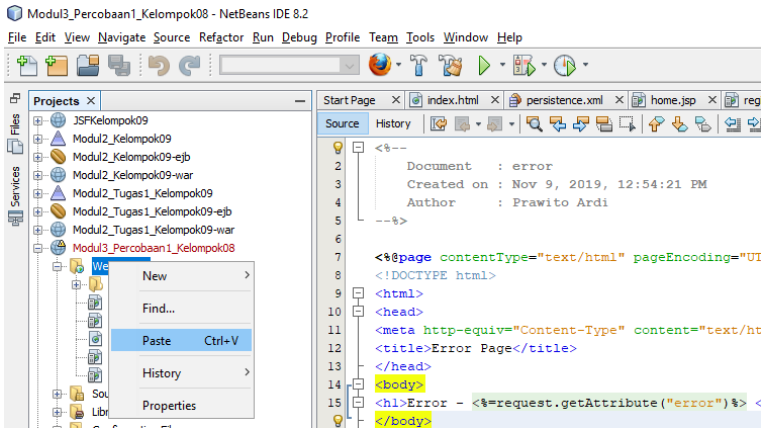
```
      }
      .daftar h3{
                margin-top: 50px;
            margin-bottom: 25px;
      }
      </style>
</head>
<div class="container">
    <div class="jumbotron row">
      <center><h2><b>Data Mahasiswa</b></h2>
      <p>Modul RSBK – Kelompok08</p>
        <h5>Selamat                                          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

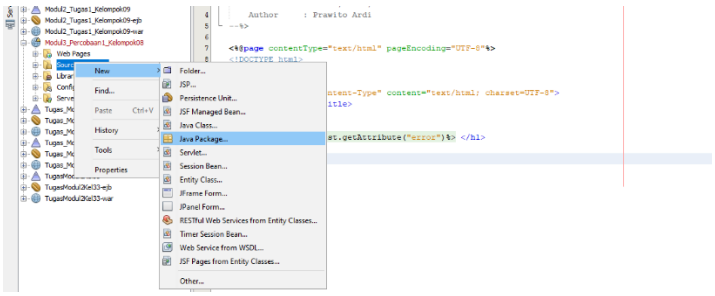
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title>Error Page</title>
</head>
<body>
<h1>Error - <%=request.getAttribute("error")%> </h1>
</body>
</html>
```

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

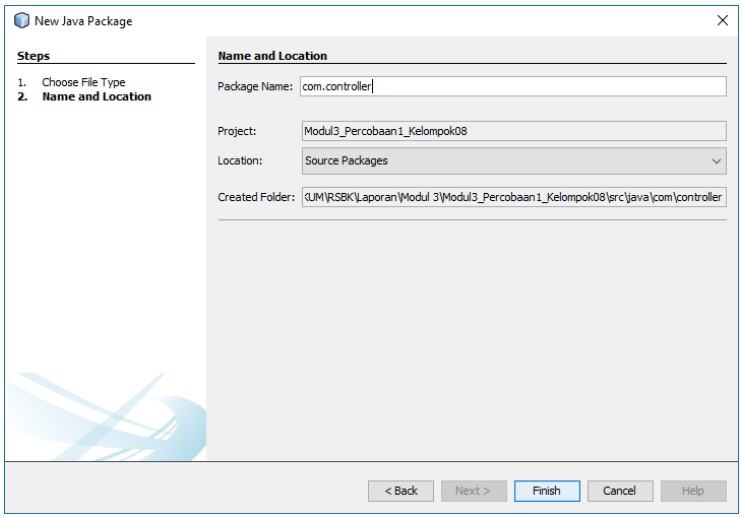


Gambar 4.24 Copy file css dan js pada folder Web Pages

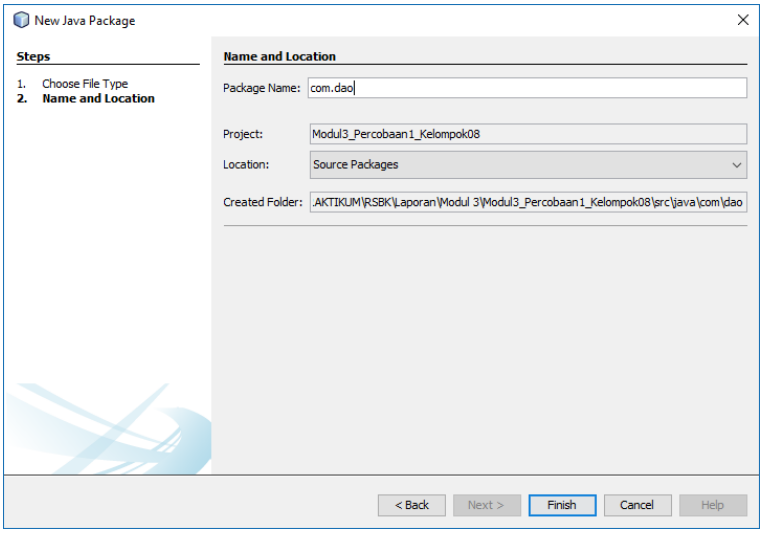
14. Buat 3 java package pada folder “Source Packages”, beri nama: “com.controller”, “com.dao”, dan “com.model”.



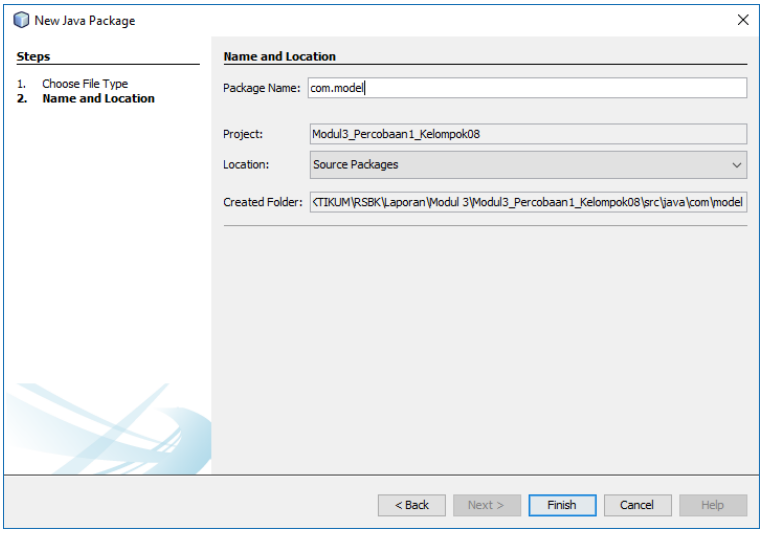
Gambar 4.25 Langkah membuat folder pada project NetBeans



Gambar 4.26 Create New Java Package com.controller

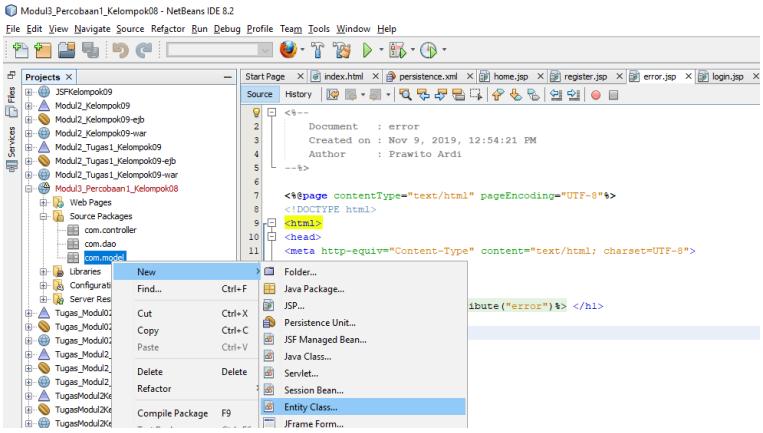


Gambar 4.27 Create New Java Package com.dao

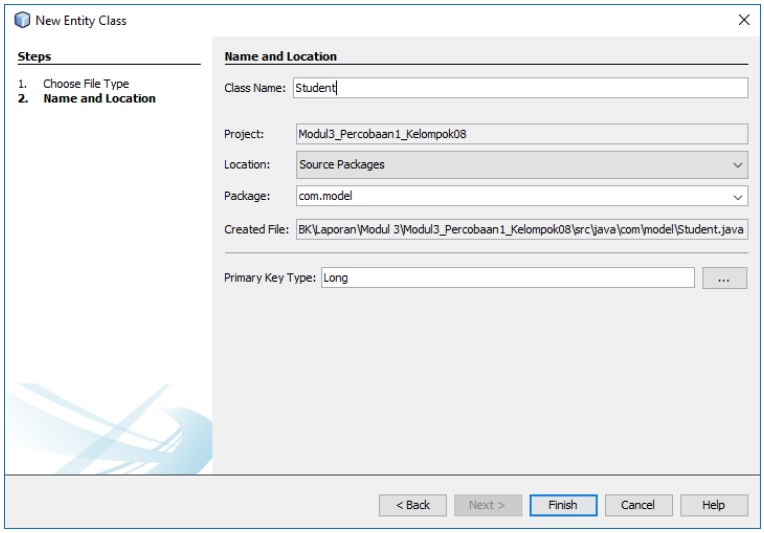


Gambar 4.28 Create New Java Package com.model

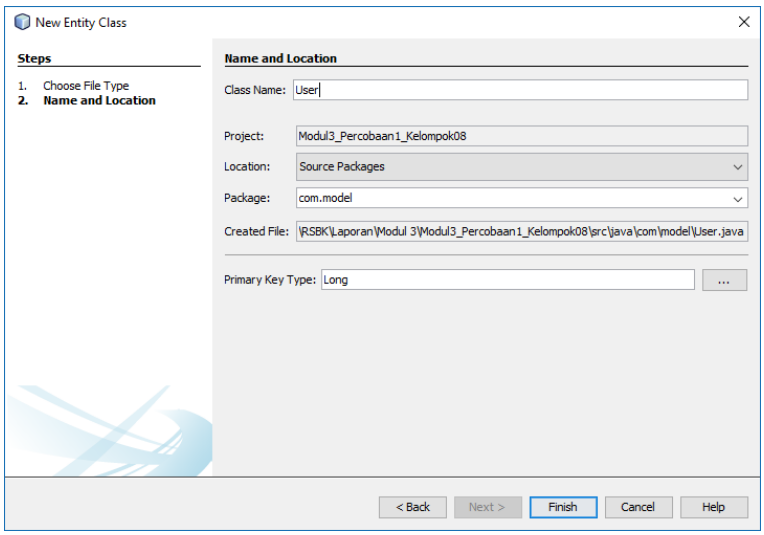
15. Buat 2 Entity Class pada package “com.model”, beri nama : “User” dan “Student”. Masukkan source code yang tersedia.



Gambar 4.29 Langkah membuat New Entity Class



Gambar 4.30 Create New Entity Class Student



Gambar 4.31 Create New Entity Class User

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;
```

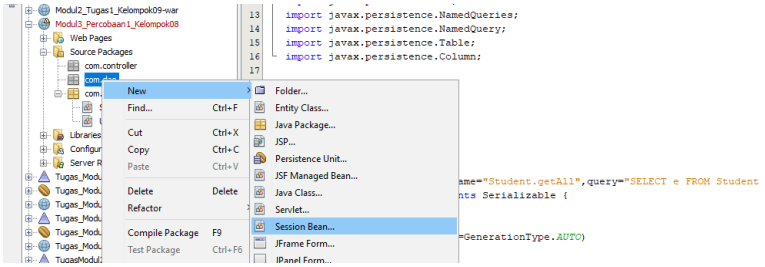
```
import javax.persistence.Table;
import javax.persistence.Column;
/**
 *
 * @author USER
 */
@Entity
@Table
@NamedQueries({@NamedQuery(name="User.getAll",query="SELECT
e FROM User e")})
public class User implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = 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

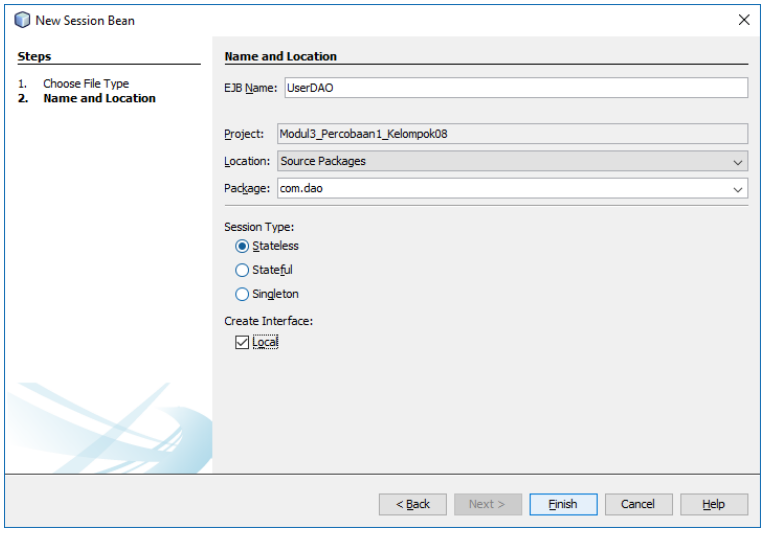
```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.model;
import java.io.Serializable;
```

```
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="SELE
CT 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;
    }
}
```

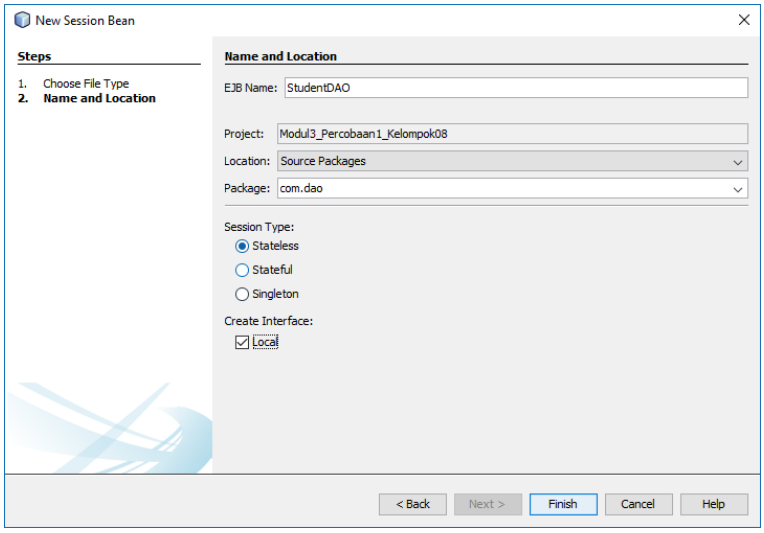
16. Buat 2 Session Beans pada package “com.dao”, beri nama: “UserDAO” dan “StudentDAO”. Ketika membuat session bean, pilih Session Type ‘**Stateless**’ dan Create Interface ‘**Local’**. Masukkan source code yang tersedia.



Gambar 4.32 Create New Session Bean pada folder com.dao



Gambar 4.33 Create New Session Bean UserDAO



Gambar 4.34 Create New Session Bean 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

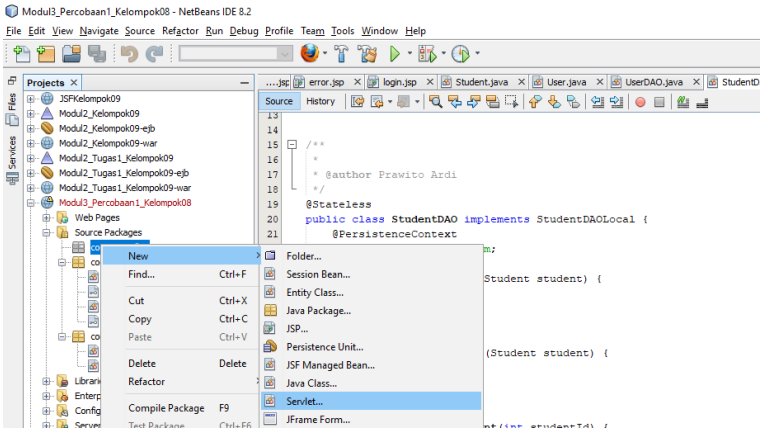
```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.dao;
import com.model.Student;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
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();
    }
}
```

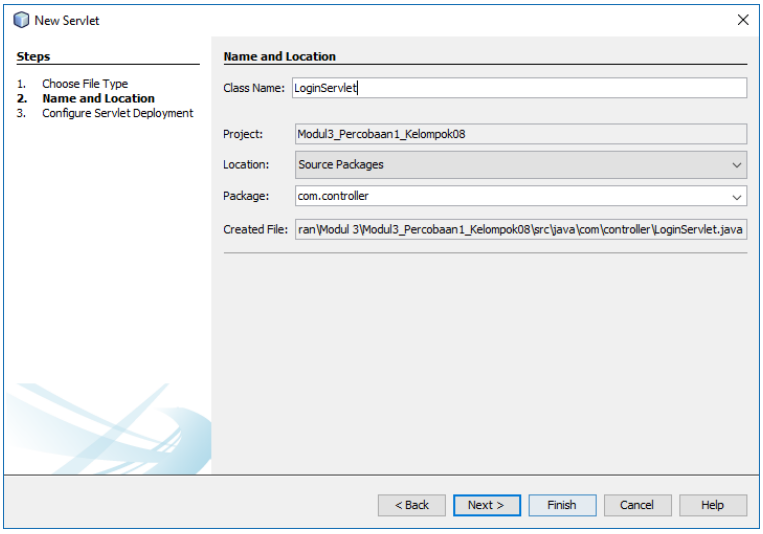
StudentDAOLocal.java

```
/*
 * 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();
}
```

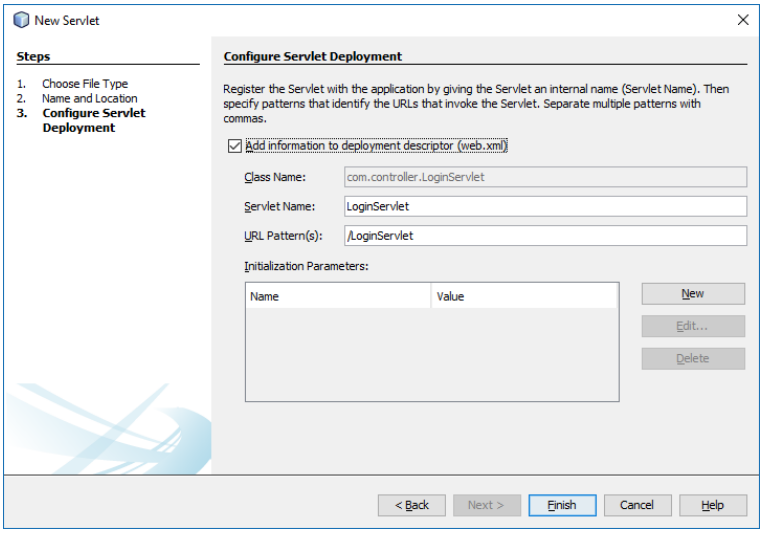
17. Buat 3 Servlet pada package “com.controller”, beri nama : “LoginServlet”, “RegisterServlet”, “StudentServlet”. Ketika membuat servlet, centang pada ‘Add information to deployment descriptor (web.xml)’. Kemudian masukkan source code yang tersedia.



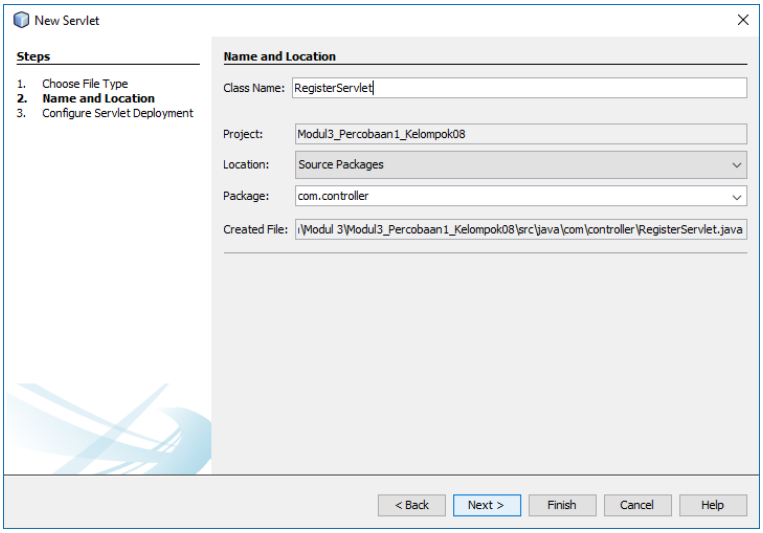
Gambar 4. 35 Create New Servlet pada folder com.controller



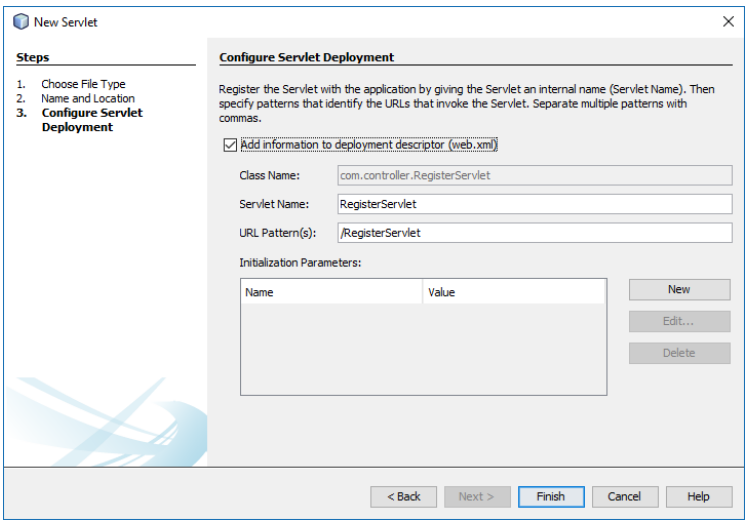
Gambar 4.36 Create New Servlet LoginServlet



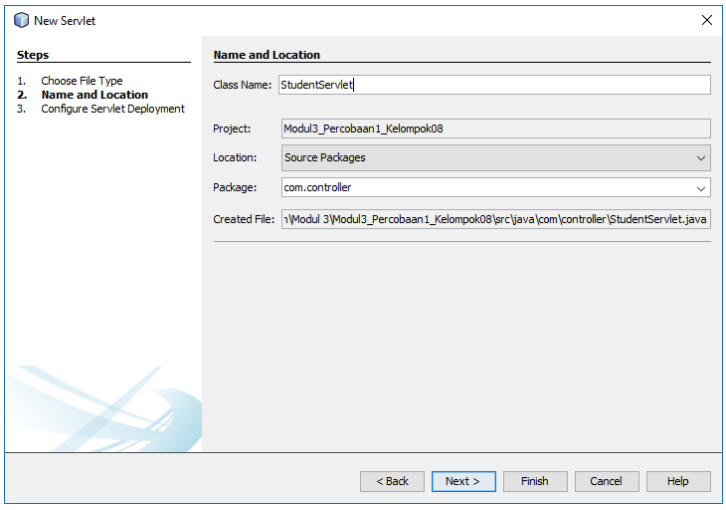
Gambar 4.37 Configure New Servlet LoginServlet



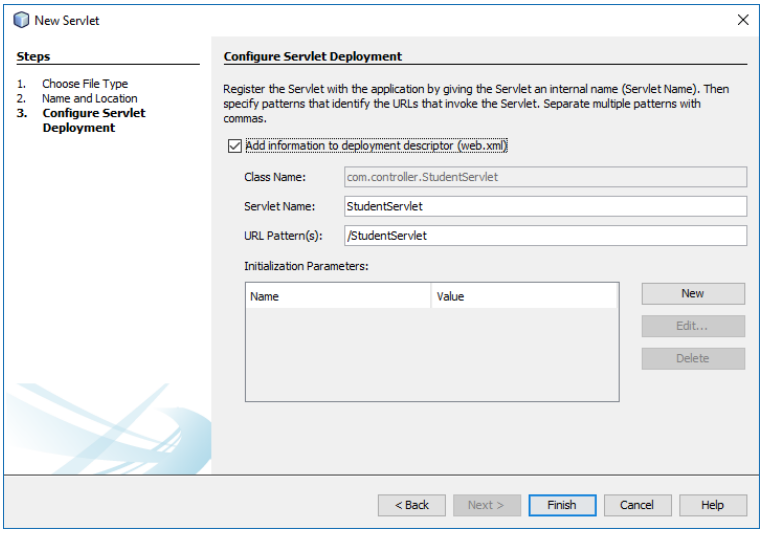
Gambar 4.38 Create New Servlet RegisterServlet



Gambar 4.39 Configure New Servlet RegisterServlet



Gambar 4.40 Create New Servlet StudentServlet



Gambar 4.41 Configure New Servlet StudentServlet

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(req
uest, 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);
```

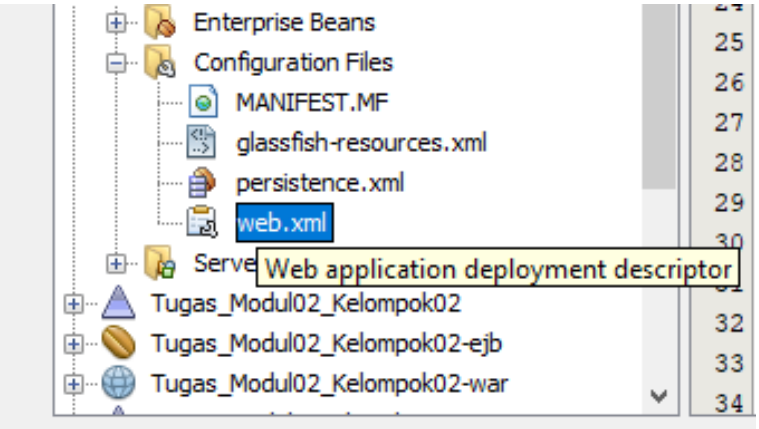
```
    }
    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

StudentServlet.java

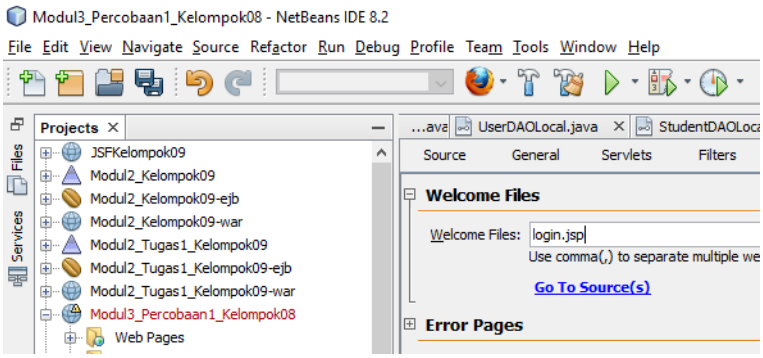
```
/*
 * To change this license header, choose License Headers in
Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package com.controller;

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>}
```

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

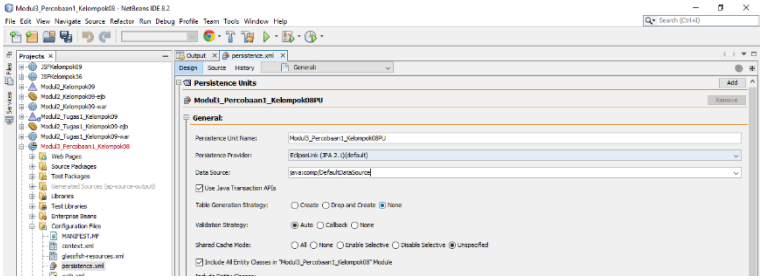


Gambar 4.42 File web.xml pada folder Configuration Files



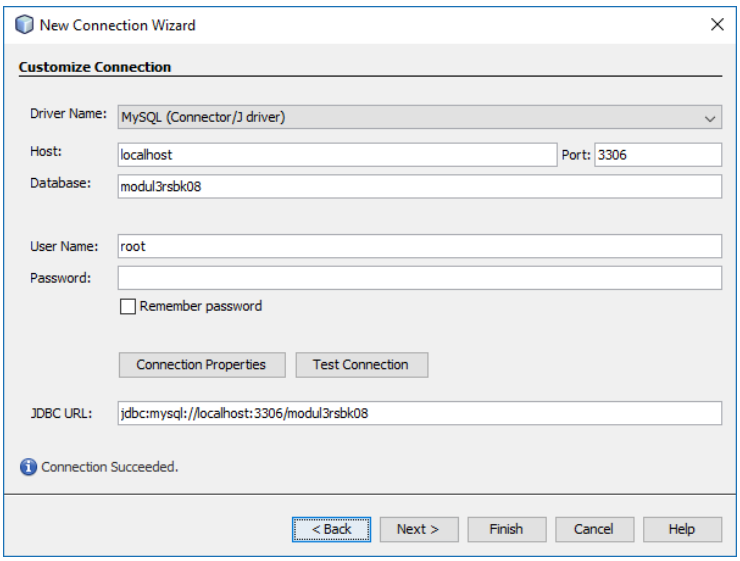
Gambar 4.43 Mengisi welcome file dengan login.jsp

19. Membuat koneksi pada MySQL, pertama masuk ke persistensi XML, kemudian ganti datasource ke yang default. Dan run program. Jika berhasil maka kemudian lanjut ke langkah berikutnya. Jika belum periksa lagi kodingannya.

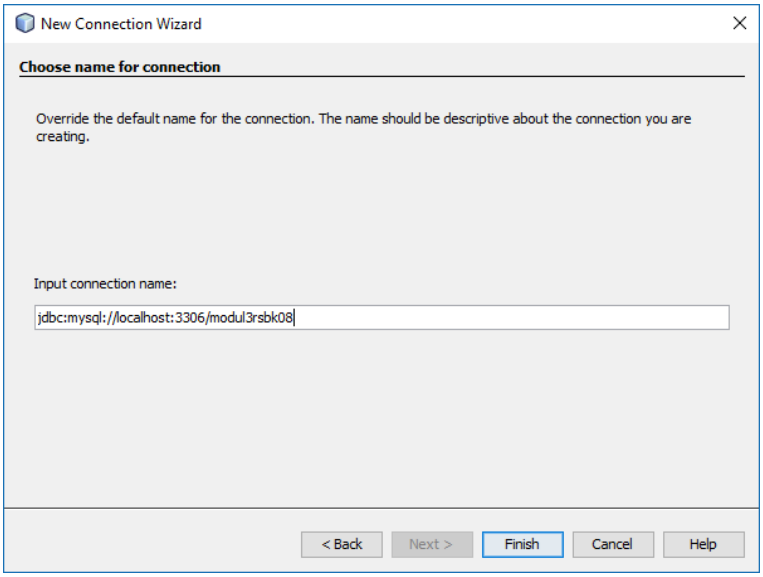


Gambar 4.44 Tampilan Persintence.xml

20. Buat data source baru



Gambar 4.45 Jendela New Connection Wizard Data Source



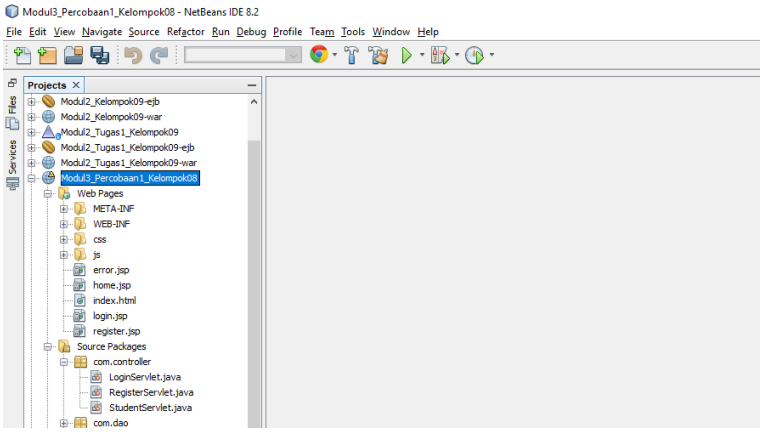
Gambar 4.46 Jendela New Connection Wizard Input Connection Name

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

4.4 Analisa Percobaan

Praktikum ini membuat sebuah program sederhana CRUD dengan JPA, EJB, dan MYSQL. Terdapat beberapa halaman yang akan menampilkan fungsi crud mulai dari halaman login, registrasi, home dan halaman lainnya. Percobaan ini menggunakan MySQL sebagai database yang digunakan untuk menyimpan data yang digunakan untuk register, login maupun fungsi crud lain.

Terdapat beberapa file folder mulai dari com.controller atau *Controller* merupakan bagian yang diimplementasikan menggunakan servlet yang berfungsi untuk mengatur aliran *request-response* dan penghubung antara *model* dan *view*. Kemudian com.dao, hingga com.model.



Gambar 4.47 Beberapa file pada Project Modul3_Percobaan1_Kelompo08

1. Folder com.controller

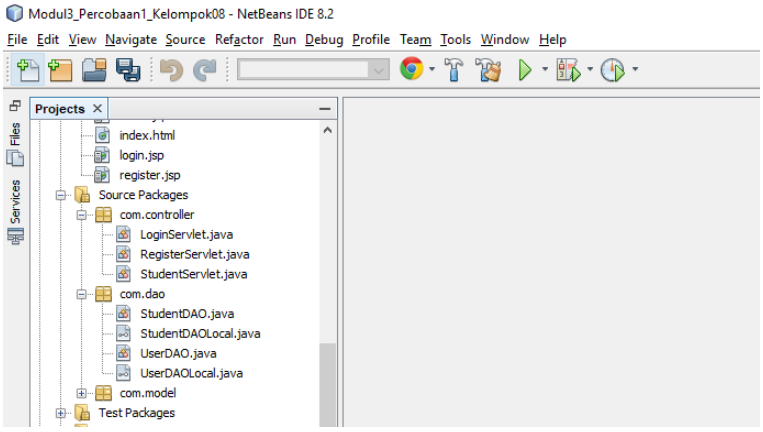
Pada folder com.controller terdapat 3 file controller yaitu LoginServlet.java, StudentServlet.java dan RegisterServlet.java. LoginServlet.java berfungsi untuk menghubungkan login.jsp dan DAOnya yang digunakan untuk *login* yaitu LoginDAO.java. LoginServlet akan menerima dan merespon jika ada *request* baik dari login.jsp maupun DAOnya.

Selanjutnya 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.

Serta RegisterServlet.java berfungsi untuk menghubungkan register.jsp dan DAOnya yang digunakan untuk proses register yaitu StudentDAO.java. StudentServlet akan menerima dan merespon jika ada data masuk baik dari Student.jsp maupun DAOnya.

2. Folder com.dao

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



Gambar 4.48 Folder package com.dao

Pada file UserDAO terdapat beberapa *listing code* yang akan menjalankan fungsi untuk melalukan login yang mana *username* dan *password* akan dicocokan dengan data yang ada pada *database* melalui method checkUser dan credential. Kemudian pada file ini juga terdapat method addUser yang digunakan untuk fungsi register ketika data ditambahkan, dimana fungsi ini juga berhubungan dengan file register.jsp/RegisterServlet yang dibuat.

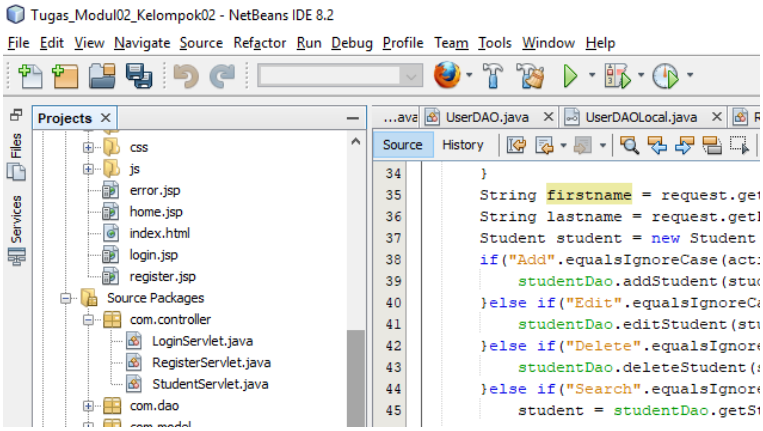
Pada file UserDAOLocal.java terdapat sebuah *method* dengan nama checkUser yang memiliki parameter *username* dan *password*, credential yang memiliki parameter *username* dan *password*, serta method addUser dengan parameter user, parameter tersebut akan divalidasi oleh UserDAO.java.

Selanjutnya terdapat file StudentDOA pada folder com.dao, berfungsi untuk memanggil data pada tabel *student* yang kemudian akan ditampilkan pada halaman

home. Data tersebut dapat dimanipulasi sesuai dengan keinginan *user* mulai dari add, delete, edit dan lain-lain yang artinya file ini juga menjalankan method fungsi crud.

3. Folder com.model

Berikutnya adalah com.model yang berisi file *entity class* yang dengan method-method untuk mengambil data serta memberi nilai dari data. Terdapat 3 model yaitu LoginServlet.java, RegisterServlet dan StudentServlet.java.



Gambar 4.49 Folder package com.dao

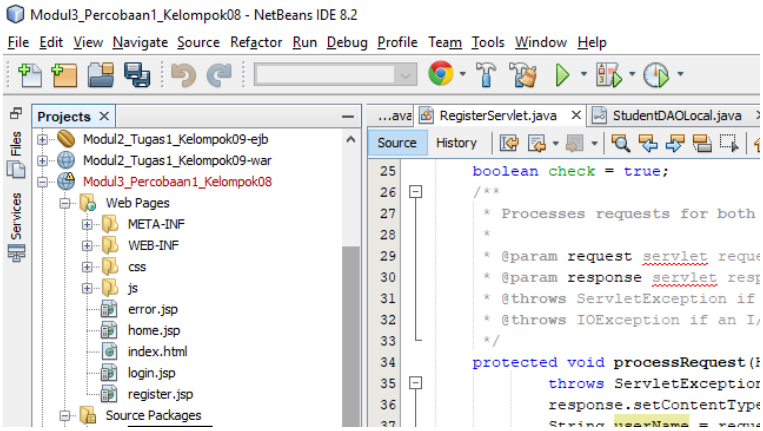
Pada LoginServlet.java berfungsi untuk memberi nilai pada data yang nantinya akan dihubungkan ke database tabel *login* dengan *view* login.jsp. File ini berisi kondisi dimana inputan yang dimasukan akan di cek mulai dari username dan password. LoginServlet ini juga terhubung dengan file userDAO method credential serta terdapat pengkondisian ketika username dan password yang di input sesuai dengan database maka akan menjalankan fungsi sesuai dengan yang ditentukan.

Selanjutnya StudentServlet.java berfungsi untuk memberi nilai pada data yang nantinya akan dihubungkan ke database tabel *student* dengan *view* student.jsp. Pada file ini juga terdapat beberapa pengkodisan yang berhubungan dengan pengolahan data seperti add, delete, edit dan lain-lain. Terdapat method doPost dan doGet yang akan menjalankan fungsi yang ada pada home.jsp dan register.jsp

Kemudian RegisterServlet.java berfungsi untuk memberi atau menambahkan data user baru pada database. Pada file ini juga terdapat beberapa method misal check, apakah data username sudah ada pada database apa belum.

4. Folder Web Pages (jsp)

Pada percobaan ini juga terdapat file JSP yang merupakan file yang digunakan untuk membuat tampilan halaman web pada percobaan ini. Terdapat 4 file .jsp yaitu login.jsp, home.jsp, register.jsp dan error.jsp. Login sebagai tampilan halaman ketika *login* menggunakan *username* dan *password*. *Home* sebagai tampilan halaman setelah *login* yang digunakan untuk memanipulasi data seperti add, delete, edit dan lain-lain. Register yang digunakan sebagai halaman untuk menambahkan data username dan password ke database yang nantinya digunakan untuk login juga. Serta *Error* sebagai tampilan halaman *error* misal ketika salah memasukkan data.



Gambar 4.50 Folder package file .jsp

Login.jsp adalah tampilan atau *view* halaman *login* dengan fungsi memberikan input *username* dan *password* pada form yang ada yang nantinya data atau input yang di berikan akan di cek/di koreksi pada DAO dan model yang dihubungkan oleh *controller* yang kemudian akan di hubungkan dengan database apakah ada atau tidak datanya, yang kemudian apabila ada dan sesuai kondisi yang diberikan, user akan masuk halaman home.

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

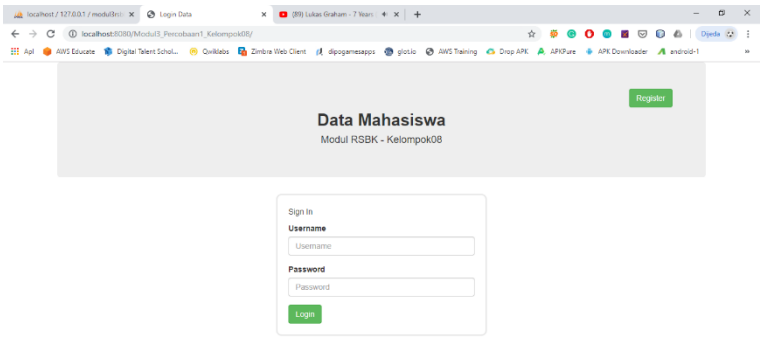
Register.jsp adalah tampilan atau view halaman dengan fungsi menambahkan user baru dengan memberikan username dan password yang kemudian akan masuk pada database serta digunakan login pada halaman login.jsp.

Error.jsp adalah tampilan atau *view* halaman yang menampilkan pesan *error* apabila ada kesalahan ketika input yang diberikan tidak sesuai dengan database yang ada, misal salah memasukkan *username* atau *password*.

Dan berikut adalah hasil dari program sederhana crud:

a. Halaman Login

Halaman view login ini berasalah dari file login.jsp yang telah dibuat, terdapat 2 texfield yang digunakan untuk memberikan input username dan password agar proses login dapat dilakukan. Kemudian terdapat button login yang apabila ditekan akan memberikan kondisi yaitu ketika username dan password benar maka akan masuk ke halaman home, dan apabila username/password salah login tidak dapat dilakukan dengan terdapat text pemberitahuan. Pada halaman login ini juga terdapat Data Mahasiswa Modul RSBK – Kelompok 08.

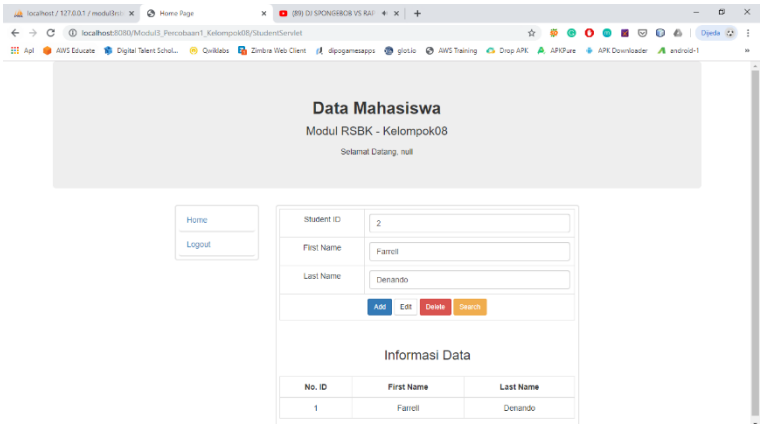


Gambar 4.51 Tampilan Halaman Login

b. Halaman Home

Pada halaman home ini terdapat tampilan berupa tabel yang berisikan nilai atau atribut berupa Informasi Data mulai dari No.ID, First Name, Last Name. Dan pada halaman home ini terdapat beberapa button yang memiliki fungsi, mulai dari button add yang digunakan untuk menambahkan data ketika text sudah memiliki nilai. Kemudian terdapat button *edit* yang digunakan untuk mengubah nilai atribut sesuai dengan StudentID. Selanjutnya button delete yang digunakan untuk menghapus

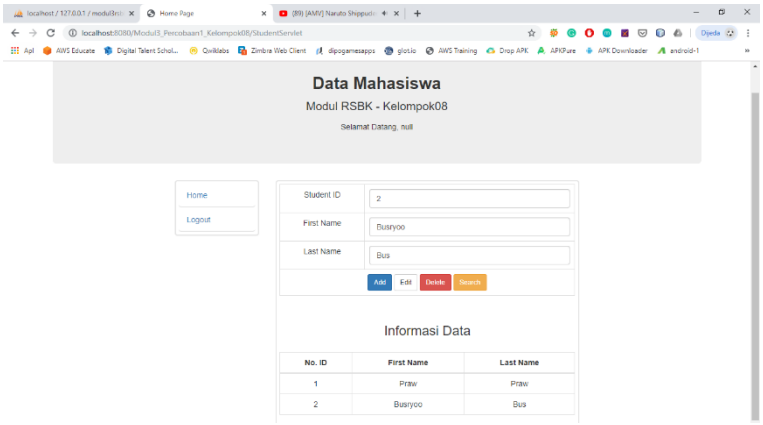
data sesuai dengan ID yang diisikan pada text field StudentID. Dan button search yang digunakan untuk mencari data sesuai dengan StudentID.



Gambar 4.52 Halaman Home

- Fungsi *Add*

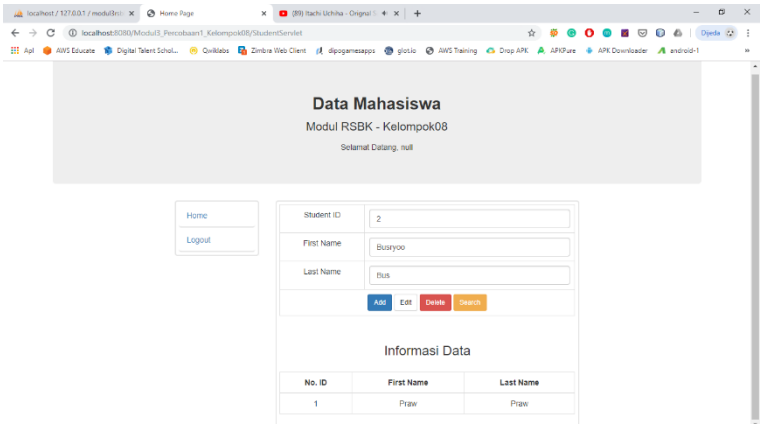
Fungsi *add* digunakan untuk menambahkan data pada tabel Informasi Data, dan pada percobaan ini kita menambahkan 2 data dengan ID 1 dan 2. Dengan mengisi *text field* yang tersedia dengan ID yang berbeda sebelumnya, maka data akan secara otomatis bertambah pada tabel Informasi Data ketika button add di klik, serta data ini juga akan bertambah pada MySQL database.



Gambar 4.53 Halaman *Home* Fungsi *Add*

- Fungsi *Delete*

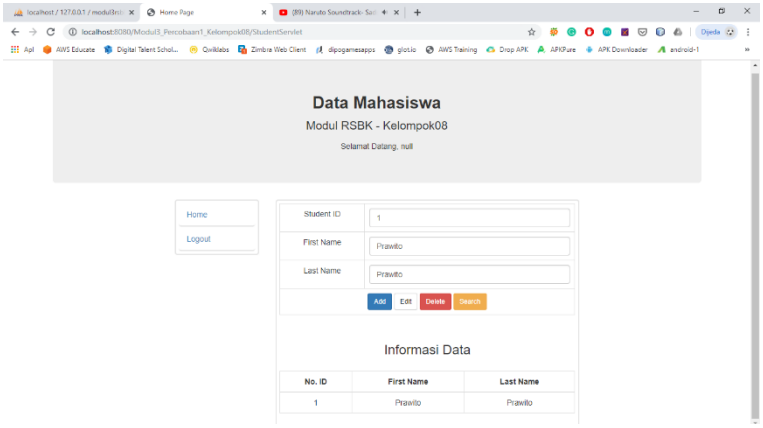
Kemudian button atau fungsi delete ini untuk menghapus data berdasarkan nilai StudentID, dimana sebelumnya yang terdapat 2 data dengan id 1 dan 2, kemudian mencoba fungsi button delete dengan mengisi text field Student ID 2, sehingga data dengan ID 2 terhapus dan akhirnya menyisakan 1 data pada table Informasi Data.



Gambar 4. 54 Halaman Home Fungsi *Delete*

- Fungsi *Edit*

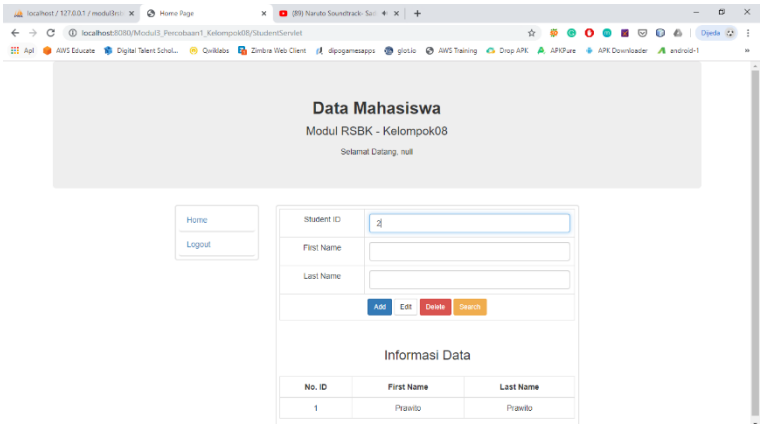
Fungsi dari edit atau button edit ini adalah untuk mengubah value atau nilai yang telah ada yaitu Id, First Name dan Last Name, dan pada fungsi ini juga terdapat form atau *text field* yang mengambil data sebelumnya yang kemudian ketika di save dengan mengubah nilai dari *text field*, maka data akan berubah dan tersimpan dalam database. Dimana sebelumnya terdapat 1 data yaitu ID 1 dangan First Name “Praw” dan Last Name “Praw”, kemudian ketika mencoba mengganti *text field* First Name dan Last Name “Prawito” dengan text field Student ID 1, maka data akan berubah ketika *button edit* ini di klik.



Gambar 4. 55 Halaman Home Fungsi *Edit*

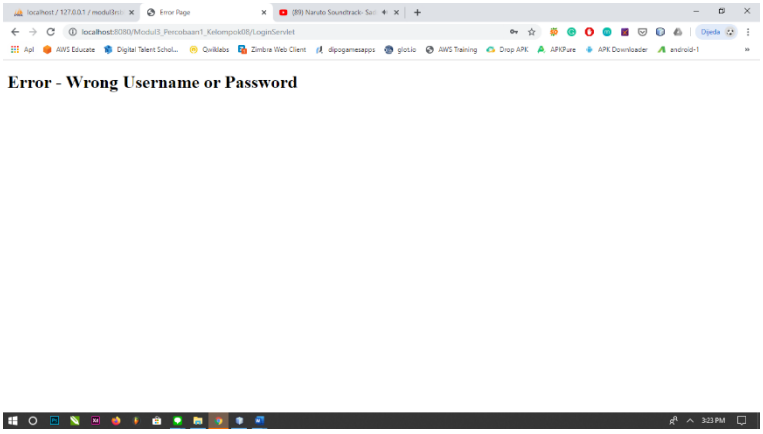
- Fungsi *Search*

Mencari data sesuai dengan nilai ID yang ada pada *text field* StudentID, yang nantinya *text field* akan terisi dengan nilai data yang di *search* serta ketika data (ID) yang di cari tidak ada maka *text field* akan kosong kembali.



Gambar 4.56 Halaman Home Fungsi *Search*

c. Halaman *Error*

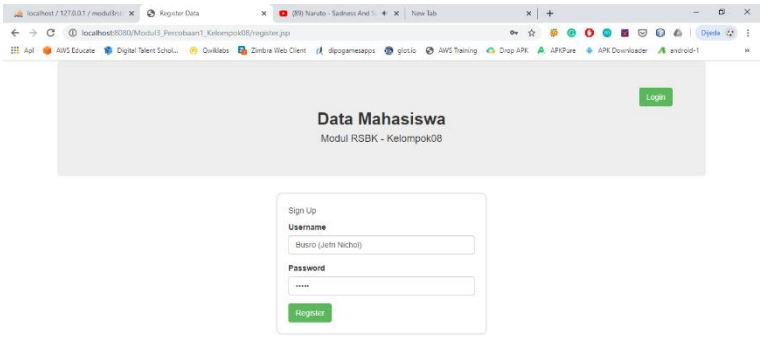


Gambar 4.57 Halaman *Error*

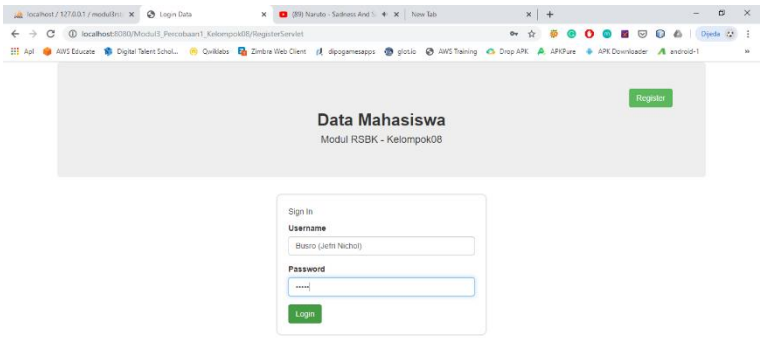
Halaman *error* yang diatur *source code* nya di error.jsp berfungsi untuk menampilkan pesan *error* ketika ada kesalahan dalam login atau register, misal salah memasukkan *username* atau *password*.

d. Halaman Register (belum selesai)

Halaman register ini digunakan untuk menambahkan user baru yang digunakan untuk proses *login*, data atau value yang ditambahkan pada *text field* nantinya akan masuk ke dalam *database user*. Pada percobaan ini mencoba untuk menambahkan user baru yaitu dengan Username “Busro (Jefri Nichol)” dan password “busro” dan ketika masuk ke halaman login menggunakan user ini, proses login dapat dilakukan.



Gambar 4.58 Tampilan Halaman Register



Gambar 4.59 Tampilan Halaman Login dengan masuk menggunakan user baru

4.5 Tugas

Pada tugas praktikum ini adalah mencoba untuk menambahkan 1 kolom alamat pada table student, dimana data ini akan ditampilkan pada halaman home, dan nilai yang ada juga akan tersimpan dalam database serta dapat diterapkan fungsi-fungsi crud.

Kemudian tugas selanjutnya adalah mencoba untuk menambahkan halaman profile dengan membuat file jsp baru yaitu profile.jsp. Pada halaman profile.jsp ini akan menampilkan informasi praktikan dan beberapa informasi lainnya yang berhubungan dengan praktikum ini.

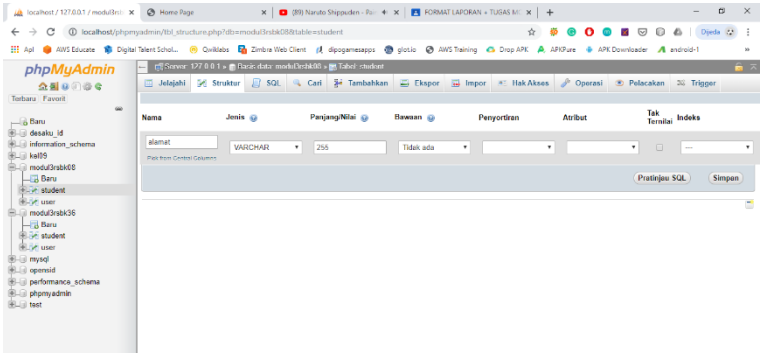
Terakhir adalah mencoba untuk memperindah tampilan, sehingga program crud sederhana ini menjadi lebih menarik dan mudah dimengerti oleh user yang berhubungan dengan fungsi *user interface* tentunya.

Pembahasan Tugas:

1. Tugas Penambahan Kolom Alamat

a. Membuat Kolom Alamat pada Database MySQL

Pada percobaan ini mencoba untuk membuat atau menambahkan atribut alamat pada table student yaitu dengan menambahkan kolom alamat di MySQL *database* pada tabel *student* pada *menu structure* terlebih dahulu. Kemudian mengisi value dengan tipe varchar dengan panjang nilai 255.

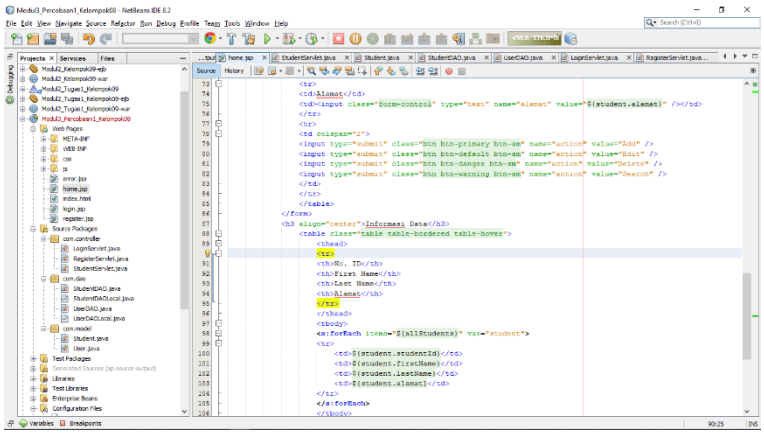


Gambar 4.60 Membuat Kolom Alamat pada tabel student di *database* MySQL

b. Membuat Beberapa source code pada home.jsp, StudentServlet.java dan student.java

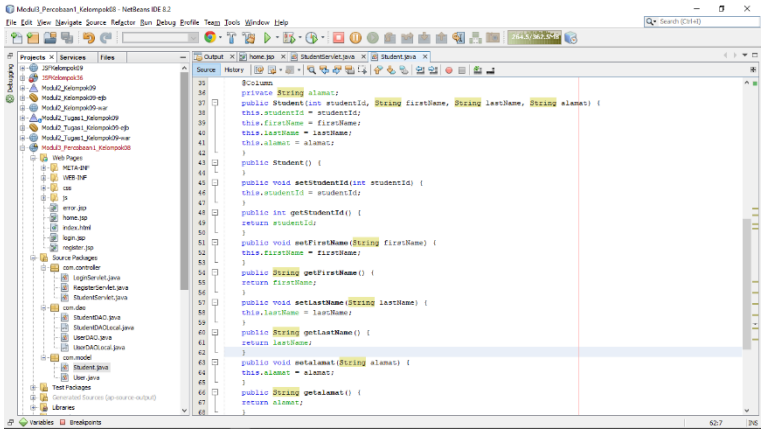
Kemudian menambahkan form atau text field alamat pada home.jsp dan menambahkan kolom alamat pada table Informasi Data, dan pada Netbeans, tambah

source code untuk kolom alamat pada StudentServlet.java dan Student.java. Dan berikut adalah hasilnya:



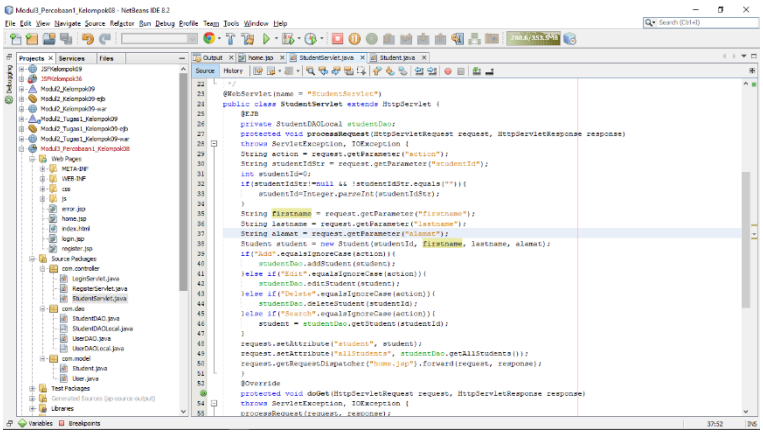
Gambar 4.61 Menambahkan beberapa *source code* pada file home.jsp

Pada gambar ini menampilkan halaman home.jsp dengan menambahkan text field alamat berupa *form control* dengan *type text*, name alamat serta value student.alamat, kemudian mencoba untuk menambahkan kolom pada table informasi data agar data di database dapat ditampilkan pada halaman ini yaitu dengan menambahkan source code <th> Alamat </th> (untuk kolom Alamat), kemudian pada bagian forEach menambahkan <td>$(student.alamat)</td> agar data dalam *database* bisa ditampilkan.



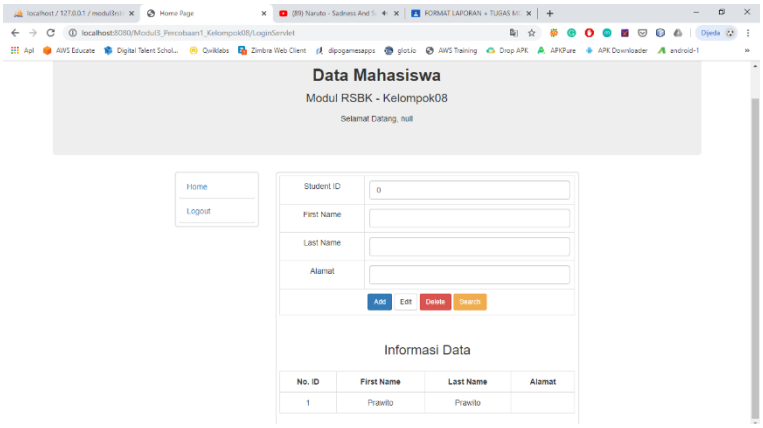
Gambar 4.62 Menambahkan beberapa *source code* pada file student.java

Selanjutnya menambahkan private String alamat dan menambahkan atribut alamat pada *method* public student. Pada student.java ini ditambahkan *setter getter* alamat yang digunakan untuk menerima dan mengembalikan nilai alamat.



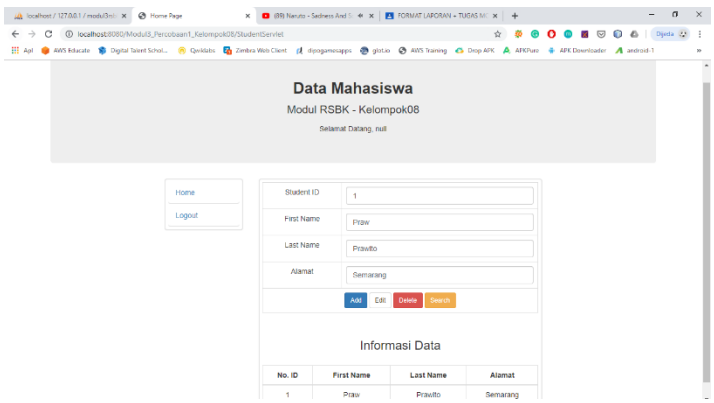
Gambar 4.63 Menambahkan beberapa *source code* pada file StudentServlet.java

Menambahkan beberapa source code pada method yaitu getParameter “alamat”, dan menambahkan atribut pada object baru yang dibuat.



Gambar 4.64 Halaman Home dengan penambahan kolom alamat

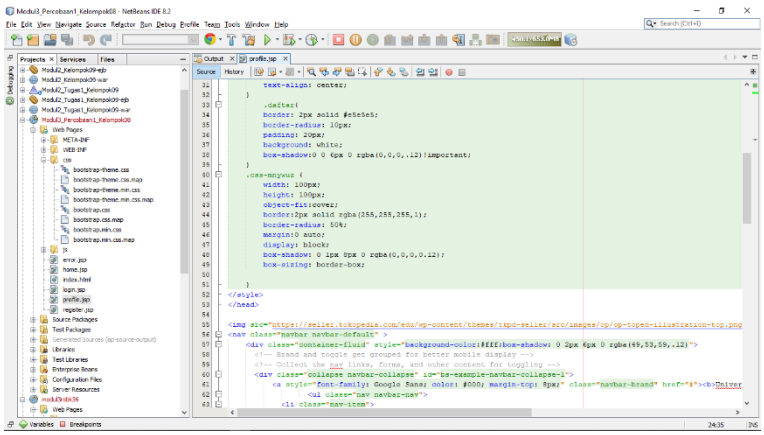
Dan ini adalah tampilan ketika halaman home informasi data ditambahkan kolom alamat serta adanya text field baru alamat yang digunakan untuk melakukan fungsi crud.



Gambar 4.65 Halaman home dengan kolom alamat terisi *value*

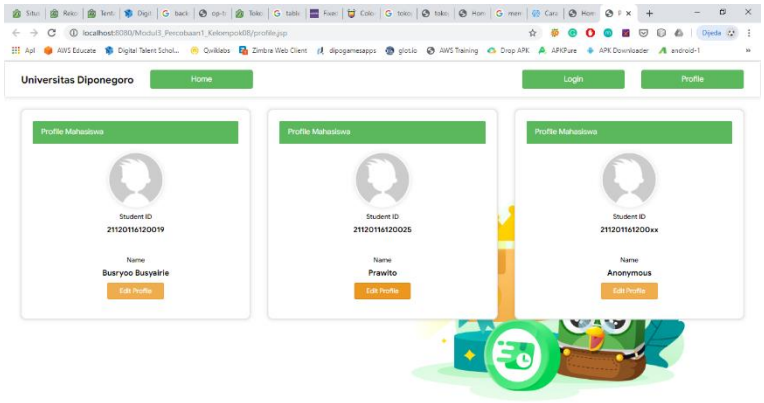
Pada gambar ini menampilkan ketika kolom alamat ini sudah terisi value dan text field juga dapat berfungsi sebagai form untuk menambahkan nilai.

2. Tugas Penambahan Halaman Profile



Gambar 4.66 Tampilan file profile.jsp

Membuat file profile.jsp yang digunakan sebagai untuk menampilkan tampilan profile dari praktikan.

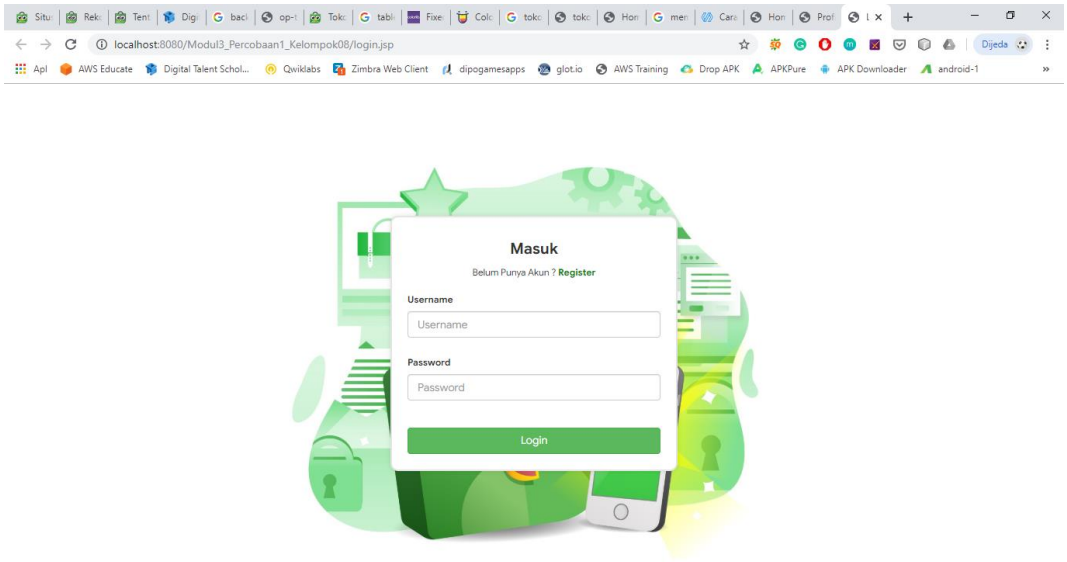


Gambar 4.67 Halaman profile dengan beberapa perubahan tampilan

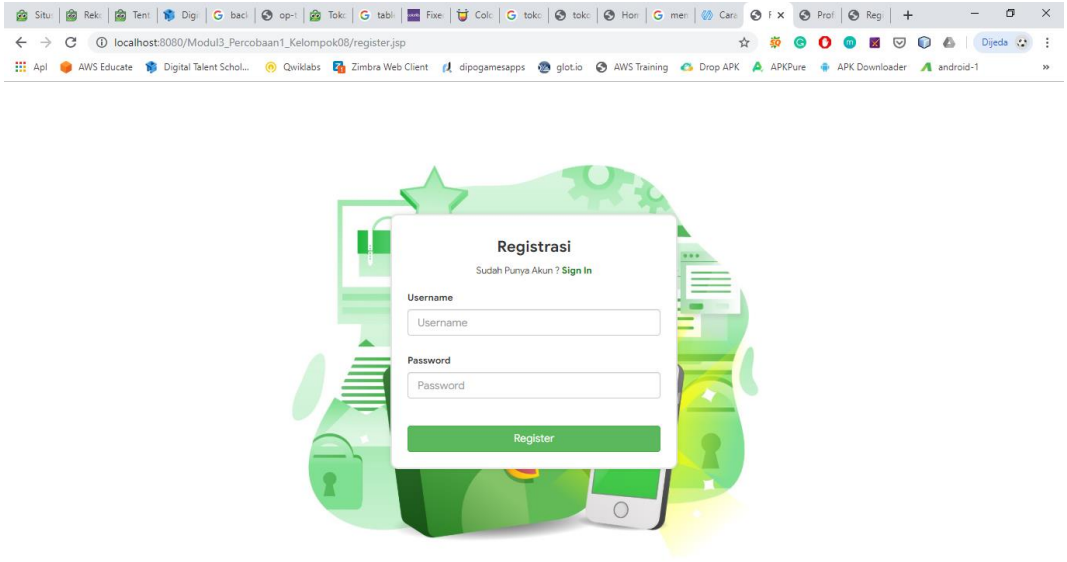
Dan ini adalah tampilan dari halaman profile.jsp dimana pada halaman ini di buat menjadi lebih menarik dengan tema Tokopedia, jadi menggunakan warna dasar hijau dan beberapa gambar dari Tokopedia.

3. Memperbaiki Tampilan

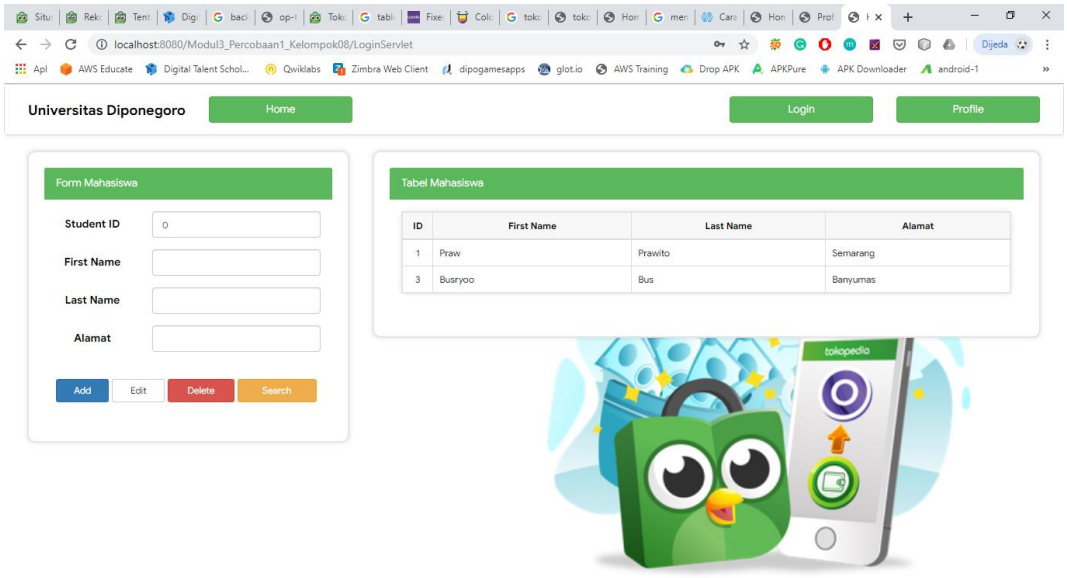
Semua menggunakan tema tokopedia, jadi memang ini disamakan dengan tokopedia.



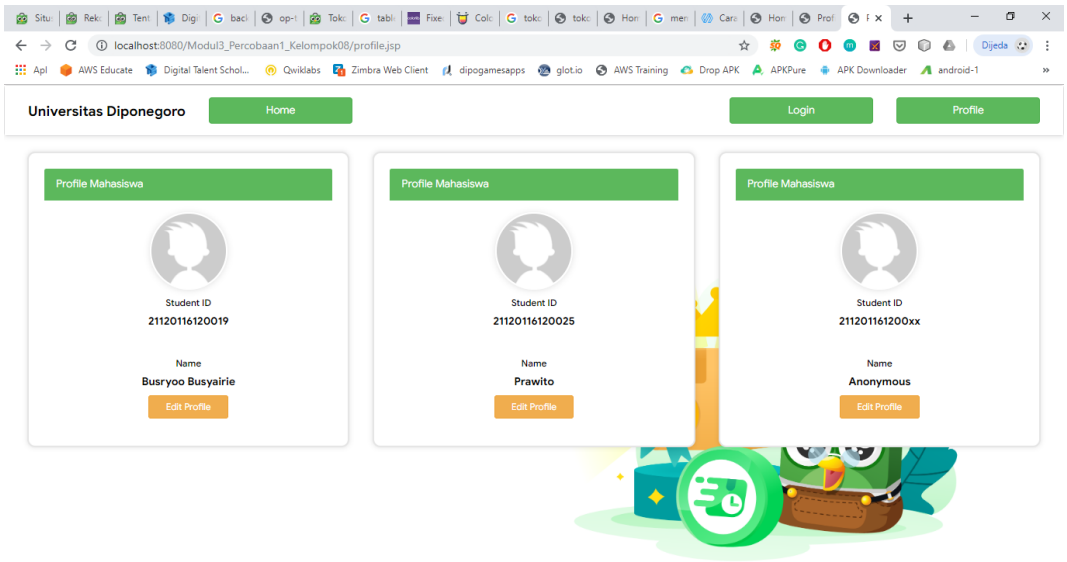
Gambar 4.68 Halaman Login



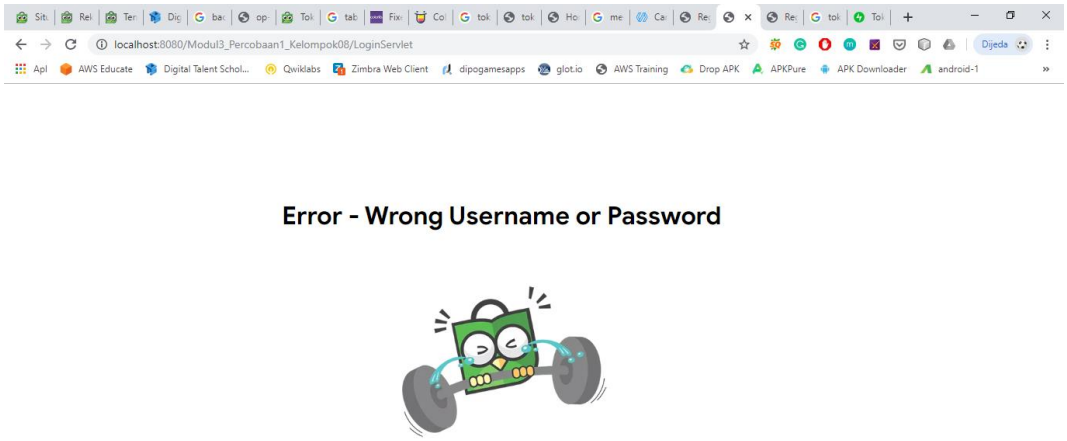
Gambar 4.69 Halaman Registrasi



Gambar 4.70 Halaman Home



Gambar 4.71 Halaman Profile



Gambar 4.72 Halaman Error

Link Github : https://github.com/Prawito-17/PraktikumRSBKKel08.git

Nama File : MODUL 3 APLIKASI CRUD DENGAN EJB, JPA, dan MySQL

4.6 Kesimpulan

1. Selain itu, manfaat JPA juga bisa dapat dengan mudah mengelola transaksi dengan API.
2. *Entity class* ini berisi tentang entity-entity apa saja yang digunakan dalam sebuah *class*.
3. Kelebihan JPA yang cukup bermanfaat adalah tidak perlu membuat *query* untuk manipulasi data.
4. Fungsi servlet adalah untuk membuat web dinamis bagi *user* serta fungsi jsp digunakan untuk memberikan kode untuk tampilan dan beberapa method post atau get.
5. *Session bean* digunakan untuk membuat bean yang memiliki bisnis proses per *session* saja. Serta *controller* digunakan untuk menghubungkan antara model dengan dao.