**BAB V**

**JAVA SERVER FACES**

* 1. Tujuan

1. Praktikan mengenal *Framework* JSF
2. Praktikan mengetahui penggunaan Java Server Faces
3. Praktikan mampu membuat aplikasi CRUD sederhana dengan JSF
4. Praktikan memahami penggunaan Tomcat Server
5. Praktikan memahami cara menghubungkan program pada java dengan MySQL Connector
   1. Dasar Teori
      1. NetBeans IDE

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama. Saat ini terdapat dua produk : *NetBeans IDE* dan *NetBeans Platform.*

The NetBeans *IDE* adalah sebuah lingkungan pengembangan - sebuah kakas untuk pemrogram menulis, mengompilasi, mencari kesalahan dan menyebarkan program. *Netbeans IDE* ditulis dalam Java - namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan.

(Sumber: https://netbeans.org/index\_id.html diakses pada 17 November 2019)

* + 1. Java Server Faces

*JavaServer Faces (JSF)* adalah spesifikasi Java untuk membangun antarmuka pengguna untuk aplikasi web. JSF merupakan bagian dari *Java* *Platform, Enterprise Edition. JSF 2* menggunakan Facelets sebagai sistem templat defaultnya. Teknologi tampilan lainnya, seperti XUL, juga dapat digunakan. Sementara itu, JSF 1.x menggunakan *JavaServer Pages* (JSP) sebagai sistem templat *defaultnya*.

(Sumber: https://id.wikipedia.org/wiki/JavaServer\_Faces diakses pada 17 November 2019)

* + 1. Tomcat Server

Dijelaskan bahwa apache tomcat open source merupakan suatu wadah aplikasi web berbasis java yang diciptakan untuk menjalankan servlet dan JSP (*Java Server Pages*) aplikasi web. *Apache* tomcat dibuat oleh Apache-Jakarta namun, karena popularitasnya sekarang proyek apache tomcat terpisah dimana apache tomcat didukung dan kembangkan oleh sekelompok relawan dari *open source java* komunitas. (Vukotic & Goodwill, 2011)

(Sumber: https://aminawm.wordpress.com/pengertian-apache-tomcat-server/ diakses pada 17 November 2019)

* + 1. Session Bean

Session beans ditujukan untuk menangani interaksi dengan aplikasi client. Ada dua jenis *session* beans, yaitu *stateless* dan *stateful*. Stateless, sesuai dengan namanya, tidak menyimpan variabel yang sedang digunakan *user*. *Variabel* apapun yang dipassing ke *stateless session beans* akan hilang pada saat *method* selesai dijalankan. *Stateful*, sebaliknya, mirip dengan aplikasi web yang mengaktifkan dukungan *session*. *Stateful session beans* dapat menyimpan data user.

(Sumber: http://alibastomi.blogspot.com/2008/06/enterprise-java-beans-ejb.html diakses pada 17 November 2019)

* + 1. MySQL Connector

MYSQL Connector adalah merupakan *Framework* yang dibuat oleh MYSQL sendiri yang dapat mendukung beberapa bahasa pemrograman, dengan menggunakan MYSQL *Connector* kita dapat mengubungkan MYSQL dengan berbagai bahasa pemrograman yang lain

Berikut adalah beberapa pemrograman yang bisa di koneksikan dengan MYSQL server menggunakan MYSQL Connector :

* ADO.NET Driver for MySQL (Connector/NET)
* ODBC Driver for MySQL (Connector/ODBC)
* JDBC Driver for MySQL (Connector/J)
* Python Driver for MySQL (Connector/Python)
* C++ Driver for MySQL (Connector/C++)
* C Driver for MySQL (Connector/C)
* C API for MySQL (mysqlclient)

(Sumber: http://mysqlvisualbasic.blogspot.com/2015/01/mysql-connector.html diakses pada 17 November 2019)

* + 1. GitHub

GitHub adalah layanan penginangan web bersama untuk proyek pengembangan perangkat lunak yang menggunakan sistem pengontrol versi Git dan layanan hosting internet. Hal ini banyak digunakan untuk kode komputer. Ini memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan bug, permintaan fitur, manajemen tugas, dan wiki untuk setiap proyek.

GitHub menawarkan paket repositori pribadi dan gratis pada akun yang sama dan digunakan untuk proyek perangkat lunak sumber terbuka. Pada bulan April 2017, GitHub melaporkan bahwa mereka mempunyai lebih dari 20 juta pengguna dan lebih dari 57 juta repositori, menjadikannya layanan terbesar dari kode sumber di dunia.

(Sumber : https://id.wikipedia.org/wiki/GitHub diakses pada 17 November 2019)

* + 1. Xampp

XAMPP adalah software *web server apache* yang di dalamnya tertanam server MySQL yang didukung dengan bahasa pemrograman PHP untuk membuat *website* yang dinamis. XAMPP dikembangkan dari sebuah tim proyek bernama *Apache* *Friends*, yang terdiri dari Tim Inti (Core Team), Tim Pengembang (Development Team) &amp; Tim Dukungan (Support Team). XAMPP sendiri mendukung dua sistem operasi yaitu windows dan Linux.

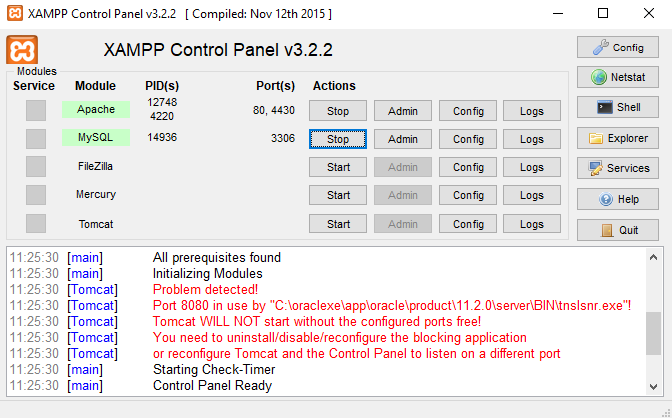
Mengenal bagian XAMPP yang biasa digunakan pada umumnya:

* htdoc adalah folder tempat meletakkan berkas-berkas yang akan dijalankan, seperti berkas PHP, HTML dan skrip lain.
* phpMyAdmin merupakan bagian untuk mengelola basis data MySQL yang ada dikomputer. Untuk membukanya, buka browser lalu ketikkan alamat http://localhost/phpMyAdmin, maka akan muncul halaman phpMyAdmin.
* Control Panel yang berfungsi untuk mengelola layanan (*service*) XAMPP. Seperti menghentikan (*stop*) layanan, ataupun memulai (*start*).

(Sumber : https://id.wikipedia.org/wiki/XAMPP diakses pada 17 November 2019)

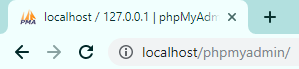
* 1. Langkah Kerja

1. Buka XAMPP, lalu Aktifkan Apache dan MySQL



Gambar 5.1 XAMPP

1. Buka Browser lalu ketikkan: localhost/phpMyAdmin



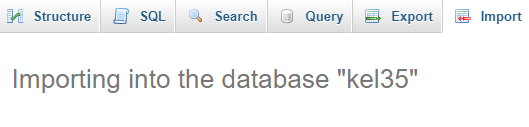
Gambar 5.2 localhost/phpMyAdmin

1. Buat database terlebih dahulu, database tersebut diberi nama kel35



Gambar 5.3 Create database

1. Kemudian, buka Tab impor



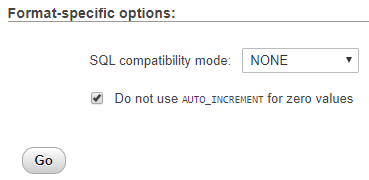
Gambar 5.4 Membuka tab import

1. Lalu, pilih Choose File dan masukkan Database yang sudah disediakan



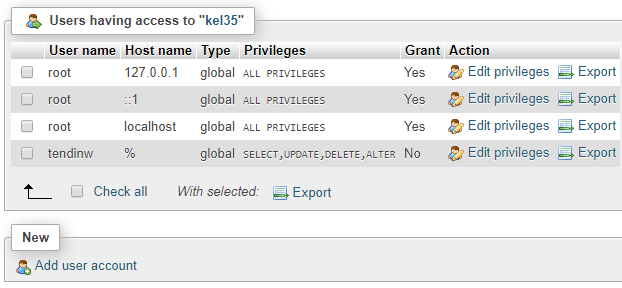
Gambar 5.5 db.sql

1. Dan kemudian pilih Go



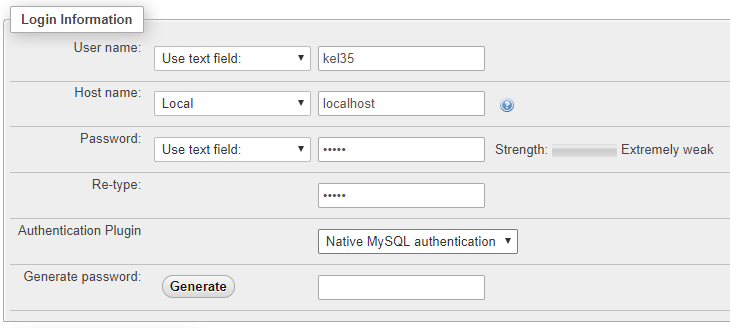
Gambar 5.6 Import database

1. Setelah itu buka tab Hak Akses/*privilege* dan pilih *add user account*



Gambar 5.7 Add user account

1. Setelah masuk ke halaman add user account, ikuti langkah seperti digambar

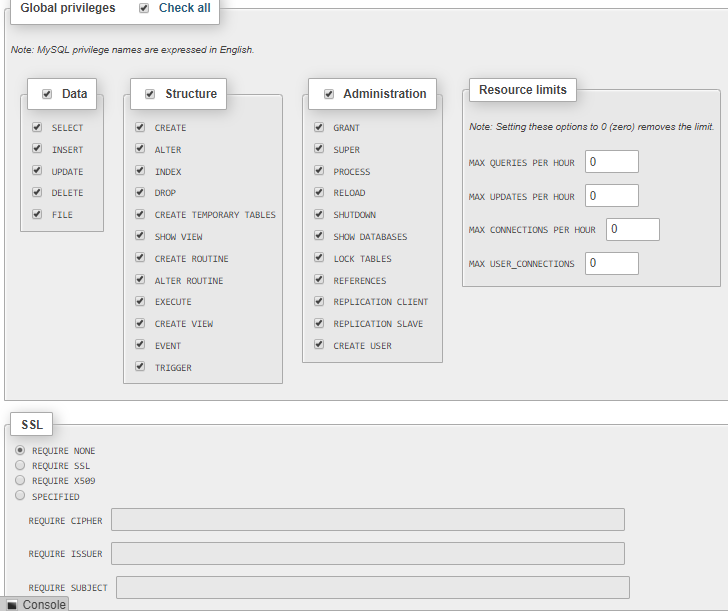


Gambar 5.8 Login information

Keterangan:

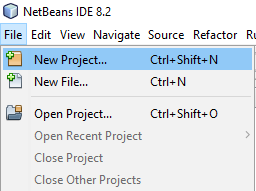
* Nama Pengguna : kel35
* Nama Pemilik : localhost (form sampingnya harus lokal)
* Kata sandi : kel35

Setelah itu pilih semua dan kirim



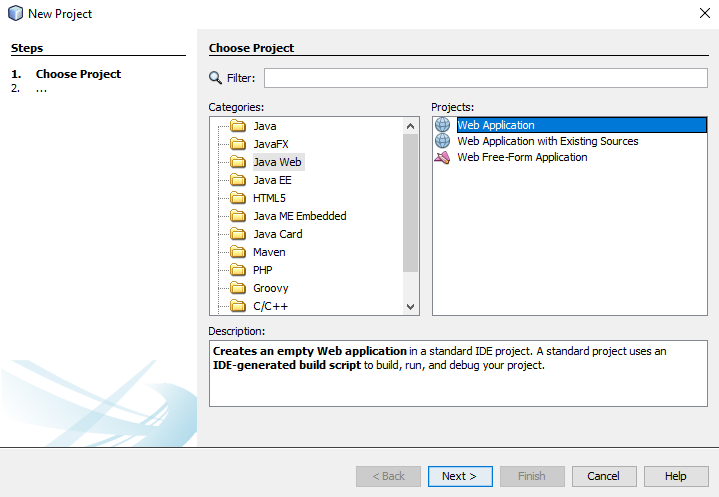
Gambar 5.9 Global privileges

1. Kemudian, buka Netbeans kalian dan pastikan sudah terinstall Tomcat
2. Kemudian pilih File, lalu New Project



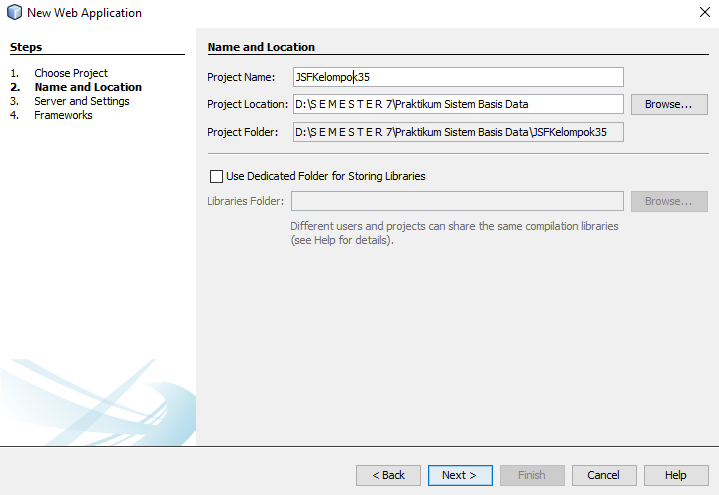
Gambar 5.10 New Project

1. Lalu pilih Java Web, kemudian pilih Web Application, kemudian Next



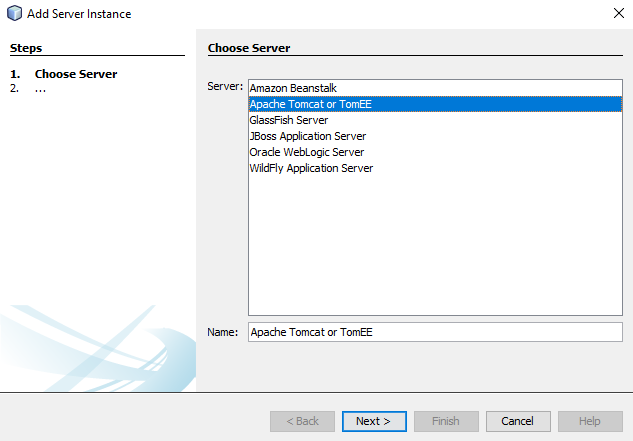
Gambar 5.11 Memilih projek Java Web – Web Application

1. Lalu, beri nama ‘JSFKelompokXX’, [ Ganti XX dengan No. Kelompok kalian ], kemudian Next lagi



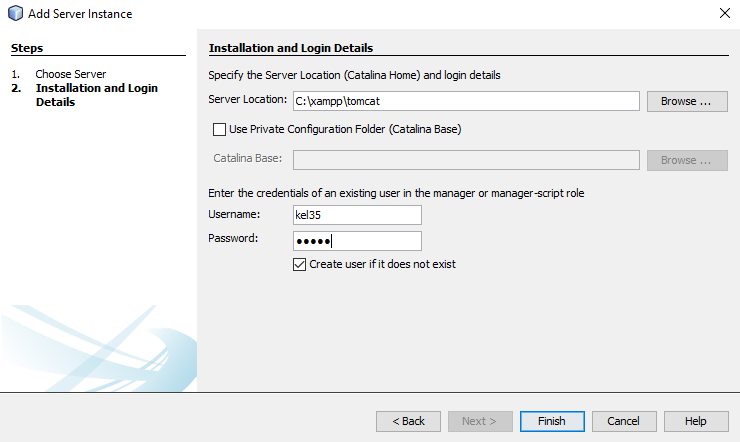
Gambar 5.12 Project name

1. Kemudian untuk Server, pilih Apache Tomcat or TomEE, dan untuk Java EE Version pilih Java EE 6 Web, jika belum ada tekan tombol add.
2. Terus pilih Apache Tomcat or TomEE. next



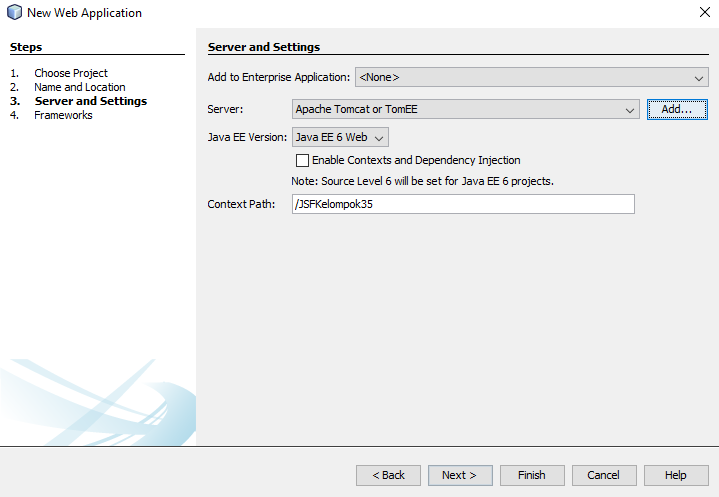
Gambar 5.13 Apache Tomcat

1. Maka akan ada form seperti gambar dibawah dan jangan lupa tekan tombol finish



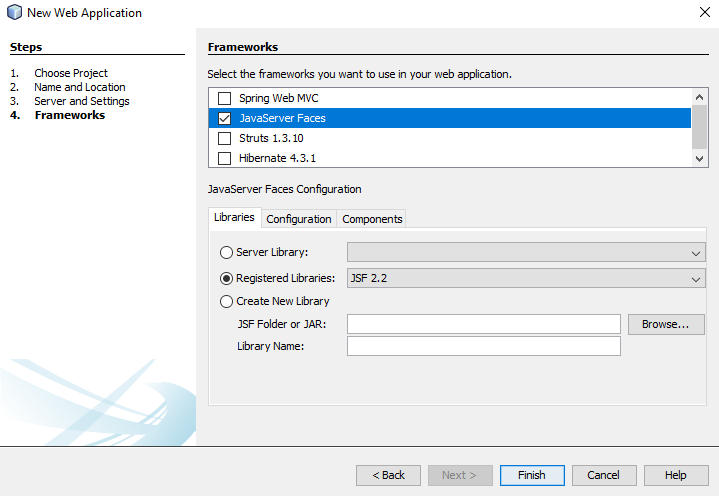
Gambar 5.14 Mengatur direktori lokasiserver

1. Setelah berhasil menambahkan server Apache Tomcat or TomEE, tekan next



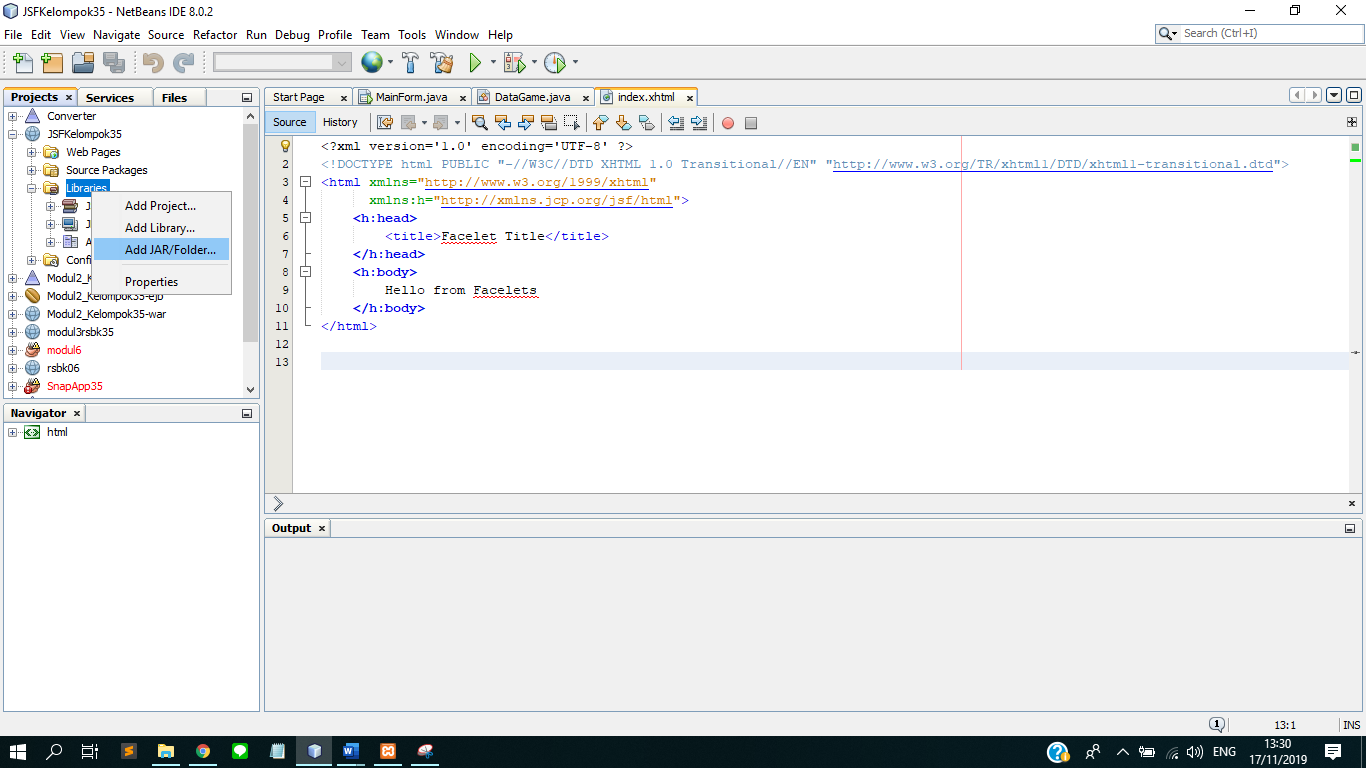
Gambar 5.15 Mengganti server dengan Apache Tomcat

1. Kemudian untuk Framework, centang JavaServer Faces, dan untuk pengaturan ikuti gambar di bawah. Lalu tekan Finish



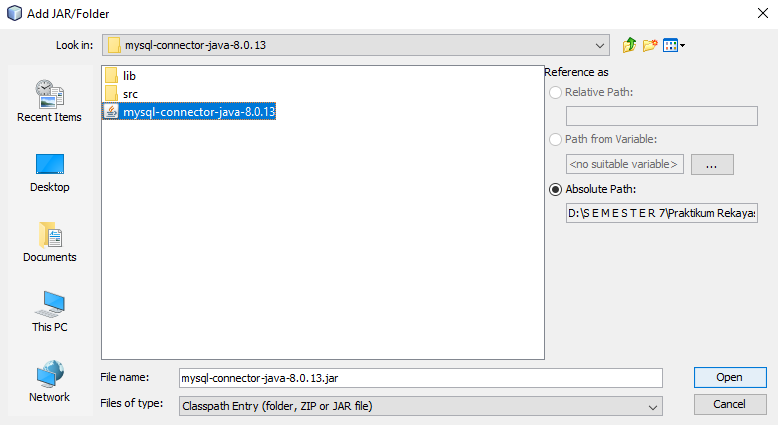
Gambar 5.16 JavaServer Faces

1. Selanjutnya, tambahkan MySQLConnector dengan cara klik kanan pada Folder Libraries dan pilih Add JAR/Folder



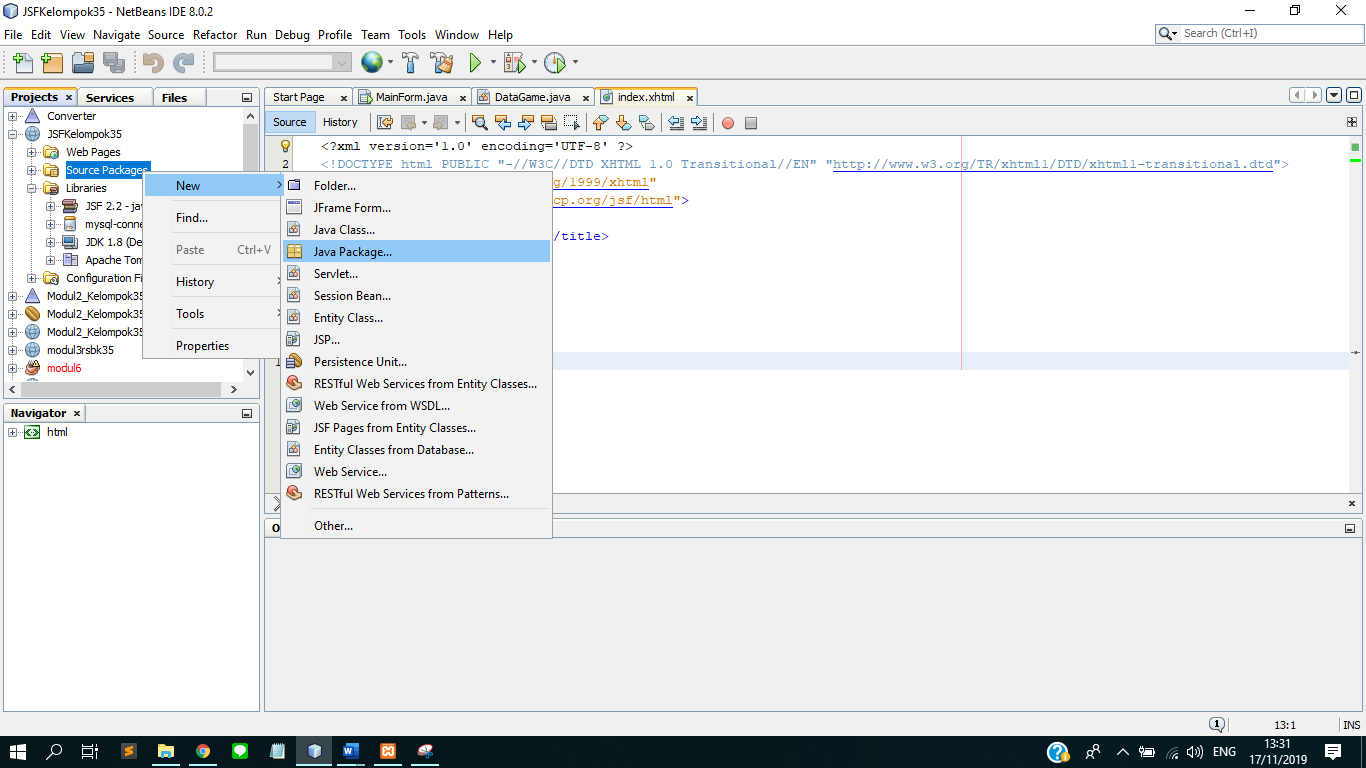
Gambar 5.17 *Add* JAR

1. Kemudian, cari lokasi dimana kalian menyimpan MySQL Connectornya, lalu tekan Open



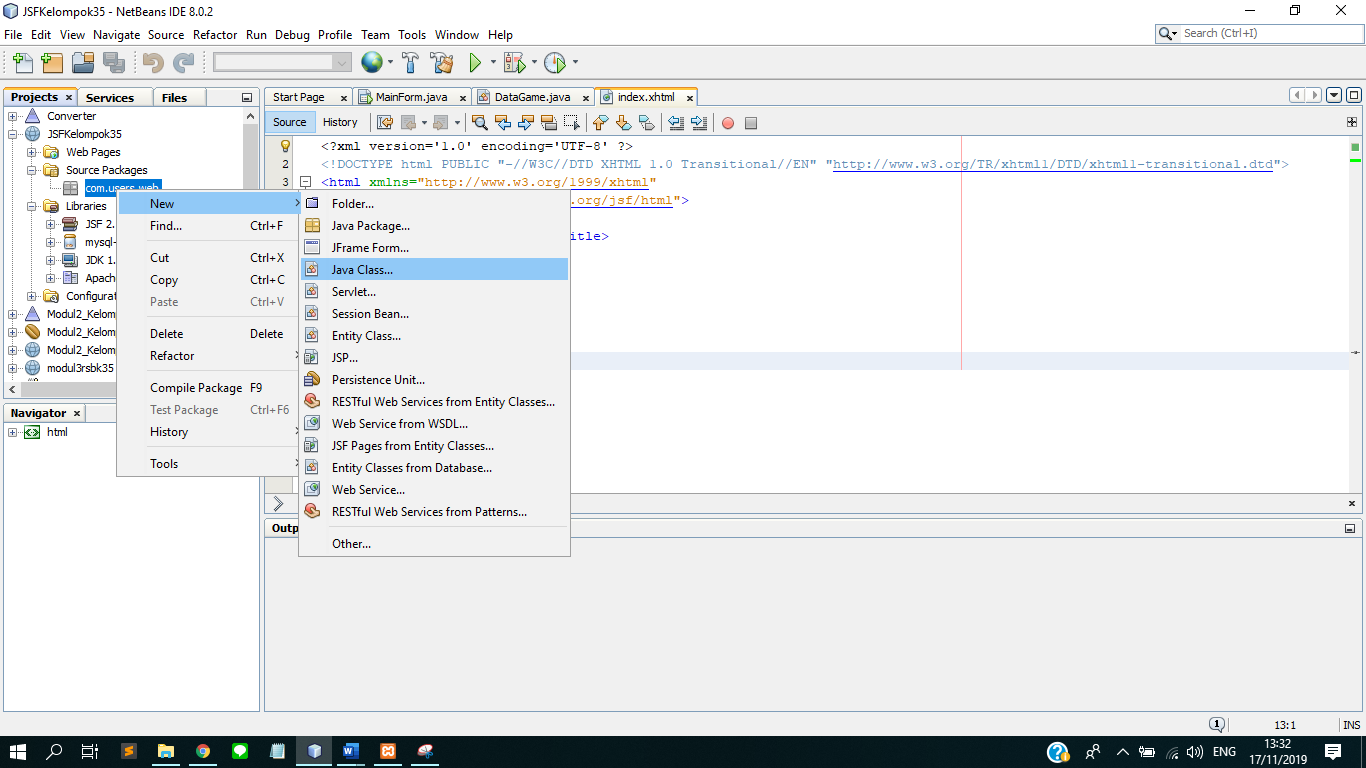
Gambar 5.18 MySQL Connector

1. Kemudian, pada bagian Source Package, klik kanan dan pilih New, lalu pilih Java Package



Gambar 5.19 *Create Java Package*

1. Lalu beri nama ‘com.users.web’, kemudian klik Finish. Lalu pada Folder com.users.web, klik kanan lalu pilih New dan pilih Java Class

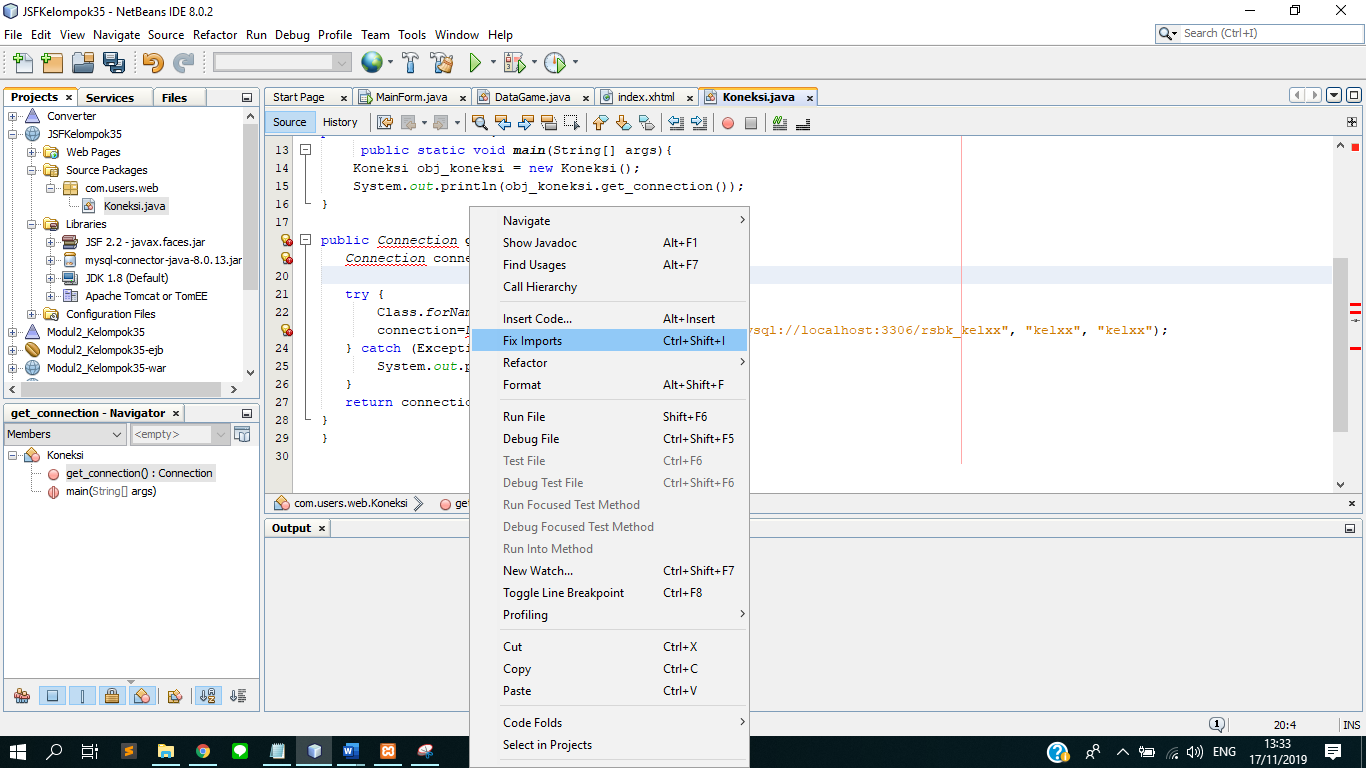


Gambar 5.20 *Create Java Class*

1. Kemudian berikana nama ‘Koneksi’, lalu tekan Finish
2. Lalu, selanjutnya. Copy kodingan ‘Koneksi.txt’ yang sudah disediakan ke dalam file Koneksi.java. Tepatnya, di dalam perintah public class Koneksi

|  |
| --- |
| public static void main(String[] args){  Koneksi obj\_koneksi = new Koneksi();  System.out.println(obj\_koneksi.get\_connection());  }    public Connection get\_connection(){  Connection connection = null;    try {  Class.forName("com.mysql.cj.jdbc.Driver");  connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/rsbk\_kelxx", "kelxx", "kelxx");  } catch (Exception e) {  System.out.println(e);  }  return connection;  } |

1. Lalu apabila ditemukan Error ketika selesai mengcopy, maka dapat klik kanan pada area kerja, dan pilih Fix Import



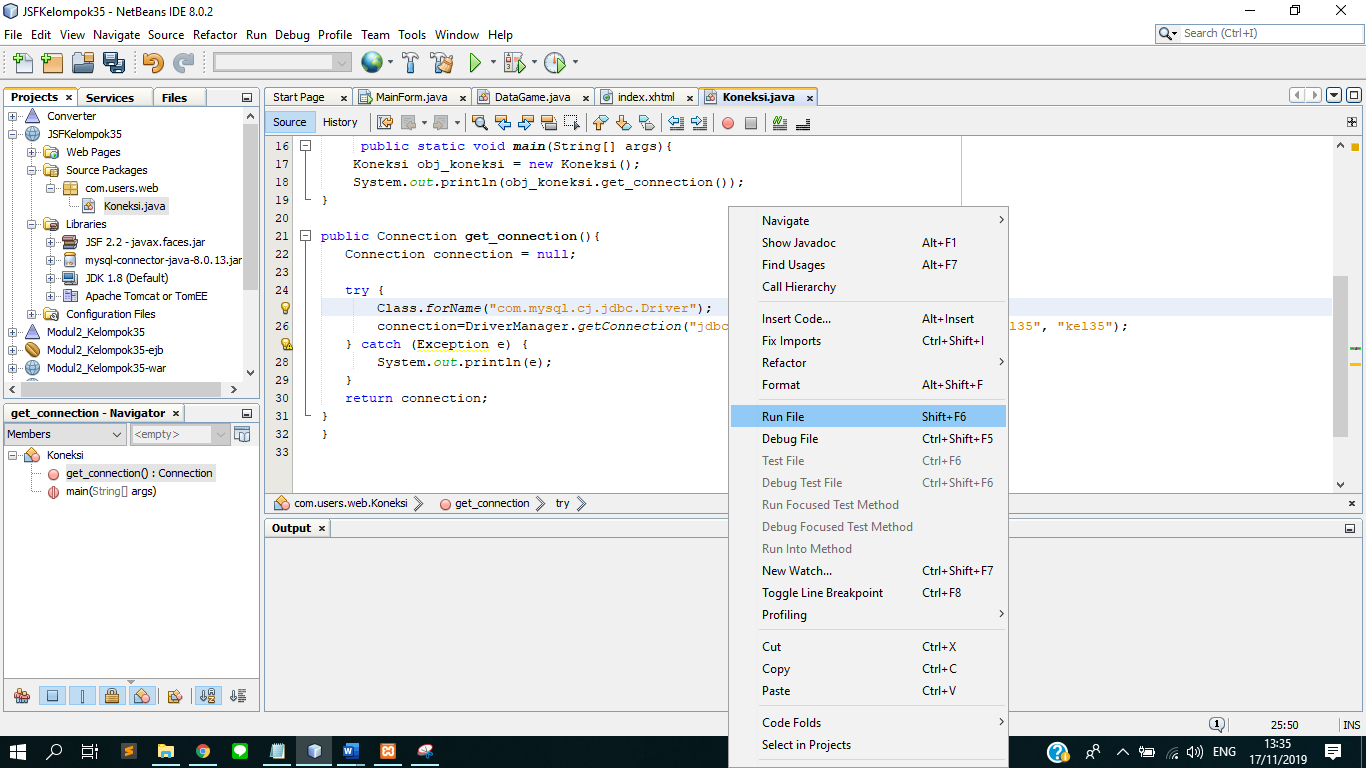
Gambar 5.21 *Fix Imports*

1. Selanjutnya sesuaikan source code dengan SS-an yang ada diatas dan sesuaikan dengan user hak akses yang sudah di buat sebelumnya



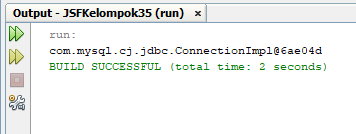
Gambar 5.22 *Username* dan *Password Database*

1. Kemudian klik kanan pada Area kerja dan pilih Run File untuk memastikan bahwa berhasil konek ke Database



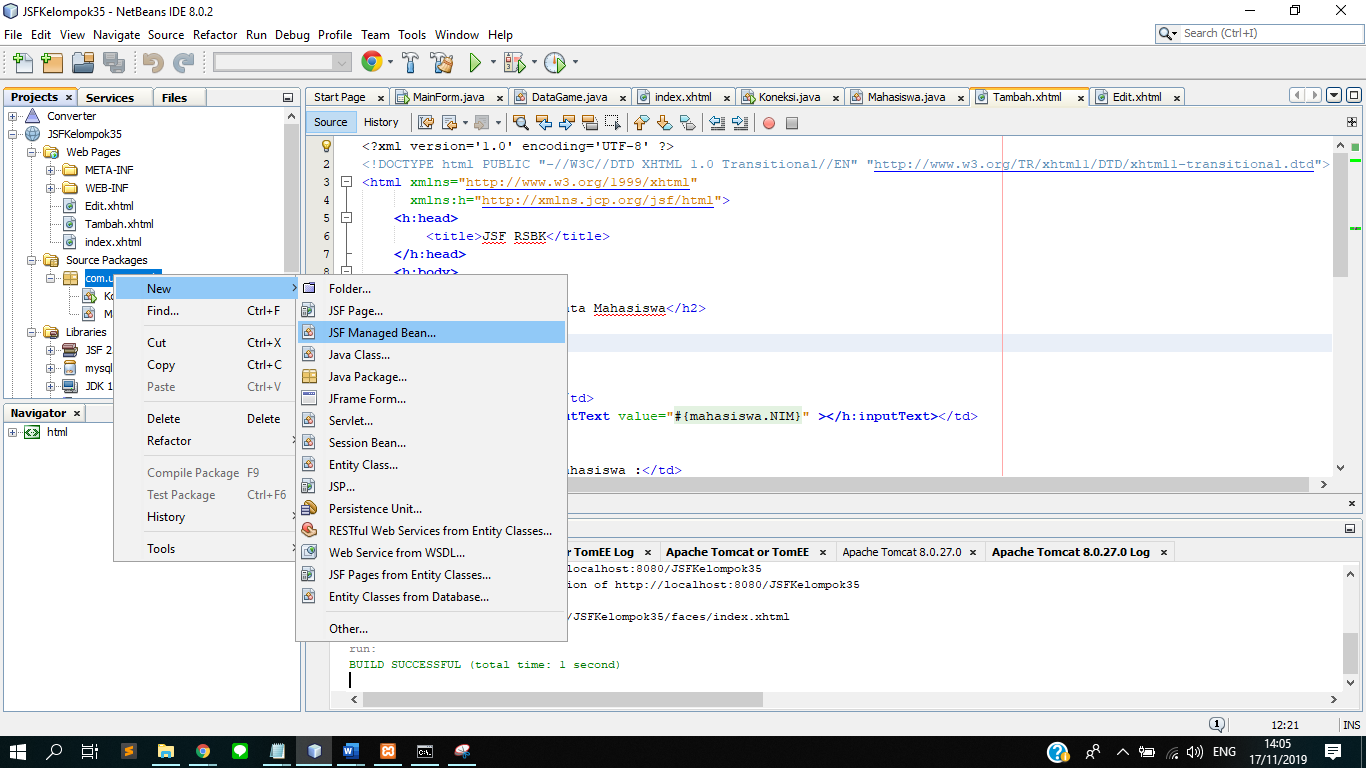
Gambar 5.23 *Run File*

1. Apabila berhasil, maka akan muncul tulisan seperti berikut :



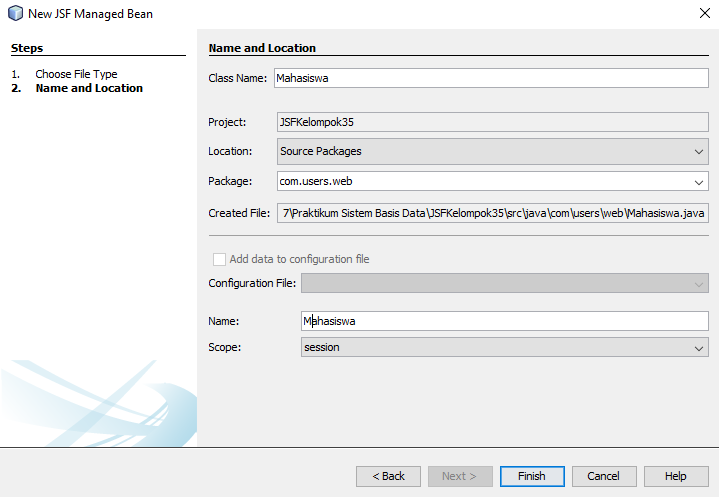
Gambar 5.24 *Build Successful*

1. Jika terjadi suatu kesalahan / error (TIME ZONE ISSUE) dapat menggunakan perintah yang ada di file solve.txt dengan cara dipaste ke database masing2
2. Atau dapat mengganti file mySql jdbc rar nya
3. Selanjutnya, klik kanan lagi pada: com.users.web. Lalu pilih New, dan pilih JSF Managed Bean. Jika pilihan tidak tersedia klik pilihan “Other”



Gambar 5.25 JSF Managed Bean

1. Lalu beri nama Mahasiswa, dan pada bagian Scope. Ubah menjadi Session. Kemudian klik Finish

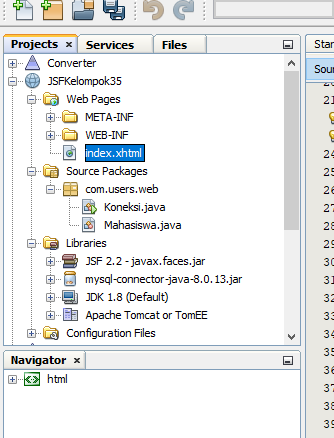


Gambar 5.26 *Class Name*

1. Selanjutnya, pada Area Kerja ‘Mahasiswa.java’ yang terletak dibawah tulisan 'package com.users.web', isi dengan Kodingan yang telah disediakan pada ‘Mahasiswa.txt’

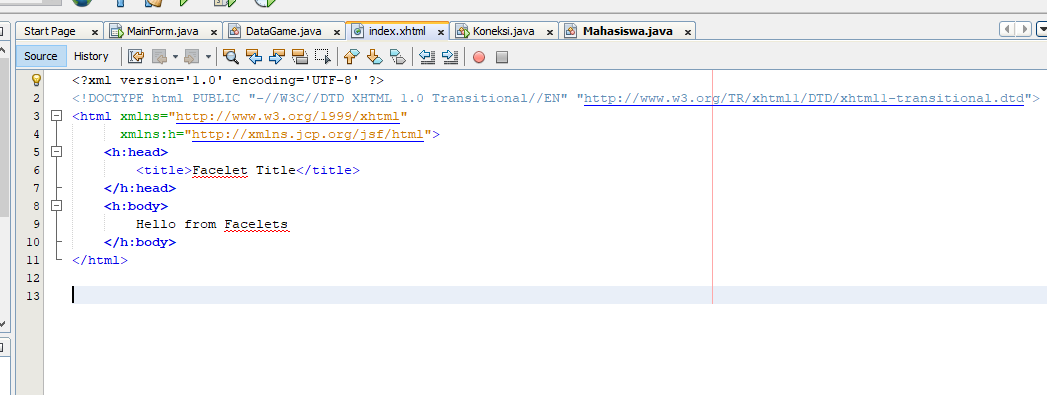
|  |
| --- |
| import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.Statement;  import java.util.ArrayList;  import java.util.Map;  import javax.faces.bean.ManagedBean;  import javax.faces.bean.RequestScoped;  import javax.faces.context.FacesContext;  /\*\*  \*  \* @author Hakushuu  \*/  @ManagedBean  @RequestScoped  public class Mahasiswa {  /\*\*  \* Creates a new instance of Mahasiswa  \*/    private String NIM;  public void setNIM(String NIM) {  this.NIM = NIM;  }  public String getNIM() {  return NIM;  }  private String NAMA;  public void setNAMA(String NAMA) {  this.NAMA = NAMA;  }  public String getNAMA() {  return NAMA;  }    private String PENJURUSAN;  public void setPENJURUSAN(String PENJURUSAN) {  this.PENJURUSAN = PENJURUSAN;  }  public String getPENJURUSAN() {  return PENJURUSAN;  }    private Map<String,Object> sessionMap = FacesContext.getCurrentInstance().getExternalContext().getSessionMap();  public String Edit\_Mahasiswa(){  FacesContext fc = FacesContext.getCurrentInstance();  Map<String,String > params = fc.getExternalContext().getRequestParameterMap();  String Field\_NIM = params.get("action");  try {  Koneksi obj\_koneksi = new Koneksi();  Connection connection = obj\_koneksi.get\_connection();  Statement st = connection.createStatement();  ResultSet rs = st.executeQuery("select \* from mahasiswa where NIM="+Field\_NIM);  Mahasiswa obj\_Mahasiswa = new Mahasiswa();  rs.next();  obj\_Mahasiswa.setNIM(rs.getString("NIM"));  obj\_Mahasiswa.setNAMA(rs.getString("Nama"));  obj\_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));  sessionMap.put("EditMahasiswa", obj\_Mahasiswa);  } catch (Exception e) {  System.out.println(e);  }  return "/Edit.xhtml?faces-redirect=true";  }  public String Delete\_Mahasiswa(){  FacesContext fc = FacesContext.getCurrentInstance();  Map<String,String> params = fc.getExternalContext().getRequestParameterMap();  String Field\_NIM = params.get("action");  try {  Koneksi obj\_koneksi = new Koneksi();  Connection connection = obj\_koneksi.get\_connection();  PreparedStatement ps = connection.prepareStatement("delete from mahasiswa where NIM=?");  ps.setString(1, Field\_NIM);  System.out.println(ps);  ps.executeUpdate();  } catch (Exception e) {  System.out.println(e);  }  return "/index.xhtml?faces-redirect=true";  }  public String Update\_Mahasiswa(){  FacesContext fc = FacesContext.getCurrentInstance();  Map<String,String> params = fc.getExternalContext().getRequestParameterMap();  String Update\_NIM= params.get("Update\_NIM");    try {  Koneksi obj\_koneksi = new Koneksi();  Connection connection = obj\_koneksi.get\_connection();  PreparedStatement ps = connection.prepareStatement("update mahasiswa set NIM=?, Nama=?, Penjurusan=? where NIM=?");  ps.setString(1, NIM);  ps.setString(2, NAMA);  ps.setString(3, PENJURUSAN);  ps.setString(4, Update\_NIM);  System.out.println(ps);  ps.executeUpdate();  } catch (Exception e) {  System.out.println(e);  }  return "/index.xhtml?faces-redirect=true";  }    public ArrayList getGet\_all\_mahasiswa() throws Exception{  ArrayList list\_of\_mahasiswa=new ArrayList();  Connection connection=null;  try {  Koneksi obj\_koneksi = new Koneksi();  connection = obj\_koneksi.get\_connection();  Statement st = connection.createStatement();  ResultSet rs = st.executeQuery("Select \* from mahasiswa");  while(rs.next()){  Mahasiswa obj\_Mahasiswa = new Mahasiswa();  obj\_Mahasiswa.setNIM(rs.getString("NIM"));  obj\_Mahasiswa.setNAMA(rs.getString("Nama"));  obj\_Mahasiswa.setPENJURUSAN(rs.getString("Penjurusan"));  list\_of\_mahasiswa.add(obj\_Mahasiswa);  }  } catch (Exception e) {  System.out.println(e);  }finally{  if(connection!=null){  connection.close();  }  }  return list\_of\_mahasiswa;  }    public String Tambah\_Mahasiswa(){  try {  Connection connection=null;  Koneksi obj\_koneksi = new Koneksi();  connection = obj\_koneksi.get\_connection();  PreparedStatement ps=connection.prepareStatement("insert into mahasiswa(NIM, Nama, Penjurusan) value('"+NIM+"','"+NAMA+"','"+PENJURUSAN+"')");  ps.executeUpdate();  } catch (Exception e) {  System.out.println(e);  }  return "/index.xhtml?faces-redirect=true";  }    public Mahasiswa() {}    } |

1. Selanjutnya, Expand bagian Web Pages, lalu kalian akan melihat file ‘index.xhtml’. Kemudian klik 2 kali pada File tersebut



Gambar 5.27 index.xhtml

1. Lalu akan tampil, tampilan seperti berikut

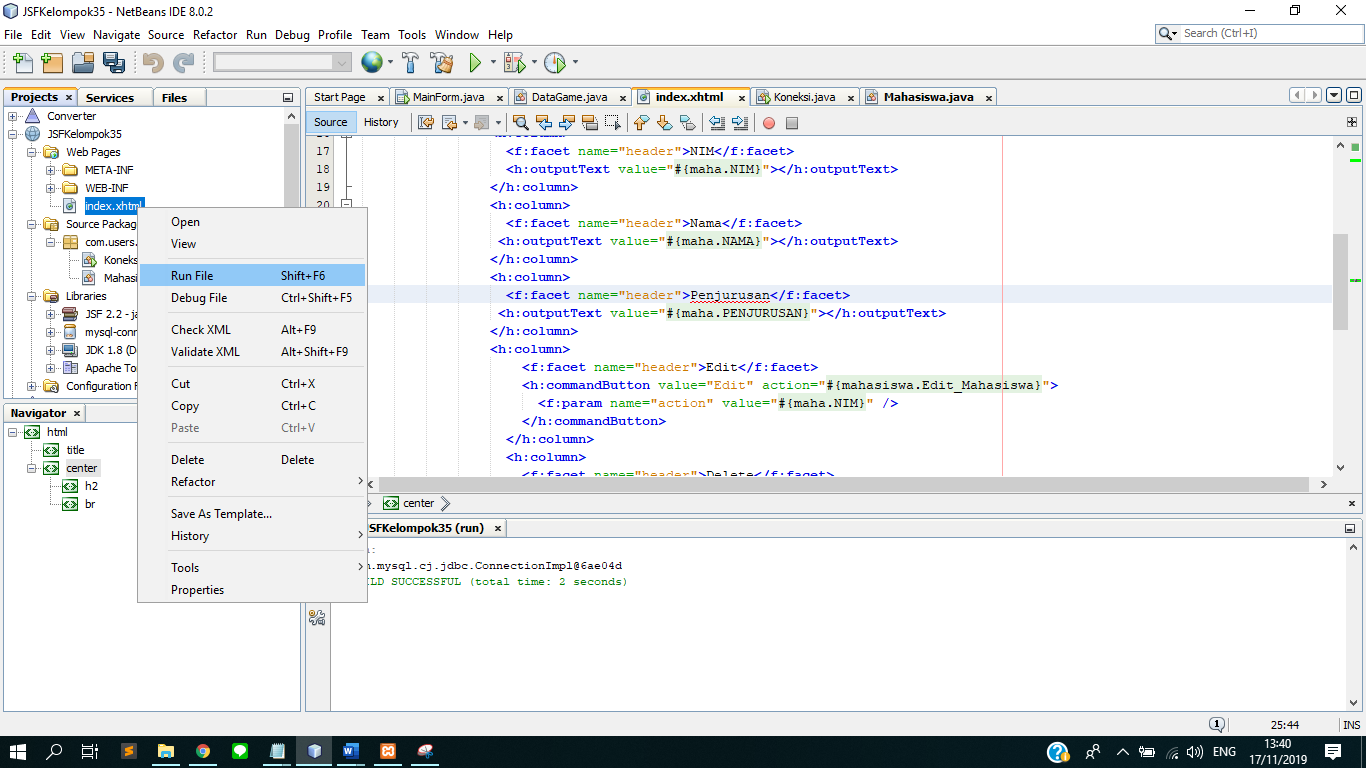


Gambar 5.28 *Source code* XHTML

1. Lalu copy kodingan yang ada pada file ‘Index.txt’ ke dalam file ‘index.xhtml’ tersebut

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:f="http://java.sun.com/jsf/core"  xmlns:h="http://xmlns.jcp.org/jsf/html">  <h:head>  <title>JSF RSBK</title>  </h:head>  <h:body>    <center>  <h2>Daftar Mahasiswa</h2>  <h:form>  <h:dataTable value="#{mahasiswa.get\_all\_mahasiswa}" var="maha" width="800px">  <h:outputText value="#{mahasiswa.NAMA}"/>  <h:column>  <f:facet name="header">NIM</f:facet>  <h:outputText value="#{maha.NIM}"></h:outputText>  </h:column>  <h:column>  <f:facet name="header">Nama</f:facet>  <h:outputText value="#{maha.NAMA}"></h:outputText>  </h:column>  <h:column>  <f:facet name="header">Penjurusan</f:facet>  <h:outputText value="#{maha.PENJURUSAN}"></h:outputText>  </h:column>  <h:column>  <f:facet name="header">Edit</f:facet>  <h:commandButton value="Edit" action="#{mahasiswa.Edit\_Mahasiswa}">  <f:param name="action" value="#{maha.NIM}" />  </h:commandButton>  </h:column>  <h:column>  <f:facet name="header">Delete</f:facet>  <h:commandButton value="Delete" action="#{mahasiswa.Delete\_Mahasiswa}">  <f:param name="action" value="#{maha.NIM}" />  </h:commandButton>  </h:column>  </h:dataTable>  <br></br>  </h:form>  <h:form>  <h:form>  Tambahkan Data Mahasiswa &nbsp;  <h:commandButton value="Tambahkan" action="Tambah.xhtml"></h:commandButton>  </h:form>  </h:form>  </center>  </h:body>  </html> |

1. Kemudian klik kanan pada file ‘Index.xhtml’ lalu pilih Run File



Gambar 5.29 *Run Index*

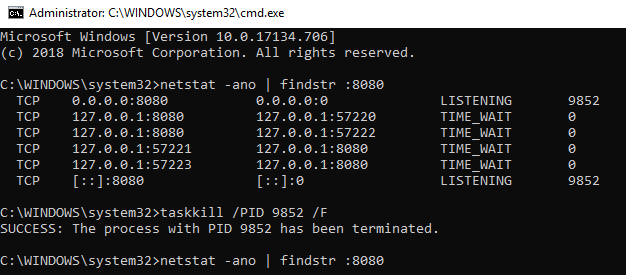
Jika terjadi suatu kesalahan / error “port is used” gunakan perintah

Run cmd as administrator

netstat -ano | findstr :8080

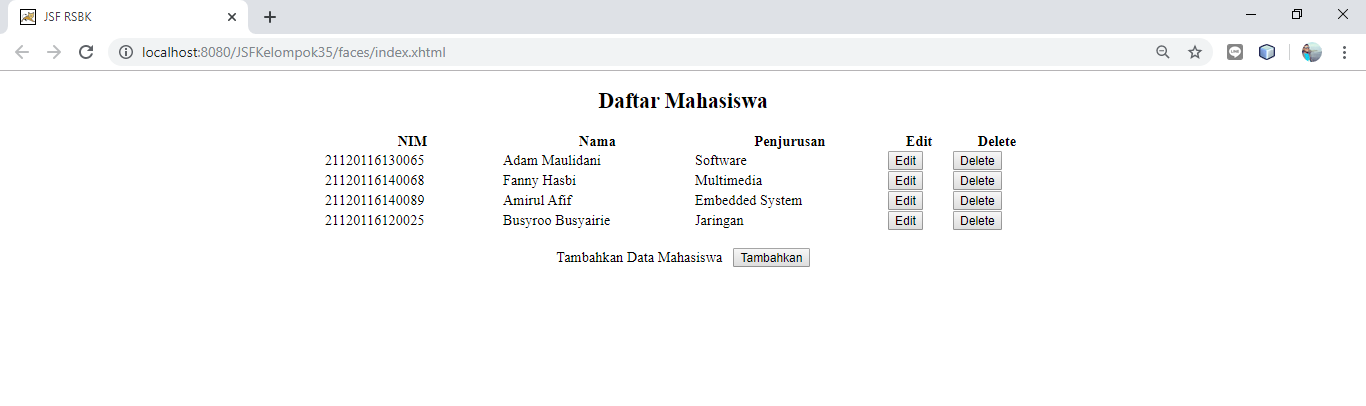
taskkill /PID 3748 /F

# 3748 adalah process id



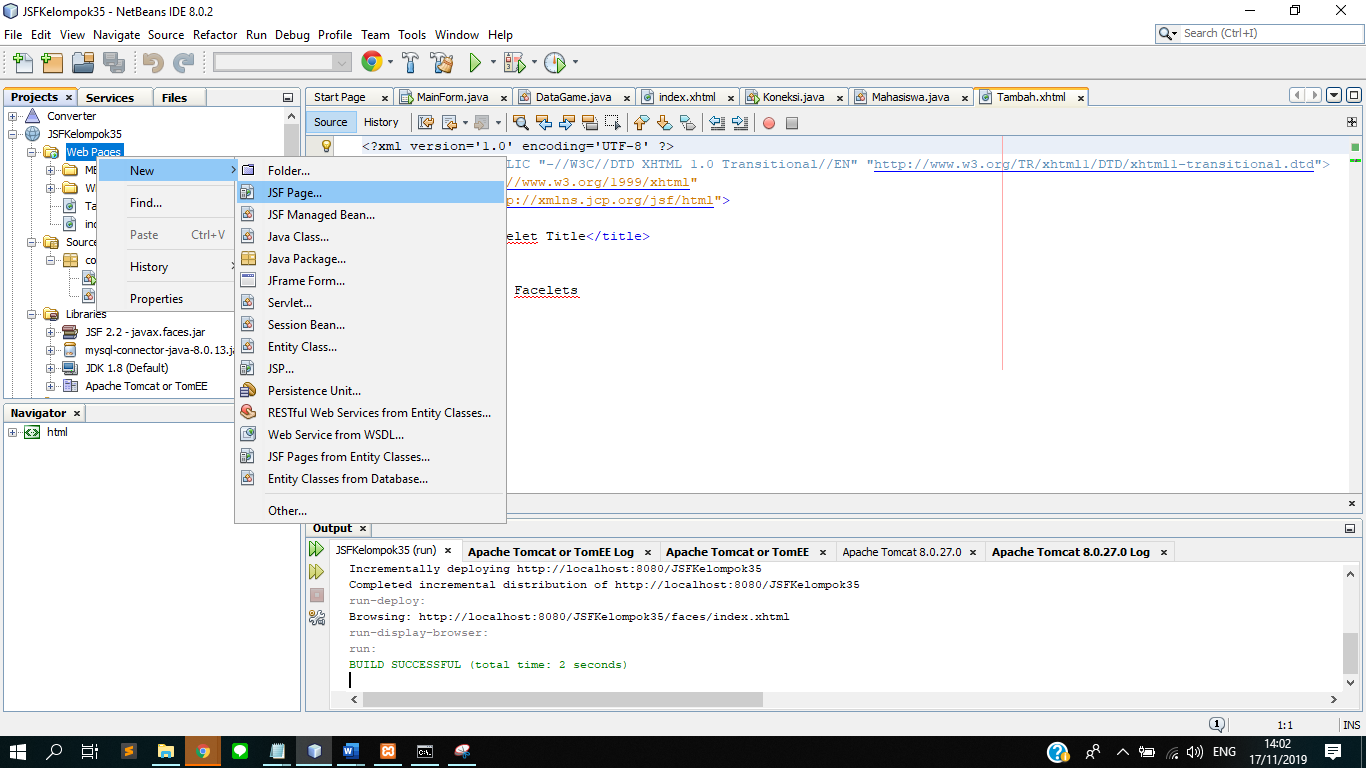
Gambar 5.30 Konfigurasi port

1. Maka, akan muncul tampilan sebagai berikut pada browser kalian:



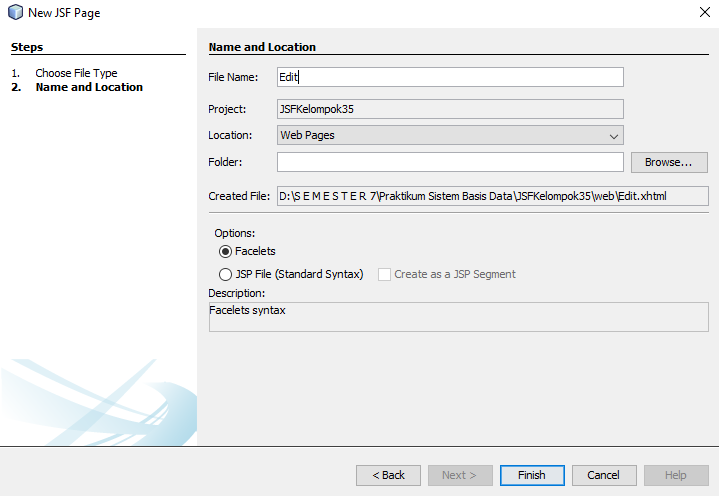
Gambar 5.31 Tampilan index.xhtml

1. Pada tampilan tersebut, button Edit dan Tambahkan, belum dapat digunakan, maka kita lanjut ke langkah selanjutnya
2. Kembali lagi ke Netbeans kalian, lalu klik kanan pada Folder Web Pages, lalu pilih New, dan pilih JSF Page

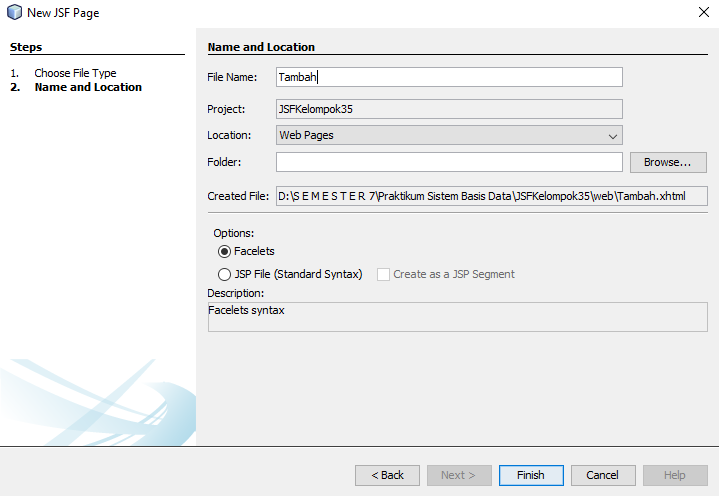


Gambar 5.32 *New* JSF Page

1. Buat 2 buah File JSF Page, dengan nama ‘Edit’ dan ‘Tambah’



Gambar 5.33 JSF *Edit*



Gambar 5.34 JSF Tambah

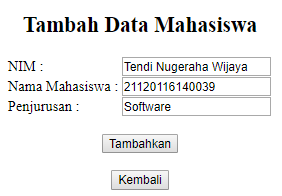
1. Kemudian copy kodingan yang ada pada file ‘Edit.txt’ ke dalam file ‘Edit.xhtml.

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:f="http://java.sun.com/jsf/core"  xmlns:h="http://xmlns.jcp.org/jsf/html">  <h:head>  <title>JSF RSBK</title>  </h:head>  <h:body>  <center>  <h:form>  <h1>Edit Data Mahasiswa</h1>    <table>  <tr>  <td>NIM :</td>  <td><h:inputText value="#{EditMahasiswa.NIM}" ></h:inputText></td>  </tr>  <tr>  <td>Nama Mahasiswa :</td>  <td><h:inputText value="#{EditMahasiswa.NAMA}" ></h:inputText></td>  </tr>  <tr>  <td>Penjurusan :</td>  <td><h:inputText value="#{EditMahasiswa.PENJURUSAN}"></h:inputText></td>  </tr>  </table>  <br></br>  <h:commandButton value="Update" action="#{EditMahasiswa.Update\_Mahasiswa}">  <f:param name="Update\_NIM" value="#{EditMahasiswa.NIM}" /></h:commandButton>  <br></br>  <br></br>  </h:form>  <h:form>  <h:commandButton value="Kembali" action="index.xhtml"></h:commandButton>  </h:form>    </center>  </h:body>  </html> |

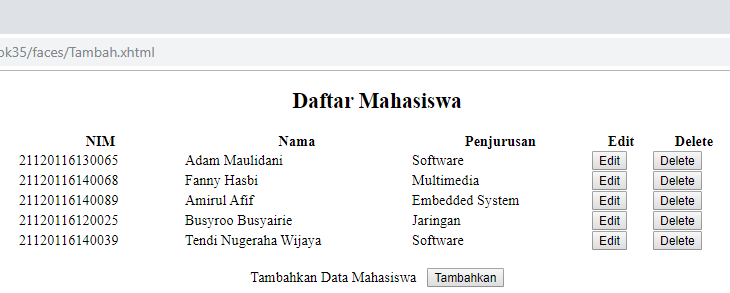
1. Dan juga copy kodingan yang ada pada file ‘Tambah.txt’ ke dalam file ‘Tambah.xhtml.

|  |
| --- |
| <?xml version='1.0' encoding='UTF-8' ?>  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:h="http://xmlns.jcp.org/jsf/html">  <h:head>  <title>JSF RSBK</title>  </h:head>  <h:body>  <center>  <h2>Tambah Data Mahasiswa</h2>    <h:form>  <table>  <tr>  <td>NIM :</td>  <td><h:inputText value="#{mahasiswa.NIM}" ></h:inputText></td>  </tr>  <tr>  <td>Nama Mahasiswa :</td>  <td><h:inputText value="#{mahasiswa.NAMA}" ></h:inputText></td>  </tr>  <tr>  <td>Penjurusan :</td>  <td><h:inputText value="#{mahasiswa.PENJURUSAN}"></h:inputText></td>  </tr>  </table>  <br></br>  <h:commandButton value="Tambahkan" action="#{mahasiswa.Tambah\_Mahasiswa}"></h:commandButton>  <br></br>  <br></br>  </h:form>  <h:form>  <h:commandButton value="Kembali" action="index.xhtml"></h:commandButton>  </h:form>  </center>    </h:body>  </html> |

1. Kemudian, coba kalian lakukan lagi Run File ‘index.xhtml’ . Dan, coba tekan button Edit atau Tambahkan. Maka button tersebut kini dapat digunakan



Gambar 5.35 Tambah data mahasiswa



Gambar 5.36 Data mahasiswa berhasil ditambahkan

* 1. Hasil dan Analisa

Pada Praktikum Modul 4 RSBK, praktikan diminta untuk membuat suatu projek berbasis JSF *file* yang mana pada file ini akan membuat suatu tampilan dengan *format* xhtml. Adanya c*lass* Koneksi.java dibuat dan digunakan untuk membangun koneksi antara aplikasi dengan *database*. *Database* yang digunakan yaitu MySQL. Beriktut adalah *source* *code* dari file koneksi.

|  |
| --- |
| public static void main(String[] args){  Koneksi obj\_koneksi = new Koneksi();  System.out.println(obj\_koneksi.get\_connection());  }    public Connection get\_connection(){  Connection connection = null;    try {  Class.forName("com.mysql.cj.jdbc.Driver");  connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/kel35", "kel35", "kel35");  } catch (Exception e) {  System.out.println(e);  }  return connection;  } |

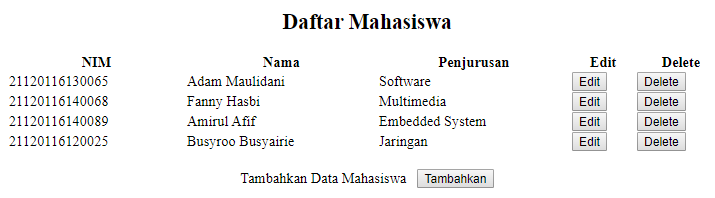
Kemudian pada *project* ini terdapat pula *file* Java *Managed Bean* yang digunakan sebagai *class* yang terdiri dari beberapa method untuk melakukan fungsi standar aplikasi yaitu CRUD (*Create* *Delete* dan *Update*). Terdapat pula 3 *file JSF Page* yang digunakan ntuk membuat suatu tampilan, yaitu index.xhtml, edit.xhtml dan tambah.xthml.

*File* index.xhtml memuat suatu tabel yang akan menampilkan data mahasiswa dan juga terdapat beberapa *button* dengan fungsi untuk tambah, *update* dan *delete*. Terdapat perintah <h:dataTable value="#{mahasiswa.get\_all\_mahasiswa}" var="maha" width="800px"> digunakan untuk membuat suatu tabel yang mana *value* pada tabel tersebut akan diisi dengan nilai dari class mahasiswa dan menjalankan *method* get\_all\_mahasiswa, pada *method* tersebut berisi *query* *select* *from* mahasiswa sehingga data yang akan ditampilkan sesuai dengan data yang ada pada tabel mahasiswa di *database*. Data pada kolom akan ditampilkan dengan cara <h:outputText value="#{mahasiswa.NAMA}"/> sesuai dengan parameter yang akan ditampilkan.

Terdapat juga *file* JSF *page* yang digunakan untuk melakukan tambah data yaitu tambah.xhtml pada *file* tersebut terdapat parameter data yang akan dijadikan sebagai inputan. Adanya *input* teks digunakan sebagai *input*an data yang akan dimasukkan kedalam *database* <td><h:inputText value="#{mahasiswa.NIM}" ></h:inputText></td> potongan source code ini berfungsi untuk menambahkan data pada variabel NIM yang sudah di integrasikan dengan *database*. Kemudian terdapat juga perintah <h:commandButton value="Tambahkan"action="#{mahasiswa.Tambah\_Mahasiswa}"></h:commandButton> untuk membuat aksi pada *button* tambah, sehingga ketika *button* ini diklik maka akan menjalankan *method* Tambah\_Mahasiswa pada *class* mahasiswa yang didalam *method* tersebut terdapat *query* untuk *insert* ke *database*.

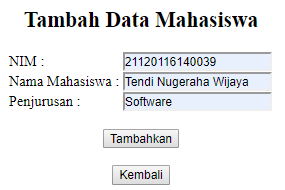
Pada *file* *edit*.xhtml terdapat *textfield* yang digunakan untuk *input*an teks dengan beberapa variabel sesuai dengan kolom yang ada pada *database*. Adanya perintah <td><h:inputText value="#{EditMahasiswa.NIM }" ></h:inputText></td> berfungsi untuk memanggil *value* dari variabel yang akan di*edit*, pada perintah tersebut berarti input teks tersebut akan diisi dengan NIM yang akan di*edit*, serta method Edit\_Mahasiswa akan dipanggil yang akan menjalankan fungsi *query* *select* dengan diikuti *where* NIM. *Button* <h:commandButton value="Update" action="#{EditMahasiswa.Update\_Mahasiswa}"> <f:param name="Update\_NIM" value="#{EditMahasiswa.NIM}" akan melakukan suatu *trigger* untuk menjalankan fungsi dari *method* Update\_Mahasiswa yang ada pada *class* Mahasiswa.java yang mana pada *method* tersebut terdapat *query* untuk melakukan *update* pada *database*. Dalam melakukan update diperlukan dengan adanya suatu parameter yang *unique* atau tidak dapat sama nilainya jadi yang akan dijadikan sebagai parameter untuk melakukan *update* yaitu pada NIM.

Untuk memastikan apakah *file* – *file* yang dibuat sudah benar maka dapat dilakukan dengan menjalankan file index.xhtml, ketik *file* *index*.xhtml dijalankan maka akan menampilkan data tabel mahasiswa dengan beberapa *button* yang akan menjalankan fungsi CRUD. Berikut adalah tampilan dari halaman *index*.xhtml.



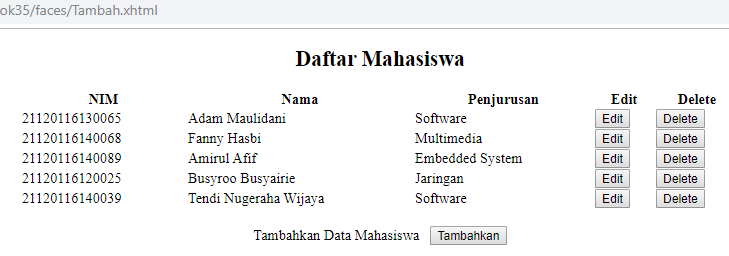
Gambar 5.37 Tampilan dari halaman index.xhtml

Ketika ingin menambahkan daftar mahasiswa maka dapat dilakukan dengan cara menekan pada *button* Tambahkan maka akan muncul halaman dengan 3 *input* teks yaitu NIM, Nama dan Penjurusan. Kemudian ketika *button* Tambahkan ditekan maka akan menjalankan fungsi dari Tambah\_Mahasiswa yang berisi *query* *insert* ke *database*.



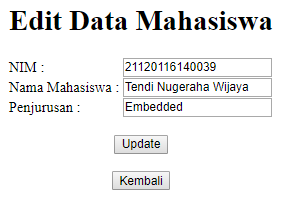
Gambar 5.38 Tampilan antarmuka halaman tambah

Kemudian setelah menambahkan data dan di klik *button* tambahkan maka data akan bertambah ke *database* dan muncul pada tabel di halaman *index*. Berikut adalah gambar setelah tambah data.



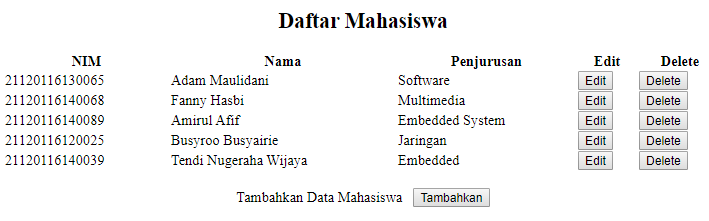
Gambar 5.39 Tampilan halaman setelah melakukan tambah data

Ketika ingin melakukan *edit* data maka dapat dilakukan dengan menekan tombol *edit* pada bagian kolom edit data yang ingin diubah. Kemudian ketika sudah ditekan tombol *edit* tersebut maka akan berpindah halaman ke *form* *edit* data. Terdapat NIM, Nama Mahasiswa dan Penjurusan yang akan diubah (secara otomatis data akan terisikan pada ke tiga input teks sesuai dengan data yang akan di *edit*). *Button* tersebut akan menjalankan method *edit* data sesuai dengan query yang ada pada class.



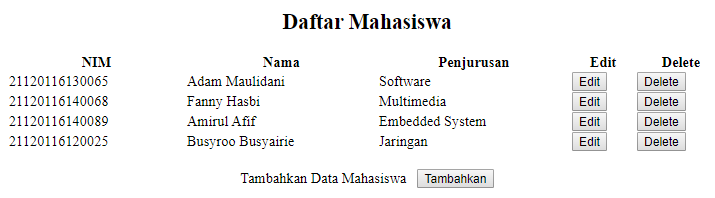
Gambar 5.40 Tampilan antarmuka halaman edit

Kemudian setelah menambahkan data dan di klik *button* *update* maka data akan diubah dan simpan ke database dan muncul pada tabel di halaman *index*. Berikut adalah gambar setelah *edit* data.



Gambar 5.41 Tampilan setelah melakukan *update* data

Delete dapat dilakukan dengan menekan *button* yang ada pada bagian kolom *delete* pada tabel. Ketika *button* tersebut ditekan maka akan menjalankan fungsi *query* *delete* pada *class*, parameter yang dijadikan untuk proses *delete* yaitu bagian NIM. Setelah *delete* dilakukan maka data akan berubah dan disimpan kedalam database. Berikut ini adalah contoh ketika *delete* pada data Tendi Nugeraha Wijaya.

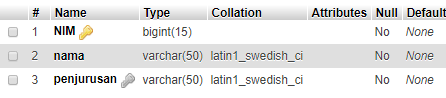


Gambar 5.42 Tampilan setelah *delete* data

* 1. Tugas

Pada bagian tugas, diminta untuk melakukan beberapa perubahan serta penambahan antara lain sebagai berikut:

* Membuat tabel baru bernama “penjurusan” yang nantinya akan menjadi penghubung dari *foreign key* pada tabel “mahasiswa” dengan kolom “penjurusan”



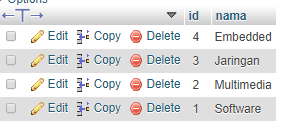
Gambar 5. 43 Struktur Tabel Mahasiswa



Gambar 5. 44 Struktur Tabel Penjurusan

Pada bagian ini, dibuat tabel baru guna menjadi referensi *foreign key* penjurusan. Sehingga setiap data penjurusan akan berasal dari tabel penjurusan.

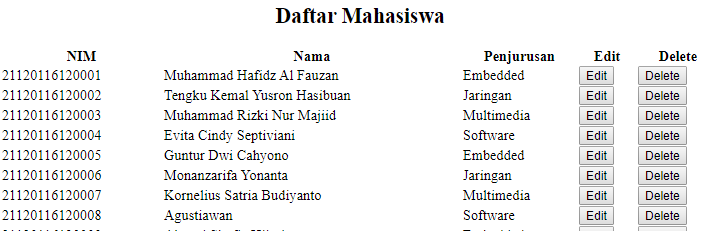
* Tabel Mempunyai 4 Penjurusan



Gambar 5. 45 Terdapat 4 Data pada Tabel Penjurusan

Pada bagian ini, ditambahkan 4 buah data pada tabel penjurusan beserta masing-masing id nya yang akan menjadi rujukan data penjurusan disetiap data mahasiswa.

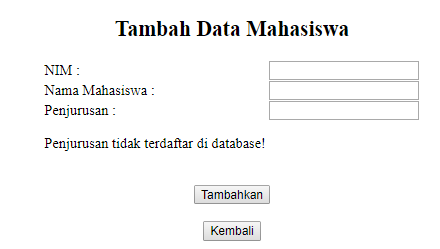
* Data perubahan penjurusan harus berasal dari tabel penjurusan



Gambar 5. 46 Semua Data Penjurusan Sesuai Dengan Tabel Penjurusan

Pada bagian ini, dibuktikan bahwa setiap data penjurusan yang ada pada data mahasiswa diambil dari data tabel penjurusan.

* Menunjukkan pemberitahuan bila penjurusan tidak ada di database maka data tidak dapat tersimpan.



Gambar 5. 47 Notifikasi Gagal Menemukan Data Penjurusan

Pada bagian ini, dibuat notifikasi sederhana ketika user salah memasukan data penjurusan, yang nantinya akan menampilkan notifikasi tersebut.

* Mengisi minimal 20 data mahasiswa



Gambar 5. 48Terdapat 20+ Data Mahasiswa

Pada bagian ini, sudah ditambahkan data mahasiswa sebanyak 20+ sehingga dapat membuktikan bahwa sistem berjalan dengan baik.

*Link GitHub Tugas :* [*https://github.com/perdanafavo/PraktikumRSBKkel35/tree/master/MODUL%204*](https://github.com/perdanafavo/PraktikumRSBKkel35/tree/master/MODUL%204)

* 1. Kesimpulan

1. *Connector* MySQL digunakan untuk membangun suatu koneksi antara program dengan *database* MySQL.
2. Untuk menambahkan *connector* MySQL dilakukan dengan cara menambahkannya pada *library* program.
3. Pada JSF xhtml untuk menampilkan *table* dilakukan dengan cara <dataTable></dataTable>.
4. Bootstrap CSS dapat diimplementasikan pada program dengan cara menambahkannya pada *folder* *resource* kemudian untuk memanggilnya dengan cara <h:openStylesheet>.
5. File JSF *Managed Bean* berguna sebagai *back-end* yang mana pada *file* ini terdapat *method* untuk menjalankan *query* CRUD.
6. Untuk menampilkan suatu *join* dapat dilakukan dengan cara membuat *method* dengan *query* *join*.
7. Fungsi dari required = true agar input teks tidak boleh dibiarkan kosong.
8. Untuk menambahkan pesan peringatan ketika *input* teks kosong yaitu dapat dilakukan dengan fungsi requiredMessage =” “.