

Membangun CRUD di Platform J2EE Dengan Menggunakan Hibernate-JSF-Primefaces

Fadlika Dita Nurjanto

fadlikadn@gmail.com

<http://fadlikadn.wordpress.com>

Lisensi Dokumen:

Copyright © 2003-2013 IlmuKomputer.Com

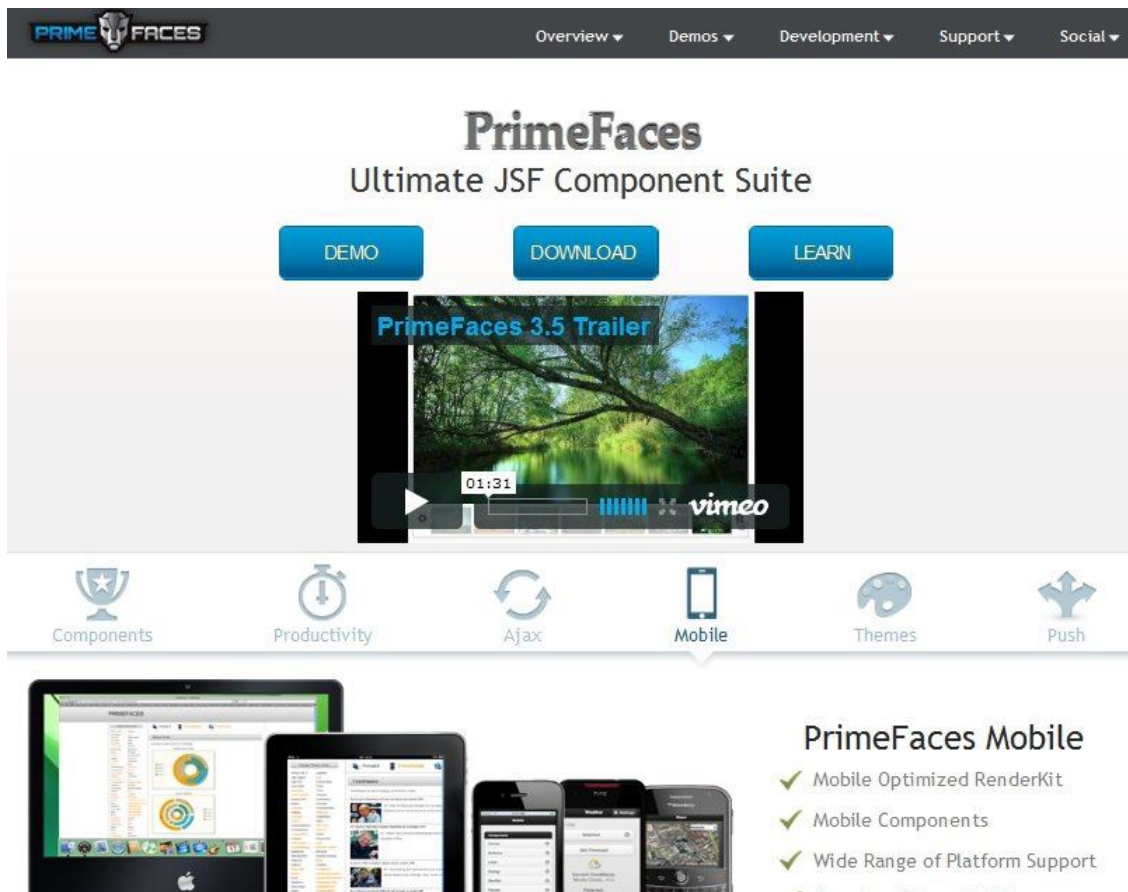
Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.

Seiring berkembangnya dunia usaha dan industri di Indonesia, semakin terbuka pula peluang penerapan teknologi di dalamnya. Dan teknologi informasi merupakan salah satu teknologi yang terbuka lebar untuk diimplementasikan di berbagai bidang.

Platform J2EE merupakan salah satu platform handal yang bisa digunakan dalam berbagai aplikasi enterprise. Platform J2EE menyediakan berbagai pilihan teknologi dan arsitektur bagi para *developer* untuk mengembangkan aplikasinya. Salah satu yang bisa digunakan adalah Hibernate, JSF, dan Primefaces. Hibernate merupakan sebuah library yang digunakan untuk Object Relation Mapping (ORM) antara objek-objek yang ada pada sebuah aplikasi. Untuk mempelajari hibernate, sumber lengkap yang bisa digunakan bisa dilihat di <http://www.hibernate.org/>. Pada arsitektur MVC, Hibernate menempati bagian Model yang berfungsi me-mapping-kan class model pada relasi database. Sedangkan JSF dan Primefaces merupakan bagian View yang nantinya akan berinteraksi langsung dengan user. JSF (Java Server Faces) didesain dengan tujuan mempermudah developer membangun aplikasi web dengan mudah. Dokumentasi lengkap tentang JSF bisa dilihat di <http://www.oracle.com/technetwork/java/javasee/javaserverfaces-139869.html>. Primefaces merupakan salah satu komponen JSF yang dapat digunakan untuk membangun aplikasi web J2EE yang memiliki tampilan interaktif dan menarik. Dokumentasi Primefaces dapat dilihat di <http://primefaces.org/>.

Dokumentasi Hibernate di <http://www.hibernate.org/>

Dokumentasi JSF di <http://www.oracle.com>



Dokumentasi Primefaces di <http://primefaces.org/>

Kombinasi Hibernate-JSF-Primefaces dapat digunakan untuk membangun aplikasi enterprise berbasis web yang solid. Dengan arsitektur MVC, aplikasi ini menjadi mudah untuk dikembangkan dan dimaintenance jika mengalami perubahan.

Untuk mempermudah penjelasan, akan ditunjukkan langsung dengan studi kasus langsung. Tool yang digunakan antara lain sebagai berikut :

- Netbeans IDE 7.2
- MySQL Database

Netbeans yang digunakan sebaiknya yang full, sehingga sudah support library Hibernate, JSF, dan Primefaces sekaligus. Buat database dengan mysql dengan menggunakan sintaks berikut :

Nama database : sekolah

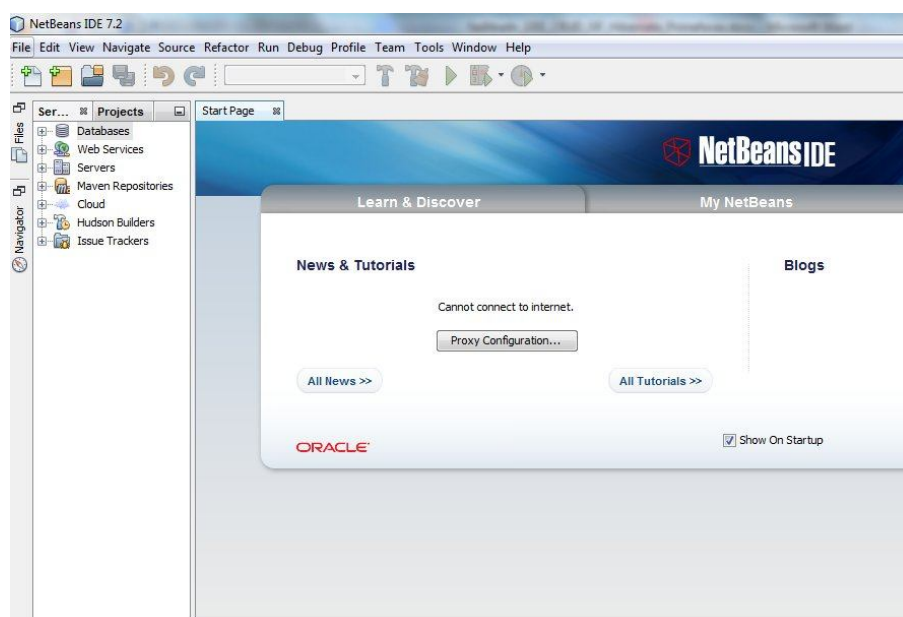
```
CREATE TABLE IF NOT EXISTS `siswa` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `nama` varchar(50) NOT NULL,  
  `sekolah` varchar(50) NOT NULL,  
  `telp` varchar(30) NOT NULL,  
  `email` varchar(30) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;  
  
--  
-- Dumping data untuk tabel `siswa`  
--  
  
INSERT INTO `siswa` (`id`, `nama`, `sekolah`, `telp`, `email`) VALUES  
(1, 'Fadlika Dita Nurjanto', 'SMK 1 Wonosobo', '085729666888',  
'fadlikadn@gmail.com'),  
(2, 'Hani Ramadhan', 'SMA 5 Surabaya', '085731781982', 'hani42@gmail.com'),  
(4, 'Muhammad Aminudin Rahman', 'SMA 2 Surabaya', '085731892012',  
'muara@gmail.com');
```

Database yang dihasilkan :

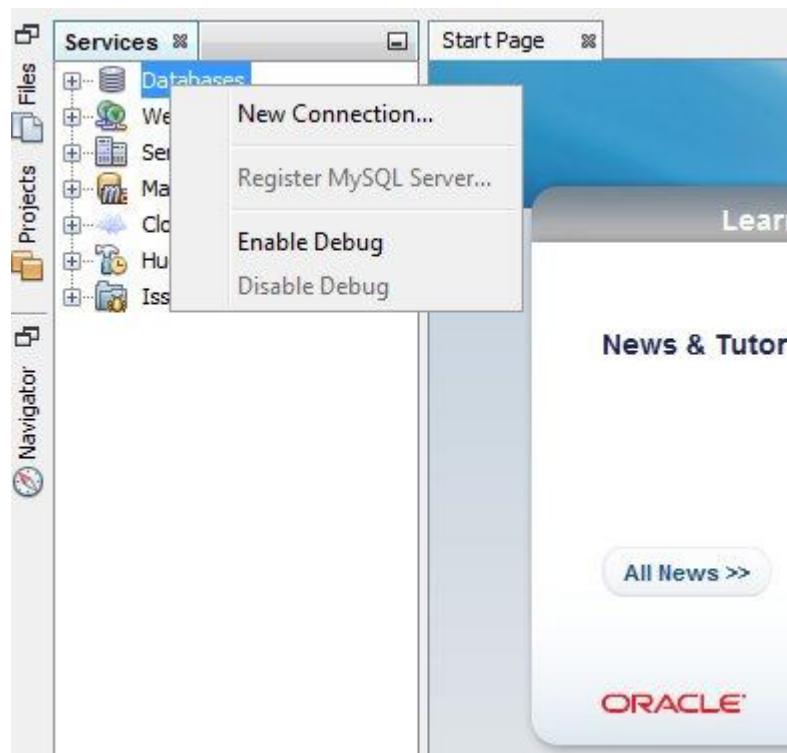


	id	nama	sekolah	telp	email
▶	1	Fadlika Dita Nurjanto	SMK 1 Wonosobo	085729666888	fadlikadn@gmail.com
	2	Hani Ramadhan	SMA 5 Surabaya	085731671852	hani42@gmail.com
	4	Muhammad Aminudin Rahman	SMA 6 Surabaya	085731671852	muara@gmail.com
*		HULL	HULL	HULL	HULL

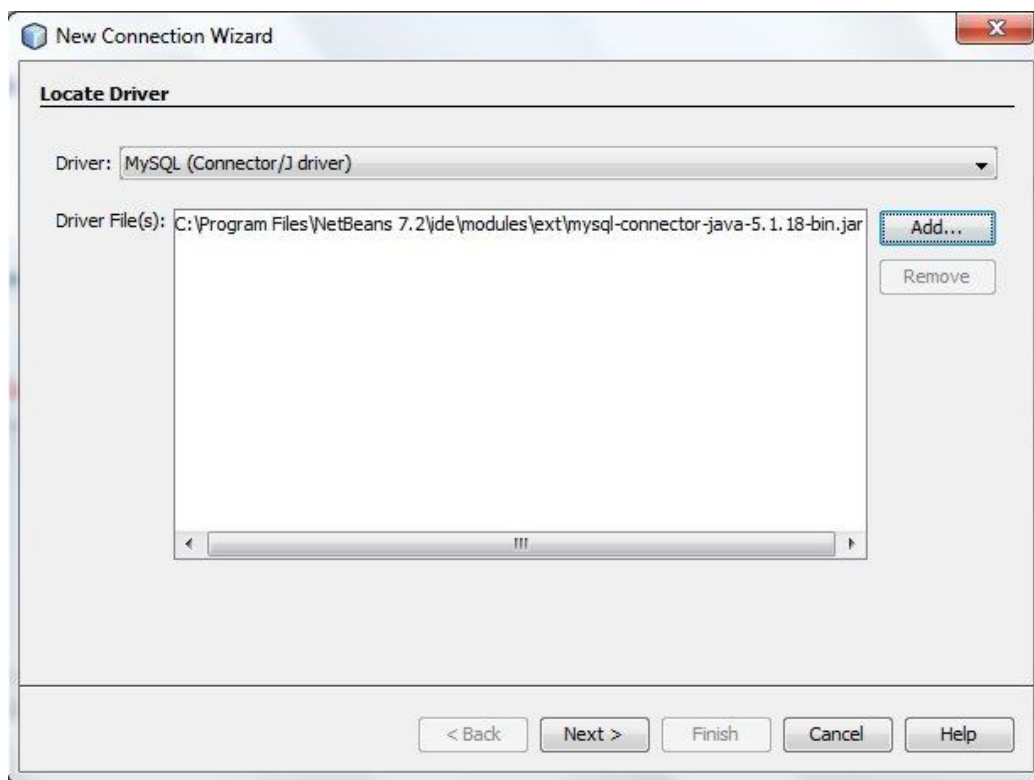
Setelah itu, buka Netbeans 7.2



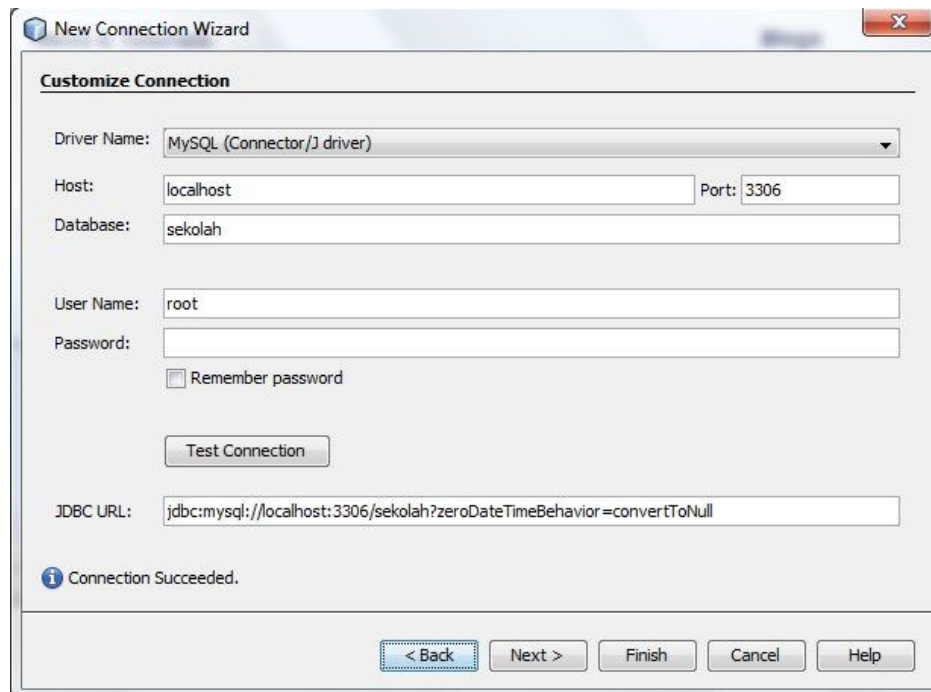
Pada jendela Service di sebelah kiri, klik kanan Database-pilih new Connection



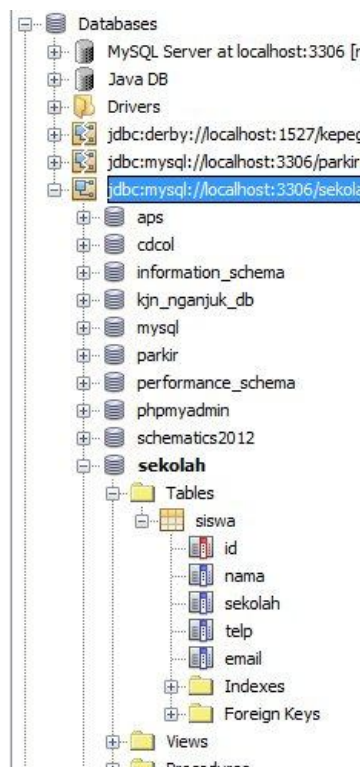
Pada kotak pilihan Driver, pilih MySQL Connector, kemudian klik next



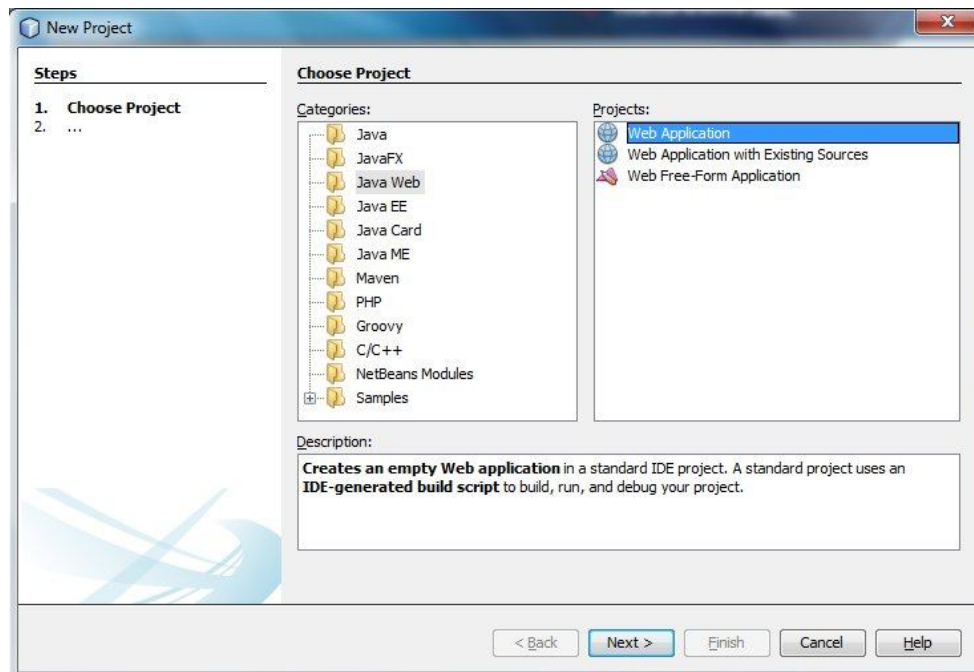
Pada jendela Customize Connection, sesuaikan konfigurasi dengan server dan database yang akan digunakan kemudian klik Finish.



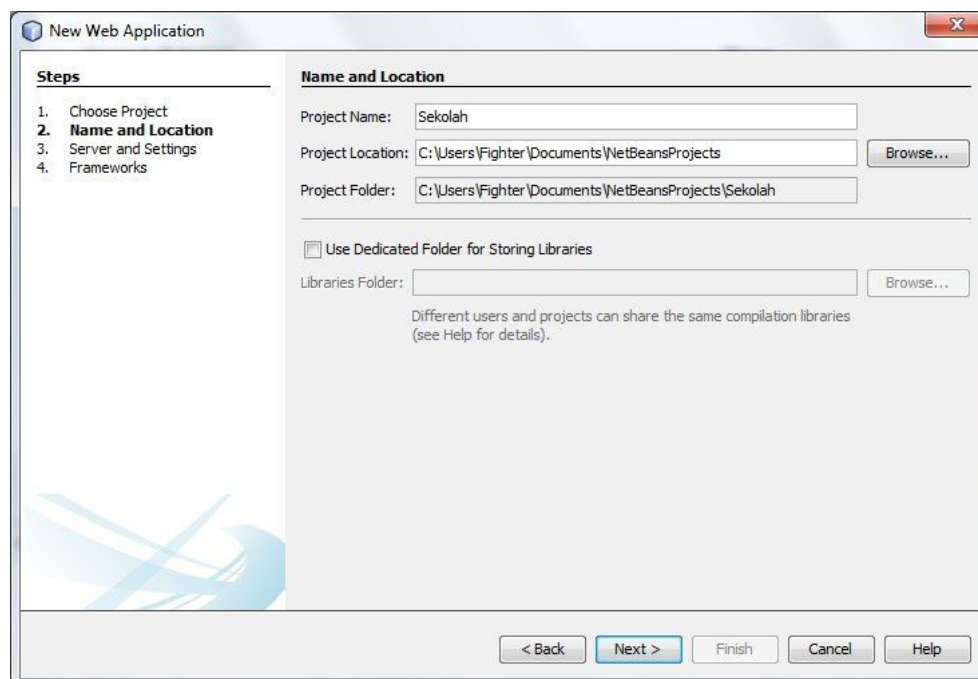
Maka, akan terbentuk koneksi dengan database MySQL di Netbeans.



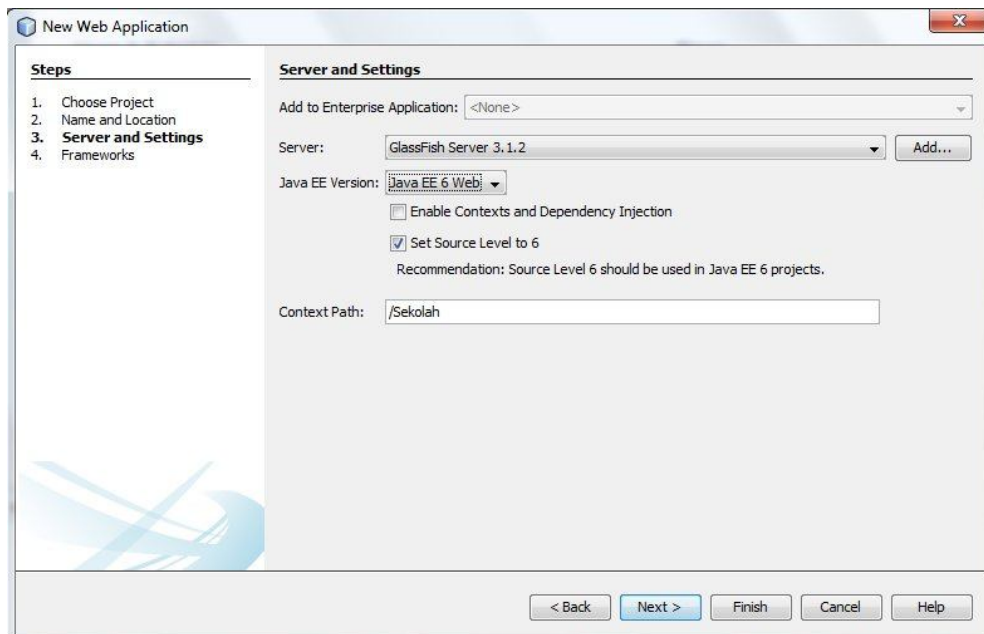
Konfigurasi database selesai dan siap digunakan. Setelah itu buat project baru di Netbeans dengan memilih menu File-New Project. Pilih Java Web - Web Application



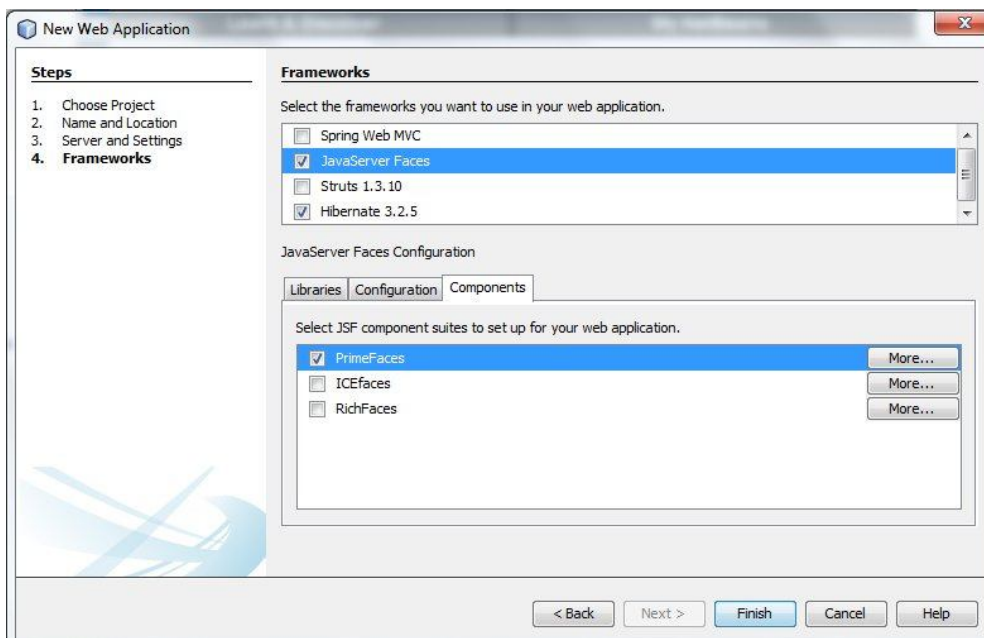
Pada Project Name, isi dengan "Sekolah"



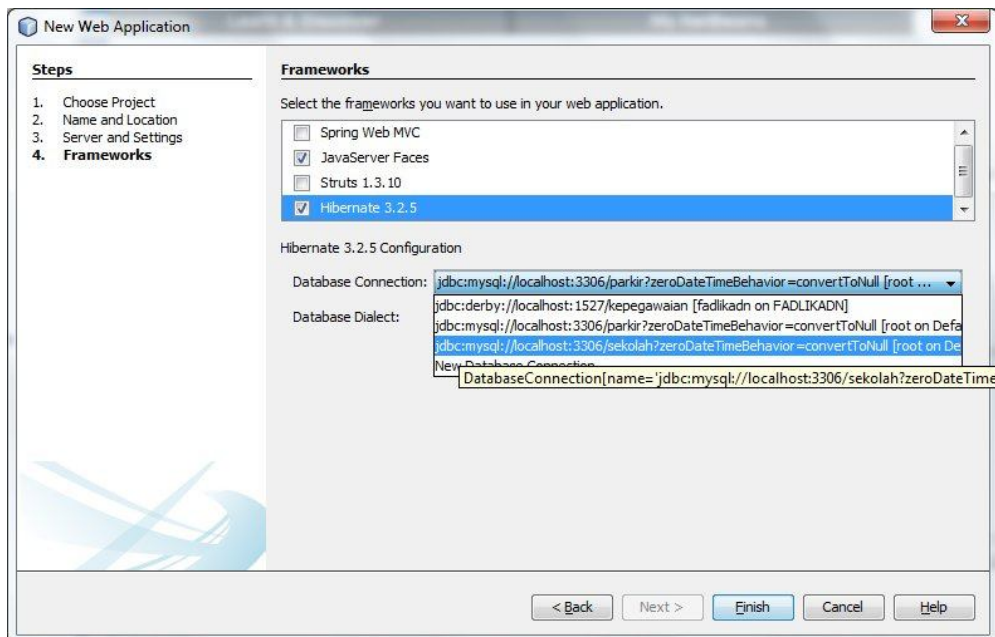
Pada Server and Settings, setting konfigurasi seperti pada gambar



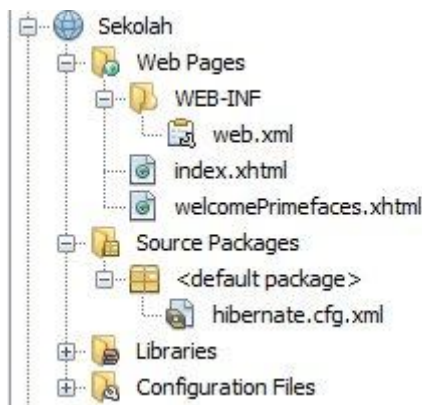
Pada Frameworks, berikan tanda centang pada JavaServer Faces dan Hibernate. Versi yang digunakan di sini adalah Hibernate 3.2.5. Pilih JavaServer Faces, kemudian pilih tab Components, beri tanda centang pada Primefaces.



Klik Hibernate, kemudian pilih Database Connection yang telah dibuat.



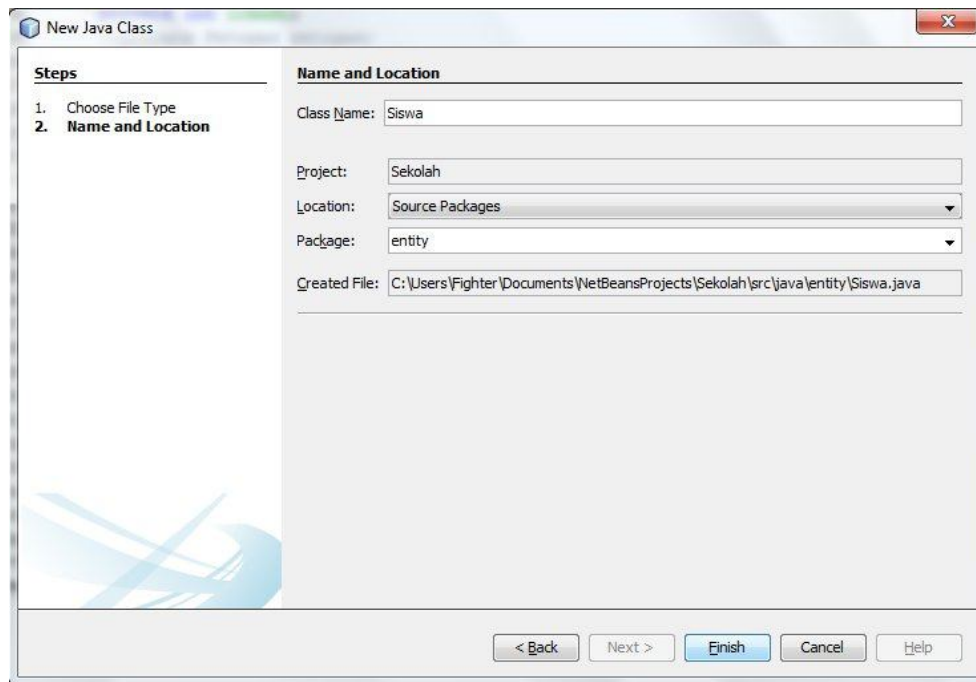
Klik finih, dan project sekolah telah siap.



Pada file hibernate.cfg.xml, ubah sehingga menjadi sebagai berikut

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/sekolah?zeroDateTimeBehavior=con
vertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">root</property>
    <property name="hibernate.show_sql">true</property>
  </session-factory>
</hibernate-configuration>
```

Kemudian tambahkan Entity Class yang digunakan sebagai ORM dengan cara klik kanan pada project – pilih New - Java Class. Beri nama class Siswa dan ketikkan “entity” pada package.



Isi dari file Siswa.java

```
package entity;

public class Siswa {
    private int id;
    private String nama;
    private String sekolah;
    private String telp;
    private String email;

    public Siswa() {
    }

    public Siswa(String nama, String sekolah, String telp, String email) {
        this.nama = nama;
        this.sekolah = sekolah;
        this.telp = telp;
        this.email = email;
    }

    public int getId() {
        return id;
    }

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

    public String getNama() {
```

```
        return nama;
    }

    public void setName(String nama) {
        this.nama = nama;
    }

    public String getSekolah() {
        return sekolah;
    }

    public void setSekolah(String sekolah) {
        this.sekolah = sekolah;
    }

    public String getTelp() {
        return telp;
    }

    public void setTelp(String telp) {
        this.telp = telp;
    }

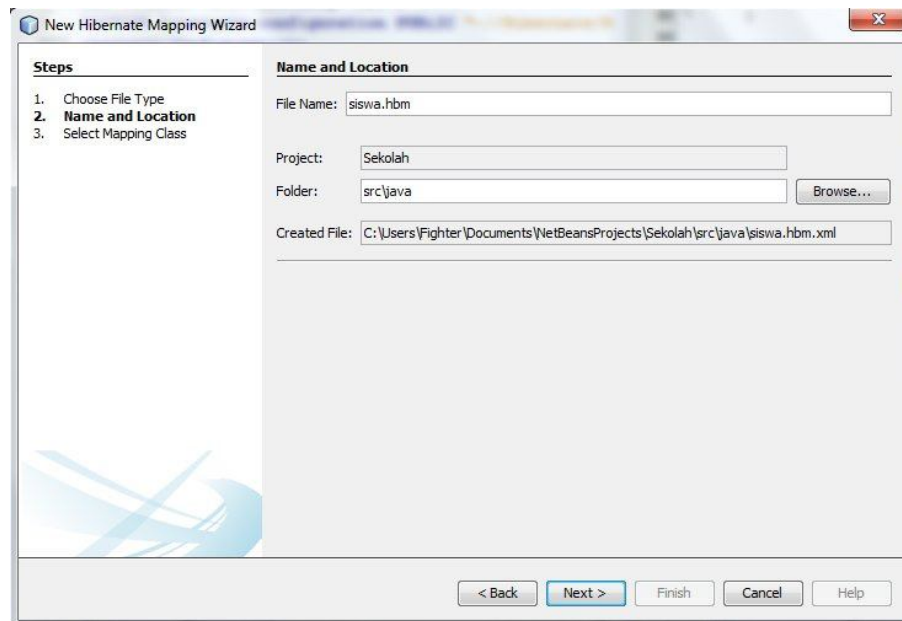
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

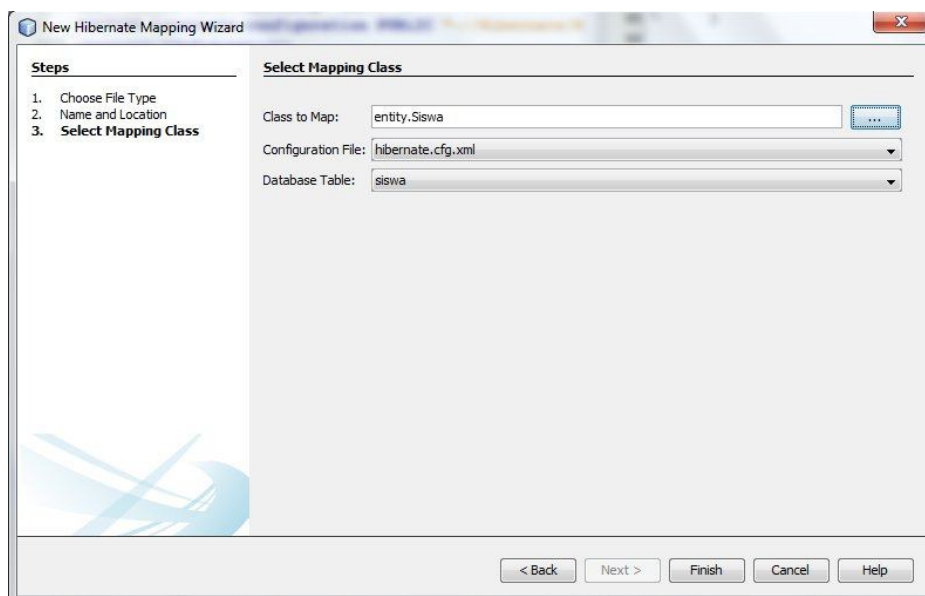
    @Override
    public int hashCode(){
        int hash = 3;
        hash = 23 * hash + this.id;
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
        }
        final Siswa other = (Siswa) obj;
        if (this.id != other.id) {
            return false;
        }
        return true;
    }
}
```

Setelah itu, buat Hibernate mapping dengan cara klik kanan pada project – Hibernate Mapping Wizard. Beri nama siswa pada jendela New Hibernate Mapping Wizard.



Pada Select Mapping Class, pilih Class to Map dan ketikkan entity Siswa, maka secara otomatis Netbeans akan mencari class yang dimaksud. Kemudian pada database table, arahkan ke tabel siswa.



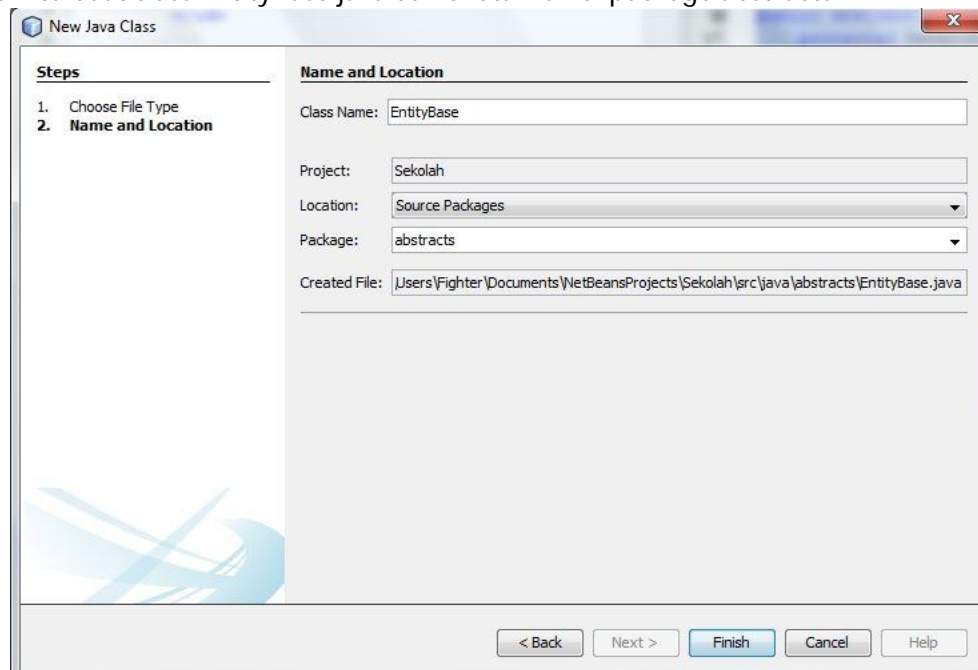
Pada file siswa.hbm.xml, ketikkan sintaks berikut. Sintaks di bawah ini merupakan sintaks yang digunakan untuk mapping database dengan class java.

```
<?xml versi on="1.0" encodi ng="UTF-8"?>
<!DOCTYPE hi bernate-mappi ng PUBLIC "-//Hi bernate/Hi bernate Mappi ng DTD 3.0//EN"
"http://hi bernate.sourceforge.net/hi bernate-mappi ng-3.0.dtd">
<hi bernate-mappi ng>
  <cl ass name="enti ty.Si swa" tabl e="si swa">
    <i d name="i d" col umn="i d">
      <generator cl ass="i ncrement"/>
    </i d>

    <property name="nama">
      <col umn name="nama"/>
    </property>
    <property name="sekol ah">
      <col umn name="sekol ah"/>
    </property>
    <property name="tel p">
      <col umn name="tel p"/>
    </property>
    <property name="emai l">
      <col umn name="emai l"/>
    </property>

  </cl ass>
</hi bernate-mappi ng>
```

Setelah itu buat class EntityBase.java dan diletakkan di package abstracts



Isi class EntityBase.java dengan sintaks berikut :

```
package abstracts;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public abstract class EntityBase {
    protected SessionFactory factory;
    protected Session session;
    protected Transaction transaction;

    public final void connect(){
        factory = new Configuration().configure().buildSessionFactory();
        session = factory.openSession();
        transaction = session.beginTransaction();
    }

    public final void disconnect(){
        transaction.commit();
        session.close();
    }
}
```

Kemudian, buat class IDataAccess di bawah package abstract

```
package abstracts;

import java.io.Serializable;
import java.util.List;

public interface IDataAccess<T> extends Serializable{

    public void insert(T obj);

    public void update(T obj);

    public void delete(T obj);

    public List<T> getAll();

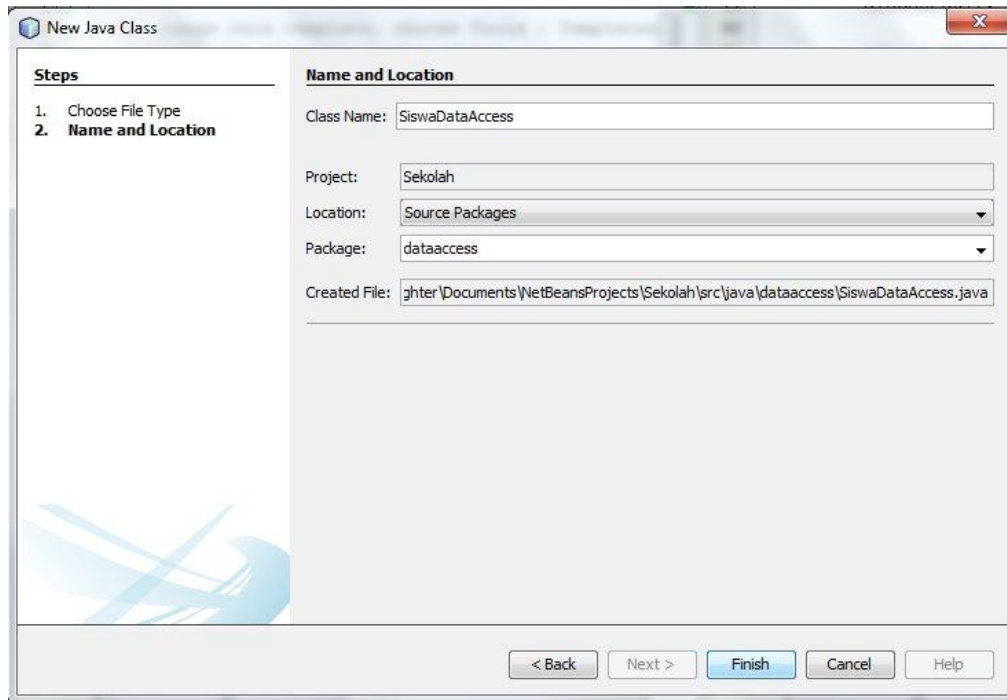
    public T getById(int id);

    public T getById(String id);

    public List<T> getByProperty(String name, Object value);

    public List<T> search(String name, Object value);
}
```

Class IDataAccess dan EntityBase merupakan class interface dan class abstract yang nantinya digunakan saat melakukan operasi CRUD. Setelah keduanya selesai, sekarang buat 1 class yang bernama SiswaDataAccess. Class ini digunakan sebagai pattern proses-proses yang dapat dilakukan pada tabel siswa. Klik kanan pada project, pilih New – Java Class. Beri nama “SiswaDataAccess” dan pada package ketik “dataaccess”.



Isi dari file SiswaDataAccess.java

```
package dataaccess;

import abstracts.EntityBase;
import abstracts.IDataAccess;
import entity.Siswa;
import java.util.List;
import org.hibernate.Query;

public class SiswaDataAccess extends EntityBase implements IDataAccess<Siswa> {

    public SiswaDataAccess() {
    }

    @Override
    public void insert(Siswa obj){
        connect();
        session.save(obj);
        session.flush();
        disconnect();
    }

    @Override
    public void update(Siswa obj){
        connect();
        session.update(obj);
    }
}
```

```
        session.flush();
        disconnect();
    }

    @Override
    public void delete(Siswa obj){
        connect();
        session.delete(obj);
        session.flush();
        disconnect();
    }

    @Override
    public List<Siswa> getAll(){
        connect();
        List<Siswa> siswaList = session.createQuery("from Siswa").list();
        disconnect();

        return siswaList;
    }

    @Override
    public Siswa getByld(String id) {
        connect();
        List<Siswa> siswaList = getByProperty("id", id);
        disconnect();

        if(siswaList != null && siswaList.size()>0){
            return siswaList.get(0);
        }
        return null;
    }

    @Override
    public List<Siswa> getByProperty(String name, Object value) {
        connect();
        Query query = session.createQuery("from Siswa where " + name + " =
: value");
        query.setParameter("value", value);
        List<Siswa> siswaList = query.list();
        disconnect();

        return siswaList;
    }

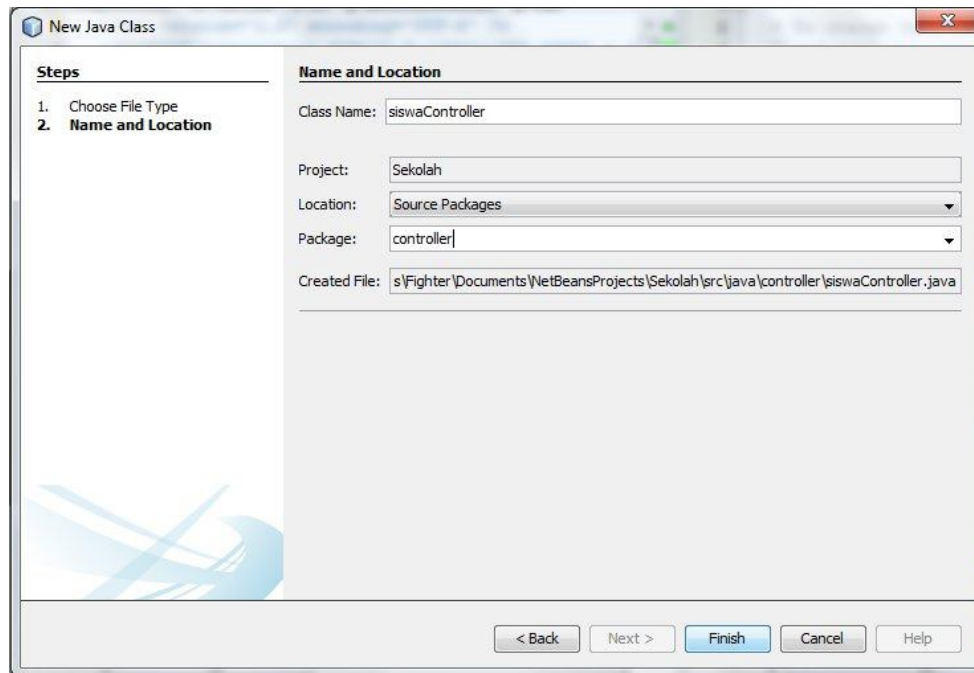
    @Override
    public List<Siswa> search(String name, Object value) {
        connect();
        Query query = session.createQuery("select * from Siswa where " + name
+ " like '%" + value + "%'");
        List<Siswa> kendaraanList = query.list();
        disconnect();

        return kendaraanList;
    }

    @Override
    public Siswa getByld(int id) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

Sampai pada tahap ini, seluruh class dasar sudah berhasil diimplementasikan. Langkah berikutnya adalah membuat class Controller dan View yang berfungsi untuk menampilkan data kepada user.

Controller yang digunakan bernama siswaController dan berada di package controller. Untuk membuatnya, klik kanan pada project, pilih New – Java Class.



Isi dari file siswaController:

```
package controller;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import java.util.ArrayList;
import java.util.List;
import javax.faces.application.FacesMessage;
import javax.faces.bean.RequestScoped;
import javax.faces.context.FacesContext;
import entity.Siswa;
import dataaccess.SiswaDataAccess;

@ManagedBean
@SessionScoped
public class siswaController {
    Siswa siswa;
    List<Siswa> siswaList;
    private SiswaDataAccess dataAccess;
    private String searchBy;
    private String searchValue;
}
```

```
public siswaController() {
    siswaList = new ArrayList<Siswa>();
    siswa = new Siswa();
    dataAccess = new SiswaDataAccess();
    siswaList = dataAccess.getAll();
}

public String displayAll(){
    siswaList = dataAccess.getAll();
    return "List";
}

public String searchSiswa(){
    siswaList = dataAccess.getByProperty(this.searchBy,
this.searchValue);
    return "List";
}

public String addSiswa(){
    dataAccess.insert(siswa);

    FacesMessage message = new FacesMessage("Data berhasil
di tambahkan");
    message.setSeverity((FacesMessage.SEVERITY_INFO));

    FacesContext.getCurrentInstance().addMessage(null, message);
    siswaList = dataAccess.getAll();
    return "List";
}

public String updateSiswa(){
    dataAccess.update(siswa);

    FacesMessage message = new FacesMessage("Data berhasil di ubah");
    message.setSeverity((FacesMessage.SEVERITY_INFO));

    FacesContext.getCurrentInstance().addMessage(null, message);
    siswaList = dataAccess.getAll();
    return "List";
}

public String deleteSiswa(){
    dataAccess.delete(siswa);

    FacesMessage message = new FacesMessage("Data berhasil di hapus");
    message.setSeverity((FacesMessage.SEVERITY_INFO));

    FacesContext.getCurrentInstance().addMessage(null, message);
    siswaList = dataAccess.getAll();
    return "List";
}

public Siswa getSiswa() {
    return siswa;
}

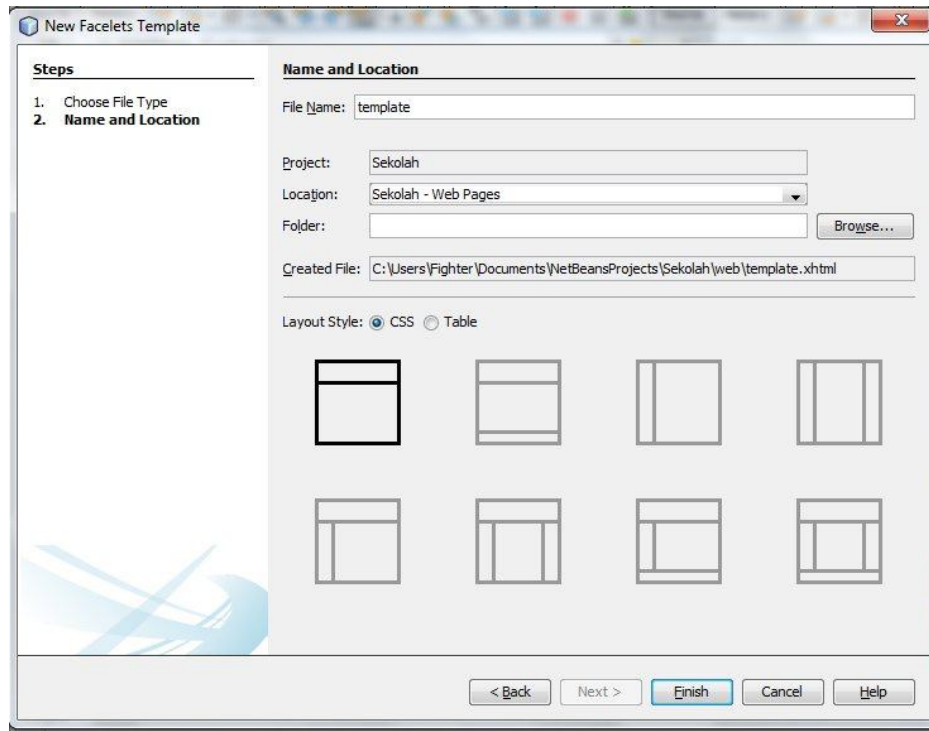
public void setSiswa(Siswa siswa) {
    this.siswa = siswa;
}
```



```
public List<Siswa> getSiswaList() {  
    return siswaList;  
}  
  
public void setSiswaList(List<Siswa> siswaList) {  
    this.siswaList = siswaList;  
}  
  
public SiswaDataAccess getDataAccess() {  
    return dataAccess;  
}  
  
public void setDataAccess(SiswaDataAccess dataAccess) {  
    this.dataAccess = dataAccess;  
}  
  
public String getSearchBy() {  
    return searchBy;  
}  
  
public void setSearchBy(String searchBy) {  
    this.searchBy = searchBy;  
}  
  
public String getSearchValue() {  
    return searchValue;  
}  
  
public void setSearchValue(String searchValue) {  
    this.searchValue = searchValue;  
}  
}
```

Langkah berikutnya adalah membuat View yang akan menampilkan data-data dari di database pada web browser. Sebelum membuat view, buat Faces Template terlebih dahulu. Ini adalah salah satu kelebihan di JSF, di mana kita bisa dengan mudah mendefinisikan template yang akan dipakai di aplikasi kita.

Untuk membuat template, klik kanan pada project, kemudian pilih New – Facelets Template. Di sini tersedia berbagai layout yang siap digunakan. Beri nama file dengan “template”.



Kemudian klik finish. Akan terbentuk sebuah file bernama template.xhtml yang berisi sintaks sebagai berikut :

```
<?xml versi on=' 1.0' encodi ng=' UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transi tional //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transi tional .dtd">
<html xml ns="http://www.w3.org/1999/xhtml "
      xml ns: ui ="http://j ava. sun. com/j sf/facel ets"
      xml ns: h ="http://j ava. sun. com/j sf/html ">

    <h: head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <link href="/resources/css/default.css" rel="styl esheet" type="text/css"
    />
        <link href="/resources/css/cssLayout.css" rel="styl esheet"
type="text/css" />
        <ti tle>Facel ets Templ ate</ti tle>
    </h: head>

    <h: body>

        <di v id="top" cl ass="top">
            <ui : i nsert name="top">Top</ui : i nsert>
        </di v>

        <di v id="content" cl ass="center_content">
            <ui : i nsert name="content">Content</ui : i nsert>
        </di v>

    </h: body>

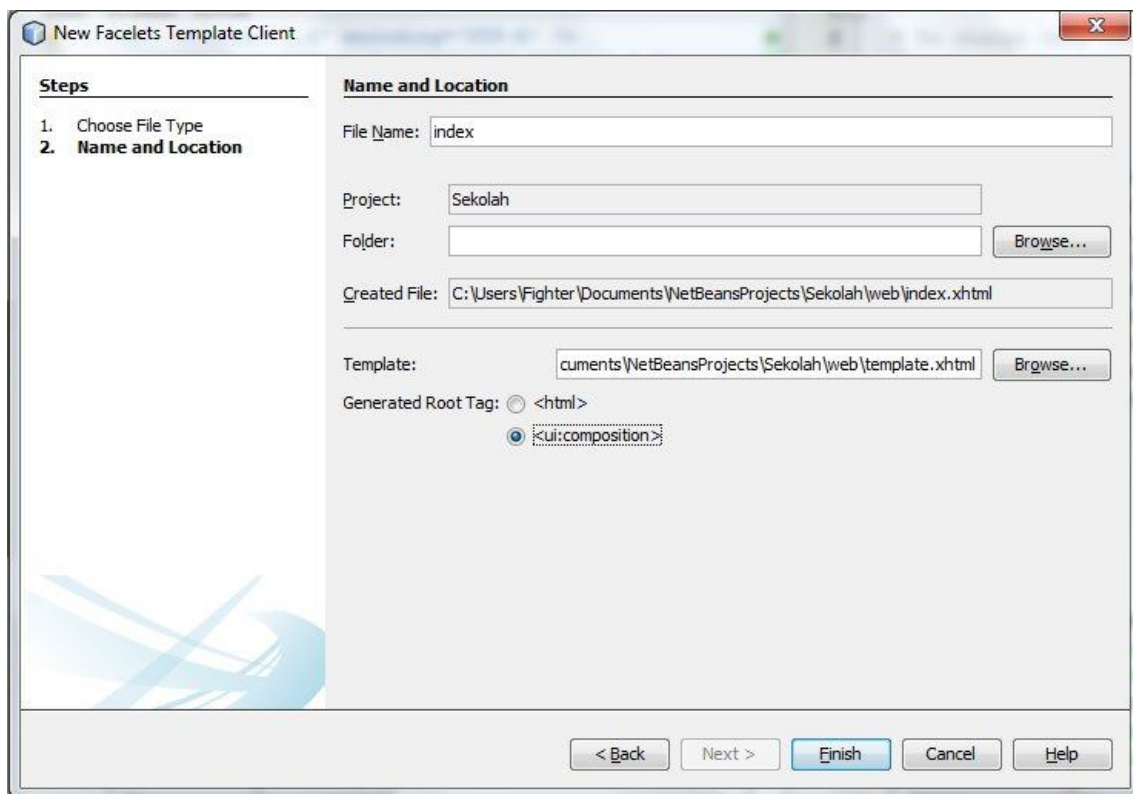
</html>
```

Sekilas sintaks-sintaks JSF memiliki kemiripan dengan HTML, tetapi yang membedakannya adalah adanya library yang bisa dipanggil sesuai dengan objek yang dibutuhkan. Konsep kerja pada JSF, GlassFish selaku web server akan menerjemahkan JSF yang ada ke dalam bentuk HTML. Maka dari itu, view dari web yang tampak pada user adalah hasil proses yang dilakukan oleh GlassFish.

Berikutnya adalah membuat Facelets Template Client yang bernama index.xhtml. File ini akan mewariskan template yang sudah disusun oleh file template.xhtml. File ini jugalah yang akan menampung data-data pada database untuk ditampilkan kepada user.

Langkah-langkah untuk membuat sebuah Facelets Template Client adalah sebagai berikut :

Pada project, klik kanan kemudian pilih New – Facelets Template Client. Beri nama “index” pada kolom File Name. Kemudian arahkan Template pada file template.xhtml yang sudah dibuat. Pada Generated Root Tag, pilih <ui:composition>. Untuk bagian ini, kita bebas memilih <html> atau <ui:composition>.



Isi dari file index.xhtml adalah sebagai berikut :

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE composition PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:composition xmlns:ui="http://java.sun.com/jsf/facets"
    template="/template.xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:c="http://java.sun.com/jsp/jstl/core">

    <ui:define name="top">
        top
    </ui:define>

    <ui:define name="content">
        <h:form>
            <p:growl id="loginmessage" showDetail="true" sticky="true"/>
            <p:panel>
                <p:panelGrid columns="5">
                    <h:outputText value="Search By"/>
                    <p:selectOneMenu value="#{siwaControlIer.searchBy}">
                        <f:selectItem itemLabel="Select One" itemValue="" />
                        <f:selectItem itemLabel="ID Si swa" itemValue="id" />
                        <f:selectItem itemLabel="Nama Si swa" itemValue="nama" />
                        <f:selectItem itemLabel="Sekolah" itemValue="sekolah" />
                        <f:selectItem itemLabel="Tel epon" itemValue="tel p" />
                        <f:selectItem itemLabel="Email" itemValue="email" />
                    </p:selectOneMenu>
                    <h:inputText value="#{siwaControlIer.searchValue}"/>
                    <h:commandButton value="Search"
action="#{siwaControlIer.searchSi swa()}" />
                    <h:commandButton value="Tampilkan Semua"
action="#{siwaControlIer.displayAll()}" />
                </p:panelGrid>
            </p:panel>
        </h:form>
        <h:form>
            <p:dataTable id="dataTable" var="si swa"
value="#{siwaControlIer.si swaList}">
                <f:facet name="header">
                    Data Si swa
                </f:facet>

                <p:column id="modelHeader" sortBy="#{si swa.id}">
                    <f:facet name="header">
                        <h:outputText value="ID Si swa" />
                    </f:facet>
                    <h:outputText value="#{si swa.id}" />
                </p:column>

                <p:column sortBy="#{si swa.nama}">
                    <f:facet name="header">
                        <h:outputText value="Nama Si swa" />
                    </f:facet>
                    <h:outputText value="#{si swa.nama}" />
                </p:column>
            </p:dataTable>
        </h:form>
    </ui:define>
</ui:composition>
```

```

<p: column sortBy="#{si swa. sekolah}">
  <f: facet name="header">
    <h: outputText value="Sekol ah" />
  </f: facet>
  <h: outputText value="#{si swa. sekolah}" />
</p: column>

<p: column sortBy="#{si swa. tel p}">
  <f: facet name="header">
    <h: outputText value="Nomor Tel epon" />
  </f: facet>
  <h: outputText value="#{si swa. tel p}" />
</p: column>

<p: column sortBy="#{si swa. email}">
  <f: facet name="header">
    <h: outputText value="Email" />
  </f: facet>
  <h: outputText value="#{si swa. email}" />
</p: column>

</p: dataTable>
</h: form>

<p: panel >
  <h: form>
    <h: panelGrid columns="2">
      <h: outputText value="ID Si swa" />
      <p: inputText value="#{si swaControl l er. si swa. id}"
requir ed="true" />

      <h: outputText value="Nama Si swa" />
      <p: inputText value="#{si swaControl l er. si swa. nama}"
requir ed="true" />

      <h: outputText value="Sekol ah" />
      <p: inputText value="#{si swaControl l er. si swa. sekolah}"
requir ed="true" />

      <h: outputText value="Nomor Tel epon" />
      <p: inputText value="#{si swaControl l er. si swa. tel p}"
requir ed="true" />

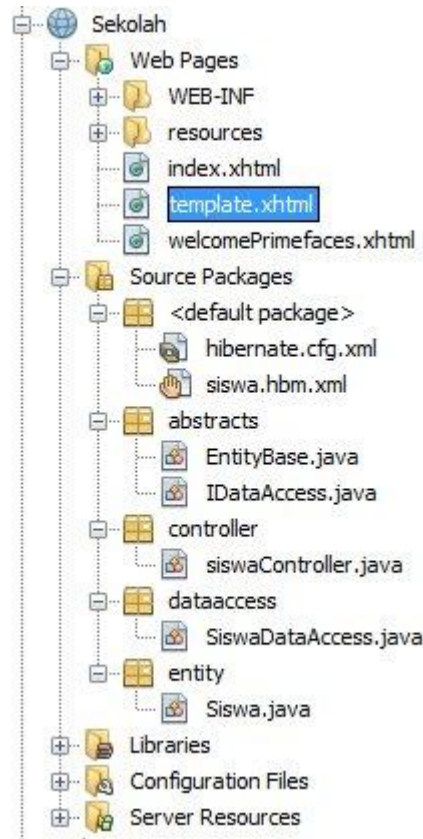
      <h: outputText value="Email" />
      <p: inputText value="#{si swaControl l er. si swa. email}"
requir ed="true" />

      <p: commandButton value="Si mpan"
acti on="#{si swaControl l er. addSi swa()}" ajax="fal se" />
      <p: commandButton value="Ubah"
acti on="#{si swaControl l er. updateSi swa()}" ajax="fal se" />
      <p: commandButton value="Hapus"
acti on="#{si swaControl l er. deleteSi swa()}" ajax="fal se" />
    </h: panelGrid>
  </h: form>
</p: panel >
</ui: defi ne>

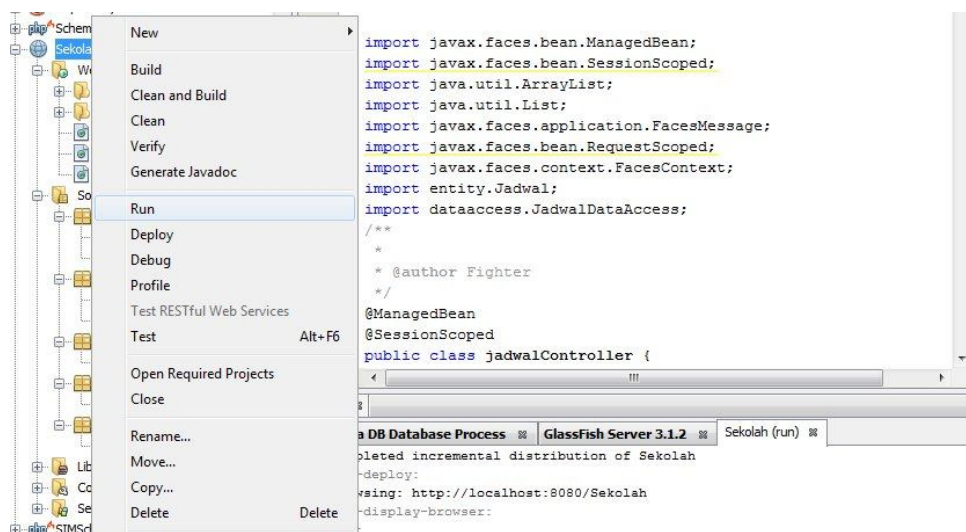
</ui: composi ti on>

```

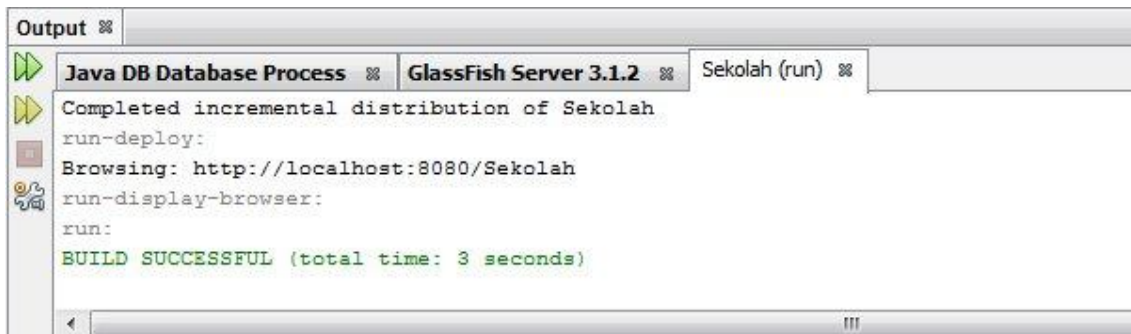

View yang dibuat sudah menggunakan library primefaces, untuk dokumentasi lengkap bisa dilihat di <http://primefaces.org/>. Perlu diperhatikan, bahwa struktur project yang dibuat dari awal menjadi seperti berikut. Sebelum di-deploy dan di-run, pastikan semua class sudah terhubung 1 sama lain sesuai dengan langkah-langkah sebelumnya. Struktur project yang sudah jadi akan menjadi seperti berikut :



Kemudian, jalankan project dengan cara klik kanan pada project, pilih Run dan tunggu beberapa saat sampai project selesai di-build dan akan tampil pada browser.

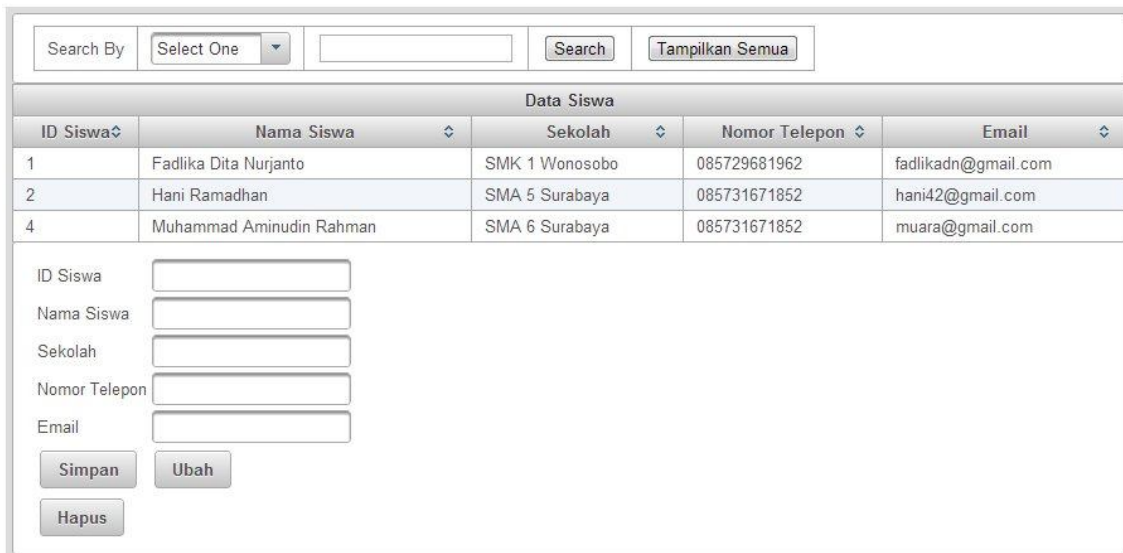


Setelah selesai di-build akan muncul pesan berikut



```
Output
Java DB Database Process  GlassFish Server 3.1.2  Sekolah (run)
Completed incremental distribution of Sekolah
run-deploy:
Browsing: http://localhost:8080/Sekolah
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 3 seconds)
```

Setelah aplikasi dijalankan pada browser.



Data Siswa				
ID Siswa	Nama Siswa	Sekolah	Nomor Telepon	Email
1	Fadlika Dita Nurjanto	SMK 1 Wonosobo	085729681962	fadlikadn@gmail.com
2	Hani Ramadhan	SMA 5 Surabaya	085731671852	hani42@gmail.com
4	Muhammad Aminudin Rahman	SMA 6 Surabaya	085731671852	muara@gmail.com

ID Siswa:
Nama Siswa:
Sekolah:
Nomor Telepon:
Email:

<http://localhost:8080/Sekolah/faces/index.xhtml>

Biografi Penulis



Fadlika Dita Nurjanto. Menyelesaikan pendidikan di SD 2 Wonosobo tahun 2004, SMP 1 Wonosobo tahun 2007, dan SMK 1 tahun 2010. Sekarang berstatus sebagai mahasiswa Teknik Informatika Institut Teknologi Sepuluh Nopember, Surabaya. Di kampus perjuangan ini penulis aktif dalam berbagai organisasi, antara lain Kopma dr.Angka ITS, HMTC Teknik Informatika ITS, dan Keluarga Muslim Informatika ITS. Selain itu penulis juga aktif menjadi salah satu pengembang di Dieng Cyber (Komunitas IT Wonosobo, Jawa Tengah), Microsoft Student Partner ITS Regional Jawa Timur, dan juga freelance pada berbagai project pengembangan web. Artikel menarik lainnya bisa ditemukan di <http://fadlikadn.wordpress.com> ataupun sharing dengan mengirim email ke fadlikadn@gmail.com.