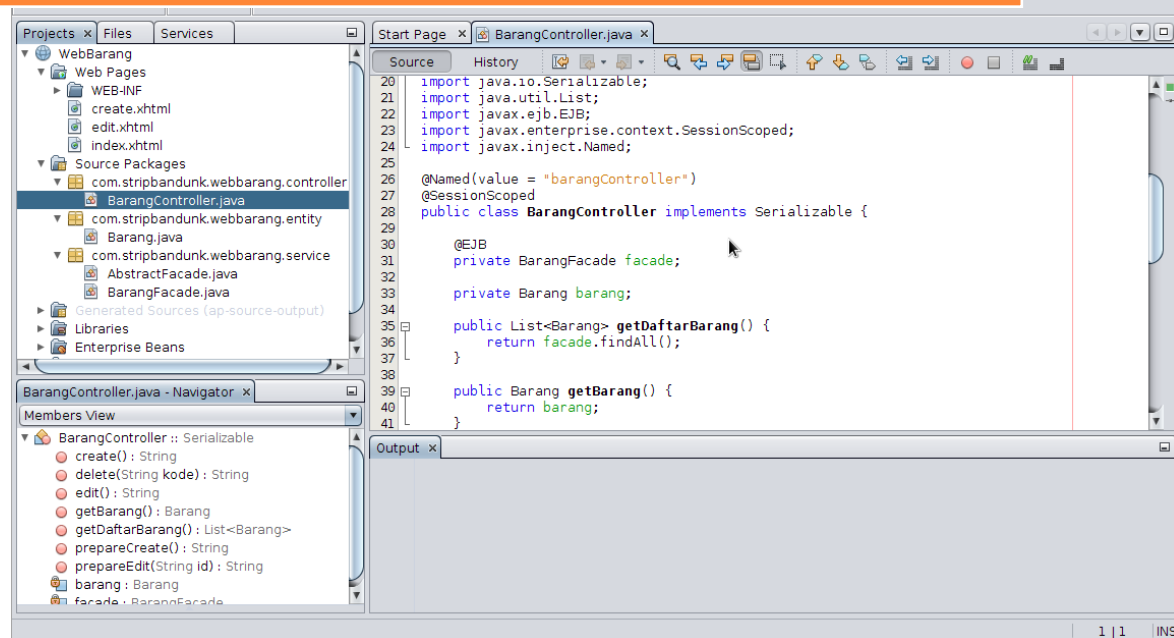


# 2012

## Membuat Aplikasi Java Web Enterprise Sederhana



Eko Kurniawan Khannedy

StripBandunk.com

4/1/2012

## **Persembahan**

Buku ini dipersembahkan untuk Indonesia yang lebih baik

Ilmu, teknologi dan sumber daya manusia yang lebih baik

Semoga buku ini bisa menjadi amal

Untuk saya dan keluarga

## Katakan...

Semoga tidak bosan-bosannya membaca buku yang saya buat :D walaupun agak semrawut bukunya, semoga bermanfaat :D

Sebenarnya banyak sih yang mencibir, menghina dan mengutuk atas apa yang saya lakukan selama ini, berbagi ilmu, saya juga gak tau kenapa :D Tapi persetan dengan mereka :D Yang penting saya akan tetap terus berbagi ilmu, sampai akhir hayat, yeah, lebay mode on :D

Buku ini adalah buku saku yang membahas tentang pembuatan aplikasi java web enterprise sederhana, dimana disini saya bakal pake teknologi java enterprise, wuih keren kedengerannya :D Teknologi apa aja? Diantarnya Java Server Faces, Java Persistence API, Enterprise Java Beans dan Bean Validation. Wuih sebanyak itu?

Percaya deh, gak seserem kedengerannya :D Malah menyenangkan kok :D Banyak kemudahan yang ditawarkan oleh teknologi-teknologi yang tadi saya sebutin :D

So, gak usah banyak cing cong, si acong aja gak banyak cing cong :D

Let's Rock!!!

## Daftar Isi

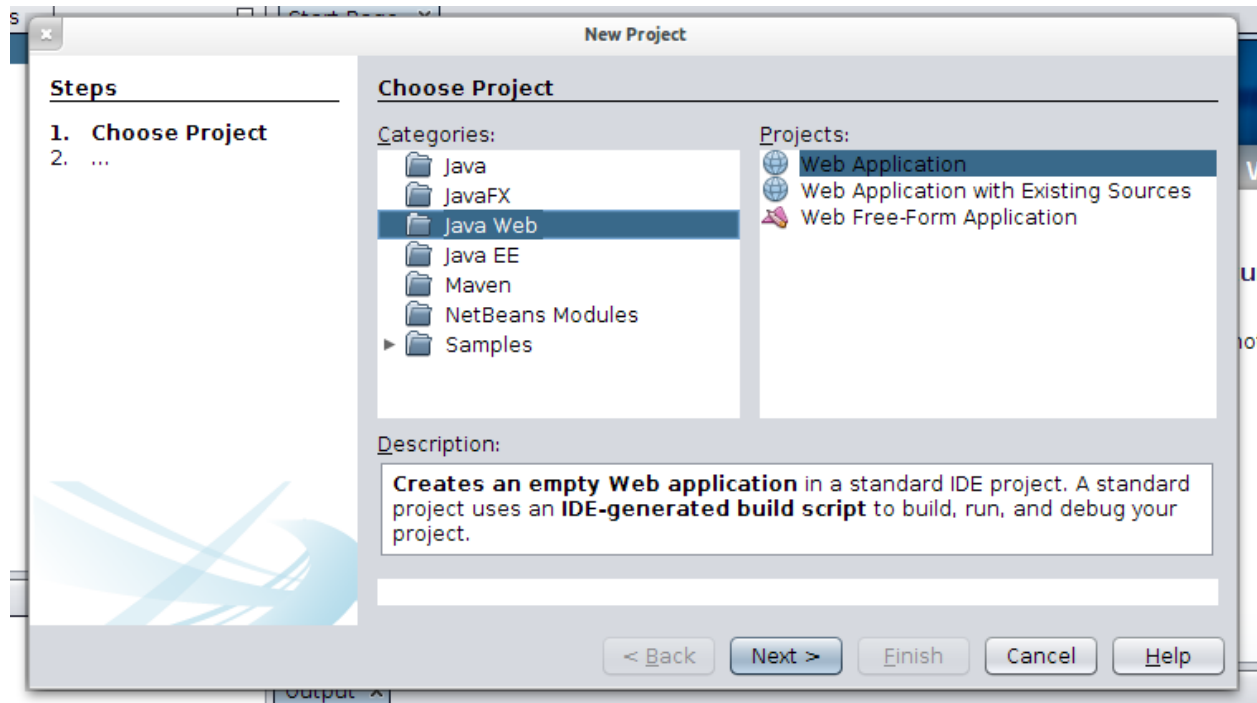
Persembahan .....	2
Katakan.....	3
Daftar Isi .....	4
Bikin Project Dulu.....	5
Bikin Konfigurasi JPA.....	9
Membuat Entitas Barang .....	13
Service Menggunakan Enterprise Java Beans.....	17
Validasi Menggunakan Bean Validation.....	20
Validasi Id Barang.....	20
Validasi Nama Barang .....	20
Validasi Kategori Barang .....	20
Validasi Harga Barang .....	21
Validasi Stok Barang.....	21
Controller Menggunakan Manage Bean .....	22
Membuat Manage Bean .....	22
Aksi Daftar Barang .....	24
Aksi Tambah Barang.....	24
Aksi Ubah Barang .....	25
Aksi Hapus Barang.....	26
View Menggunakan Java Server Faces .....	27
Nampilin Data Barang .....	27
Nambah Data Barang .....	30
Ngedit Data Barang.....	35
Ngapus Barang .....	37
Yuk Coba .....	39
Ngejalanin Web.....	39
Misi Selanjutnya.....	43
Tentang Saya.....	44

## Bikin Project Dulu

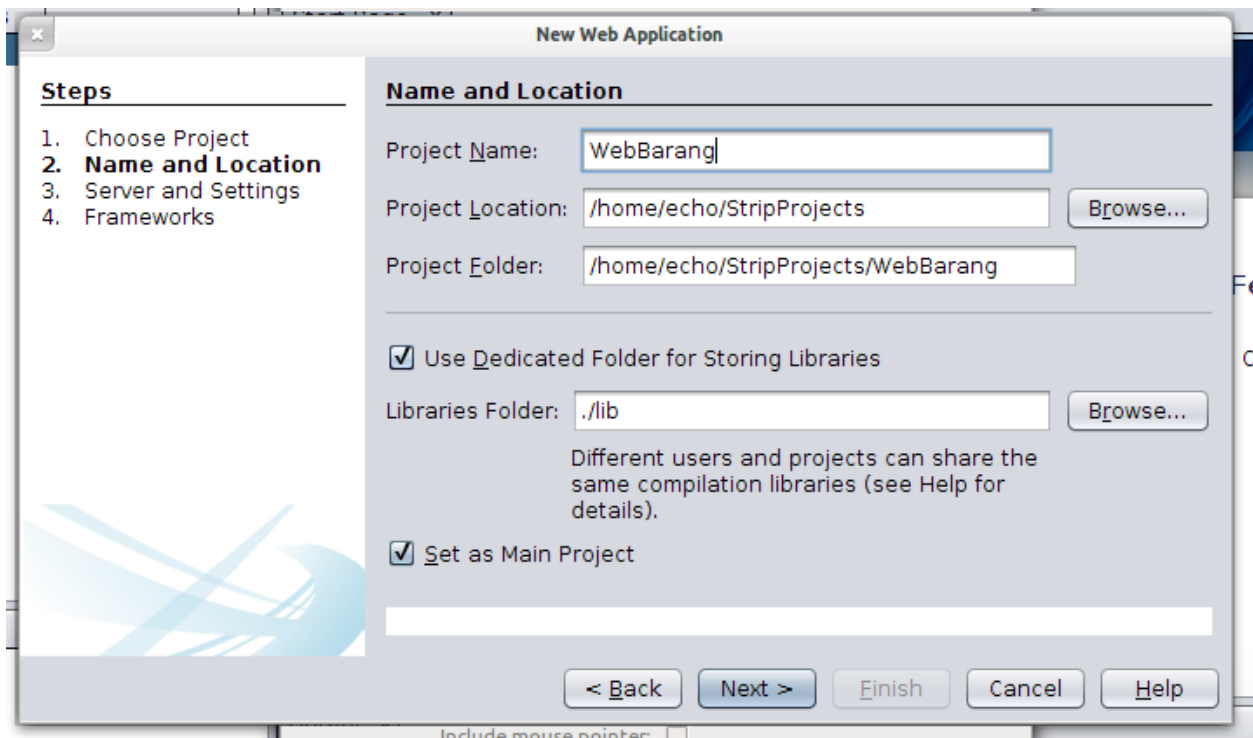
Gak seperti di PHP yang bisa langsung bikin file .PHP, di Java gak seperti itu. Saat mau bikin aplikasi web, di java harus bikin project dulu.

Disini saya pake NetBeans buat editor-nya dan Glassfish buat servernya.

Jadi sekarang kita bikin project java web dulu di NetBeans, tinggal pilih **File -> New Project**, trus keluar dialog kayak gini :

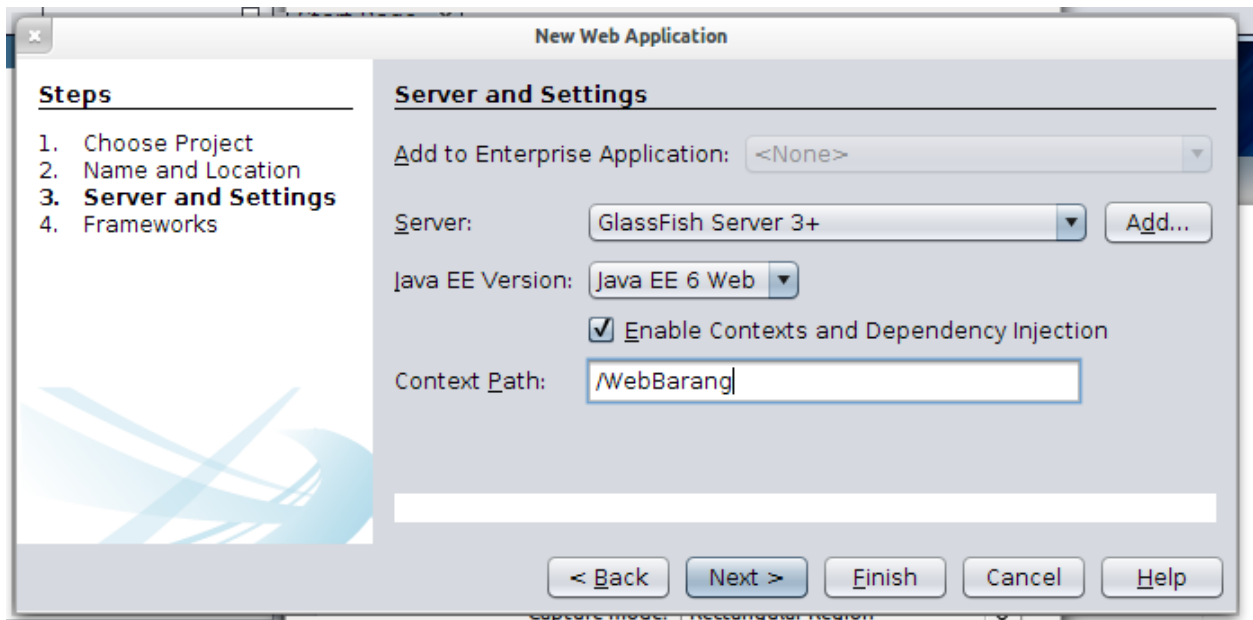


Trus pilih kategori Java Web dan proyek Web Application, kalo udah klik **Next >**



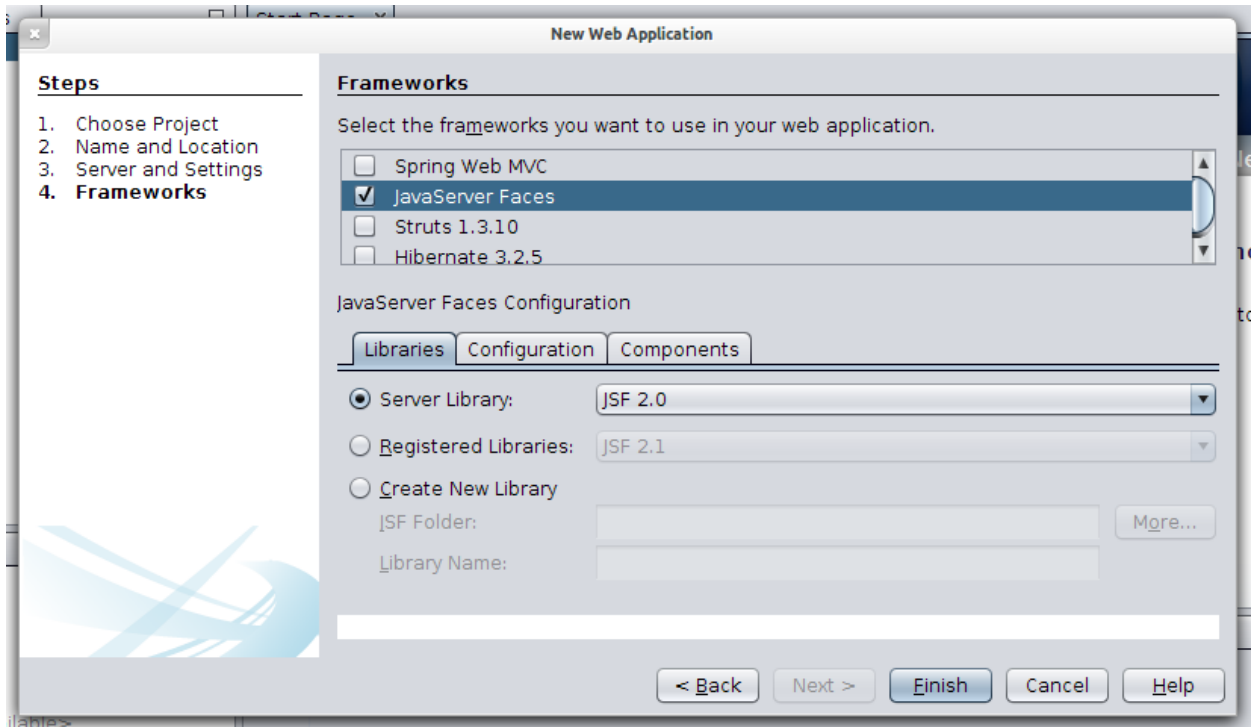
Kalo udah tinggal masukin nama proyek nya, misal disini saya kasih nama **WebBarang**. KokWebBarang? Yup soalnya disini saya mau bikin web buat manipulasi data barang. Cukup sederhana kan? Jangan terlalu kompleks, nanti bukunya kepanjangan, saya males ngetik nya :P

Kalo udah klik **Next >**

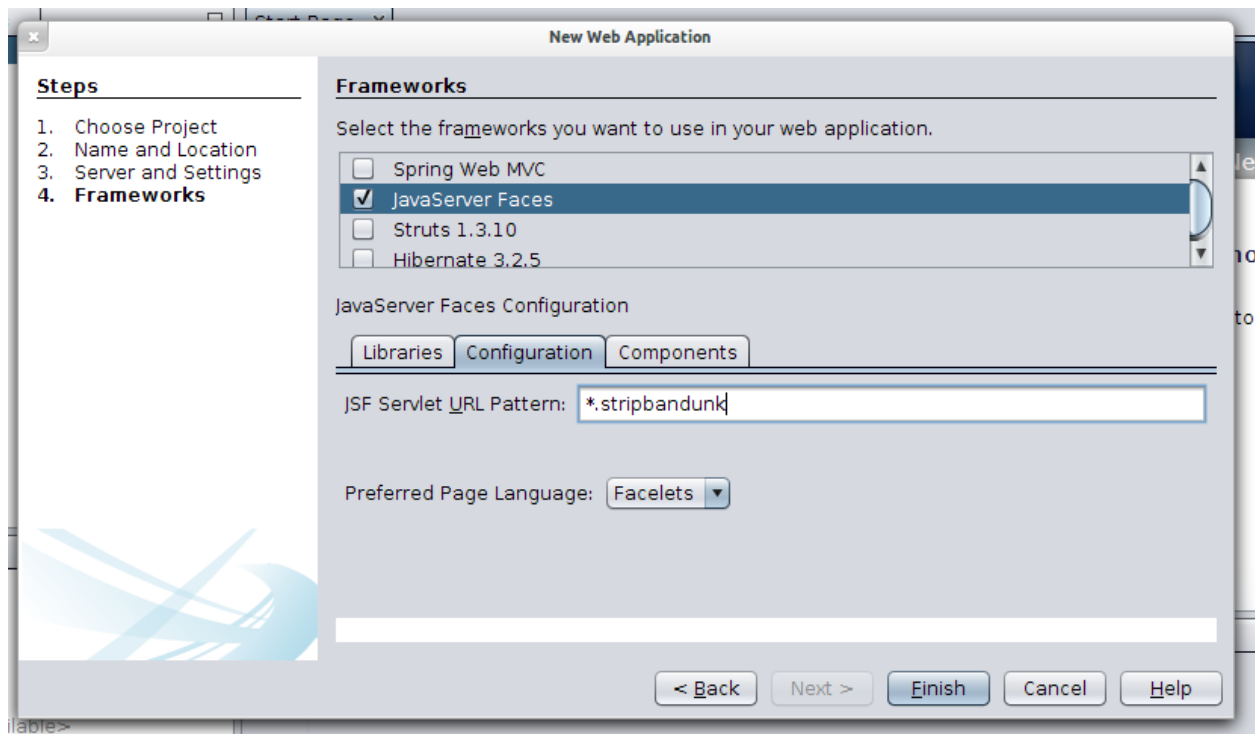


Pilih server GlassFish, trus versi Java EE nya pilih yang Java EE 6 Web, lalu ceklis Enabled CDI, lalu masukin Context Path, ContextPath itu url untuk ngakses aplikasi web kita, defaultnya sama ama nama project yang kita buat.

Kalo udah klik **Next >**



Pilih Java Server Faces, soalnya kita mau pake JSF, lalu di configuration nya ubah juga kayak gambar dibawah ini :

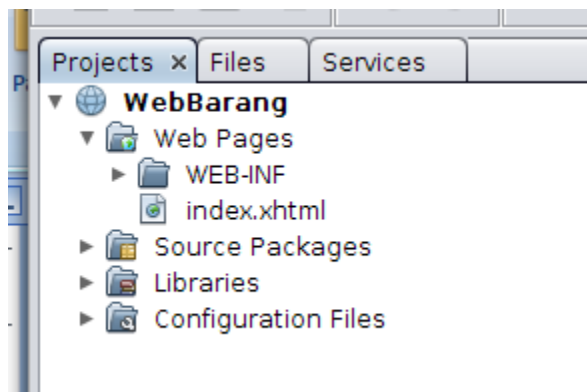


JSF Servlet URL Pattern nya saya ubah jadi \*.stripbandunk, artinya nanti setiap file JSF akan diakses dengan diakhiri .stripbandunk, misal :

- Index.stripbandunk
- Create.stripbandunk
- Dan lain-lain

Keren kan? Bosen kalo .php, .jsp, .html, mending .namakita :D

Kalo udah klik Finish, sekarang NetBeans akan bikinin projectnya buat kita





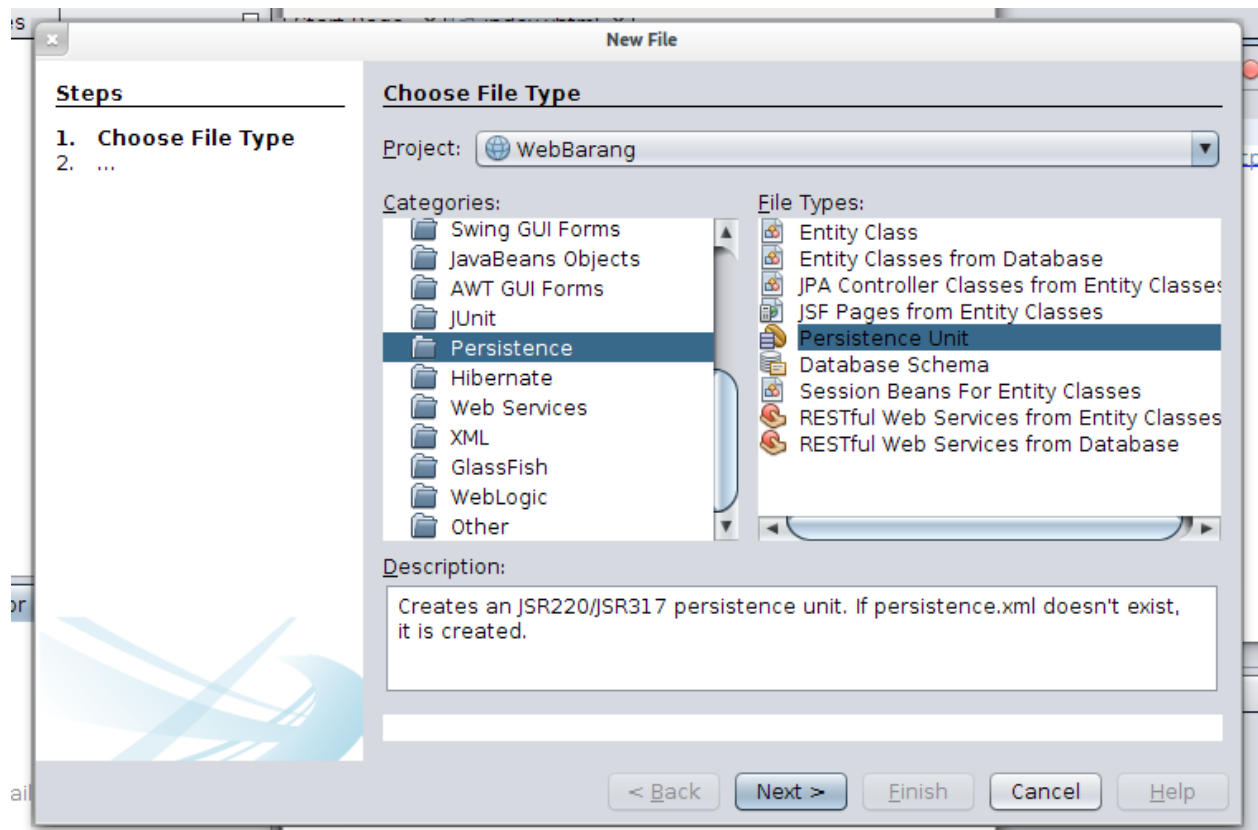
## Bikin Konfigurasi JPA

Kalo udah bikin project nya, sekarang kita bikin konfigurasi Java Persistence API nya dulu, jadi gak langsung bikin halaman webnya dulu, itu nanti belakangan. Yang paling utama itu bikin data tier nya dulu, lalu service tier, dan terakhir adalah presentation tier.

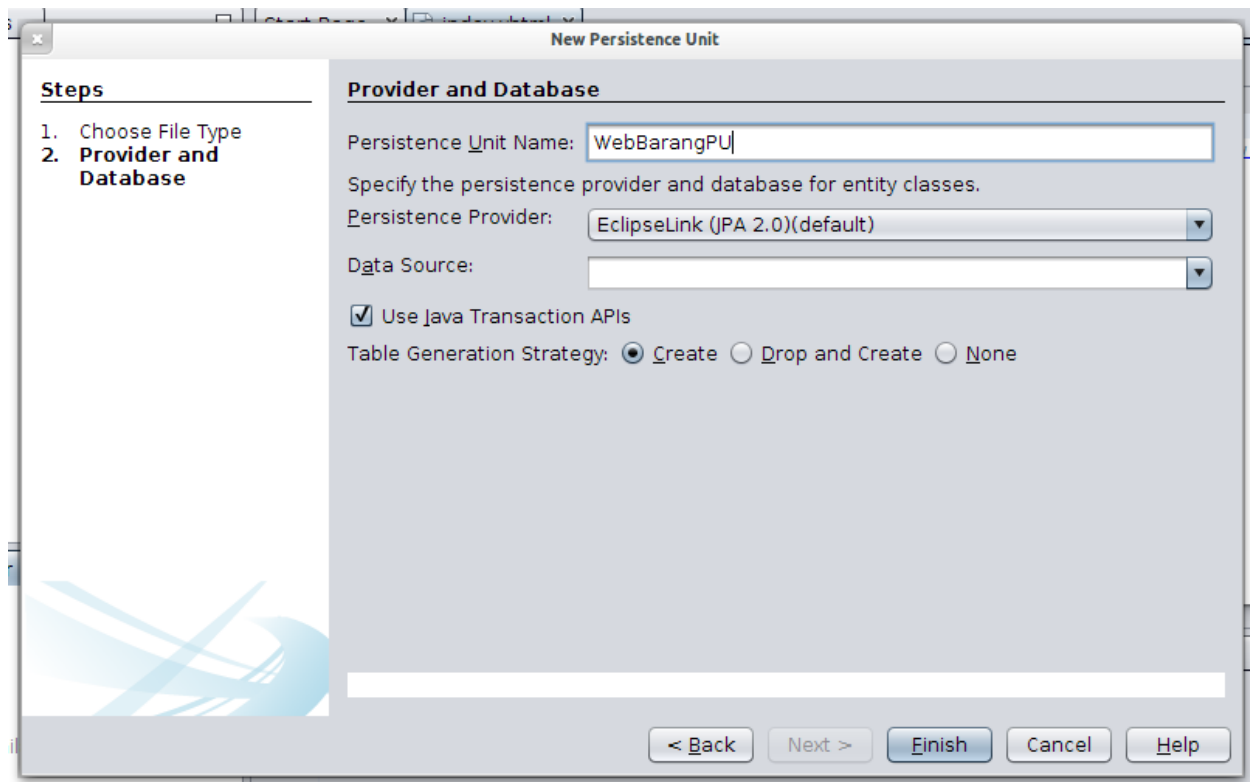
JPA adalah data tier, EJB adalah service tier dan JSF adalah presentation tier.

Masih inget kan aplikasi 3-Tier? Kalo lupa, buka Wikipedia lagi sana :P

Ok, back to topic, buat bikin konfigurasi JPA, pilih **File -> New File**.

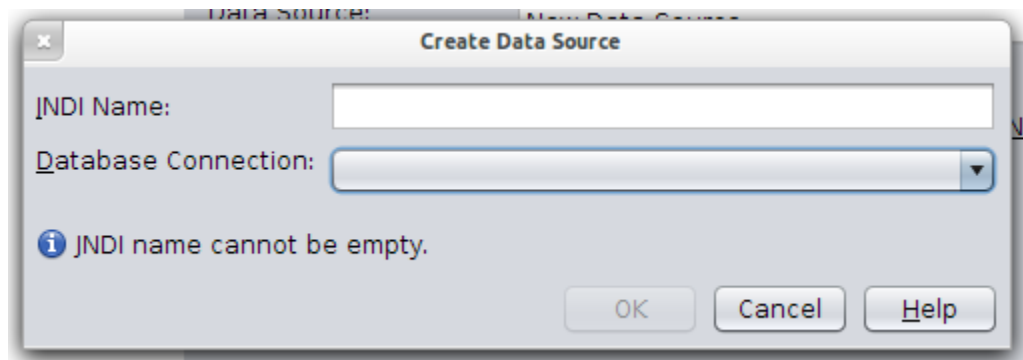


Trus pilih kategori Persistence dan tipe file nya Persistence Unit, kalo udah klik **Next >**.



Buat nama persistence unit nya, trus pilih persistence providernya itu EclipseLink, soalnya itu default bawaan Glassfish.

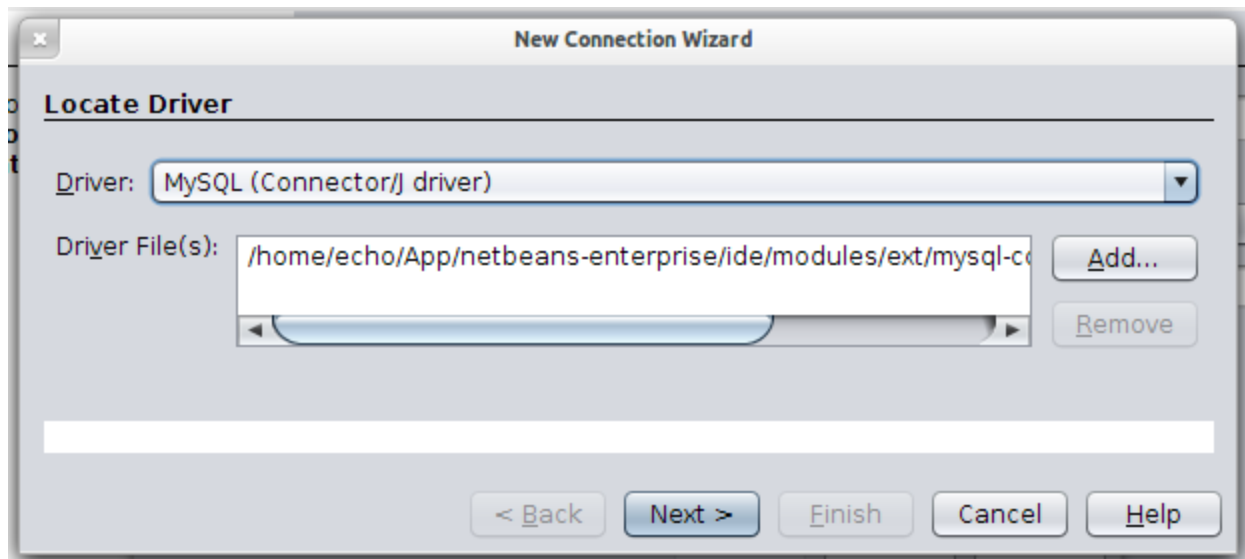
Nah tinggal kita buat datasource nya, pilih combobox si **DataSource** nya, trus pilih **New Data Source**.



Sekarang kita buat datasource nya dulu, datasource itu merupakan objek untuk membuat koneksi ke database.

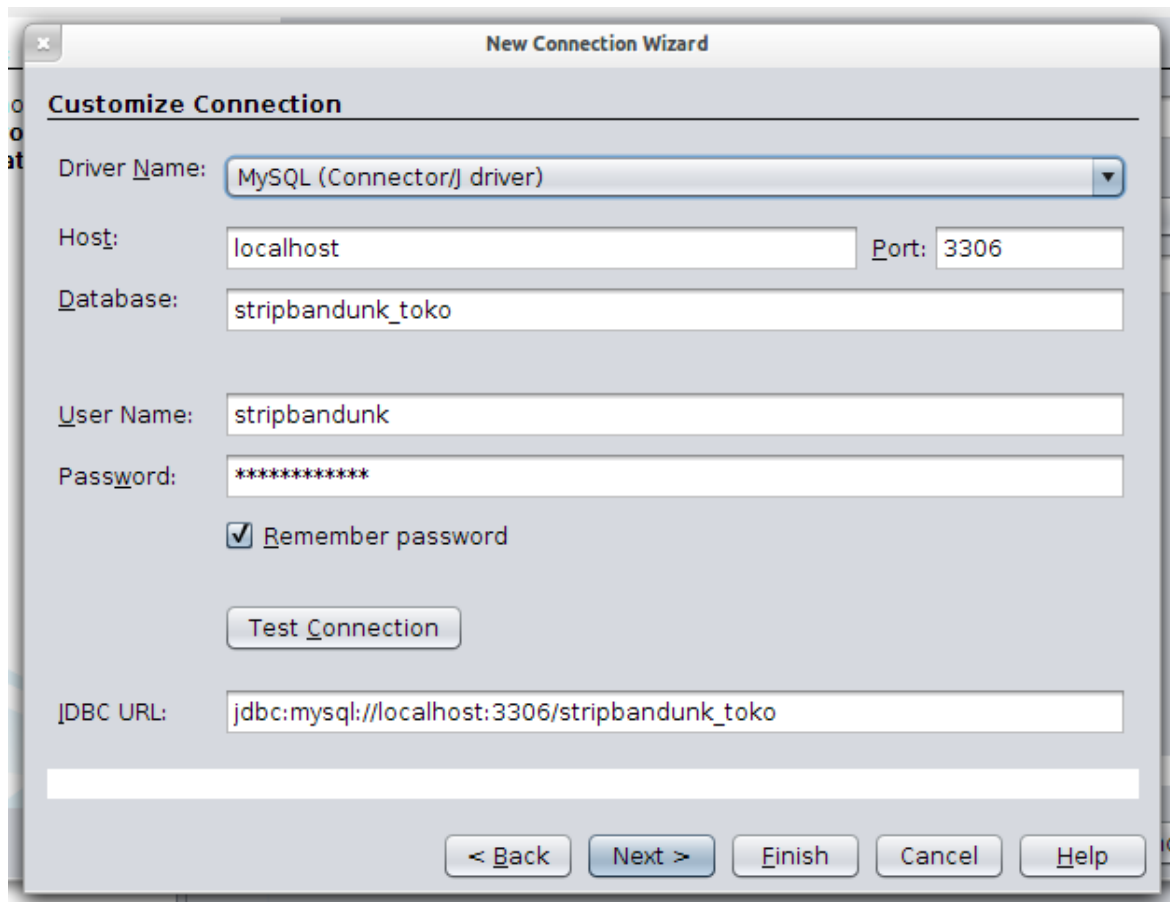
Untuk membuat datasource, pertama pilih dulu database connectionnya.

Sekarang saya anggap belum ada database connectionnya, jadi silahkan pilih combobox nya trus pilih **New Database Connection**.

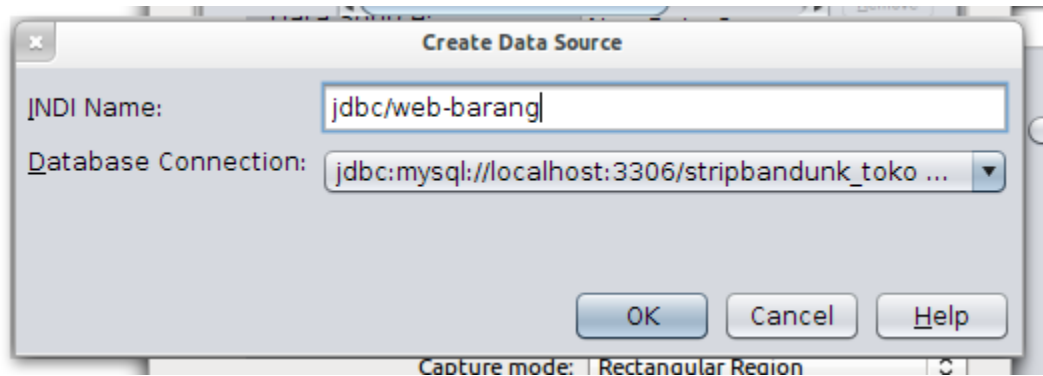


Silahkan pilih Driver yang mau digunakan untuk membuat koneksi database nya, misal saya pake driver MySQL, soalnya saya mau pake database MySQL, kalo kamu sih terserah mau pake database apa aja :P

Kalo udah klik **Next >**.



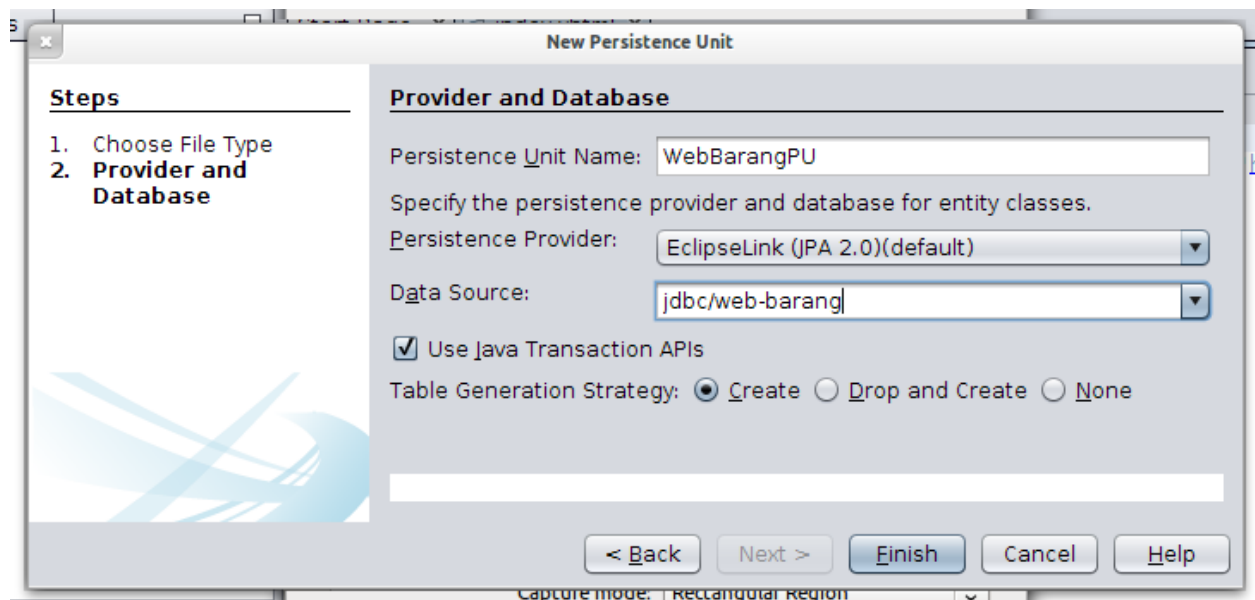
Silahkan isi konfigurasi koneksi databasenya, kalo sudah klik tombol **Finish**.



Balik lagi ke pembuatan datasource, sekarang silahkan tambahkan nama JNDI nya, ini adalah nama yang digunakan sebagai identitas atau unik id nya si objek datasource.

Biasanya diawali dengan jdbc/ misal jdbc/web-barang

Kalo udah klik tombol OK



Balik lagi ke pembuatan Persistence Unit, sekarang pilih table Use Java Transaction API, dan cek radio button Create pada Table Generation Strategy, hal ini agar nanti tabel akan otomatis dibuatkan oleh si JPA, jadi saya gak perlu lagi buat table pake SQL, keren kan? :D

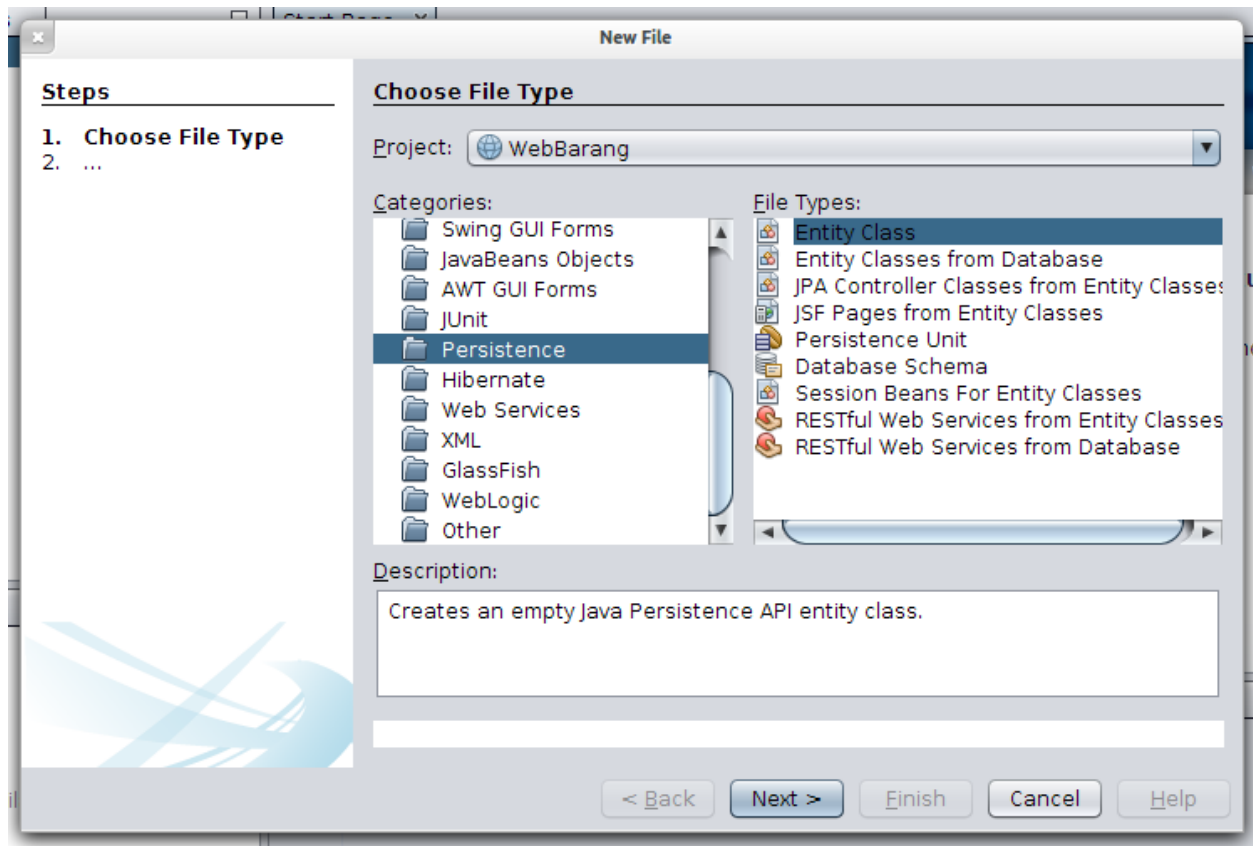
Kalo udah klik **Finish**, dan selesai :D

## Membuat Entitas Barang

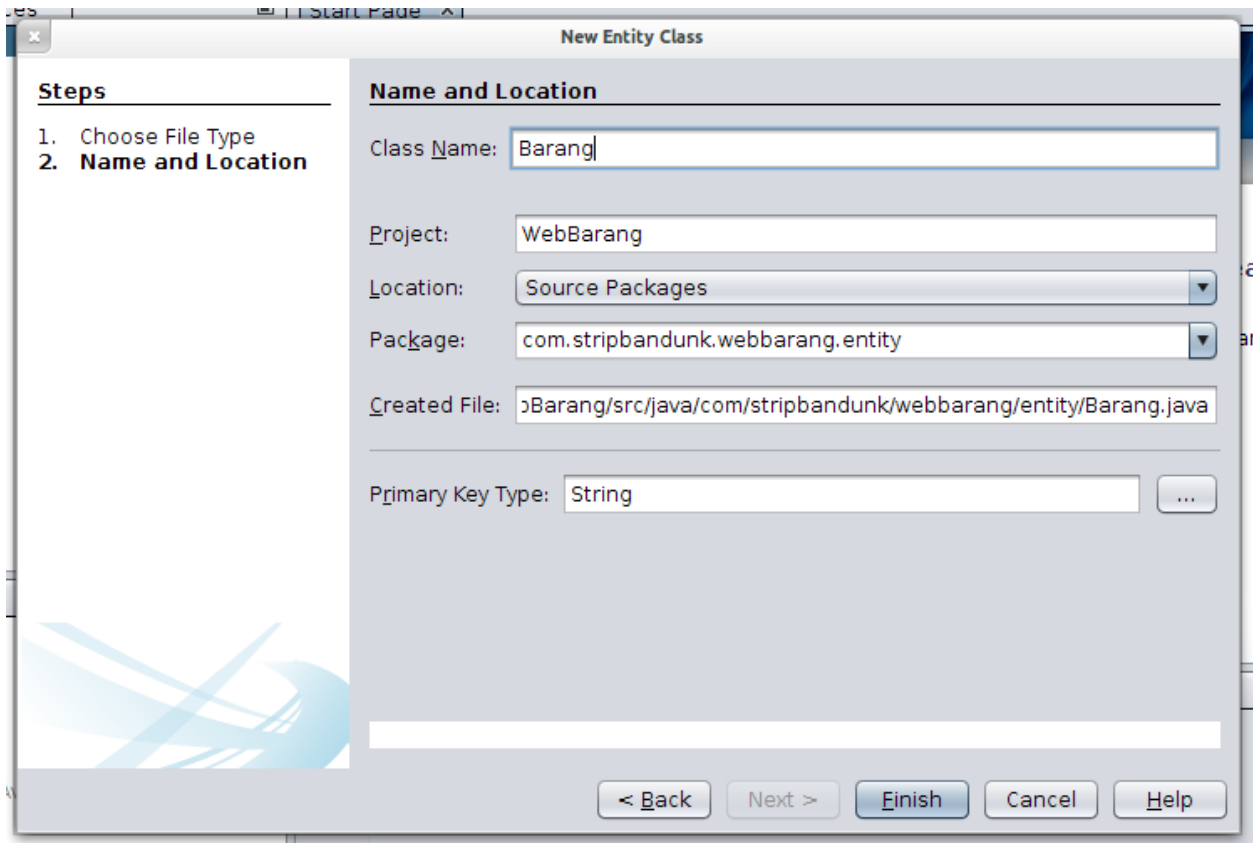
Sekarang setelah membuat konfigurasi JPA, saatnya membuat entitas barang. Kelas entitas merupakan kelas yang merepresentasikan tabel yang ada di database.

Jadi anggap saja kalo sekarang kita sedang membuat tabel di database.

Untuk membuat kelas entitas, kita bisa menggunakan menu **File -> New File**.



Pilih kategori Persistence dan tipe file nya adalah Entity Class, lalu klik **Next**.



Beri nama kelasnya adalah **Barang**, trus jangan lupa kasih **package** juga. Untuk **Primary Key Type**, disini saya menggunakan String, karena nanti kode barang atau id barang akan saya buat berupa String.

Setelah itu klik tombol **Finish**.

Hasilnya adalah seperti pada gambar dibawah ini :

```

27  L  */
28  @Entity
29  public class Barang implements Serializable {
30
31      private static final long serialVersionUID = 1L;
32
33      @Id
34      @GeneratedValue(strategy = GenerationType.AUTO)
35      private String id;
36
37      public String getId() {
38          return id;
39      }
40
41      public void setId(String id) {
42          this.id = id;
43      }

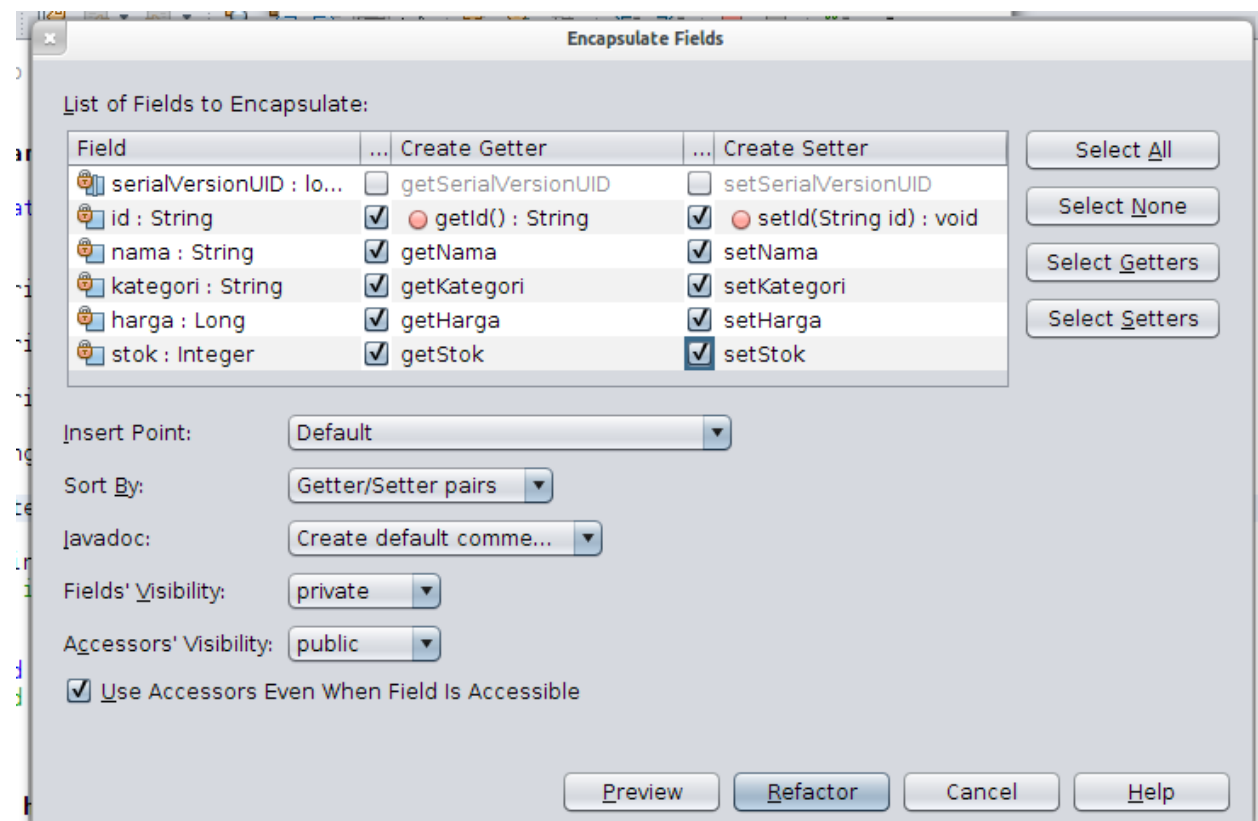
```

Hapus **@GeneratedValue()** yang ada di atribut id, kenapa? Soalnya itu hanya untuk auto increment angka, sedangkan id barang ini saya menggunakan string.

Setelah itu tambahkan atribut-atribut yang diperlukan, seperti nama, harga, stok dan kategori :

```
25  */
26  @Entity
27  public class Barang implements Serializable {
28
29      private static final long serialVersionUID = 1L;
30
31      @Id
32      private String id;
33
34      private String nama;
35
36      private String kategori;
37
38      private Long harga;
39
40      private Integer stok;
41  }
```

Setelah itu buat getter dan setter untuk semua atributnya. Supaya mudah kita dapat menggunakan generator milik NetBeans, caranya klik menu **Refactor -> Encapsulate Fields**.



Ceklist getter dan setter untuk id, nama, kategori, harga dan stok, setelah itu klik tombol **Refactor**.

Dan otomatis getter dan setter akan terbuatkan semuanya :D cool....

```
74
75     /**
76      * @return the nama
77      */
78     public String getNama() {
79         return nama;
80     }
81
82     /**
83      * @param nama the nama to set
84      */
85     public void setNama(String nama) {
86         this.nama = nama;
87     }
88
89     /**
90      * @return the kategori
91      */
92     public String getKategori() {
93         return kategori;
94     }
95
96     /**
97      * @param kategori the kategori to set
98      */
99     public void setKategori(String kategori) {
100         this.kategori = kategori;

```

Selesai, sekarang kita sudah membuat entitas Barang.

Lanjut ke tahapan selanjutnya :D



## Service Menggunakan Enterprise Java Beans

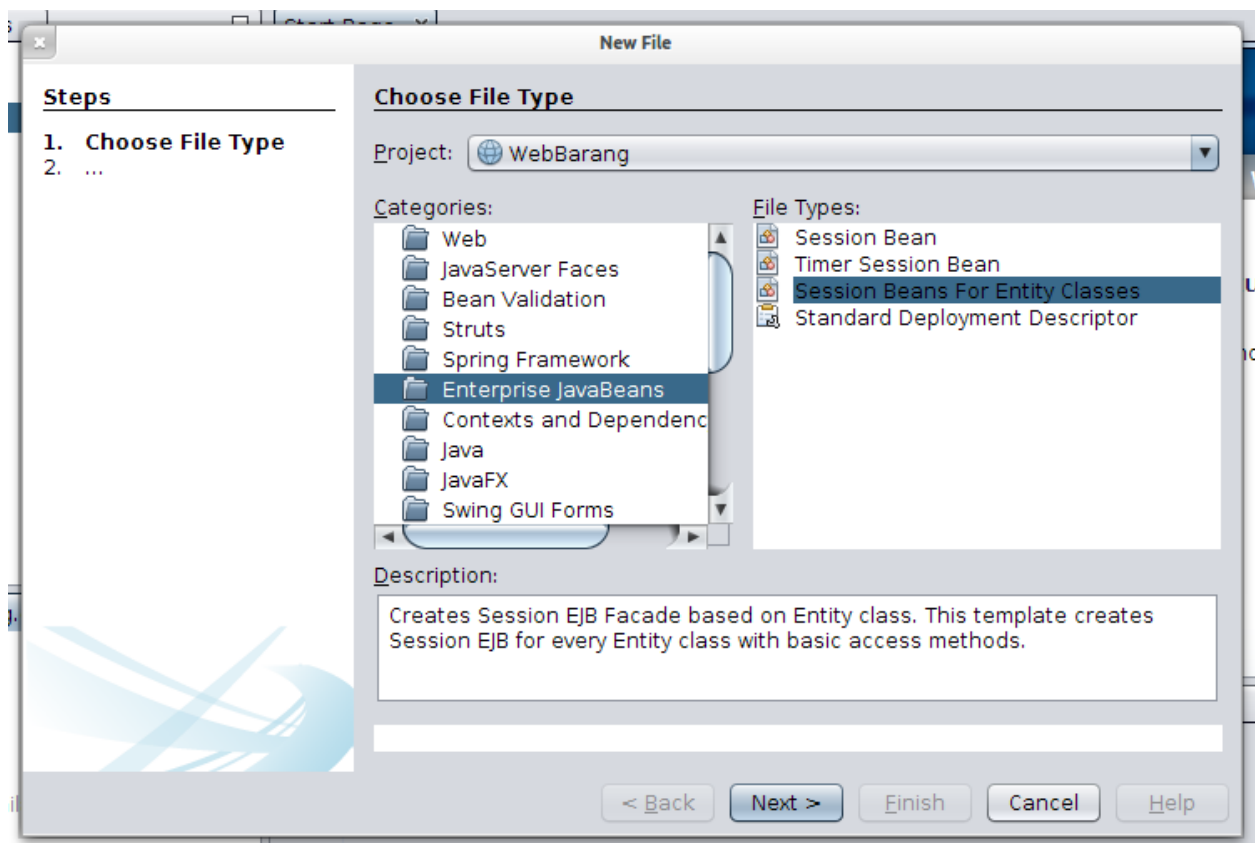
Bagi yang sudah mengenal tentang Data Access Object, bab ini juga kita membuat hal seperti itu. Namun bedanya namanya kita ganti dengan Service, dan disini kita menggunakan EJB dan JPA.

Berterima kasih lah paka NetBeans, karena pembuatan Service menggunakan EJB di NetBeans sangatlah mudah :D

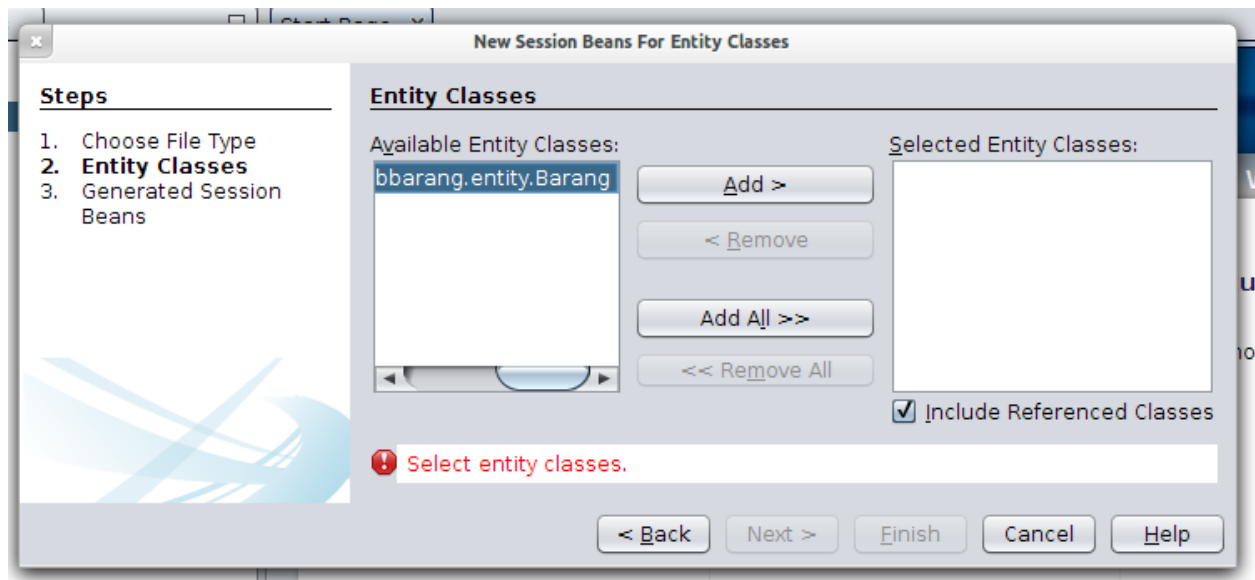
Mudah sekali, hampir seperti membalikkan telapak tangan :D

Masa? Sumpeh deh :P

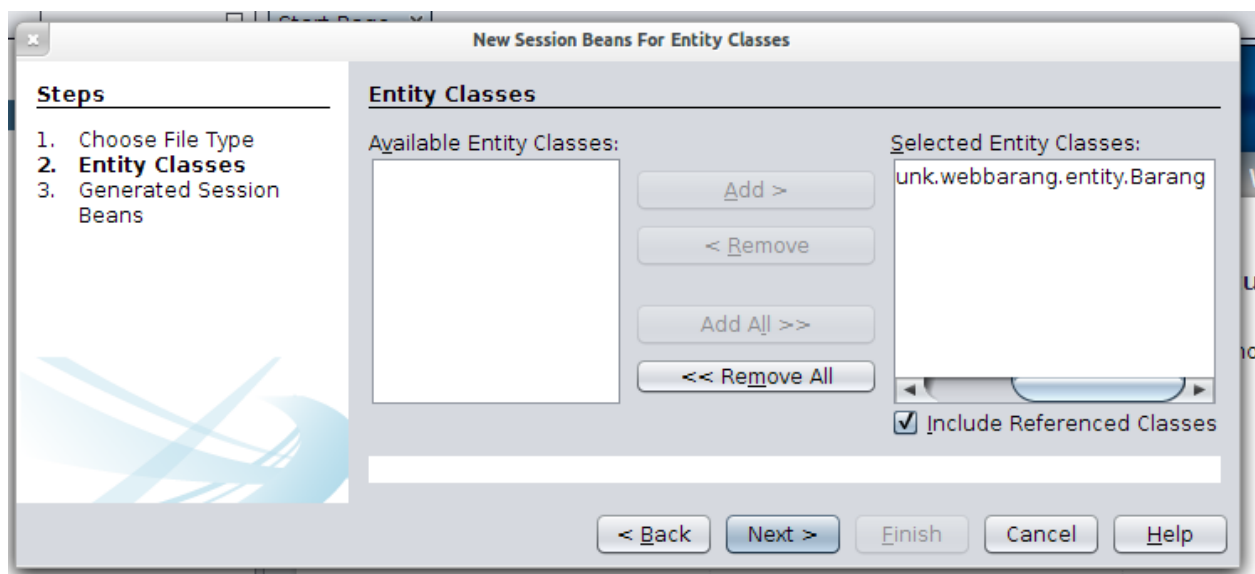
Ok, back to topic, pertama buat sebuah file baru, pilih menu **File -> New File**.



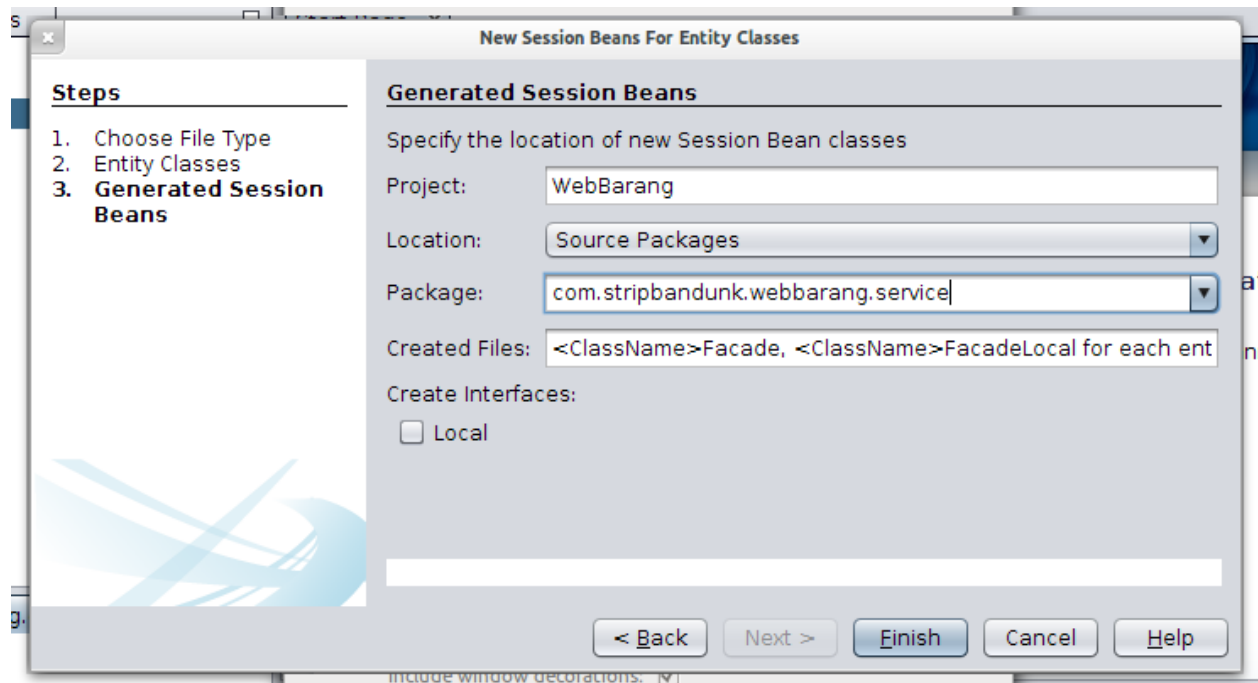
Pilih katgori Enterprise JavaBeans dan tipe file nya Session Beans for Entity Classes. Lalu klik **Next >**.



Pilih entitas Barang, lalu klik tombol **Add >**.



Setelah itu klik tombol **Next >**.



Masukkan nama package nya, disini saya menggunakan package **service**. Lalu kalo sudah, klik tombol **Finish**.

```

26  */
27  @Stateless
28  public class BarangFacade extends AbstractFacade<Barang> {
29
30      @PersistenceContext(unitName = "WebBarangPU")
31      private EntityManager em;
32
33      @Override
34      protected EntityManager getEntityManager() {
35          return em;
36      }
37
38      public BarangFacade() {
39          super(Barang.class);
40      }
41
42  }
43

```

Dan sekarang kelas service nya sudah dibuatkan oleh NetBeans dengan nama **BarangFacade**. Kita tidak perlu melakukan coding lagi :D

Hahaha... mudah kan? Sekarang tinggal lanjut ke next job.

## Validasi Menggunakan Bean Validation

Sebelum mulai membuat halaman web nya, bikin validasi dulu. Validasinya pake Bean Validation. Jangan bikin validasi manual, capek :P

Untuk melakukan validasi menggunakan Bean Validation, harus di integrasikan dengan kelas entitasnya. Jadi sekarang kita buka lagi kelas Barang nya.

### Validasi Id Barang

Pertama kita validasi id atau kode barang. Disini misal saya ingin id barang itu panjangnya tidak boleh lebih dari 10 karakter, dan tidak boleh kosong :

```
18 import java.io.Serializable;
19 import javax.persistence.Entity;
20 import javax.persistence.Id;
21 import javax.validation.constraints.Min;
22 import javax.validation.constraints.Size;
23 import org.hibernate.validator.constraints.NotEmpty;
24
25 @Entity
26 public class Barang implements Serializable {
27
28     @Id
29     @NotEmpty(message = "Kode tidak boleh kosong")
30     @Size(max = 10, message = "Kode tidak boleh lebih 10 karakter")
31     private String id;
```

@NotEmpty digunakan agar kode tidak boleh kosong, dan @Size digunakan untuk melakukan validasi panjang karakter dimana max nya adalah 10 karakter. Untuk atribut message adalah pesan kesalahan ketika validasi gagal.

### Validasi Nama Barang

Untuk nama barang, validasinya tidak boleh kosong dan juga panjang karakter maksimal adalah 50 karakter.

```
33     @NotEmpty(message = "Nama tidak boleh kosong")
34     @Size(max = 50, message = "Nama tidak boleh lebih 50 karakter")
35     private String nama;
```

### Validasi Kategori Barang

Kalo untuk kategori barang, validasinya cuma panjang karakter aja 50 karakter, disini saya anggap kategori boleh kosong :

```
36
37     @Size(max = 50, message = "Kategori tidak boleh lebih 50 karakter")
38     private String kategori;
```

## Validasi Harga Barang

Untuk harga barang, pastinya tidak boleh kurang dari 1 rupiah, mana ada barang 0 rupiah, barang apa?

```
39
40 @Min(value = 1, message = "Harga tidak boleh kurang dari 1")
41 private Long harga;
42
```

## Validasi Stok Barang

Terakhir untuk stok, boleh 0, tapi yang pasti stok tidak boleh negative :

```
39
40 @Min(value = 1, message = "Harga tidak boleh kurang dari 1")
41 private Long harga;
42
```

Dan sekarang kita sudah selesai menambahkan validasi menggunakan Bean Validation.

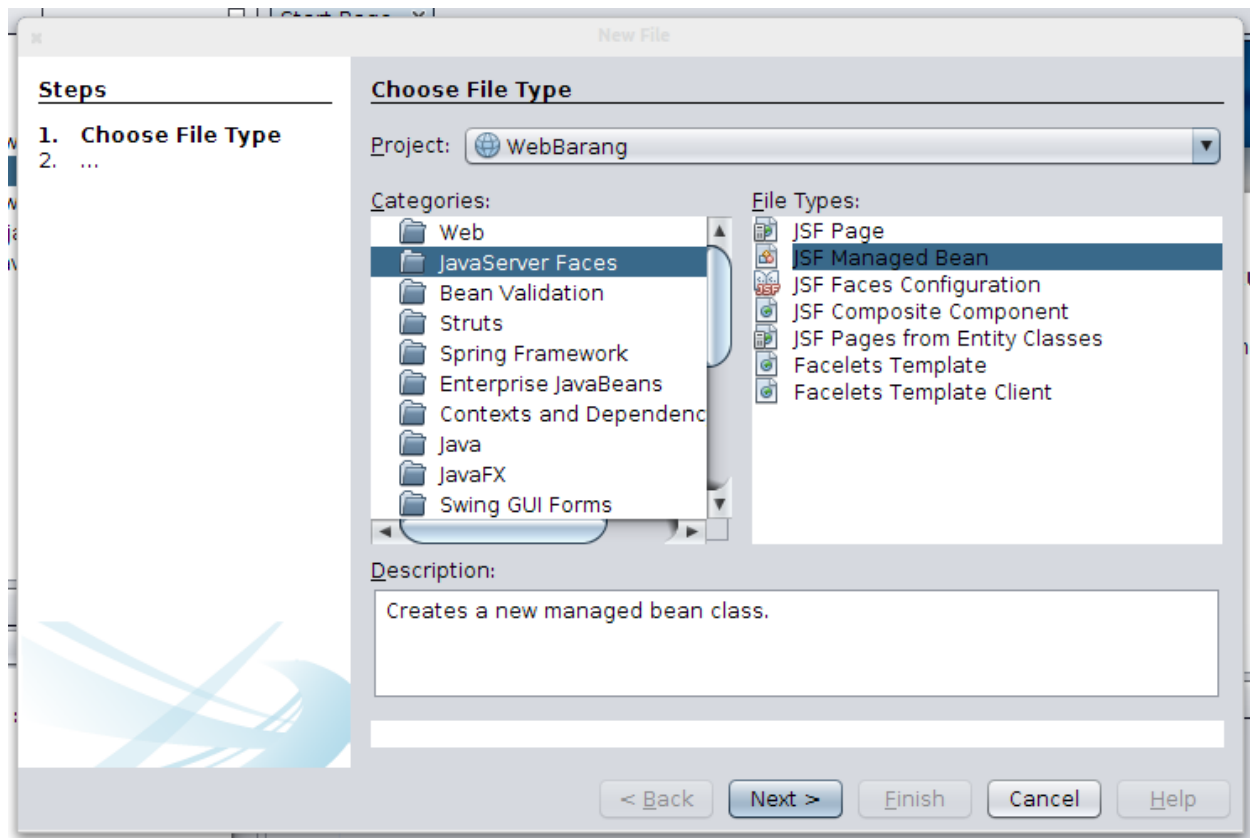
```
18 import java.io.Serializable;
19 import javax.persistence.Entity;
20 import javax.persistence.Id;
21 import javax.validation.constraints.Min;
22 import javax.validation.constraints.Size;
23 import org.hibernate.validator.constraints.NotEmpty;
24
25 @Entity
26 public class Barang implements Serializable {
27
28     @Id
29     @NotEmpty(message = "Kode tidak boleh kosong")
30     @Size(max = 10, message = "Kode tidak boleh lebih 10 karakter")
31     private String id;
32
33     @NotEmpty(message = "Nama tidak boleh kosong")
34     @Size(max = 50, message = "Nama tidak boleh lebih 50 karakter")
35     private String nama;
36
37     @Size(max = 50, message = "Kategori tidak boleh lebih 50 karakter")
38     private String kategori;
39
40     @Min(value = 1, message = "Harga tidak boleh kurang dari 1")
41     private Long harga;
42
43     @Min(value = 0, message = "Stok tidak boleh kurang dari 0")
44     private Integer stok;
45 }
```

## Controller Menggunakan Manage Bean

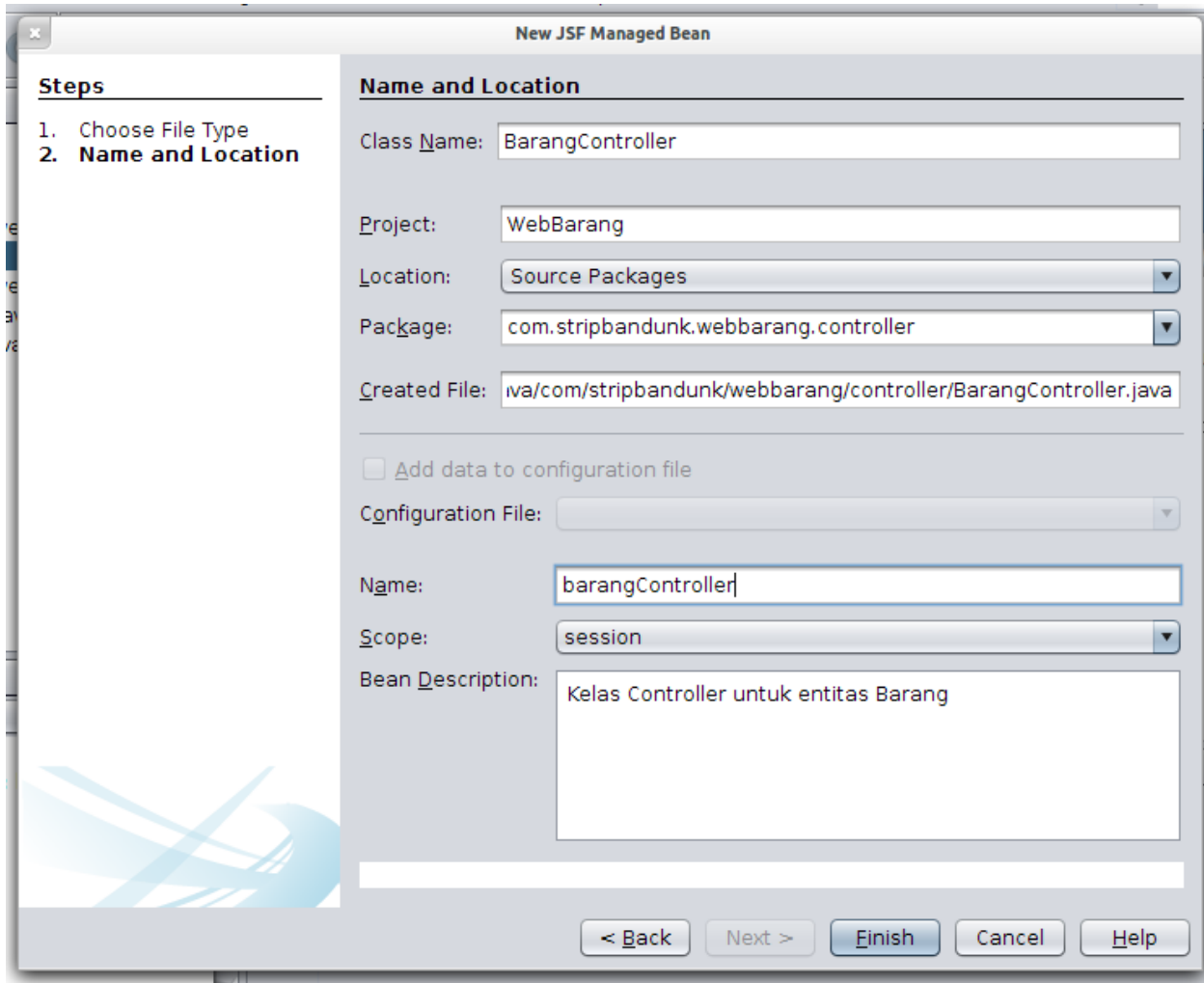
Controller adalah kelas yang digunakan untuk melakukan kontrol terhadap aksi-aksi di web. Controller bertugas seluruhnya untuk memberikan data dan merespon aksi yang diminta oleh pengguna.

### Membuat Manage Bean

Pertama kita perlu membuat kelas Manage Bean terlebih dahulu, caranya buat file baru **File -> New File** :



Pilih kategori JavaServer Faces dan tipe file nya JSF Managed Bean, setelah itu klik **Next >**



Tambahkan nama kelasnya **BarangController**, lalu jangan lupa tambahkan package nya, saya buat package yang berakhiran **controller**. Lalu tambahkan **Name**, **Scope** nya ubah jadi **session**, supaya hanya dibuat satu objek per session, lalu silahkan tambahkan deskripsinya. Selanjutnya klik tombol **Finish**.

```

16 package com.stripbandunk.webbarang.controller;
17
18 import java.io.Serializable;
19 import javax.enterprise.context.SessionScoped;
20 import javax.inject.Named;
21
22 @Named(value = "barangController")
23 @SessionScoped
24 public class BarangController implements Serializable {
25     |
26
27 }
28

```

Hasilnya seperti pada gambar diatas.

Sekarang saatnya kita tambahkan metode-metode untuk aksi-aksi di web.

### Aksi Daftar Barang

Untuk aksi daftar data barang, nanti di webnya, halamannya akan berisikan tabel yang menampilkan data barang. Untuk melakukan itu, kita hanya perlu membuat sebuah metode getter yang mengembalikan data List of Barang.

Data List of Barang ini kita ambil melalui EJB, jadi seperti ini :

```
17
18 import com.stripbandunk.webbarang.entity.Barang;
19 import com.stripbandunk.webbarang.service.BarangFacade;
20 import java.io.Serializable;
21 import java.util.List;
22 import javax.ejb.EJB;
23 import javax.enterprise.context.SessionScoped;
24 import javax.inject.Named;
25
26 @Named(value = "barangController")
27 @SessionScoped
28 public class BarangController implements Serializable {
29
30     @EJB
31     private BarangFacade facade;
32
33     public List<Barang> getDaftarBarang() {
34         return facade.findAll();
35     }
36 }
37
```

### Aksi Tambah Barang

Logikanya, saat pertama kali muncul maka halaman daftar barang akan muncul, lalu ada link Tambah Barang, saat di klik mana akan masuk ke halaman tambah data barang.

Aksi tersebut perlu kita buat metodenya, dimana metodenya mengembalikan data String yang menunjukkan halaman web nya.



```

26 @Named(value = "barangController")
27 @SessionScoped
28 public class BarangController implements Serializable {
29
30     @EJB
31     private BarangFacade facade;
32
33     private Barang barang;
34
35     public List<Barang> getDaftarBarang() {
36         return facade.findAll();
37     }
38
39     public Barang getBarang() {
40         return barang;
41     }
42
43     public String prepareCreate() {
44         barang = new Barang();
45         return "/create";
46     }
47 }
48

```

Metode getBarang() digunakan untuk mengambil objek barang yang nanti akan di binding ke dalam form, dan dalam metode prepareCreate() objek barang tersebut dibuat baru dengan menggunakan **new**. Dalam metode prepareCreate() mengembalikan nilai /create, yang artinya nanti akan menampilkan halaman /create.stripbandunk

Selanjutnya pengguna memasukkan data barang, setelah itu klik tombol save dan data barang di simpan ke dalam database dan kembali lagi ke halaman daftar barang

Aksi tersebut juga perlu dibuatkan sebuah metode baru, misal create();

```

47
48 public String create() {
49     facade.create(barang);
50     return "/index";
51 }
52

```

Metode create() menghasilkan string /index yang artinya setelah di simpan nanti akan kembali ke halaman /index.stripbandunk

## Aksi Ubah Barang

Selanjutnya untuk aksi ubah, sama seperti aksi buat barang, pertama pasti pengguna harus memilih salah satu barang yang akan di ubah.

```

52
53 public String prepareEdit(String id) {
54     barang = facade.find(id);
55     return "/edit";
56 }
57

```

Dalam prepareEdit() ditambahkan parameter String id yang merupakan parameter untuk id barang yang akan diubah.

Selanjutnya halaman /edit.stripbandunk akan muncul dan setelah selesai mengedit, penggunaan akan menekan tombol simpan. Aksi ini juga akan mengakses metode edit()

```

57
58 public String edit() {
59     facade.edit(barang);
60     return "/index";
61 }

```

### Aksi Hapus Barang

Dan yang terakhir adalah aksi hapus barang, kalo ini sih gak perlu lagi pake prepare dulu, langsung aja hapus berdasarkan kode barangnya.

Untuk aksi ini, kita buat metode dengan nama delete()

```

62
63 public String delete(String kode) {
64     barang = facade.find(kode);
65     facade.remove(barang);
66     return "/index";
67 }

```

Akhirnya selesai juga membuat kelas controller :D

Tinggal sekarang kita bikin web nya pake Java Server Faces, ayo!!!

## View Menggunakan Java Server Faces

Bagian yang langsung berinteraksi dengan pemakai itu disebut presentation tier, atau simple nya sih bisa dibilang view.

Karena di project ini, saya mau buatnya itu project web, jadi saya pilih untuk view nya itu adalah java server faces.

Kenapa JSF? Enggak Struts, Spring MVC, Tapestry atau bahkan JSP?

Suka-suka saya dunk, saya yang bikin buku, kalo gak mau pake JSF, bikin aja buku sendiri :P

## Nampilin Data Barang

Pertama, saat halaman index.stripbandunk muncul, disini saya pengen nampilin dulu daftar barang pake tabel, trus ada tombol tambah barang di atas tabel nya, dan di tiap baris data barang, ada tombol buat hapus ama edit.

Jadi pertama sekarang buka halaman **index.xhtml** nya. Inget **index.xhtml**, bukan index.stripbandunk, tapi walaupun halaman nya .xhtml, tapi nanti pas buka di browser sih .stripbandunk :D

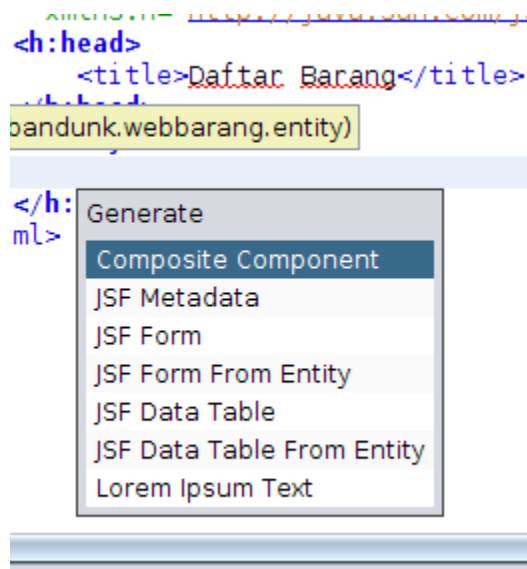
Isi index.xhtml itu default nya waktu bikin project itu kayak gini :

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
3  <html xmlns="http://www.w3.org/1999/xhtml"
4      xmlns:h="http://java.sun.com/jsf/html">
5      <h:head>
6          <title>Facelet Title</title>
7      </h:head>
8      <h:body>
9          Hello from Facelets
10     </h:body>
11 </html>
12
13
```

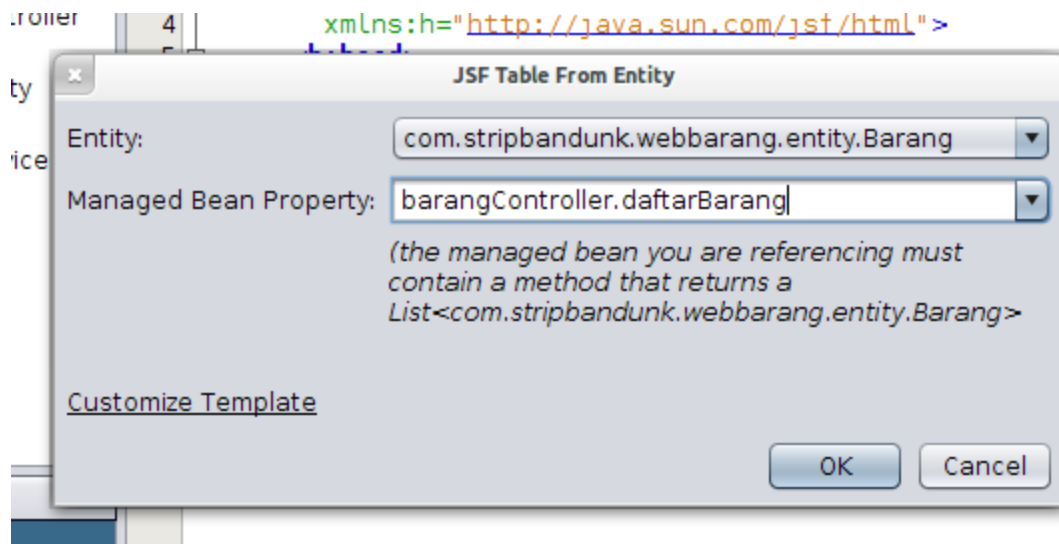
Sekarang ubah dulu jadi kayak gini :

```
html  h:body
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Tran:
3  <html xmlns="http://www.w3.org/1999/xhtml"
4      xmlns:h="http://java.sun.com/jsf/html">
5      <h:head>
6          <title>Daftar Barang</title>
7      </h:head>
8      <h:body>
9          |
10     </h:body>
11 </html>
12
13
```

Trus kursor nya simpan di dalam tag <h:body>, lalu pilih menu Source -> Insert Code, nanti bakal muncul menu kayak gini :



Pilih menu **JSF Data Table From Entity**.



Pilih entity nya kelas Barang, trus pilih manage bean property nya itu barangController.daftarBarang, trus klik **OK**.

```

html h:body f:view h:form
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
3  <html xmlns="http://www.w3.org/1999/xhtml"
4      xmlns:h="http://java.sun.com/jsf/html">
5      <h:head>
6          <title>Daftar Barang</title>
7      </h:head>
8      <h:body>
9          <f:view>
10         <h:form>
11             <h1><h:outputText value="List"/></h1>
12             <h:dataTable value="#{barangController.daftarBarang}"
13                 <h:column>
14                     <f:facet name="header">
15                         <h:outputText value="Id"/>
16                     </f:facet>
17                     <h:outputText value="#{item.id}"/>
18                 </h:column>
19                 <h:column>
20                     <f:facet name="header">
21                         <h:outputText value="Nama"/>
22                     </f:facet>
23                     <h:outputText value="#{item.nama}"/>

```

Sekarang otomatis dibuatin data table oleh NetBeans, jadi kita gak perlu buat manual data table nya :D

Ada error di <f:view> dan <f:facet>, ini soalnya tag f belum di import, caranya klik logo lampu bohlam, trus pilih import :

```

6      <title>Daftar Barang</title>
7  </h:head>
8  <h:body>
9      <f:view>
10         <h1>Daftar Barang</h1>
11         <h:dataTable value="#{barangController.daftarBarang}">
12             <h:column>
13

```

Dan sekarang akan ditambahkan import tag untuk f :

```

2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:h="http://java.sun.com/jsf/html"
5        xmlns:f="http://java.sun.com/jsf/core">
6      <h:head>

```

## Nambah Data Barang

Kalo udah nampilin data barang, saatnya sekarang nambah data barang.

Seperti yang sudah dibahas di controller, ada dua aksi di tambah data barang ini, pertama prepare() yaitu aksi untuk menampilkan form tambah barang, dan aksi menyimpan data barang di form.

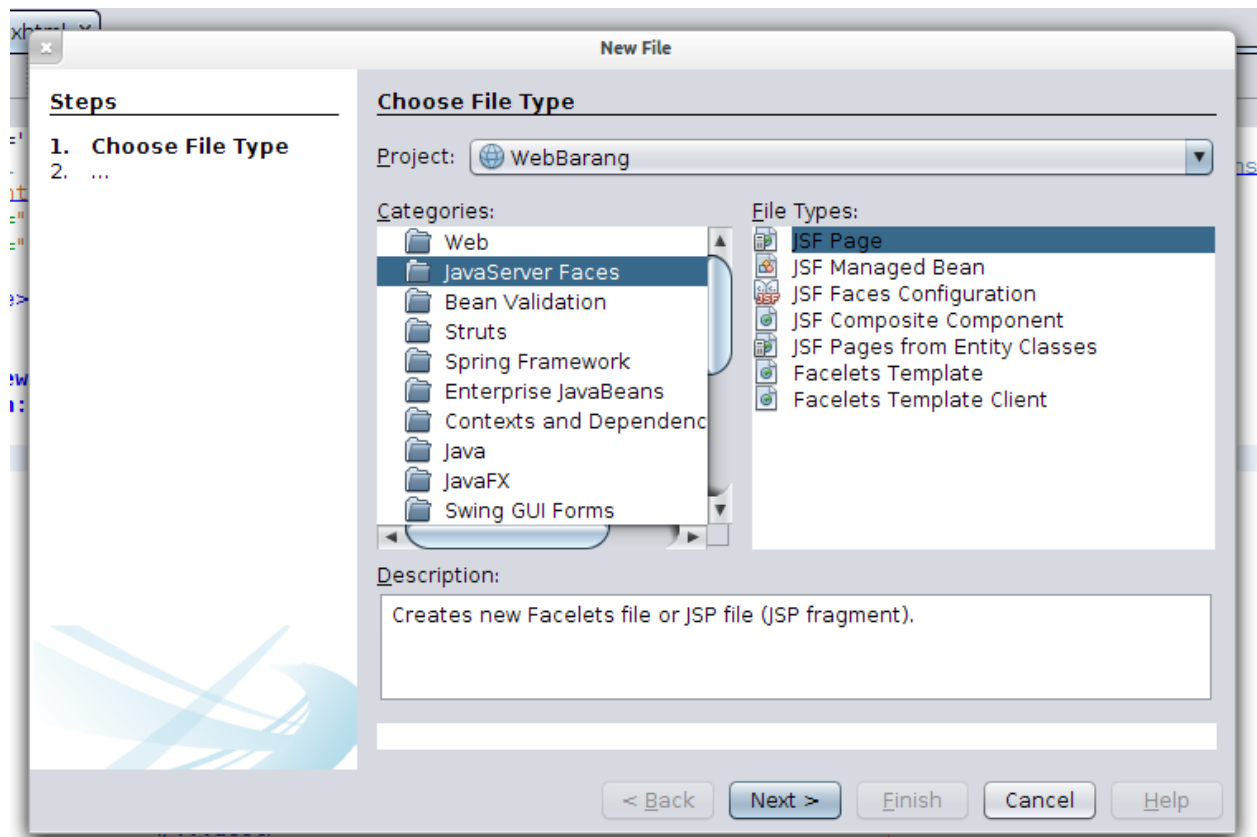
Sekarang tinggal tambahkan link diatas data table untuk membuka form tambah data barang.

```

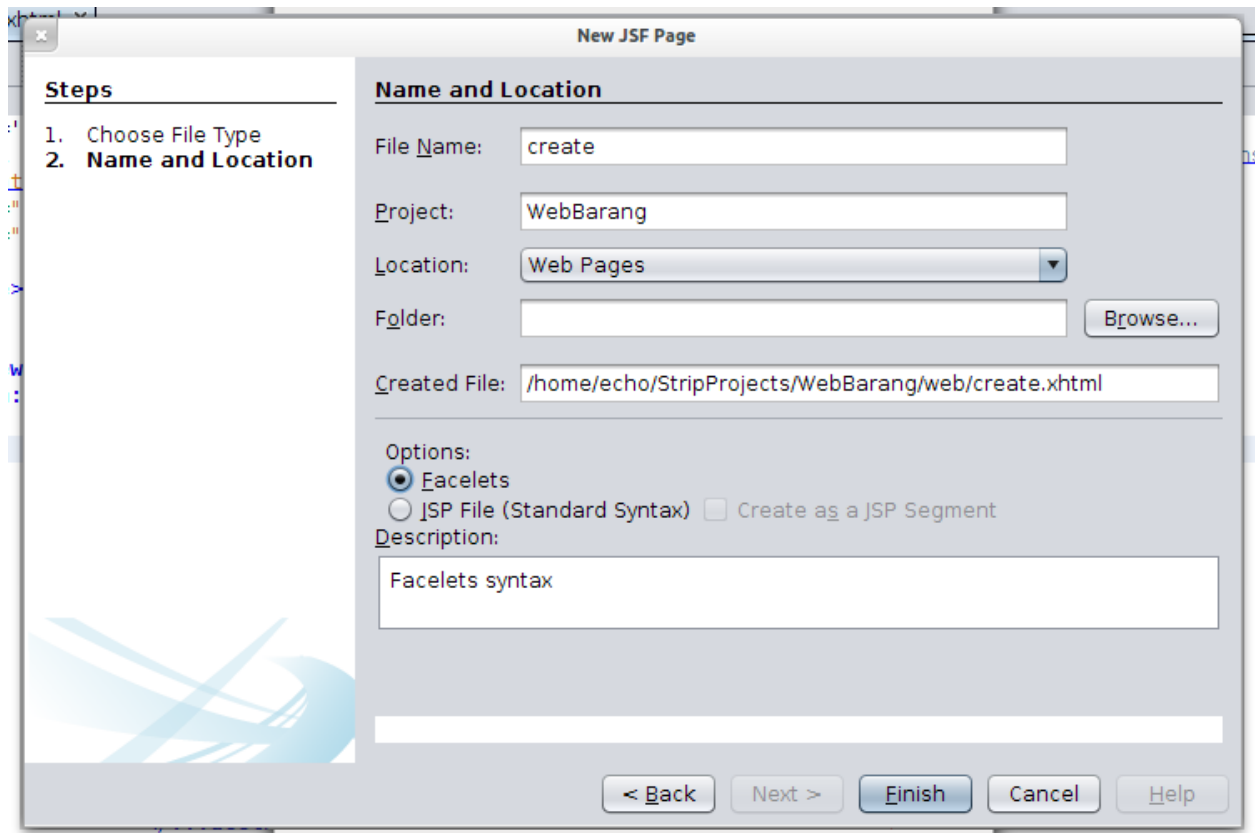
8  </h:head>
9  <h:body>
10     <f:view>
11         <h:form>
12             <h:commandButton value="Tambah Barang" |
13                             action="#{barangController.prepareCreate()}" />
14             <h1><h:outputText value="List" /></h1>
15             <h:dataTable value="#{barangController.daftarBarang}">
16                 <h:column>
17                     <h:commandButton value="Tambah" />
18

```

Selanjutnya, buat file JSF baru dengan nama create.xhtml, caranya pilih menu **File -> New File**.

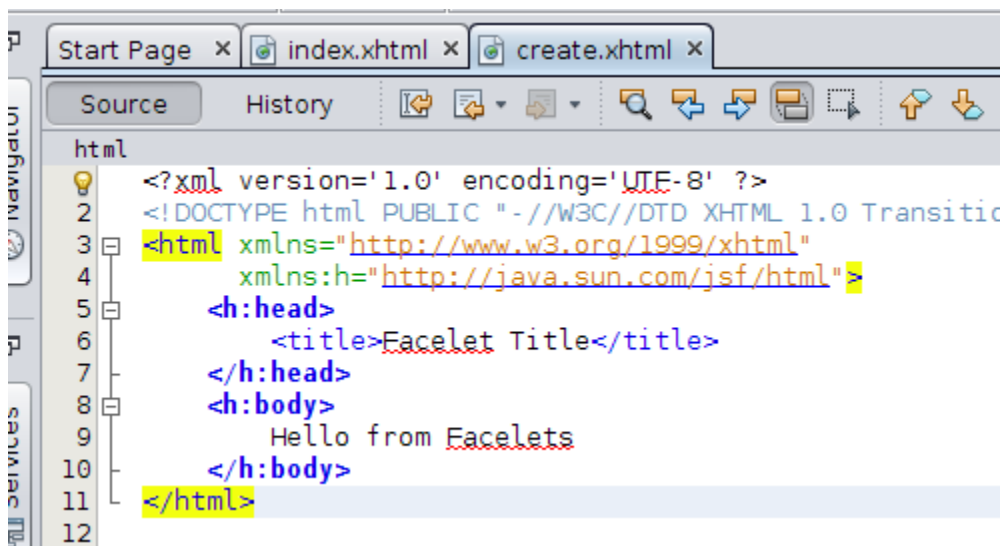


Pilih kategori JavaServer Faces dan tipe file nya JSF Page, trus klik **Next >**.



Beri nama file nya, create dan options nya itu facelets. Kalo udah, klik **Finish**.

Dan ini isi file nya :



Sekarang tinggal ubah jadi seperti ini :

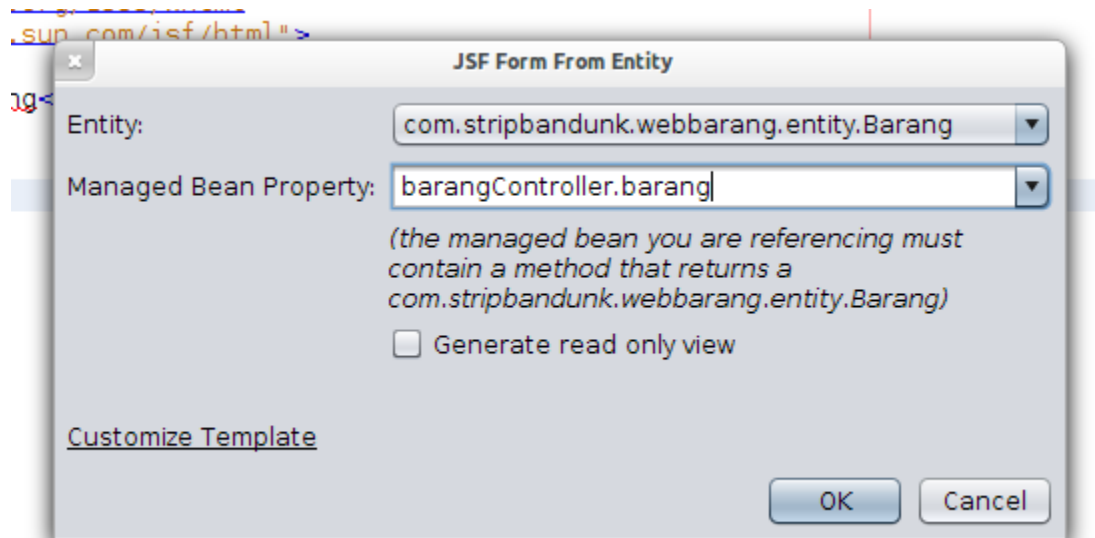


```

html h:body
1 <?xml version='1.0' encoding='UTF-8' ?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Tran
3 <html xmlns="http://www.w3.org/1999/xhtml"
4     xmlns:h="http://java.sun.com/jsf/html">
5     <h:head>
6         <title>Tambah Barang</title>
7     </h:head>
8     <h:body>
9         |
10    </h:body>
11 </html>
12

```

Lalu masukkan cursor di dalam tag <h:body> trus pilih menu **Source -> Insert Code**, setelah muncul dialog seperti sebelumnya di daftar barang, pilih **JSF Form From Entity**.



Pilih entity kelas Barang, dan manage beans property nya barangController.barang, lalu klik **OK**.

```

6      <title>Tambah Barang</title>
7      </h:head>
8      <h:body>
9          <f:view>
10             <h:form>
11                 <h1><h:outputText value="Create/Edit" /></h1>
12                 <h:panelGrid columns="2">
13                     <h:outputLabel value="Id:" for="id" />
14                     <h:inputText id="id" value="#{barangController.ba
15                     <h:outputLabel value="Nama:" for="nama" />
16                     <h:inputText id="nama" value="#{barangController.
17                     <h:outputLabel value="Kategori:" for="kategori" /
18                     <h:inputText id="kategori" value="#{barangControl
19                     <h:outputLabel value="Harga:" for="harga" />
20                     <h:inputText id="harga" value="#{barangController
21                     <h:outputLabel value="Stok:" for="stok" />
22                     <h:inputText id="stok" value="#{barangController.
23                 </h:panelGrid>
24             </h:form>
25          </f:view>
26      </h:body>
27 </html>
28

```

Sekarang otomatis NetBeans akan membuat Form untuk entitas Barang, tinggal kita pake saja :D

Kalo ada error, seperti biasa, import dulu tag f nya.

Ubah judul di Form nya jadi seperti ini :

```

<h:form>
    <h1><h:outputText value="Tambah Barang" /></h1>
    <h:panelGrid columns="2">

```

Selanjutnya dibagian bawah, tambah tombol yang mengeksekusi metode create() di controller :

```

22      <h:outputLabel value="Stok:" for="stok" />
23      <h:inputText id="stok" value="#{barangController.
24
25      <h:commandButton value="Simpan"
26                      action="#{barangController.create()}" />
27
28      </h:panelGrid>
29  </h:form>

```

Karena kita menggunakan Bean Validation, dan defaultnya itu JSF tidak melakukan validasi menggunakan BeanValidation, jadi untuk mengaktifkan BeanValidation, kita perlu menggunakan tag <f:validateBean> di dalam form nya, jadi seperti ini :

```

11 <h:form>
12 <f:validateBean>
13 <h:outputText value="Tambah Barang" />
14 <h:panelGrid columns="2">
15 <h:outputLabel value="Id:" for="id" />
16 <h:inputText id="id" value="#{barangC
17 <h:outputLabel value="Nama:" for="nam
18 <h:inputText id="nama" value="#{baran
19 <h:outputLabel value="Kategori:" for=
20 <h:inputText id="kategori" value="#{b
21 <h:outputLabel value="Harga:" for="ha
22 <h:inputText id="harga" value="#{bara
23 <h:outputLabel value="Stok:" for="sto
24 <h:inputText id="stok" value="#{baran
25
26 <h:commandButton value="Simpan"
27 action="#{barangController.cre
28
29 </h:panelGrid>
30 </f:validateBean>
31 </h:form>
32 </f:view>

```

## Ngedit Data Barang

Selanjutnya, kalo udah tambah data barang, saatnya ngedit data barang, caranya hampir sama ama nambah data barang, yang membedakan cuma aksi dan lokasi tombol edit barang nya ada di tiap baris di data table.

Sekarang tambah satu kolom baru di data table nya, trus masukkin tombol edit barang, jadi kayak gini :

```

13 <f:facet name="header">
14 <h:outputText value="Stok" />
15 </f:facet>
16 <h:outputText value="#{item.stok}" />
17 </h:column>
18
19 <h:column>
20 <f:facet name="header">
21 <h:outputText value="Aksi" />
22 </f:facet>
23 <h:commandButton value="Edit"
24 action="#{barangController.prepareEdit(item.id)}" />
25 </h:column>
26
27 </h:dataTable>
28 </h:form>
29 </f:view>

```

Terus bikin file JSF baru dengan nama **edit.xhtml**.

Gak perlu dijelasin lagi kan cara bikinnya :P Sebelumnya kan udah saya jelasin, kebangetan kalo lupa :P

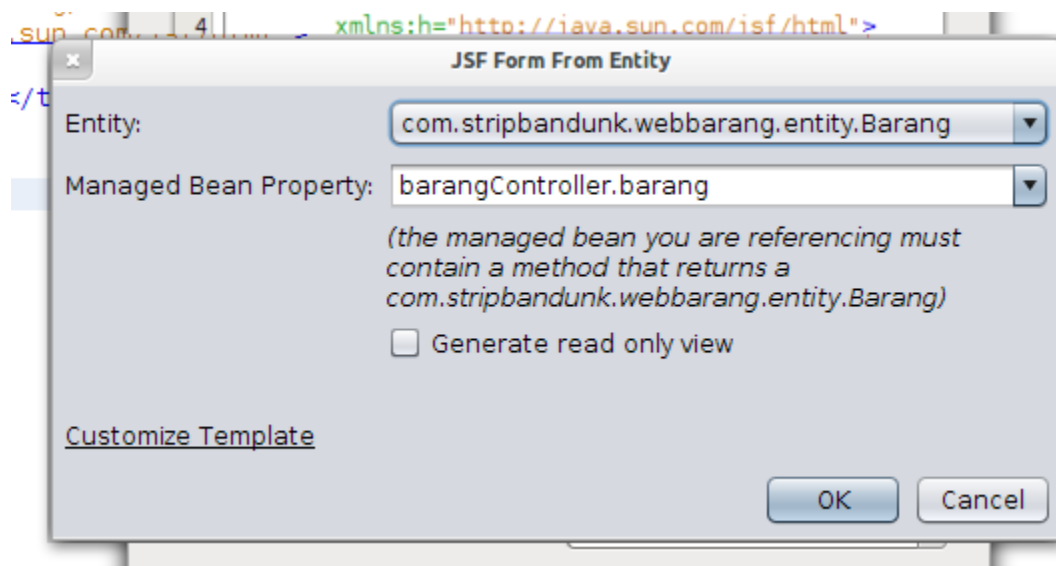
Tinggal ubah file edit.xhtml nya jadi kayak gini :

```

12  html h:body
13  <?xml version='1.0' encoding='UTF-8' ?>
14  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Trans
15  <html xmlns="http://www.w3.org/1999/xhtml"
16  xmlns:h="http://java.sun.com/jsf/html">
17  <h:head>
18  <title>Edit Barang</title>
19  </h:head>
20  <h:body>
21  |
22  </h:body>
23  </html>

```

Trus pake menu yang sama, **Source -> Insert Code**, pilih lagi **JSF Form From Entity**.



Pilih lagi entity nya Barang dan manage bean property nya itu barangController.barang :

Tahapannya sama dengan tambah data barang, yang membedakan sekarang cuma di aksi tombol simpannya, kalo di tambah data barang ke barangController.create() kalo disini barangController.edit().

Lengkapnya kayak gini nih :

```

9      <h:body>
10     <f:view>
11     <h:form>
12         <h1><h:outputText value="Edit Data Barang"/></h1>
13         <f:validateBean>
14         <h:panelGrid columns="2">
15             <h:outputLabel value="Id:" for="id" />
16             <h:inputText id="id" value="#{barangController.ba
17             <h:outputLabel value="Nama:" for="nama" />
18             <h:inputText id="nama" value="#{barangController.l
19             <h:outputLabel value="Kategori:" for="kategori" />
20             <h:inputText id="kategori" value="#{barangControl
21             <h:outputLabel value="Harga:" for="harga" />
22             <h:inputText id="harga" value="#{barangController
23             <h:outputLabel value="Stok:" for="stok" />
24             <h:inputText id="stok" value="#{barangController.l
25
26             <h:commandButton value="Simpan"
27                 action="#{barangController.edit()}" />
28
29         </h:panelGrid>
30     </f:validateBean>
31 </h:form>

```

## Ngapus Barang

Terakhir adalah hapus data barang, kalo ini gak perlu lagi pake form, cukup tambhin tombol hapus barang di tiap baris di data table.

Untuk komlomnya, digabung aja ama tombol Edit, jadi kayak gini :

```

46         <h:outputText value="#{item.stok}" />
47     </h:column>
48
49     <h:column>
50     <f:facet name="header">
51         <h:outputText value="Aksi" />
52     </f:facet>
53     <h:commandButton value="Edit"
54         action="#{barangController.prepareEdit(item.id)}"
55     <h:commandButton value="Hapus"
56         action="#{barangController.delete(item.id)}" />
57     </h:column>
58
59 </h:dataTable>
60 </h:form>
61 </f:view>

```

Untuk jaga-jaga kalo misal pengguna gak sengaja klik tombol Hapus, saya tambhin konfirmasi pake JavaScript, jadi kayak gini hasil akhirnya :

```
</t:facet>
<h:commandButton value="Edit"
    action="#{barangController.prepareEdit(item.id)}" />
<h:commandButton value="Hapus"
    action="#{barangController.delete(item.id)}"
    onclick="return confirm('Anda Yakin?');" />
</h:column>
```

Selesai deh :D

Tinggal kita uji coba, mudah-mudahan gak ada error euy :D

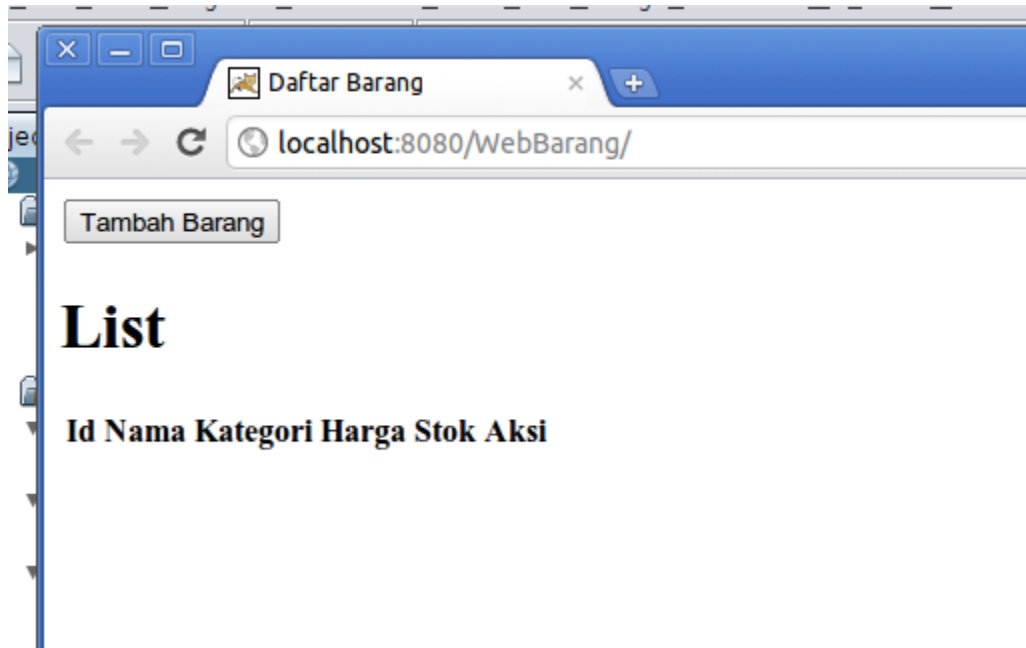
## Yuk Coba

Sekarang tinggal kita coba web nya yang sudah dibuat tadi.

## Ngejalanin Web

Untuk ngejalanin webnya, gambar di projectnya tinggal klik nanan, trus pilih menu **Run**, tinggal tunggu aja web nya jalan di browser, kayak gini hasilnya :

Daftar barang masih kosong :



Nambah barang :

Calibri (Body) 11

Tambah Barang

localhost:8080/WebBarang/index.stripbandunk

## Tambah Barang

Id:

Nama:

Kategori:

Harga:

Stok:

Validasi gagal nambah barang :

Tambah Barang

localhost:8080/WebBarang/create.stripbandunk

## Tambah Barang

Id:

Nama:

Kategori:

Harga:

Stok:

- Kode tidak boleh kosong
- Nama tidak boleh kosong



Daftar Barang :

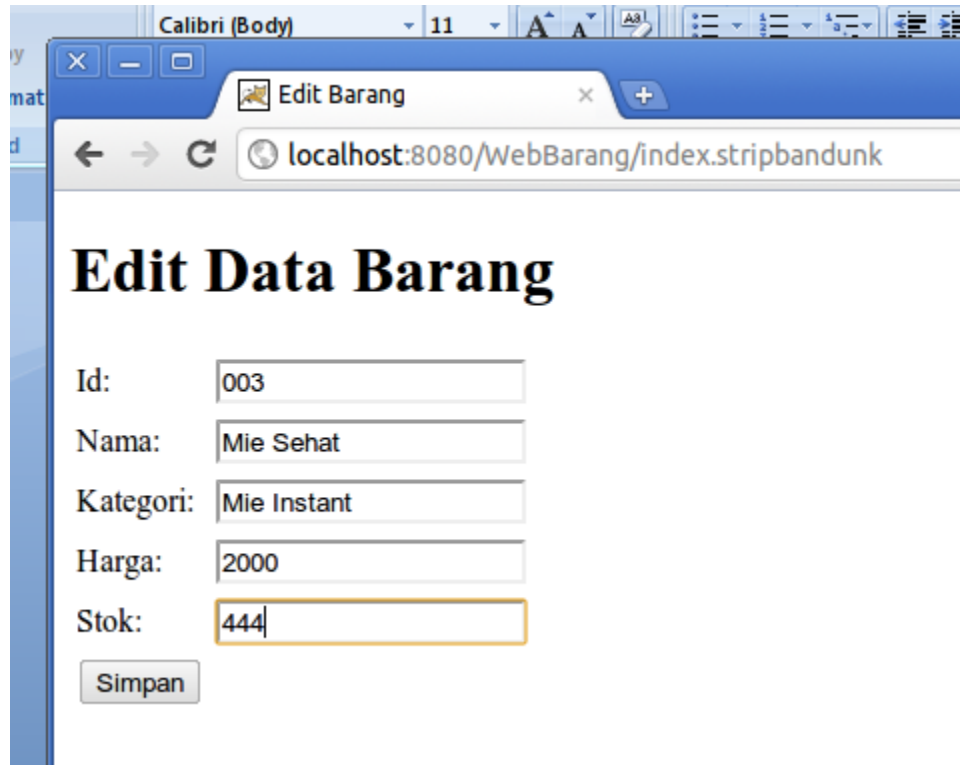


Tambah Barang

## List

<b>ID</b>	<b>Nama</b>	<b>Kategori</b>	<b>Harga</b>	<b>Stok</b>	<b>Aksi</b>
001	Kopi Ku	Kopi	1000	200	<input type="button" value="Edit"/> <input type="button" value="Hapus"/>
002	Jinggo	Rokok	8000	100	<input type="button" value="Edit"/> <input type="button" value="Hapus"/>
003	Mie Sehat	Mie Instant	2000	100	<input type="button" value="Edit"/> <input type="button" value="Hapus"/>

Edit Barang :



Calibri (Body) 11

Edit Barang

localhost:8080/WebBarang/index.stripbandunk

## Edit Data Barang

Id:

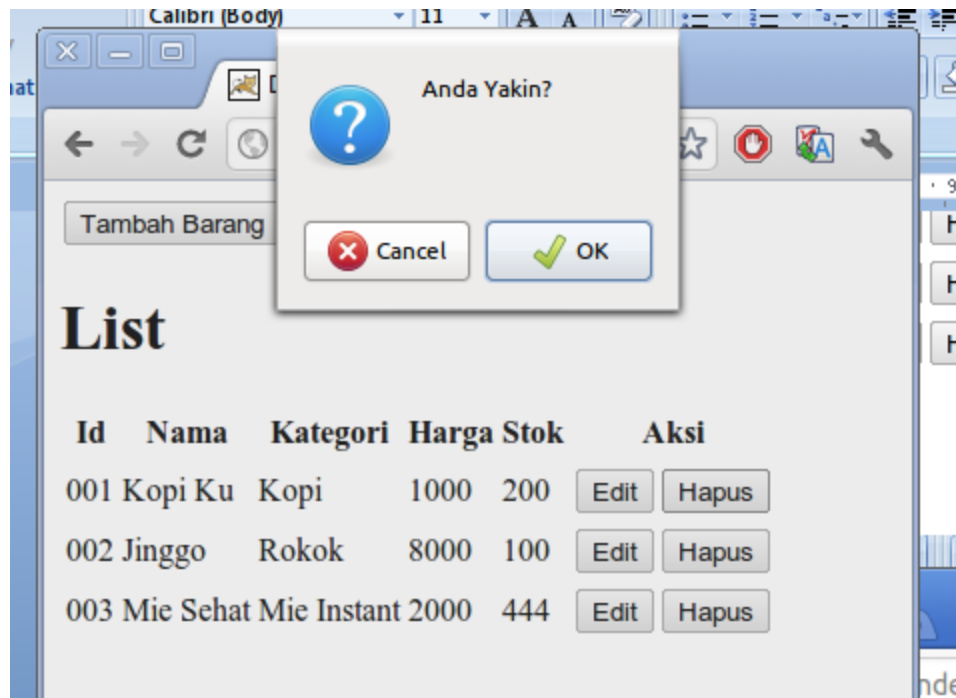
Nama:

Kategori:

Harga:

Stok:

Hapus Barang :



## Misi Selanjutnya

Sudah selesaikah?

Belum lah, masih banyak yang harus diselesaikan, misi selanjutnya adalah :

- Ubah tampilan daftar barang supaya lebih menarik
- Ubah juga tampilan form tambah dan form edit
- Pastikan kode barang gak bisa diedit oleh pengguna, biasanya memang kode itu gak bisa diubah lagi.
- Tambahin lagi atribut-atribut nya, misal harga beli, harga jual, minimum stok, dan lain-lain

Selamat menjalankan misi!!!

## Tentang Saya

Dari pada saya cape-cape ngenalin penulis, mending ngenalin saya sendiri :D

Saya **Eko Kurniawan Khannedy**, penulis buku ini. Buku ini adalah buku gratis, kalo anda harus bayar buat dapetin buku ini, berarti anda tertipu :D

Pembaca semua bisa menghubungi saya di :

- Email : [echo.khannedy@gmail.com](mailto:echo.khannedy@gmail.com)
- Twitter : <http://twitter.com/khannedy>
- Facebook : <http://facebook.com/EkoKurniawanKhannedy>
- Wordpress : <http://eecchhoo.wordpress.com/>
- Video Tutorial : <http://stripbandunk.com/>

Dan jangan lupa ikuti informasi Java terkini di :

- Twitter : <http://twitter.com/PakarJava>
- Facebook : <http://facebook.com/TanyaJava>

Terimakasih udah nyempetin baca buku saya :D