

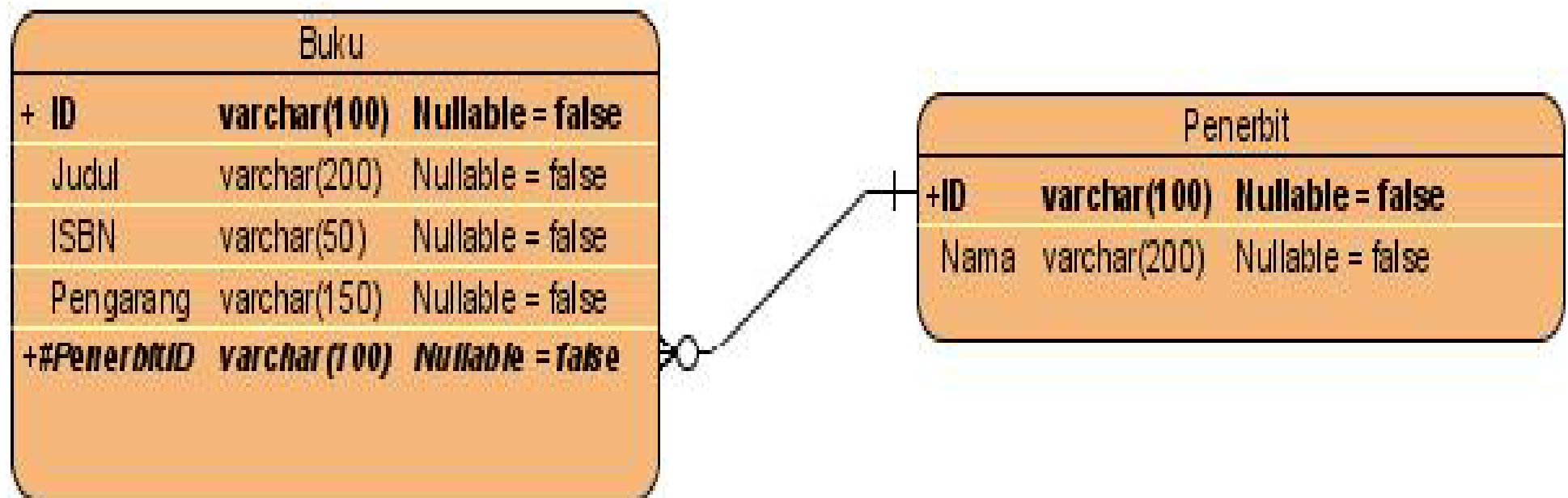
# TUTORIAL MEMBUAT APLIKASI KATALOG BUKU MENGUNAKAN SPRING DAN STRUTS

eriq.adams@gmail.com

# Intro

- Sebelum mengikuti tutorial ini, terlebih dahulu anda harus menginstall Oracle XE dan Netbeans 6.5 yang ada fitur JEE-nya dan Apache Tomcat 6.0
- Template Project ada di file **SpringStrutsTemplate.zip**
- Hasil tutorial ada di file **KatalogBuku.zip**

# Buat E-R Diagram



**Perhatikan relasi antara tabel Buku dan Penerbit !!**

# Buat Database User di Oracle XE

- ❑ Login ke Oracle HTML DB sebagai System
- ❑ Masuk ke menu Home>Administration>Manage Database Users
- ❑ Klik tombol Create>

Create Database User

Cancel Create

\* Username

\* Password

\* Confirm Password

Expire Password

Account Status

Default Tablespace **USERS**

Temporary Tablespace **TEMP**

CONNECT

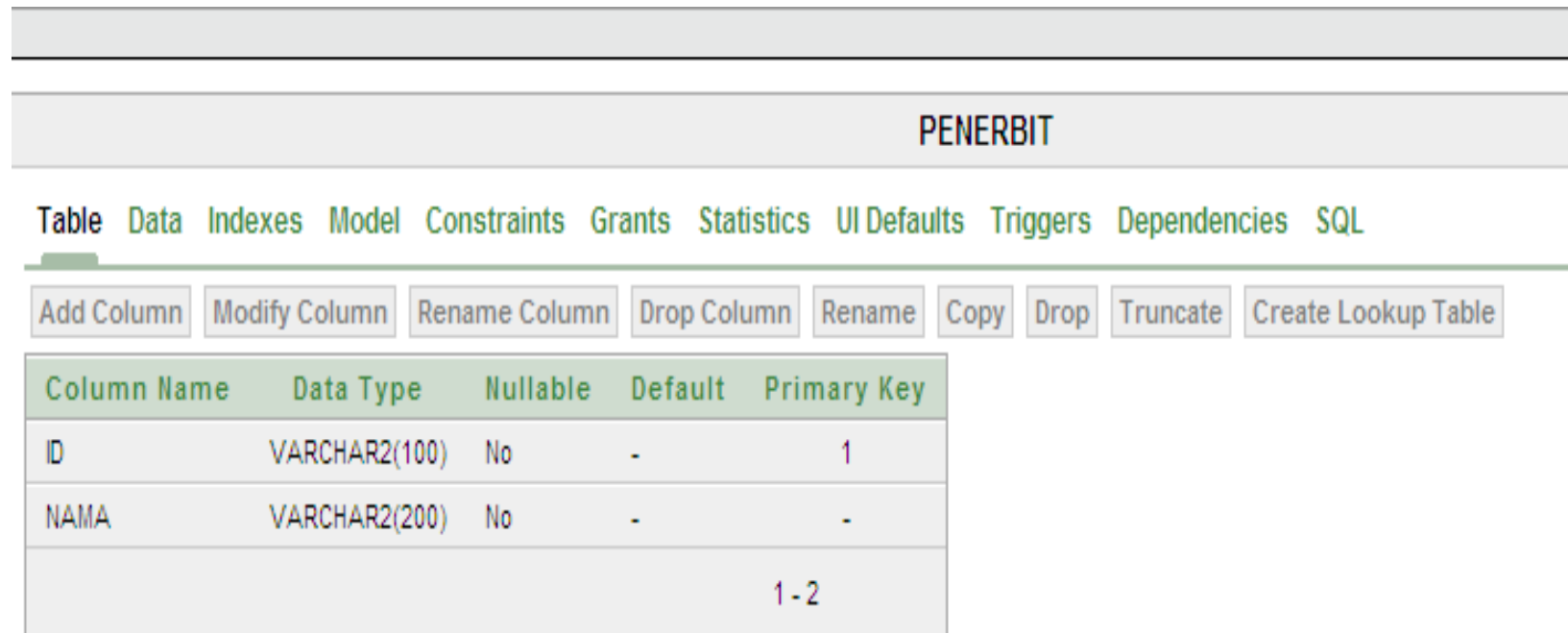
Roles  RESOURCE

DBA

Quota

# Buat Tabel Buku dan Penerbit

- ❑ Logout dari Oracle HTML DB
- ❑ Login ke Oracle HTML DB sebagai **katalogbuku**
- ❑ Pilih menu Object browser -> Create Table



The screenshot shows the Oracle HTML DB Object Browser interface for a table named 'PENERBIT'. The table structure is displayed with columns: Column Name, Data Type, Nullable, Default, and Primary Key. The columns are ID (VARCHAR2(100), No, -, 1) and NAMA (VARCHAR2(200), No, -, -). A '1-2' label is visible at the bottom of the table structure.

Column Name	Data Type	Nullable	Default	Primary Key
ID	VARCHAR2(100)	No	-	1
NAMA	VARCHAR2(200)	No	-	-

# Buat Tabel Buku dan Penerbit

**BUKU**

Table Data Indexes Model Constraints Grants Statistics UI Defaults Triggers Dependencies SQL

Add Column Modify Column Rename Column Drop Column Rename Copy Drop Truncate Create Lookup Table

Column Name	Data Type	Nullable	Default	Primary Key
ID	VARCHAR2(100)	No	-	1
ISBN	VARCHAR2(50)	No	-	-
JUDUL	VARCHAR2(200)	No	-	-
PENGARANG	VARCHAR2(150)	No	-	-
PENERBIT_ID	VARCHAR2(100)	No	-	-
				1 - 5

## Tambahkan Constraint Primary Key

Add Constraint Cancel Next >

Schema: KATALOGBUKU  
Table: BUKU

\* Constraint Name   
 Preserve Case

Constraint Type

\* Primary Key Column 1   
Primary Key Column 2   
Primary Key Column 3

# Buat Tabel Buku dan Penerbit

- Tambahkan constraint foreign key

The screenshot shows the 'Add Constraint' dialog box with the following fields and values:

- Schema: KATALOGBUKU
- Table: BUKU
- \* Constraint Name: BUKU\_PENERBIT
- Preserve Case
- Constraint Type: Foreign Key
- On Delete Cascade
- \* Foreign Key Column(s): PENERBIT\_ID
- \* Reference Table Name: PENERBIT
- \* Reference Table Column List: (empty)

- Nah, selamat anda mempunyai 2 tabel yg berrelasi

# Buat Koneksi Oracle XE di Netbeans 6.5

- Pilih TAB **Services** (sebelah kanan TAB **Projects**)
- Pilih **Databases**, klik kanan **new Database Connection**

New Database Connection

Basic setting Advanced

Data Input Mode:  Field Entry  Direct URL Entry

Name: Orade Thin (with Service ID (SID))

Host: localhost

Port: 1521

Service ID (SID): xe

User Name: katalogbuku

Password: ●●●●●●●●

Remember password  
(see help for information on security risks)

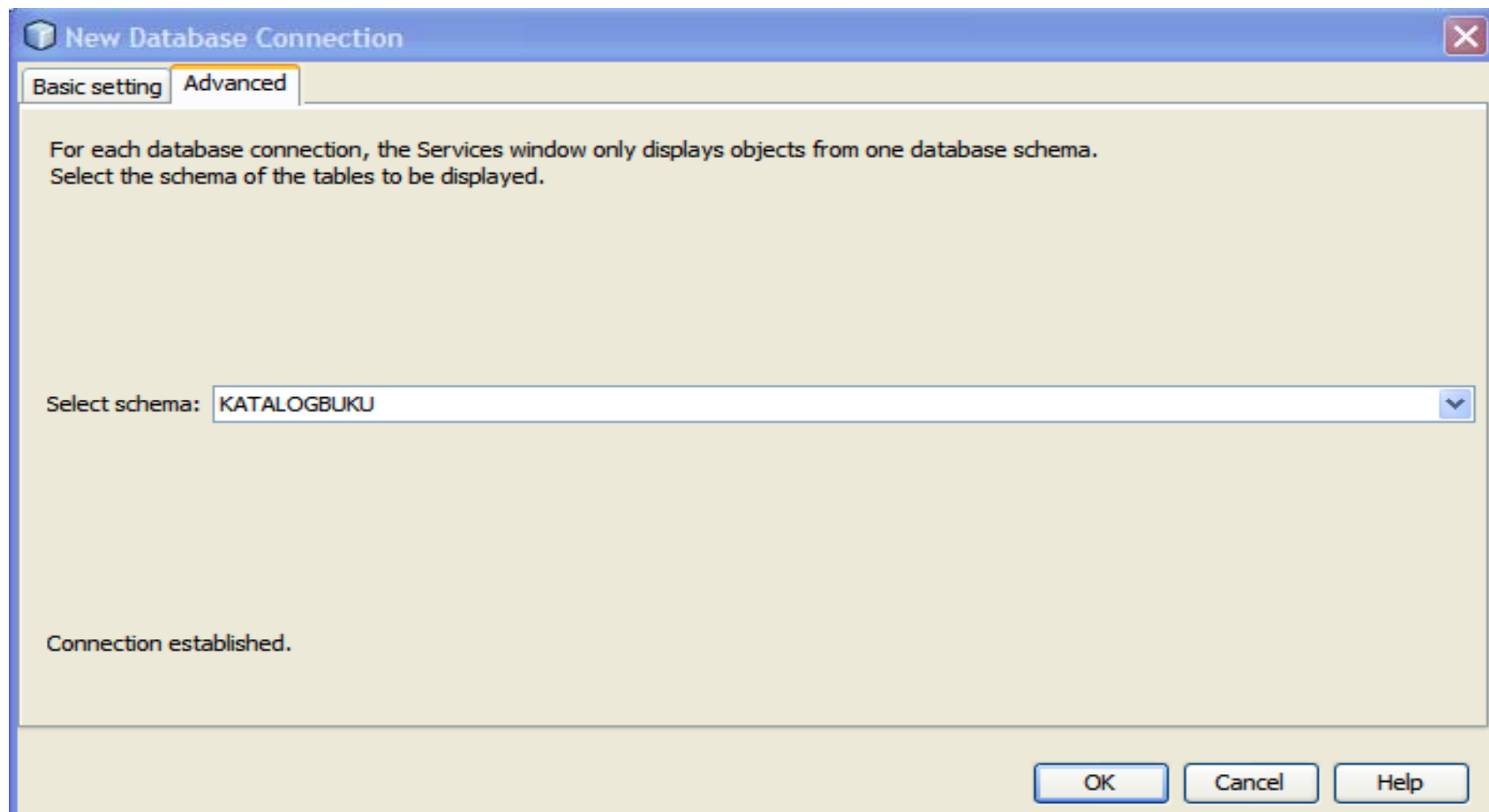
Additional Props:

Show JDBC URL jdbc:orade:thin:@localhost:1521:xe

OK Cancel Help

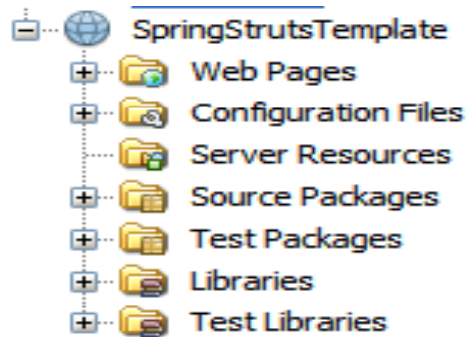
# Buat Koneksi Oracle XE di Netbeans 6.5

Pilih Schema **KATALOGBUKU** dan klik OK

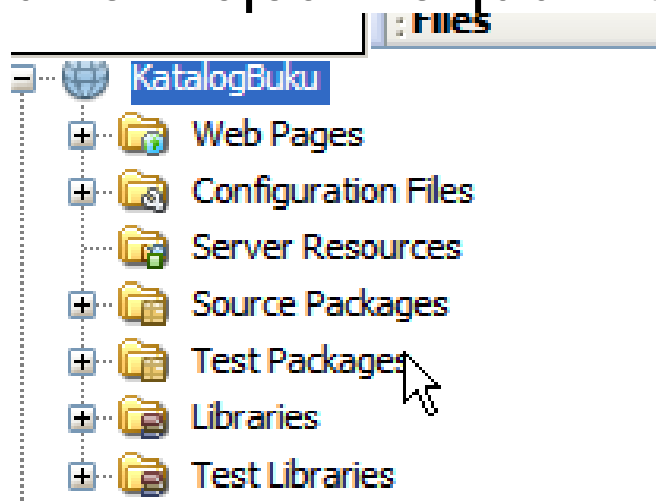


# Open Template Project di Netbeans 6.5

## □ Buka Project **SpringStrutsTemplate**

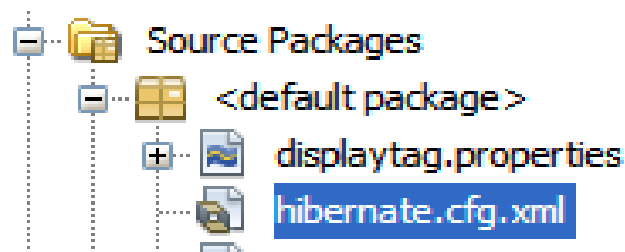


## Rename Project menjadi **KatalogBuku**



# Setting file hibernate.cfg.xml

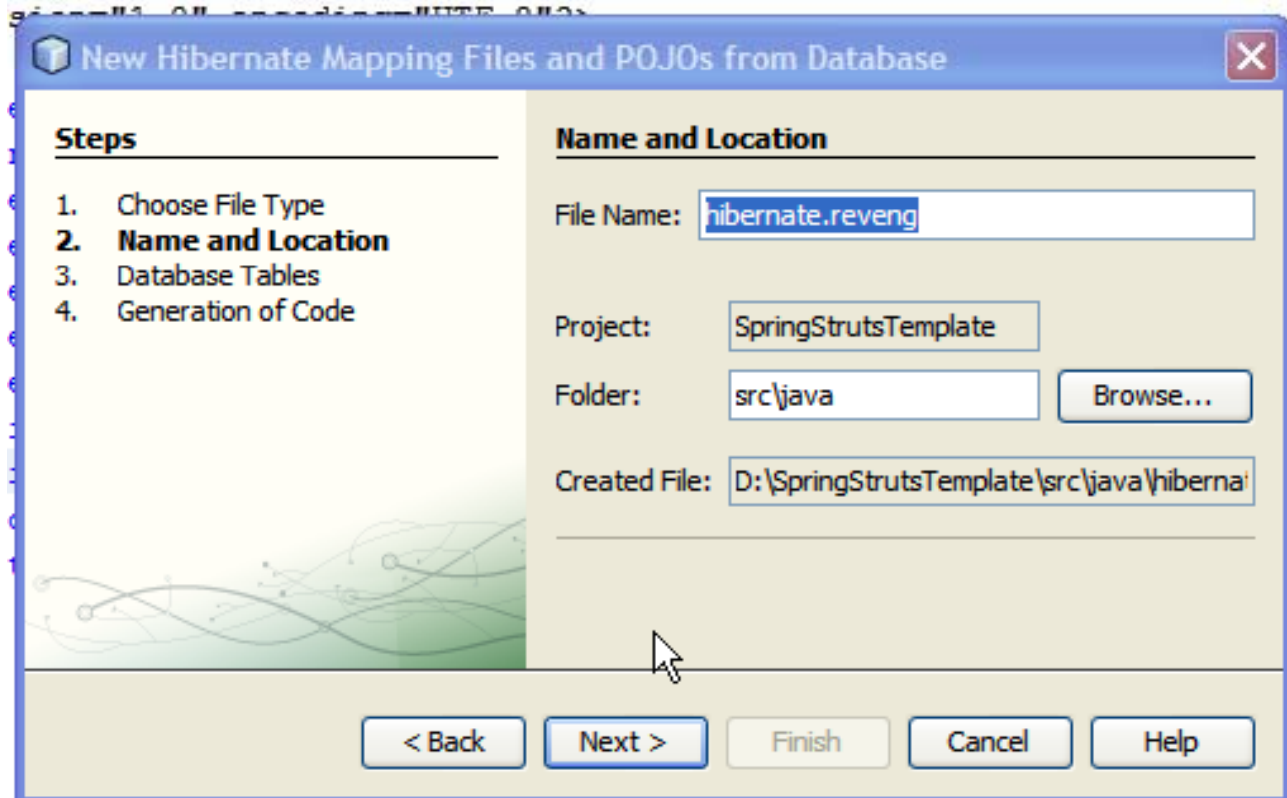
- Ubah URL , username dan password utk koneksi.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "htt
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>
    <property name="hibernate.connection.driver_class">oracle.jdbc.OracleDriver</property>
    <property name="hibernate.connection.url">jdbc:oracle:thin:@//localhost:1521/xe</property>
    <property name="hibernate.connection.username">katalogbuku</property>
    <property name="hibernate.connection.password">katalogbuku</property>
  </session-factory>
</hibernate-configuration>
```

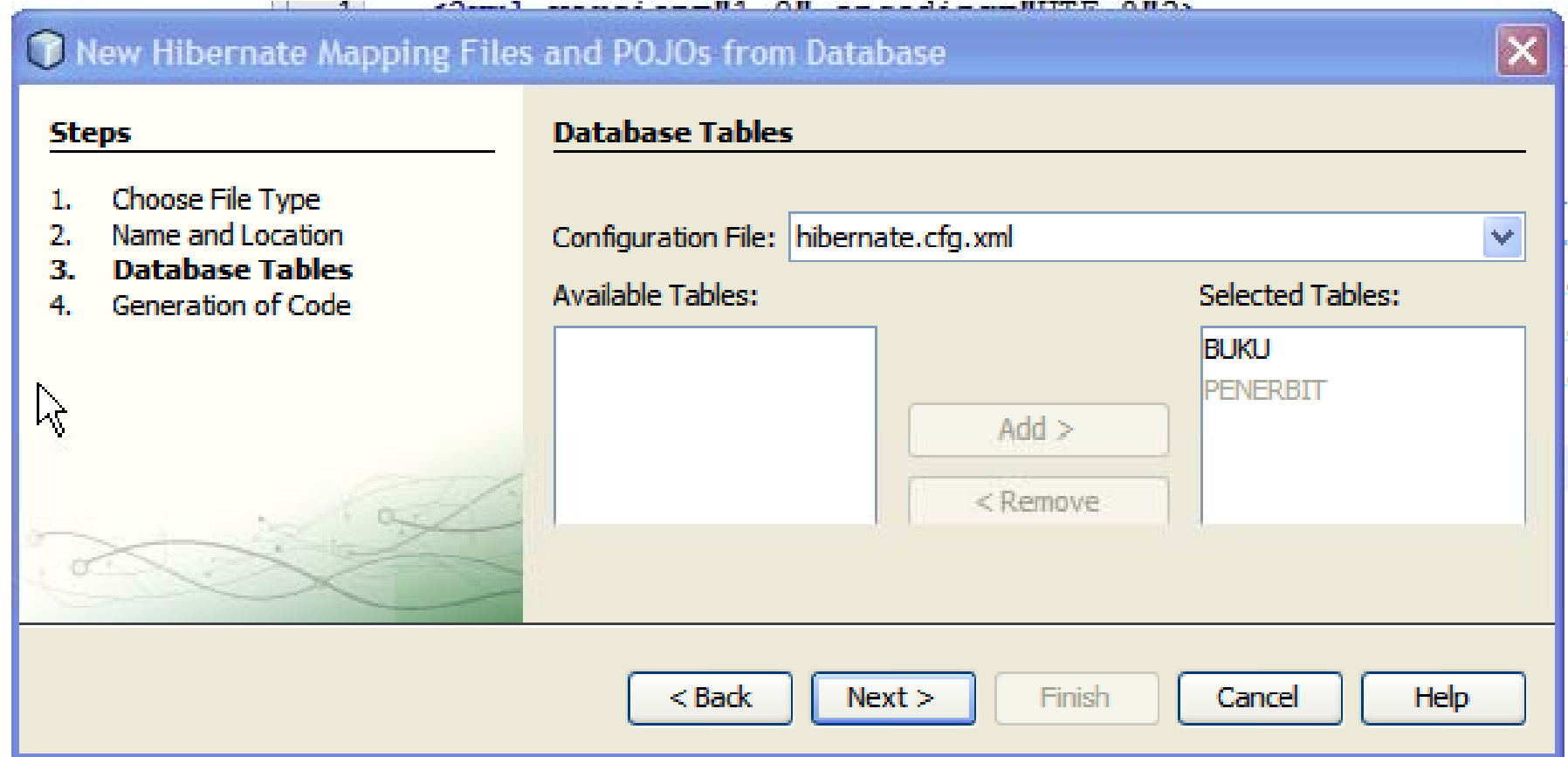
# Generate POJO dan Hibernate Mapping

- Menggenerate POJO (Plain Old Java Object) , klas yg mencerminkan Tabel-tabel dan file aturan mapping hibernate dari Oracle XE. Klik kanan → **New Hibernate Mapping Files and POJOs from Database**



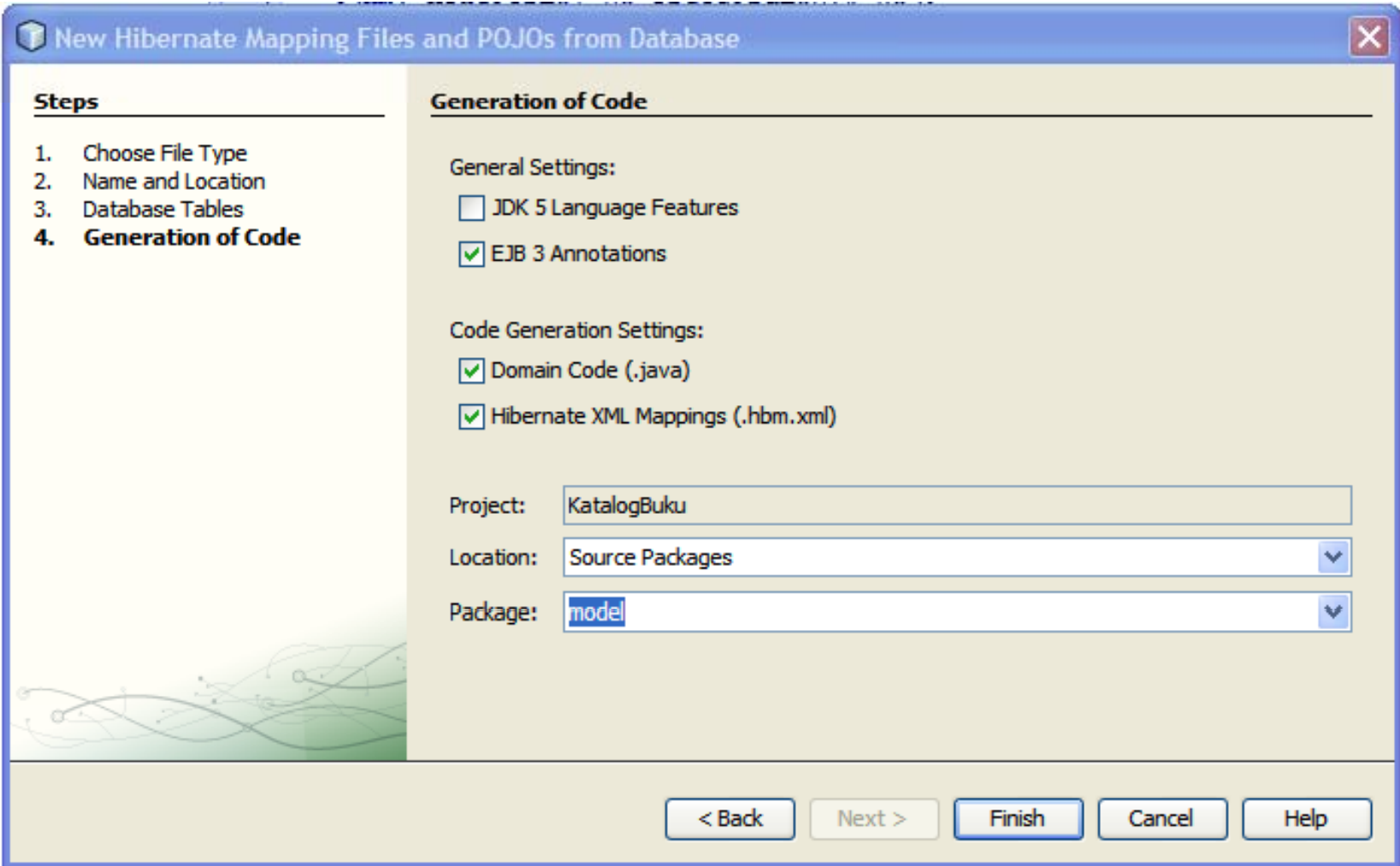
# Generate POJO dan Hibernate Mapping

- Klik tombol **Add All >**



# Generate POJO dan Hibernate Mapping

## Pilih Package : model



**New Hibernate Mapping Files and POJOs from Database**

**Steps**

1. Choose File Type
2. Name and Location
3. Database Tables
4. **Generation of Code**

**Generation of Code**

General Settings:

- JDK 5 Language Features
- EJB 3 Annotations

Code Generation Settings:

- Domain Code (.java)
- Hibernate XML Mappings (.hbm.xml)

Project: KatalogBuku

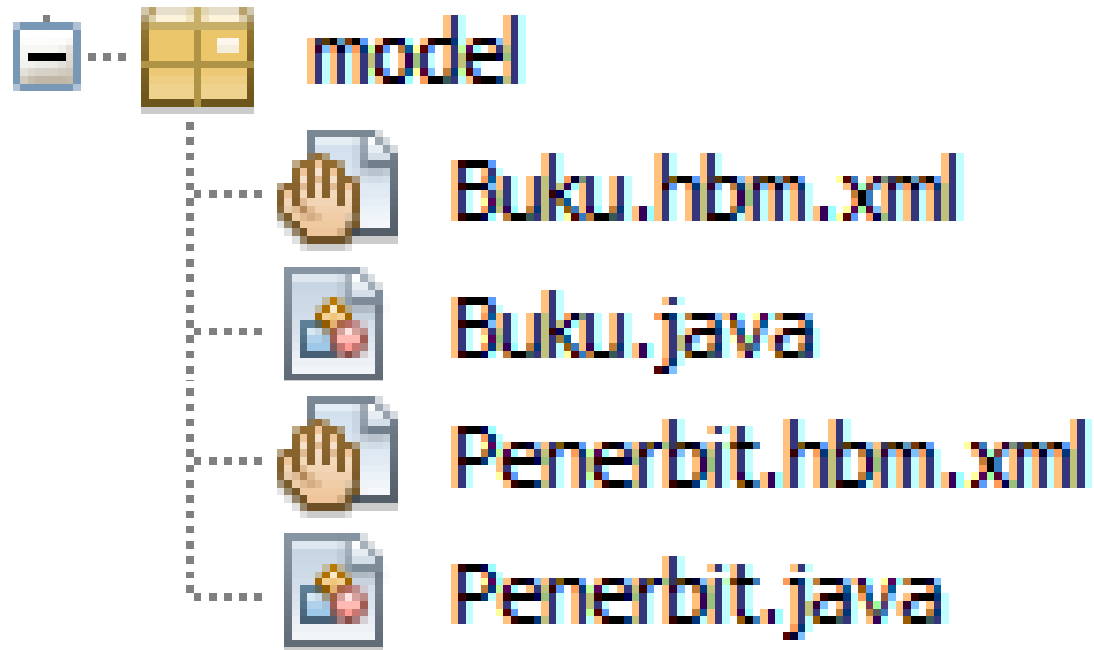
Location: Source Packages

Package: model

< Back   Next >   **Finish**   Cancel   Help

# Generate POJO dan Hibernate Mapping

- Nah, selamat anda sudah memiliki file \*.hbm.xml dan POJOs - nya



# Generate POJO dan Hibernate Mapping

- Misal kita ingin melakukan penambahan kolom ID pada tabel Buku dan Penerbit secara otomatis maka kita perlu mengubah tipe generationnya menjadi **uuid** pada file **Buku.hbm.xml** dan **Penerbit.hbm.xml** .

```
<hibernate-mapping>
```

```
  <class name="model.Buku" schema="KATALOGBUKU" table="BUKU">
```

```
    <id name="id" type="string">
```

```
      <column length="100" name="ID"/>
```

```
      <generator class="uuid"/>
```

```
    </id>
```

# Generate POJO dan Hibernate Mapping

- Pada relasi klas dan tabel **many-to-one** perlu diubah atribut **lazy** menjadi **false**. (Mengapa? Silakan baca buku pembuat hibernate **Hibernate In Action, Christian Bauer dan Gavin King**) .

```
<many-to-one class="model.Penerbit" fetch="select" name="penerbit" lazy="false">  
  <column length="100" name="PENERBIT_ID" not-null="true"/>  
</many-to-one>
```

# Buat DAO (Data Access Object)

- DAO berguna untuk memudahkan kita dalam melakukan operasi database CRUD (Pelajari Design Pattern **Data Access Object**). Di dalam project ini sudah terdapat klas **dao.DAOHibernate** yg berisi method2 umum utk melakukan operasi database. Hal yg kita lakukan selanjutnya adalah membuat DAO (model.**BukuDAO** dan model.**PenerbitDAO**) utk masing2 POJOs (model.Buku dan model.Penerbit) dengan cara meng-**extends** dari klas **model.DAOHibernate**.

# Buat DAO (Data Access Object)

```
public class BukuDAO extends DAOHibernate<Buku, String>{  
}
```

- (1) Merupakan Tipe Data POJO , Misal untuk membuat BukuDAO kita menggunakan Tipe Data **model.Buku**
- (2) Merupakan Tipe Data Id POJO atau tipe data Primary Key dari Tabel , Misal untuk tabel Buku Primary Key-nya ada pada kolom **ID** dan representasi pada klasnya ada pada Atribut **id** yang bertipe **java.lang.String**.

# Buat Controller dg Struts

- Nah, tahapan berikut adalah membuat controller menggunakan struts atau **StrutsAction** yg berguna untuk memproses HTTP Request. Kita akan membuat 2 buah controller (**controller.BukuAction** dan **controller.PenerbitAction**) yang bertipe **DispatchAction** (bisa dilihat di Buku **Beginning Apache Struts from Novice to Professional** ttg **DispatchAction**) untuk klas **model.Buku** dan **model.Penerbit**. (penjelasan listing program ada pada kode program **BukuAction.java** dan **PenerbitAction.java**) .

# Buat aturan mapping struts di struts-config.xml

- Buka file struts-config.xml dan tambahkan aturan mappingnya.



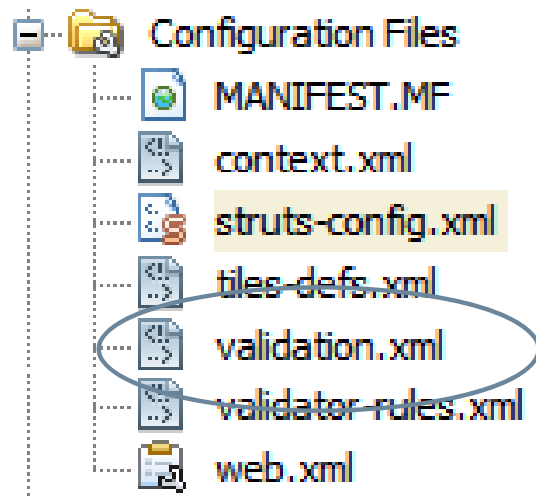
## Buat aturan mapping struts di struts-config.xml

- Buat definisi **StrutsForm** dulu sebelum membuat definisi **StrutsAction**. Ketika kita membuat form penambahan atau edit buku maka kita perlu perlu form, oleh karena itu kita perlu membuat **StrutsForm**-nya . Berikut strutsform dari **BukuAction**. (penjelasan detail ada di file **struts-config.xml**).

```
<form-bean name="bukuForm"  
    type="org.apache.struts.validator.DynaValidatorForm">  
    <form-property name="buku" type="model.Buku" />  
    <form-property name="penerbitId" type="java.lang.String" />  
</form-bean>
```

## Buat aturan mapping struts di struts-config.xml

- Selanjutnya kita dapat membuat **validasi errors bukuForm** dengan mendefinisikannya pada file **validation.xml**. Untuk mendefinisikan validasi error sebuah struts form kita dapat merujuk pada constraint tabel tsb(tabel **BUKU**) seperti **NOT NULL, MAX LENGTH**.



# Buat aturan mapping struts di struts-config.xml

```
<form name="bukuForm">
  <field property="buku.judul"
    depends="required,minlength,maxlength,mask">
    <arg0 name="Judul Buku" />
    <arg1 key="{var:maxlength}" name="maxlength"
      resource="false" />
    <arg1 key="{var:minlength}" name="minlength"
      resource="false" />
    <var>
      <var-name>minlength</var-name>
      <var-value>3</var-value>
    </var>
    <var>
```

Penjelasan lebih detail dapat dilihat di file **validation.xml**

# Buat aturan mapping struts di struts-config.xml

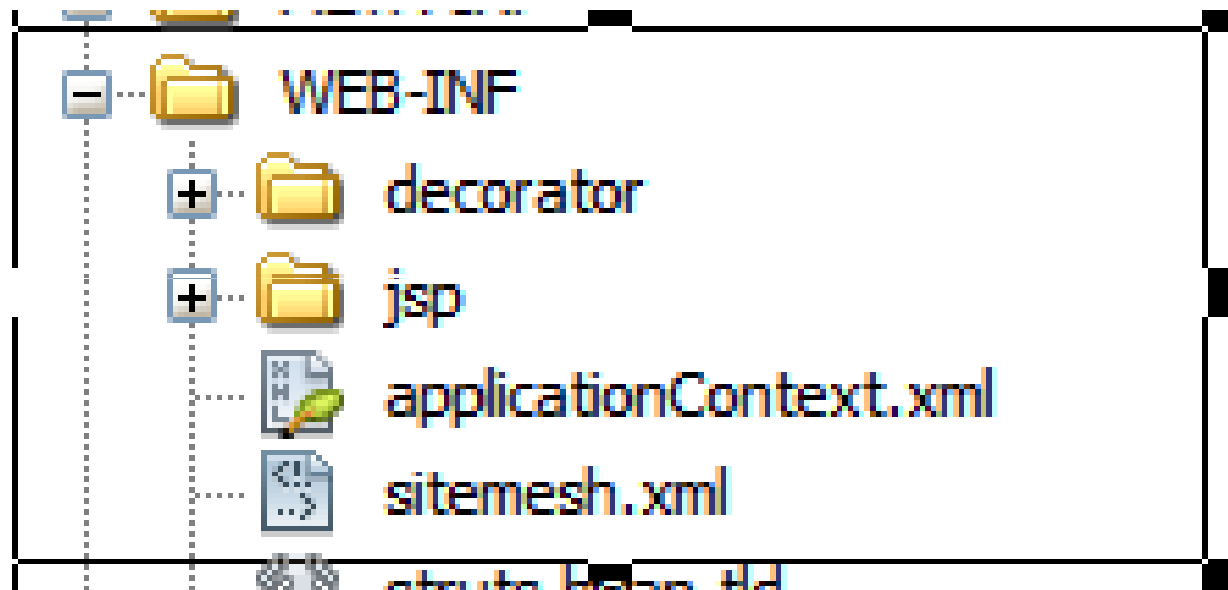
## □ Buat definisi **StrutsAction**

```
<action-mappings>
  <!-- Setelah men-setting struts action disini cobalah untuk mensettingnya lagi -->
  <!-- di file applicationContext.xml untuk menggunakan fitur Spring IoC Dependency Inject
  <action path="/penerbit" type="org.springframework.web.struts.DelegatingActionProxy"
  name="penerbitForm" scope="request" parameter="action" validate="false">
    <forward name="formPenerbit" path="/WEB-INF/jsp/penerbitForm.jsp" />
    <forward name="daftarPenerbit" path="/WEB-INF/jsp/daftarPenerbit.jsp" />
  </action>

  <action path="/buku" type="org.springframework.web.struts.DelegatingActionProxy"
  name="bukuForm" scope="request" parameter="action" validate="false">
    <forward name="formBuku" path="/WEB-INF/jsp/bukuForm.jsp" />
    <forward name="daftarBuku" path="/WEB-INF/jsp/daftarBuku.jsp" />
  </action>
</action-mappings>
```

## Setting DataSource di file applicationContext.xml

- Karena kita melakukan koneksi melalui spring, maka kita perlu mengubah setting data source dan koneksi-nya pada file applicationContext.xml



# Setting DataSource di file applicationContext.xml

```
<bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName">
    <value>oracle.jdbc.OracleDriver</value>
  </property>
  <property name="url">
    <value>jdbc:oracle:thin:@//localhost:1521/xe</value>
  </property>
  <property name="username">
    <value>katalogbuku</value>
  </property>
  <!-- Make sure <value> tags are on same line - if they're not,
        authentication will fail -->
  <property name="password">
    <value>katalogbuku</value>
  </property>
</bean>
```

# Setting DataSource di file applicationContext.xml

- Masukkan \*.hbm.xml pada definisi bean **SessionFactory**

```
<!-- Hibernate SessionFactory -->
<bean id="sessionFactory"
      class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
  <property name="dataSource">
    <ref local="dataSource" />
  </property>
  <property name="mappingResources">
    <list>
      <value>model/Buku.hbm.xml</value>
      <value>model/Penerbit.hbm.xml</value>
    </list>
  </property>
</bean>
```

# Saatnya Dependency Injection

- Selanjutnya kita melakukan **Dependency Injection** melalui Spring IoC Container (pelajari Inversion of Control design pattern). Jika kita lihat dalam **controller.BukuAction** memiliki atribut/dependency **bukuDao : BukuDAO** dan **penerbitDao: PenerbitDAO**. Kita dapat mengisi object / instance **penerbitDao** dan **bukuDao** dengan melakukan dependency injection. Dengan cara mendefinisikan ketergantungan / dependency tsb pada file **applicationContext.xml**.

# Saatnya Dependency Injection

Definisikan dulu dependency utk bukuDao dan penerbitDao

```
<bean id="bukuDao" class="dao.BukuDAO">
  <property name="sessionFactory">
    <ref bean="sessionFactory" />
  </property>
</bean>
```

```
<bean id="penerbitDao" class="dao.PenerbitDAO">
  <property name="sessionFactory">
    <ref bean="sessionFactory" />
  </property>
</bean>
```

# Saatnya Dependency Injection

Definisikan controller controller.BukuAction dan controller.PenerbitAction beserta dependencies-nya

```
<bean name="/penerbit" class="controller.PenerbitAction">
  <property name="bukuDao">
    <ref bean="bukuDao" />
  </property>
  <property name="penerbitDao">
    <ref bean="penerbitDao" />
  </property>
</bean>
```

```
<bean name="/buku" class="controller.BukuAction">
  <property name="bukuDao">
    <ref bean="bukuDao" />
  </property>
  <property name="penerbitDao">
    <ref bean="penerbitDao" />
  </property>
</bean>
```

# Membuat View

- Untuk membuat View kita harus membuat halaman JSP (Java Server Pages) -nya. Untuk memudahkan dalam desain kita menggunakan sitemesh templating framework (yang menerapkan design pattern **Decorator**, pelajari **Decorator Pattern**). Definisi sitemesh ada di file `WEB-INF/sitemesh.xml` dan `WEB-INF/decorator/decorators.xml`. Dengan menggunakan sitemesh kita hanya perlu mendefinisikan template CSS utama saja yang akan kita manfaatkan untuk seluruh halaman JSP. Detail utk konfigurasi dapat dilihat komentar di file **sitemesh.xml** dan **decorators.xml** (jika anda terbiasa dg PHP mungkin sitemesh ini hampir sama dengan Smarty framework).

# Membuat View

- Selanjutnya kita membuat file-file JSP (Ada di folder WEB-INF/jsp/).
- **penerbitForm.jsp**, form utk edit / tambah data penerbit
- **daftarPenerbit.jsp** form utk menampilkan daftar penerbit
- **bukuForm.jsp**, form utk edit / tambah data buku
- **daftarBuku.jsp** form utk menampilkan daftar buku
- **index.jsp** form halaman awal atau index

# Membuat View

- Halaman / template CSS utama ada di WEB-INF/decorator/**template.jsp** dan Style CSS-nya ada di **style.css**. Silakan anda modifikasi.

# Aplikasi KatalogBuku

## Katalog Buku

menggunakan spring dan struts

[Home](#) [Daftar Penerbit](#) [Tambah Penerbit](#) [Daftar Buku](#) [Tambah Buku](#)

### [Aplikasi Katalog Buku](#)

Selamat datang di aplikasi Katalog Buku.

Aplikasi ini adalah contoh aplikasi J2EE menggunakan spring dan struts framework.