

**MAKALAH  
PEMROGRAMAN WEB (JAVA)  
JAVA PERSISTENCE API ( JPA )**



Disusun oleh :

Dodik Murdiono	07.41.2362
Agung Susilo	07.41.2330
Khastholani	07.41.2395
Desyka Prihantara	07.41.2357
PROGRAM	: DIPLOMA-III
JURUSAN	: D3TKJ – A

**STIMIK AMIKOM YOGYAKARTA  
JURUSAN TEKNIK INFORMATIKA  
YOGYAKARTA  
2008**

# JPA (Java Persistence API)

## I . Konsep

Menurut wikipedia, Java persistence API, disebut juga JPA, merupakan bahasa pemrograman java framework yang memungkinkan developer untuk mengelola relasi data di platform Java Standard Edition (j2SE) dan Java Platform Enterprise Edition(j2EE). JPA berasal dari bagian JSR 220 Expert Group.

Persistence terdiri dari tiga bagian:

- \* API berada dalam package javax.persistence
- \* Persistence query language
- \* Objek / relasional metadata

Dalam Dokumentasi **J2SE** (Desktop application), diterangkan bahwa spesifikasi JSR 220 mendefinisikan Enterprise JavaBeans 3,0 . Salah satu tujuan utama adalah untuk menyederhanakan pembuatan, manajemen, dan menyimpan entitas. Dalam mencapai tujuan tersebut, Sun Microsystems dan komunitas pengembang aplikasi membuat program antarmuka (API) yang memungkinkan anda menggunakan "plain lama Java objek" atau POJOs sebagai persistable entities. Java persistence API memfasilitasi penggunaan POJOs sebagai entitas bean dan secara signifikan mengurangi kerumitan dalam membuat deskriptor saat deployment dan mendapat bantuan bean tambahan . Selain itu, bahkan dapat menggunakan API dalam aplikasi desktop.

Anda akan menemukan banyak alasan untuk menggunakan persistence API yang baru, disini hanya beberapa:

- Anda tidak perlu membuat complex data access objects (DAO).

- API membantu mengelola transaksi.
- Penulisan standar-kode yang berinteraksi dengan database relasional, lepas dari query database yang digunakan.
- Anda dapat menghindari SQL penulisan bahasa query yang menggunakan nama kelas dan properti.
- Anda dapat menggunakan dan mengelola POJOs.

Anda juga dapat menggunakan java persistence API untuk aplikasi desktop.

Persistence di Java masuk dalam Java Persistence Api atau JPA. JPA mempunyai kemampuan yang cukup hebat yaitu project benar-benar tidak berurusan dengan database. tablenya akan dibuatkan sendiri oleh JPA.

Beberapa hal yang perlu diperhatikan :

1. Kita hanya perlu paham pendekatan Object Oriented
2. Table dalam database akan di generate otomatis berdasarkan entity class yang kita buat.
3. Ada beberapa library JPA yang bisa digunakan antara lain TopLink dan Hibernate.

Pengertian lain dari JPA adalah salah satu *Java framework* yang menggunakan pendekatan Object-relational mapping (ORM). Lahirnya ORM dilatarbelakangi oleh cara pandang OOP dalam dunia nyata untuk melihat sistem, termasuk ke dalam sistem basis data.

Cara kerja ORM ini adalah memetakan objek ke dalam database. Objek yang dimasukkan akan dibaca oleh ORM tersebut dan diubah ke dalam sintaks SQL. selanjutnya baru kemudian dijalankan di basis data relasional dan hasilnya kembali ke ORM. Berbagai macam teknologi yang menggunakan pendekatan ini seperti Entity Beans 2.x, Toplink, Hibernate, JDO, JDBC dengan DAO.

Dengan banyaknya pilihan teknologi, membuat pakar JAVA EE terinspirasi dengan membuat suatu framework yang dikenal dengan Java Persistence API (JPA).

JPA merupakan fitur baru yang ditambahkan mulai dari Java EE 5 dan merupakan sub-spesifikasi dari EJB 3. Selain untuk Java EE, JPA juga dapat dipergunakan untuk membangun aplikasi berbasis Java SE .

### **Java Persistence FAQ**

T: Mengapa java persistence API dimasukan dalam bagian platform Java EE 5?

J: Ada 2 alasan kami memasukkan Java persistence API untuk platform Java. Pertama, API baru ini akan menyederhanakan pengembangan Java EE dan Java SE menggunakan data persistence. Kedua, kami ingin seluruh masyarakat java di bernaung dalam satu standar persistence API.

T: Apakah keuntungan dari java persistence API?

J: Java persistence API didasarkan pada teknologi persistence terbaik seperti hibernate, TopLink, dan JDO. Pelanggan sekarang tidak lagi menghadapi ketidaksesuaian menentukan non-standar model persistence untuk Object-Relational Mapping. Selain itu, Java persistence API dapat digunakan baik di dalam lingkungan Jawa SE serta Java EE, yang memungkinkan pengembang untuk mengambil banyak keuntungan dari standar persistence API.

T: Apa yang menjadi fitur utama dari Java Persistence API?

J: java persistence API adalah POJO(Plain Old Java Object) persistence API untuk untuk Object-Relational Mapping. Yang penuh berisi spesifikasi Object-Relational Mapping mendukung penggunaan java metadata annotation dan XML descriptors dalam menentukan pemetaan antara object java dan objek database

relasional. Kayan dengan bahasa SQL, seperti query(yang merupakan ekstensi signifikan atas EJB QL) statis dan dinamis. Ia juga mendukung penggunaan pluggable persistence dari provider tertentu.

T: Mengapa Java Persistence API dikembangkan sebagai bagian dari JSR-220 (EJB 3,0)?

J: Java persistence API aslinya berasal dari bagian JSR 220 Expert Group untuk menyederhanakan EJB CMP entitas beans. segera menjadi jelas expert grup, namun, penyederhanaan EJB CMP tidak cukup, dan itulah mengapa yang diperlukan adalah persistence POJO sejalan dengan kerangka lain O / R pemetaan teknologi yang tersedia di industri. Setelah Konsep EJB 3,0 termasuk java persistence API ini dirilis, JSR-220 Expert Group menerima banyak permintaan dari masyarakat akan tersedia hanya di luar ruang lingkup EJB.

T: Mengapa tidak Anda memecah Java persistence API agar bisa masuk ke dalam JSR?

J: Kami percaya bahwa pendaftaran dalam konteks JSR-220 akan meminimalkan risiko dan memberikan hasil yang berkualitas jadi lebih cepat. Lebih lanjut, mengintegrasikan API ini dengan lancar dan konsisten dengan simplifications ke EJB 3,0 API. Hal itu nampaknya terbaik untuk memperluas proyek ini berlangsung, dan mendatangkan ahli tambahan ke dalam JSR-220 sebagai grup yang perkerjaannya diperluas.

T: Mengapa tidak Anda mengadopsi hibernate atau JDO sebagai persistence API?

J: Kami memilih untuk menggabungkan ide-ide terbaik dari banyak sumber ke dalam persistence API yang baru dan membuatnya lebih praktis, mudah untuk menggunakan API dalam memenuhi kebutuhan komunitas java EE dan java SE.

Java persistence API tidak berdasar pada satu persistence framework tetapi juga mencakup - dan meningkatkan - ide kontribusi dari berbagai framework populer, termasuk hibernate, TopLink, JDO, dan lain-lain.

T: Bagaimana jika saya ingin menggunakan java persistence API diluar dari platform Java EE?

J: spesifikasi, RI, dan TCK memastikan bahwa java persistence API bekerja dengan Java SE serta Java EE. Melewati TCK untuk bagian Java SE memungkinkan vendor lain kompatibel dengan Java persistence API tanpa Java EE sertifikasi.

T: Apa yang akan terjadi dengan data persistence API yang lain ketika sekarang Java persistence API tersedia?

J: Java persistence API sekarang adalah standarisasi untuk persistence dan objek / pemetaan relasional untuk platform Java EE. Versi API sebelumnya tentu saja tidak akan ditinggalkan.

T: Bagaimana Java persistence API berkembang sekarang JSR 220 yang telah selesai?

J: Kami berharap bahwa perjalanan dari Java persistence API untuk evolusi JSR masa depan. Ini berarti, versi dari Java persistence API tidak akan terikat untuk masa depan EJB JSRs. Kami berharap dapat terus menarik keahlian dari sumber yang beragam, bisa-bisa termasuk banyak dari anggota dari JSR 220 Expert Group yang membantu menentukan Jawa persistence API.

T: Apakah Java persistence API menjadi bagian dari Jawa SE?

J: Tidak ada rencana untuk memasukkan java persistence API kedalam java SE. Seperti perkembangan Java persistence API, kemungkinan bahwa masalah ini akan dipertimbangkan oleh ahli Java SE dimasa yang akan datang.

T: Apakah perubahan yang diperlukan untuk membuat aplikasi EJB CMP?

J: Adanya aplikasi EJB CMP tidak akan merubah keadaan, dan teknologi EJB CMP akan terus didukung. Tidak perlu aplikasi untuk bermigrasi ke java persistence API. Lebih lanjut, kemungkinan adanya penggabungan aplikasi yang sama untuk melanjutkan penggunaan EJB CMP entitas dengan bean dengan komponen baru EJB yang menggunakan Java persistence API.

### **Cara Kerja JPA.**

Seperti Hibernate, JPA menggunakan annotation untuk melakukan mapping objek-objek ke dalam basis data relasional. Objek ini sering disebut dengan entitas. Entitas JPA adalah Plain Old Java Object (POJO) yang tidak meng-extend berbagai kelas atau meng-implement berbagai interface.

Salah satu kelebihan dari sini, kita tidak perlu membuat XML descriptor untuk melakukan mapping. Jika kita lihat dokumentasi API, JPA dibuat dengan menggunakan sedikit kelas dan interface. Sebagian besar package javax.persistence adalah annotation.

### **Sample Aplikasi Persistence**

Contoh dibawah ini adalah salah satu penerapan JPA dalam model sederhana aplikasi Employee, untuk menyimpan data employee ke dalam database.

Aplikasi yang digunakan IDE Netbean , database mysql.

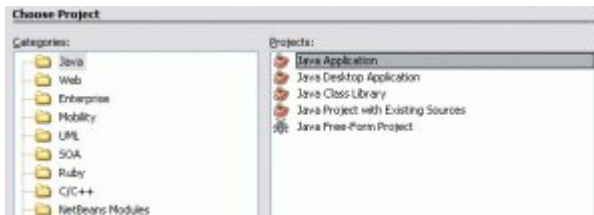
Login ke mysql dan buat sebuah database employee

view plain copy to clipboard print ?

```
create database employee;
```

## Pada Netbeans

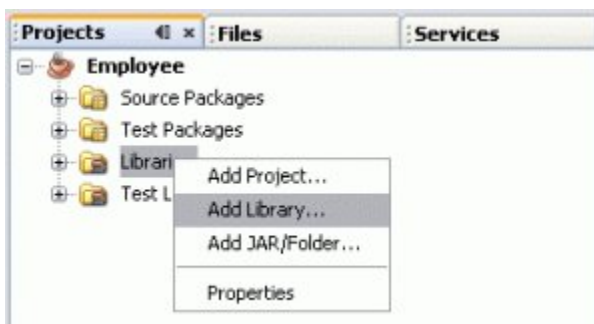
- Buat Project Baru (Ctrl+Shift+N)
- Pilih kategori Java dan Project Java Application, Next



- Project name: Employee, uncheck Create main class. Check Set as Main Project. Finish



- Tambahkan library MySQL pada project. Klik kanan pada Libraries -> Add Library

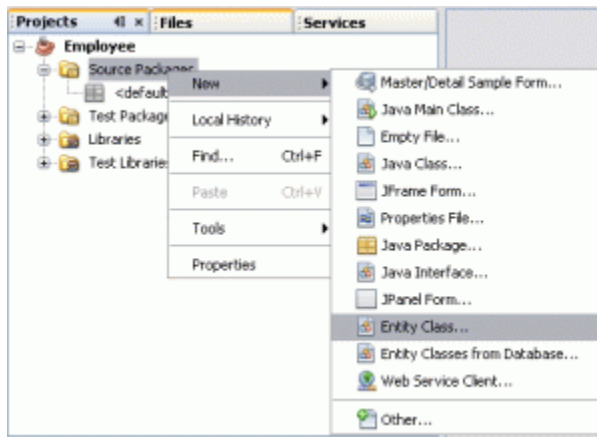


- Cari MySQL JDBC Driver, dan klik Add Library



Sekarang, kita buat Entity Class Employee

- Klik kanan pada Source Packages, pilih New -> Entity Class

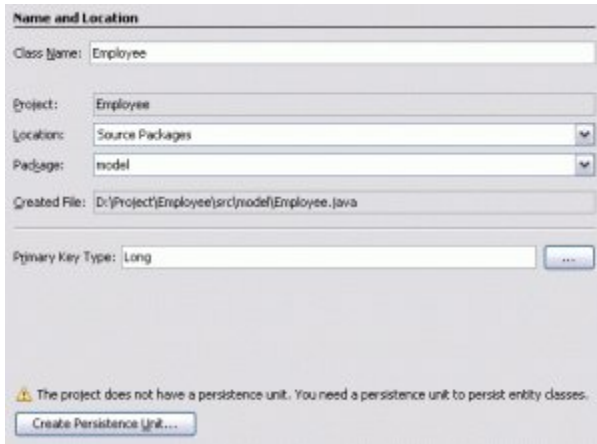


- Pada New Entity Class

Class Name: Employee

Package: model

Primary Key: Long



- Klik Create Persistence Unit.

Persistence Unit Name: EmployeePU

Persistence Library :Toplink

Database Connection: New Database Connection...

- Pada New Database Connection

Name: MySQL (Connector/J driver)

Database URL: jdbc:mysql://localhost:3306/employee

User Name: root

Password: \*\*\*\*\*

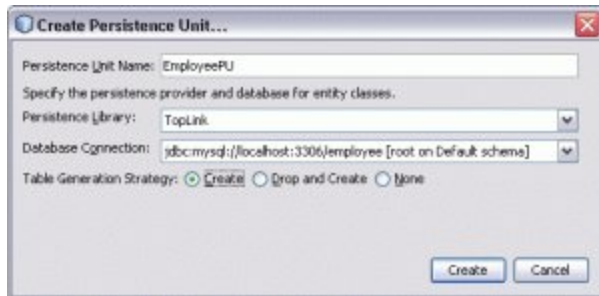
Centang Remember password

Klik Ok



Kembali pada Create Persistence Unit

- Pilih option Create pada Table Generation Strategy, hal ini agar tabel dibuat secara otomatis oleh JPA. Kemudian klik Create dan klik Finish.



Netbeans secara otomatis membuat package META-INF dan menggenerate file persistence.xml yang berisikan informasi seperti database yang akan digunakan, driver JDBC yang akan digunakan, juga provider dan berbagai macam properties lainnya.

### ***persistence.xml***

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
  <persistence-unit name="EmployeePU" transaction-type="RESOURCE_LOCAL">
    <provider>oracle.toplink.essentials.PersistenceProvider</provider>
    <class>model.Employee</class>
    <properties>
      <property name="toplink.jdbc.user" value="root"/>
      <property name="toplink.jdbc.password" value="password"/>
      <property name="toplink.jdbc.url" value="jdbc:mysql://localhost:3306/employee"/>
      <property name="toplink.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="toplink.ddl-generation" value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

Pada Employee.java tambahkan attribut seperti name, department, job\_title beserta setter dan getter method.

### ***Employee.java***

```
package model;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;
    private String department;
    private String jobTitle;

    public Long getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    public String getJobTitle() {
```

```

public String getJobTitle() {
    return jobTitle;
}

public void setJobTitle(String jobTitle) {
    this.jobTitle = jobTitle;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (id != null ? id.hashCode() : 0);
    return hash;
}

@Override
public boolean equals(Object object) {
    if (!(object instanceof Employee)) {
        return false;
    }
    Employee other = (Employee) object;
    if ((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "model.Employee[id=" + id + "]";
}
}

```

Kembali pada kode diatas, terdapat beberapa aturan yang harus diikuti, yaitu :

- Harus diidentifikasi sebagai entity dengan menggunakan `@javax.persistence.Entity` atau disingkat dengan `@Entity` annotation.
- Harus ada attribute yang bertugas sebagai identifier (Id) yang dianotasikan dengan `@javax.persistence.Id`

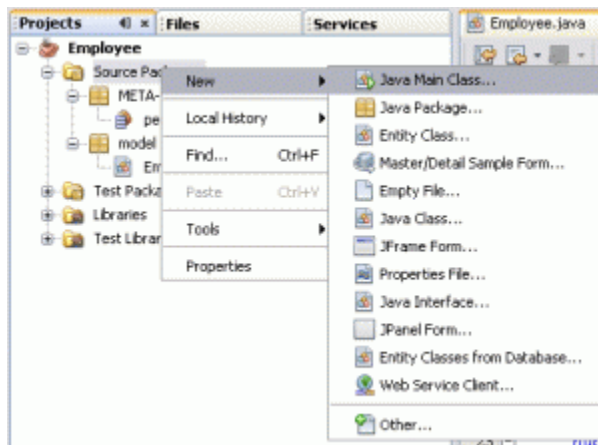
Sedangkan `@javax.persistence.GeneratedValue` adalah annotation yang akan membuat JPA akan menggenerate primary key auto increment. Ada 4 macam strategy yang digunakan, yaitu:

- AUTO (default), membiarkan provider (dalam hal ini toplink) yang menentukan 3 dari strategy dibawah ini.

- SEQUENCE, menggunakan SQL Sequence untuk mendapatkan primary key
- TABLE, toplink akan secara otomatis membuat satu table SEQUENCE dengan dua kolom: nama sequence dan nilainya (toplink default strategy)
- IDENTITY, menggunakan identity generator, seperti kolom yang di set auto\_increment pada MySQL

Sekarang mari kita buat main class

- Klik kanan pada Source Packages -> New -> Java Main Class



- Pada New Java Main Class

Class Name: MainEmployee  
Package: main

Name and Location	
Class Name:	MainEmployee
Project:	Employee
Location:	Source Packages
Package:	main
Created File: D:\Project\Employee\src\main>MainEmployee.java	

- Klik Finish

## MainEmployee.java

```
package main;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import model.Employee;

public class MainEmployee {

    public static void main(String[] args) {
        Employee emp = new Employee();
        emp.setName("Roy Abu Bakar");
        emp.setDepartment("ICT");
        emp.setJobTitle("Kuli Komputer");

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("EmployeePU");
        EntityManager em = emf.createEntityManager();
        em.getTransaction().begin();

        try {
            em.persist(emp);
            em.getTransaction().commit();
        } catch (Exception ex) {
            ex.printStackTrace();
            em.getTransaction().rollback();
        } finally {
            em.close();
        }
    }
}
```

Run Main class, lalu buka console MySQL

```
view plain copy to clipboard print ?
```

```
use employee;
show tables;
```

```
+-----+
| Tables_in_employee |
+-----+
| EMPLOYEE           |
| SEQUENCE           |
+-----+
2 rows in set (0.00 sec)
```

```
view plain copy to clipboard print ?
```

```
desc employee;
```

```

+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| ID         | bigint(20)    | NO   | PRI |          |       |
| NAME       | varchar(255)  | YES  |     | NULL     |       |
| DEPARTMENT | varchar(255)  | YES  |     | NULL     |       |
| JOBTITLE   | varchar(255)  | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

```

view plain copy to clipboard print ?
select * from employee;

```

```

+-----+-----+-----+-----+
| ID | NAME          | DEPARTMENT | JOBTITLE   |
+-----+-----+-----+-----+
| 2  | Roy Abu Bakar | ICT        | Kuli Komputer |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Bisa dilihat bahwa JPA membuat tabel secara otomatis dan menginsert objek-objek ke dalam tabel, pada contoh diatas kita sama sekali tidak menggunakan sql command .

## II. Java Persistence in the Java EE 5 Platform

Melalui pembahasan ini, akan mempelajari dasar-dasar pengembangan web menggunakan aplikasi Java™ Persistence. Java persistence API diperkenalkan sebagai bagian dari platform Java EE 5. Dalam tutorial ini akan membuat sebuah proyek aplikasi web sederhana. Dalam proyek ini menggunakan persistence, Java persistence API memungkinkan untuk menggunakan persistence tanpa perlu membuat sebuah modul EJB.

Menggunakan Java persistence disebuah project sangat memudahkan dalam pengembangan aplikasi dengan mengkonfigurasi penyebaran descriptors untuk menyediakan informasi object-relational mapping untuk persistence field atau properties. Sebagai gantinya, Anda dapat menggunakan annotation untuk menentukan properti secara langsung di class Java.

Entitas persistence dikelola oleh EntityManager API. EntityManager API menangani konteks, dan setiap isi persistence adalah sekelompok entitas Instance. Saat mengembangkan aplikasi, dapat menggunakan annotation didalam class untuk membedakan isi persistence instance dari entity instance. siklus dari entitas instance kemudian ditangani oleh container.

Pembahasan ini menggunakan IDE netbean 5.5

Note: IDE netbean terbaru adalah 6.1 jika menggunakan IDE 6.0 atau 6.1 lihat di [java persistence di platform java EE 5](#)

### Prasyarat

Dalam pembahasan ini pembaca diasumsikan memiliki dasar pengetahuan, atau pengalaman dengan pemrograman dengan aplikasi berikut:

- \* Pemrograman Java
- \* NetBeans IDE

### Perangkat lunak yang diperlukan untuk Tutorial

Untuk tutorial ini Anda perlu memiliki beberapa perangkat lunak yang diinstal pada komputer:

- NetBeans IDE 5.5
- Java Standard Development Kit (JDK) version 5.0 or version 6.0 ([download](#))
- Sun Java System Application Server Platform Edition 9 ([download](#))

Untuk tutorial ini diperlukan register di sebuah local instance dari Sun Java

Server dengan aplikasi IDE.

### **Tutorial Latihan :**

- \* Setting Proyek aplikasi web
- \* Membuat persistence Unit
- \* Membuat Enti class
- \* Membuat Web Interface
- \* Menjalankan Proyek
- \* Pemecahan masalah

### **Setting web aplikasi**

Tujuan dari latihan ini adalah untuk menciptakan aplikasi web ZooApp. Aplikasi ini akan memanfaatkan fitur baru dari model java persistence. Salah satu fiturnya adalah entitas objek dapat dibuat sederhana dengan class java, sehingga mereka tidak lagi perlu ditempatkan di sebuah EJB modul dan di simpan dalam file EAR. berarti entitas kelas dapat langsung dibuat dalam aplikasi web.

1. Pilih File> New Project (Ctrl-Shift-N). Pilih aplikasi Web dari Web kategori dan klik Next.
2. Nama proyek ZooApp, set server ke Sun Java System Application Server , set versi Java EE ke Java EE 5, dan klik Next.
3. Pilih JavaServer centang dan klik Finish.

Ringkasan

Dalam latihan ini telah membuat sebuah aplikasi web Java EE 5 yang akan berisi entiti class.

### **Membuat Persistence Unit**

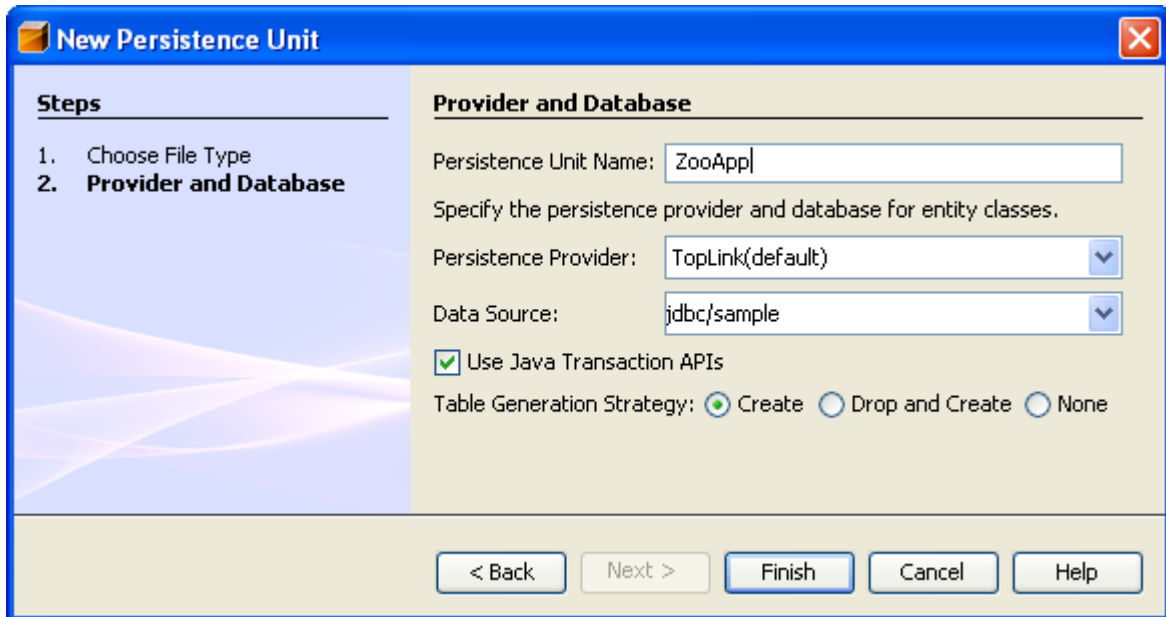
Dalam latihan ini, mencoba membuat sebuah persistence unit yang akan memberitahukan container, entiti class mana yang akan dimanage oleh entiti manager, dan juga sumber data yang digunakan oleh entitas tersebut.

Membuat persistence unit yang dibuat dalam module web dengan mendefinisikan propertinya di persistence.xml . Setelah menetapkan nama persistence unit, lalu akan menentukan *persistence provider*, berisi cotainer API yang digunakan untuk mengelola entitas instance. perlu ditentukan juga sumber data dan tabel. Mencoba membuat persistence unit dengan menggunakan *New Persistence Unit Wizard*.

**Note:** anda dapat membuat persistence unit di New Entitas class Wizard. Ketika

membuat sebuah entitas class, wizard akan meminta untuk membuat sebuah persistence unit jika tidak ada yang lain.

Membuat Persistence Unit :



1. Klik kanan aplikasi web ZooApp dalam windows dan pilih New> File / Folder Baru  
untuk membuka new file wizard.
2. Dari kategori persistence, pilih Persistence Unit dan klik Next.
3. Abaikan nama Persistence Unit.
4. Untuk Persistence provider, pilih TopLink (standar).  
Default provider adalah TopLink Essential.jar. TopLink Essential.jar berisi library persistence untuk Java. Entitas manajer terletak di TopLink Essential.jar.
5. Untuk Sumber data, menggunakan standar jdbc jdbc/sample.  
Default jdbc/sample digunakan untuk menghubungkan ke java DB database yang telah disediakan java server.
6. Pastikan persistence unit menggunakan java Transaction API dan bahwa Tabel Buat sehingga tabel berdasarkan class entitas dibuat bila aplikasi tersebut dideploy.
7. Klik Selesai.

### Menciptakan entitas class

Dalam latihan ini akan dibuat dua entitas kelas, *Animal.java* dan *Pavilion.java*, yang mewakili tabel dalam database relasional. Tentukan

beberapa field dalam class yang merepresentasikan data. Java persistence memungkinkan untuk menggunakan anotasi untuk memberikan informasi kepada container tentang field, seperti ORM.

## Membuat entiti class Animal

Pertama, kita akan menciptakan entitas class Animal. Class ini mewakili table ANIMAL dalam database. Ketika menciptakan entitas class, IDE menambahkan anotasi `@Entity` sebagai penanda class tersebut sebagai entitas class. Setelah membuat class, buatlah field dalam class untuk merepresentasikan data yang dibutuhkan dalam tabel, gunakan anotasi untuk memperjelas informasi field

Setiap entitas class harus memiliki kunci utama (primary key). Ketika menciptakan entitas kelas, IDE menambahkan anotasi `@Id` yang menyatakan field yang digunakan sebagai primary key. IDE juga menambahkan anotasi `@GeneratedValue` untuk menetapkan kunci general sebagai *primary id*.

Untuk menciptakan class Animal, lakukan hal berikut:

1. Klik kanan ZooApp dan memilih New> File / Folder.
2. Dari katagori persistence, pilih Entity class dan klik Next.
3. Nama class tulis Animal, entity untuk package, abaikan primary key bertipe Long .  
Klik Finish.

Bila sudah diklik Finish, entity class Animal.java terbuka di Editor. di Editor, tambahkan hal berikut:

1. Tambahkan deklarasi berikut dalam classs:

```
String name;  
String kind;  
String weight;  
Pavilion pavilion;
```

2. Klik kanan dalam editor pilih Refactor> Encapsulate field untuk menghasilkan getters dan setter . Dalam kotak dialog Encapsulate Field, pastikan bahwa checkbox ter pilih semua dalam getter dan setter.
3. Klik Next di kotak dialog Encapsulate Field dan kemudian klik Do Refactoring di tab Refactoring.
4. Sekarang mencoba mengubah nama salah satu kolom yang dibuat dalam tabelAnimal. Dalam colom *name* diberi nama *animalName*.

```
@Column(name="animalName")  
private String name;
```

5. lanjutkan dengan mengetik :

```
@ManyToOne  
private Pavilion pavilion;
```

6. Tekan Alt-Shift-F untuk mengenerate statement impor yang dibutuhkan.

7. Simpan perubahan Anda.

Pada langkah selanjutnya kita akan menciptakan Pavilion Entity Class.

### **Membuat Pavilion Entity Class:**

Membuat entitas clas Pavilion mewakili tabel PAVILION dalam database. kembali menggunakan anotasi di class untuk menentukan ORM dari beberapa kolom. Untuk menciptakan Pavilion class, lakukan hal berikut:

1. Klik kanan ,ZooApp dan pilih New> File / Folder.
2. Dari katagori persistance, pilih Entity class dan klik Next.
3. Nama class : Pavilion, entitas untuk package, dan abaikan primary key bertipe Long  
Klik Finish.

Bila anda klik Selesai, baru kelas entitas Pavilion.java Sumber terbuka di Editor. Sumber di Editor, melakukan hal berikut:

1. Tambahkan deklarasi berikut dalam classs:

```
String name;  
String address;  
Collection <Animal> animals;
```

2. Pilih Refactor> Encapsulate Field untuk menghasilkan yang getters dan setter untuk masing- masing field.
3. Tambahkan anotasi berikut di atas nama deklarasi untuk mengubah nama kolom yang dihasilkan:

```
@Column(name="pavilionName")  
private String name;
```

4. Tambahkan anotasi berikut di atas animal collection untuk menetapkan hubungan satu-ke-Banyak untuk entitas:

```
@OneToMany(mappedBy="pavilion")  
private Collection <Animal> animals;
```

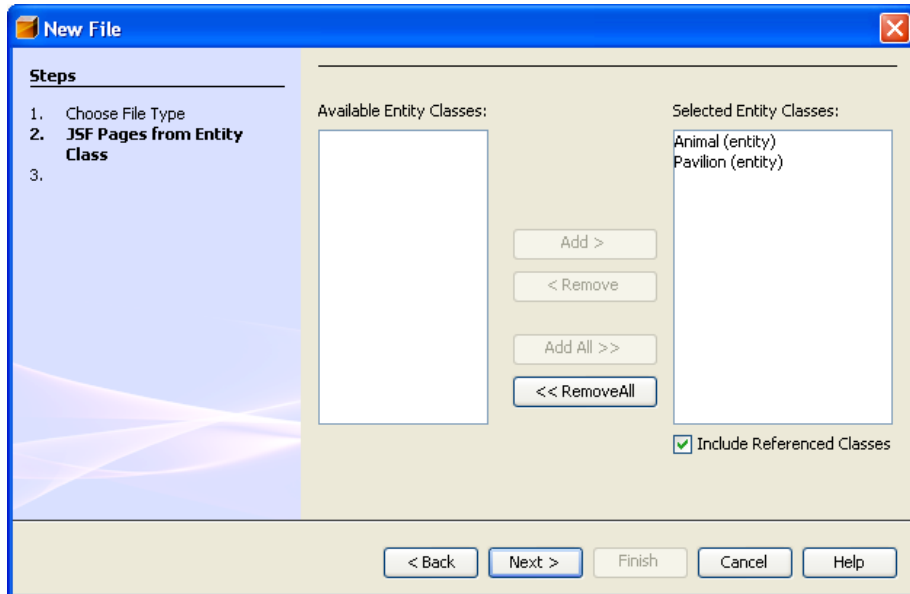
5. Tekan Alt-Shift-F untuk mengenerate statement impor yang dibutuhkan.

6. Simpan perubahan Anda.

## Ringkasan

Dalam latihan ini, Anda menciptakan dua entitas class dan menentukan field. Juga menggunakan anotasi untuk menentukan propertis dari masing kolom didalam table tabel yang akan dihasilkan ketika mendeploy aplikasi.

## Membuat web interface



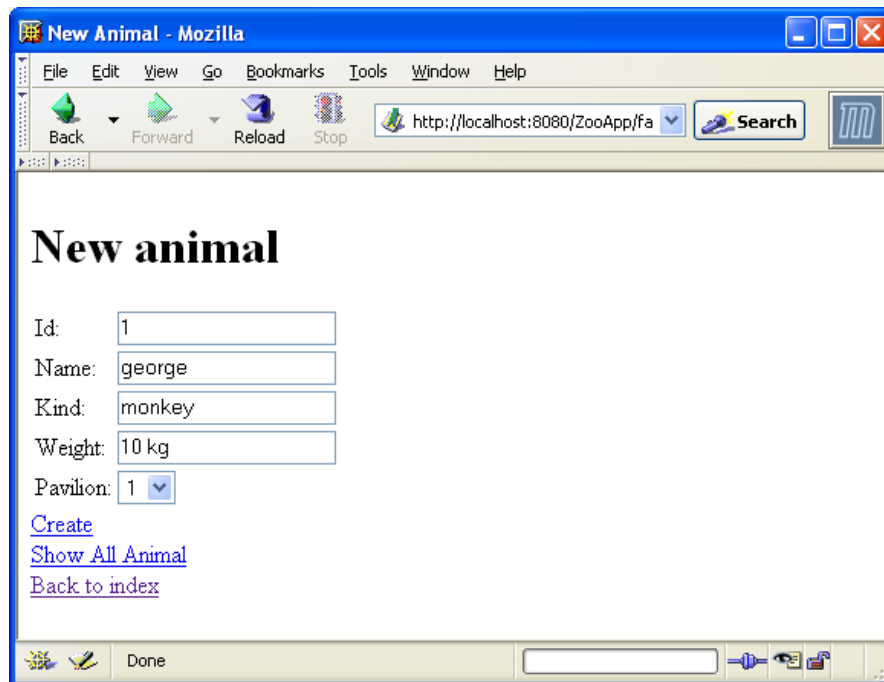
1. Pilih File> New dari menu utama. Pilih JSF pages dari Entitas clas kategori persistence dan klik Next.
2. di Halaman JSF dari Entitas class wizard, klik Tambah Semua dua entitas class dan klik Next.
3. Biarkan folder Halaman JSF text field nya kosong untuk menyimpan file JSF di lokasi default.
4. Tetapkan sebagai entitas package untuk kelas dihasilkan dan klik Selesai.

## Menjalankan project

Selanjutnya deploy ZooApp dan test aplikasi tersebut

1. start java DB database , pilih tools>java DB database> start java DB server
2. 2. klik kanan project ZooApp dan pilih Run Project .

Bila Run dijalankan, sebuah halaman web akan terbuka di browser ,menu yang memungkinkan untuk melihat daftar pavilions dan Animal.



Ringkasan :

Dalam latihan ini, telah terbangun sebuah aplikasi web ZooApp .

## Daftar Pustaka

<http://java.sun.com/javaee/overview/faq/persistence.jsp>

<http://java.sun.com/developer/technicalArticles/J2SE/Desktop/persistenceapi/>

<http://blog.rab.web.id/general/episode-1-java-persistence-api-jpa-dengan-toplink-dan-mysql/>

[http://en.wikipedia.org/wiki/Java\\_Persistence\\_API](http://en.wikipedia.org/wiki/Java_Persistence_API)

<http://jasoet.wordpress.com/2008/11/10/persistence-sederhana-dengan-netbeans-2/>

<http://www.netbeans.org/kb/55/web/customer-book.html>