Materi Pelatihan

Java Desktop

Ifnu Bima
ifnu@artivisi.com
http://ifnu.artivisi.com

http://www.artivisi.com
Jakarta, 2008

Daftar Isi

Java Foundation Class

Java Foundation Class (JFC) merupakan sekumpulan class-class Java yang digunakan untuk mengembangkan perangkat lunak berbasis GUI (Graphical User Interface). Selain itu, JFC juga mempunyai class-class yang digunakan untuk menambahkan fungsi dan kemampuan interaksi yang variatif dari pemrograman Java. Dari definisi ini, JFC tidak hanya berisi class-class GUI saja tetapi juga class-class lain yang dapat meningkatkan kemampuan pemrograman Java baik dari segi fungsionalitasnya maupun dari segi kemampuan interaksi pemrograman Java yang sangat kaya.

Feature JFC

Fitur-fitur yang dipunyai oleh JFC	
Fitur	Deskripsi
Komponen Swing	Memuat semua class-class yang dibutuhkan untuk membuat aplikasi berbasis GUI, dari tombol, table, tab, menu, toolbar dan sebagainya
Look and Feel (LaF)	Memberikan kemampuan kepada program Java yang dikembangkan menggunakan library swing untuk memilih tema tampilan. Misalnya sebuah program yang sama dapat mempunyai tampilan windows LaF atau Java LaF, atau LaF lain yang dikembangkan oleh komunitas seperti JGoodies.
Accessibility API	Faslititas untuk mengembangkan aplikasi bagi penyandang cacat, misalnya dukungan untuk membuat huruf braile, kemampuan mengambil input dari layar sentuh dan sebagainya.
Java 2D API	Berisi kumpulan class-class yang dapat digunakan untuk memanipulasi object-object 2 dimensi, sperti garis, kotak, lingkaran, kurva dan lain sebagainya. Selain itu Java 2D API juga memberikan kemampuan program yang ditulis menggunakan Java untuk mencetak output ke alat pencetak seperti printer.
Drag-and-drop	Menyediakan kemampuan drag-and-drop antara program Java dan program lain yang ditulis spesifik untuk suatu platform sistem operasi tertentu.

International ization (i18n)	Membantu pengembang perangkat lunak untuk membangun aplikasi yang dapat mendukung semua bahasa dan huruf yang ada di dunia.

Tabel Feature JFC

Modul ini akan berkonsentrasi untuk membahas komponen swing. Pemilihan komponen dan library swing yang tepat dapat mempengaruhi kualitas program yang kita buat secara signifikan. Hal ini dikarenakan, dalam dunia Java Standard Edition, lebih spesifik lagi aplikasi Java yang dibangun menggunakan swing, belum ada framework yang benar-benar komprehensif membimbing pengembang untuk membuat aplikasi yang berkualitas.

Pada umumnya aplikasi yang dikembangkan dengan Swing mempunyai kode yang sangat 'kotor', dimana kode yang berisi pengendalian terhadap event komponen swing bercampur aduk dengan kode yang berisi aturan bisnis dan kode yang berisi manipulasi terhadap data.

Swing Package

Swing API sangat bagus dan lengkap, Java 6.0 menyertakan setidaknya tujuh belas (17) buah package yang berisi class-class swing yang siap digunakan.

javax.accessibility	javax.swing.plaf	javax.swing.text
javax.swing	javax.swing.plaf.basic	javax.swing.text.html
javax.swing.border	javax.swing.plaf.metal	javax.swing.text.rtf
javax.swing.colorchooser	javax.swing.plaf.multi	javax.swing.table
javax.swing.event	javax.swing.plaf.synth	javax.swing.tree
javax.swing.filechooser		javax.swing.undo

Utungnya kita tidak akan menggunakan semua class-class dalam package swing, hanya sebagian kecil saja dari class-class tersebut yang nantinya akan benar-benar kita gunakan. Sehingga kita bisa berkonsentrasi untuk memahami beberapa komponen penting saja. Dalam modul ini nanti kita hanya akan menggunakan beberapa class komponen swing yang penting saja. Beberapa kelas ini sudah cukup sebagai bahan pemembuat perangkat lunak berkualitas.

Komunitas Java juga menyediakan banyak sekali library swing, antara lain dari Swingx dan JGoodies yang mengembangkan

library standard swing dengan menambahkan berbagai macam feature menarik. Sedangkan komunitas dari javadesktop.org mengembangkan banyak sekali library swing untuk keperluan khusus. Nyaris semua komponen yang kita perlukan baik komponen umum hingga komponen untuk tujuan khusus banyak tersedia di internet, kita tinggal mencari dan menggunakan.

Praktek yang baik dalam memilih komponen apa yang tepat adalah dengan mencari dahulu informasi di internet. Hal ini sangat bermanfaat untuk mengurangi waktu kita mengembangkan komponen, sehingga kita bisa lebih banyak berkonsentrasi untuk mengembangkan sisi bisnis dan usability dari software yang kita kembangkan. Sebaik apapun softeware yang kita buat tapi tidak memberikan nilai tambah terhadap masalah yang dihadapi adalah kesia-siaan belaka. Banyak sekali software yang dianggap gagal memberikan nilai tambah terhadap masalah yang dihadapi hanya karena tampilan GUI-nya sangat susah dipahami dan tidak intuitif.

Swing HelloWorld

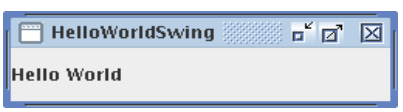
Menggunakan contoh langsung adalah cara yang tepat untuk memulai proses belajar. Cara ini memberikan gambaran kongkrit tentang subject yang akan dipelajari, sehingga proses belajar lebih cepat diserap. Untuk tujuan ini, kita akan membuat sebuah program kecil yang menampilkan kata “HelloWorld” menggunakan komponen swing. Berikut ini adalah langkah-langkah yang harus anda lakukan untuk membuat program “HelloWorld” berbasis komponen swing:

1. Install Java Development Kit (JDK)
2. Membuat program HelloWorld itu sendiri
3. Melakukan kompilasi program HelloWorld
4. Menjalankan program HelloWorld

Install Java Development Kit

Yang perlu kita lakukan dalam langkah ini hanyalah mendownload JDK dari website java.sun.com. kemudian jalankan program instalasinya dan ikuti perintah-perintah dalam langkah-langkah instalasi tersebut. Setelah proses instalasi selesai, kita siap untuk membuat program Java.

Membuat program HelloWorld



Program Java dengan tampilan seperti di atas dapat dibuat dengan dua cara. Cara yang pertama adalah dengan menggunakan text editor dan mengetik kode program. Cara yang kedua adalah dengan menggunakan Netbeans Matisse GUI Builder.

Lakukan langkah-langkah berikut ini untuk membuat program diatas menggunakan text editor:

1. Buka text editor kesayangan anda.
2. Ketikkan kode program di bawah ini dan simpan dengan nama file HelloWorld.java :

```
public class HelloWorld {
public HelloWorld(){ }
public void display(){
    JFrame.setDefaultLookAndFeelDecorated(true);
    JLabel label = new JLabel("HelloWorld");
    JFrame frame = new JFrame();
    frame.getContentPane().add(label);
    frame.setVisible(true);
    frame.pack();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
public static void main(String[] str){
    HelloWorld hello = new HelloWorld();
    hello.display();
    }
}
```

Melakukan kompilasi program HelloWorld

Kompilasi program tersebut dengan cara menjalankan program javac (java compiler). Jika anda bekerja di lingkungan windows buka command prompt, kemudian ketik program berikut ini :

```
c:\latihan> javac HelloWorld.java
```

Jika anda bekerja di lingkungan GNU/linux, buka console dan ketikkan perintah berikut ini :

```
shell$ javac HelloWorld.java
```

Menjalankan program HelloWorld

Proses kompilasi akan menghasilkan file yang berekstensi .class, file inilah yang akan kita eksekusi. Jika anda bekerja di lingkungan windows lakukan perintah berikut ini:

```
c:\latihan> java HelloWorld
```

Jika anda bekerka di lingkungan GNU/Linux jalankan perintah berikut ini:

```
shell$ java HelloWorld
```

Membuat Swing HelloWorld dengan Netbeans 6.1

Netbeans 6.1 dilengkapi dengan GUI builder yang dikenal dengan Matisse. Dalam modul ini selanjutnya, Matisse akan digunakan untuk menyebut Netbeans GUI Builder. Tools ini sangat powerful dan produktif dalam membuat komponen GUI. Langkah-langkah yang harus anda lakukan untuk membuat Swing HelloWorld dengan Matisse adalah sebagai berikut:

1. Buat project baru dalam Netbeans, caranya pilih menu :

```
File > New Project
```

2. Langkah berikutnya anda harus menentukan kategori project yang akan anda buat, caranya pilih :

```
General > Java Application
```

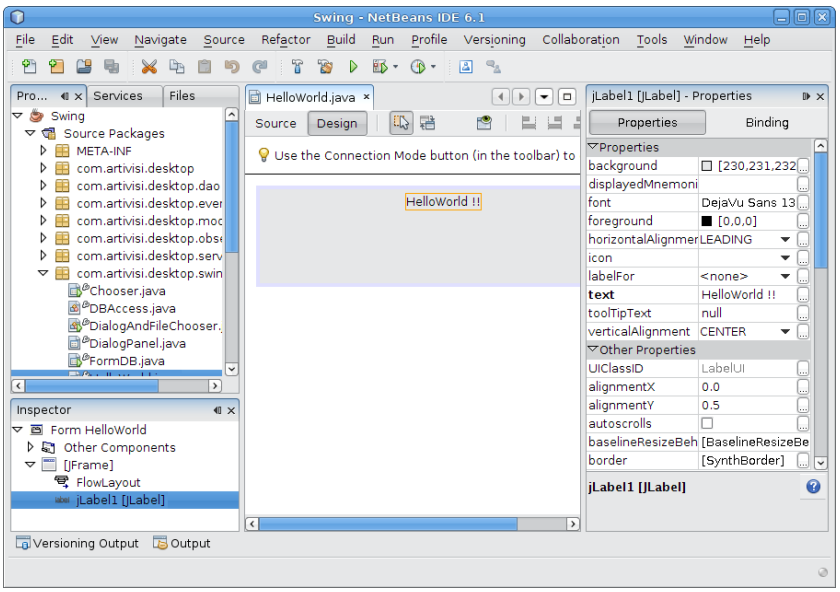
   Anda akan dibawa ke dialog untuk menentukan nama project dan folder dimana anda meletakkan project tersebut, pilih folder sesuai keinginan anda.

3. Klik kanan di project yang baru anda buat, popup menu akan muncul, kemudian pilihlah menu :

```
New > JFrame Form...
```

   Kemudian masukkan nama class JFrame yang akan dibuat, misalnya Helloworld.java, klik finish.

4. Tampilan Netbeans akan berganti dengan tampilan GUI builder, dimana di sisi kanan akan terlihat Swing Pallet. Klik item Label di Swing Pallet kemudian klik di atas JFrame, sebuah JLabel akan dibuat.



Jendela Design dan Pallete Netbeans Matisse

5. Untuk memenuhi tujuan kita membuat Swing HelloWorld, kita akan memasukkan string “HelloWorld” ke dalam JLabel yang baru saja kita buat. Caranya, dobel klik di atas JLabel tersebut, kursor muncul bersama text field dan ketikkan “Helloworld”.
6. Klik kanan di file HelloWorld.java pada jendela explorer di sebelah kiri, pilih menu **Run File...** untuk mengcompile dan menjalankan class HelloWorld.java atau tekan tombol SHIFT + F6.

Matisse mempunyai sistem Layouting yang sangat fleksible, sistem layout yang digunakan oleh Matisse adalah GroupLayout. Dalam chapter berikutnya kita akan belajar bagaimana menggunakan GroupLayout ini dengan benar dan memanfaatkan keunggulanya dalam menata component GUI dengan sangat rapi.

Swing helloworld ini hanya sebagian kecil saja dari pekerjaan yang harus dilakukan dalam membangun aplikasi desktop berbasis Java. Selanjutnya kita akan membahas penggunaan JLabel, JButton, JCheckBox, JTextField dan JRadioButton untuk membuat aplikasi GUI sederhana dengan menggunakan Matisse.

Komponen Swing

Swing toolkit menyediakan banyak sekali komponen untuk membangun aplikasi GUI desktop. Swing toolkit juga menyediakan class-class untuk menangani interaksi antara aplikasi dan user menggunakan standard input seperti keyboard dan mouse. Komponen-komponen yang disediakan swing mencakup semua GUI toolkit yang lazim digunakan dalam apilasi desktop, seperti : JTabel, JList, JTree, JButton, JLabel dan masih banyak komponen-komponen lainnya yang sudah teruji dan siap pakai.

Selain komponen GUI, swing juga menyediakan fasilitas untuk proses undo, komponen untuk mengolah text, internationalization, Komponen GUI yang mendukung penyandang cacat (accessibility support) dan fasilitas drag-and-drop.

Look and Feel merupakan fasilitas yang unik dalam swing. Dengan fasilitas Look and Feel ini kita bisa dengan mudah merubah tampilan dari program kita sesuai dengan keinginan dan tujuan kita. Misalnya, agar program terlihat fancy atau agar program terlihat konsisten dalam segala keadaan.

Swing juga menyediakan library Java 2D untuk pengolahan data secara visual, seperti mengolah gambar, object 2D, bahkan animasi. SwingLabs.org menyediakan libary Swing Painter yang merupakan pengembangan dari Java 2D, Swing Painter ini memungkinkan aplikasi swing mempunyai tampilan yang indah dan terlihat profesional.

Java 6.0 menambahkan banyak sekali fitur-fitur baru ke dalam package swing, termasuk dukungan untuk library OpenGL menggunakan JOGL, Tray Icon dan Web Service. Dengan adanya dukungan ini swing menjadi lebih poweful dan mempunyai masa depan yang cerah.

Struktur Komponen Swing

Secara arsitektur, Swing dibangun diatas arsitektur AWT (Abstract Windows Toolkit). AWT adalah GUI toolkit yang dikembangkan oleh Sun engineer sebelum swing muncul. Kelemahan utama AWT adalah fleksibilitas tampilan GUI, seperti painting method yang masih sangat primitif.

Swing dimaksudkan untuk memperbaiki kekurangan dari AWT tanpa harus membuang teknologi yang sudah dibuat dan membuat

GUI toolkit baru dari nol.

Komponen AWT diletakkan dalam satu package yaitu java.awt, didalamnya terdapat komponen-komponen GUI dasar, salah satunya adalah Component. Class Component adalah moyang dari sebagian besar komponen AWT maupun Swing. CheckBox, Label, Button dan beberapa komponen AWT lainnya adalah turunan langsung dari class Component. Namun dalam kenyataanya arsitektur demikian tidak memberikan fleksibilitas yang cukup memadai untuk membuat berbagai macam komponen baru yang dibutuhkan dalam desktop application.

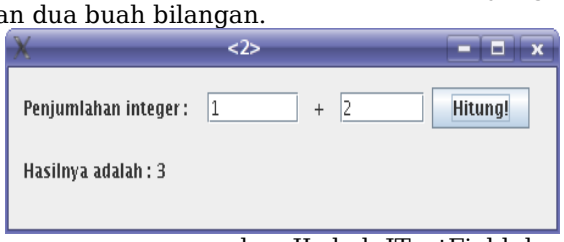
Swing muncul dengan membawa teknologi AWT yang telah ditambahkan dengan banyak kemampuan. Nyaris semua komponen GUI dari swing merupakan turunan class Container dan class Container adalah turunan dari class Component.

Bekerja dengan JLabel, JTextField dan JButton

Bekerja dengan komponen swing menggunakan Matisse sangat menyenangkan dan mudah. GroupLayout yang sangat fleksibel memungkinkan kita untuk membuat aplikasi dengan tampilan seperti yang kita harapkan.

Label, textfield dan tombol adalah komponen-komponen dasar yang selalu ada dalam setiap aplikasi berbasis desktop. Ketiga komponen ini mempunyai fungsi yang sangat sederhana, textfield menyimpan data berbentuk text (string) yang relatif pendek , label banyak digunakan untuk memberikan keterangan penjelas terhadap komponen lain dan tombol digunakan user untuk menjalankan satu instruksi tertentu.

Berikut ini adalah contoh aplikasi sederhana yang melakukan penjumlahan dua buah bilangan.



Contoh program menggunakan JLabel, JTextField dan JButton

Untuk membuat aplikasi ini menggunakan Matisse, lakukan langkah-langkah berikut ini:

1. Buat project baru di Netbeans (kalau sudah membuat project,

tidak perlu membuat lagi) dengan cara memilih menu :

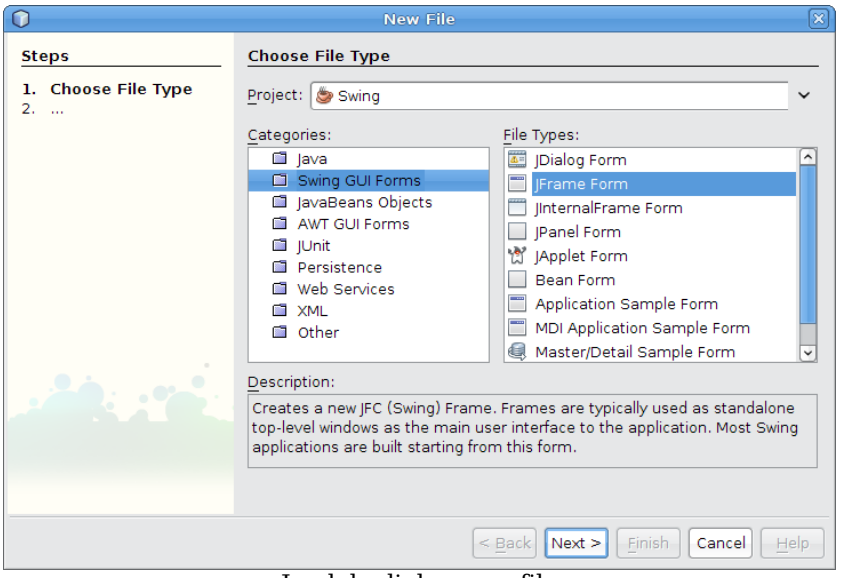
```
File > New Project
```

Kemudian ikuti petunjuk yang diberikan dialog.

2. Buat class JFrame baru, caranya dengan memilih menu :

```
File > New File
```

Kemudian akan muncul dialog seperti di bawah ini :



Jendela dialog new file

3. Pilih kategori :

```
Java GUI Forms > JFrame Form
```

Seperti terlihat di dialog New File dialog diatas, kemudian beri nama Penjumlahan.java

4. Buat tampilan form seperti gambar bawah ini, caranya dengan klik Jendela Pallete di sebalah kanan untuk memilih komponen apa yang akan dibuat, kemudian klik di jendela Design untuk menempatkan komponen yang sudah dipilih tadi ke dalam form. Hasilnya terlihat seperti pada gambar di bawah ini:

Jendela design Netbens Matisse

5. Ganti nama setiap komponen agar mudah dikenali. Klik kanan diatas setiap komponen yang ada dalam Jendela Design diatas, kemudian pilih menu :

penting

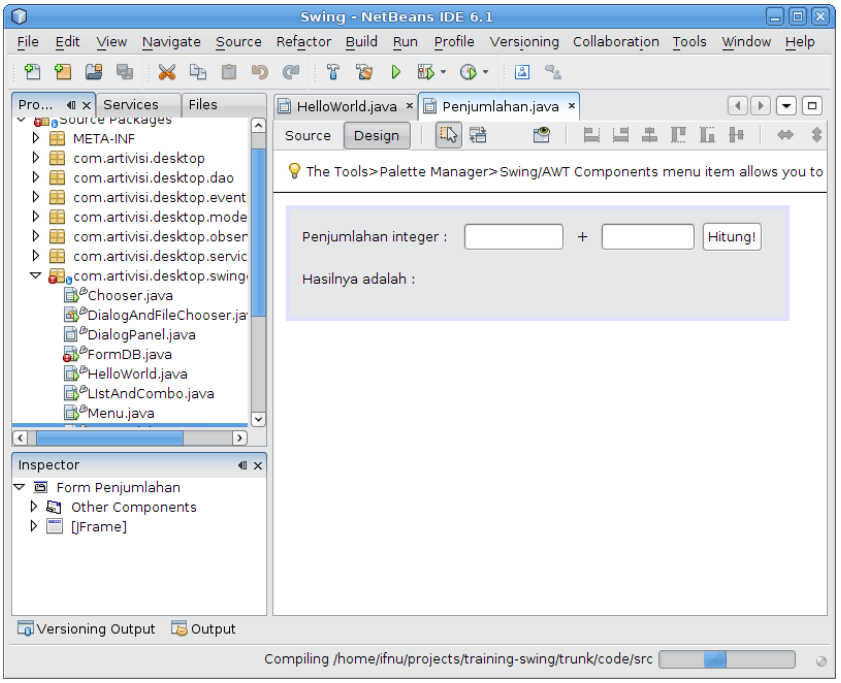
Jendela Design menampilkan visualisasi komponen GUI.

Jendela Source menampilkan kode program dari class yang sedang dibuka.

Jendela Swing Pallete berisikan komponen-komponen swing yang bisa kita drag-and-drop ke dalam jendela design.

Jendela Properties digunakan untuk mengedit properti dari komponen yang sedang aktif dalam jendela design.

Jendela Inspector menampilkan semua komponen swing dalam class yang sedang aktif baik yang kelihatan secara visual di jendela design atau tidak.




```
Klik kanan > Change Variable Name ...
```

Ganti nama komponen-komponen tersebut (sesuai urutan dari kiri ke kanan, atas ke bawah) menjadi : lblKeterangan, txtA, lblPlus, txtB, btnHitung, lblHasil.

6. Menambahkan variable untuk menampung nilai yang akan dijumlahkan. Klik tombol Source untuk membuka jendela yang menampilkan kode sumber dari program di atas kemudian tambahkan kode di bawah ini tepat dibawah definisi dari class Penjumlahan:

```
private String str = "Hasilnya adalah : ";
private int a, b;
```

7. Menangani penekanan tombol btnHitung. Klik kanan diatas komponen btnHitung kemudian pilih menu :

```
Events > Action > actionPerformed
```

Anda akan dibawa ke jendela Source, dan akan menemukan kode program seperti di bawah ini :

```
private void btnHitungActionPerformed(
      java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}
```

Ubah kode program diatas menjadi :

```
private void btnHitungActionPerformed(
      java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        a = Integer.parseInt(txtA.getText());
        b = Integer.parseInt(txtB.getText());
        int hasil = a + b;
        lblHasil.setText(str + hasil);
}
```

penting

JLabel dan JTextField mempunyai method getText dan setText untuk mengambil dan mengeset text yang ditampilkan.

8. Compile dan jalankan program. Tekan tombol SHIFT + F6, atau klik kanan file Penjumlahan.java kemudian pilih menu Run File.

Catatan :

- Method Integer.parseInt digunakan untuk merubah String menjadi Integer.
- Method btnHitungActionPerformed akan dipanggil setiap kali kita memencet tombol btnHitung.

Sekarang anda bisa melihat bahwa bekerja dengan JLabel, JTextField dan JButton sangat sederhana. Untuk latihan, silahkan rubah fungsi yang digunakan dalam program diatas, misalnya perkalian dua bilangan atau pengurangan dua bilangan.

Bekerja dengan JCheckBox dan JRadioButton

penting

JCheckBox dan JRadioButton sebaiknya digunakan hanya jika item pilihanya sedikit dan tidak bersifat dinamis

JCheckBox dan JRadioButton hanya bisa mempunyai dua buah kemungkinan nilai, benar atau salah. Kedua komponen ini digunakan untuk merepresentasikan data yang berupa pilihan. JCheckBox digunakan jika pilihanya berupa multiple selection, sedangkan JRadioButton digunakan jika pilihanya berupa single selection.

JRadioButton digunakan misalnya untuk merepresentasikan pilihan jenis kelamin. JCheckBox digunakan misalnya untuk merepresentasikan pilihan hobby.

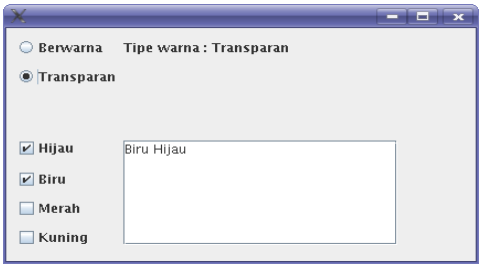
ButtonGroup diperlukan untuk mengumpulkan JRadioButton yang mempunyai grup pilihan yang sama. Misalnya grup pilihan jenis kelamin digunakan untuk mengumpulkan JRadioButton yang

merepresentasikan pilihan laki-laki dan JRadioButton yang merepresentasikan pilihan perempuan dalam satu group. Jika JRadioButton tidak diletakkan dalam satu group, maka pilihan laki-laki dan pilihan perempuan bisa dipilih bersamaan.

Status dari JRadioButton dan JCheckBox dapat diketahui dengan melihat nilai kembalian dari method isSelected, jika dipilih maka nilai kembalian method isSelected adalah benar, dan false jika sebaliknya.

Setiap JRadioButton dan JCheckBox mempunyai text yang menerangkan pilihan yang diwakilinya. Method getText dan setText digunakan untuk memanipulasi text.

Dibawah ini adalah contoh program yang menggunakan JCheckBox dan JRadioButton.



Contoh aplikasi menggunakan JCheckBox dan JRadioButton

Di bagian atas aplikasi ini, terdapat dua JRadioButton untuk merepresentasikan pilihan tipe warna, transparan atau berwarna. Dibawahnya terdapat pilihan warna yang dapat dipilih lebih dari satu buah menggunakan JCheckBox.

Untuk membuat program diatas ikuti langkah-langkah berikut ini:

1. Buat class baru bertipe JFrame Form, kemudian beri nama Pilihan.java
2. Buat tampilan diatas menggunakan Matisse. komponen yang harus dibuat adalah :
   - Dua object JRadioButton : radioBerwarna dan radioTransparan.
   - Satu object ButtonGroup : groupTipeWarna.
   - Empat object JCheckBox : chkHijau, chkBiru, chkMerah, chkKuning.
   - Satu object JTextArea : txtWarna.
   - Satu object JScrollPane : scrollWarna

Untuk melihat semua komponen yang ada dalam Jendela Design, gunakan Jendela Inspector di sisi kiri bawah.

3. Masukkan object radioBerwarna dan radioTransparan ke dalam object groupTipeWarna. Caranya dengan :
   a) Memilih komponen radioBerwarna di Jendela Design
   b) Klik tab code di Jendela Properties
   c) Pilih properti : Post-Creation Code
   d) Masukkan kode berikut ini kedalam dialog yang muncul :

```
groupTipeWarna.add(radioBerwarna);
```

   Lakukan langkah yang sama terhadap object radioTransparan.

4. Menangani event ketika JRadioButton diklik. Caranya dengan :
   a) Memilih komponen radioBerwarna di Jendela Design
   b) Klik kanan komponen radioBerwarna, kemudian pilih menu:

```
Event > Action > actionPerformed
```

   c) Anda akan dibawa ke dalam Jendela Code, dan menemukan kode berikut ini :

```
private void radioBerwarnaActionPerformed(
      java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}
```

   Ubahlah kode diatas menjadi :

```
private void radioBerwarnaActionPerformed(
      java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        if(radioBerwarna.isSelected()){
            lblTipeWarna.setText("Tipe warna : " +
                    radioBerwarna.getText());
        }
    }
```

   Lakukan langkah yang sama terhadap radioTransparan.

5. Buat sebuah private method untuk menangani event pemilihan terhadap JCheckBox. Method tampilkanWarna ini nantinya akan dipanggil setiap kali salah satu dari JCheckBox dipilih. yang dilakukan oleh metod tampilkanWarna adalah mengecek status setiap JCheckBox, apakah sedang dipilih atau tidak. Jika sedang dipilih maka text dari JCheckBox tersebut akan ditampilkan dalam txtWarna.

   Class StringBuffer digunakan untuk menampung nilai text dari

penting

JRadioButton yang mempunyai group yang sama, harus dimasukkan dalam sebuah object ButtonGroup yang sama.

JCheckBox yang statusnya terpilih.

```
private void tampilkanWarna(){
    StringBuffer warna = new StringBuffer();
    if(chkBiru.isSelected()){
        warna.append(chkBiru.getText() + " ");
    }
    if(chkHijau.isSelected()){
        warna.append(chkHijau.getText() + " ");
    }
    if(chkKuning.isSelected()){
        warna.append(chkKuning.getText() + " ");
    }
    if(chkMerah.isSelected()){
        warna.append(chkMerah.getText() + " ");
    }
    txtWarna.setText(warna.toString());
}
```

penting

Class StringBuffer sangat dianjurkan untuk digunakan sebagai class untuk memanipulasi String.

Penggabungan string menggunakan operator + sangat tidak dianjurkan, apalagi jika ukuran object String-nya sudah cukup besar.

6. Menangani event pemilihan JCheckBox. Caranya sebagai berikut :
   a) Pilih komponen chkHijau di Jendela Design.
   b) Klik kanan komponen chkHijau untuk memunculkan context (popup) menu.
   c) Pilih menu :

```
Event > Action > actionPerformed
```

   d) Anda akan dibawa ke Jendela Code, kemudian dalam method chkHijauActionPerformed tersebut panggil method tampilkanWarna. seperti di bawah ini :

```
private void chkHijauActionPerformed(
        java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        tampilkanWarna();
}
```

   Lakukan hal ini untuk semua JCheckBox.

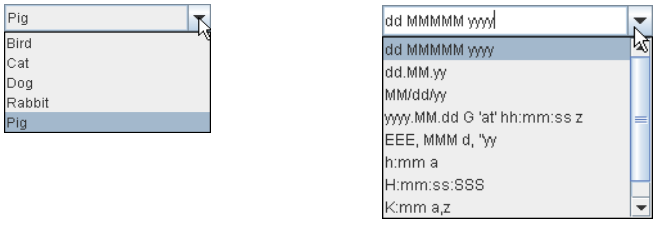
7. Compile dan jalankan program dengan menekan tombol SHIFT + F6.

Cara lain dalam menampilkan pilihan adalah dengan menggunakan JList dan JComboBox. Kedua komponen ini mempunyai fleksibilitas yang lebih tinggi dan lebih mudah digunakan jika object yang dimasukkan dalam pilihan lebih kompleks. JList dan JComboBox bisa mempunyai ComponentEditor agar pilihan yang ditampilkan tidak hanya berupa text, bisa berupa warna atau icon. Bagian berikutnya akan membahas bagaimana bekerja menggunakan JList

dan JComboBox.

Bekerja dengan JList dan JComboBox

JComboBox memerlukan tempat yang minimalis dibandingkan dengan JRadioButton, selain itu JComboBox mempunyai bentuk ComboBox yang dapat diedit, sehingga memungkinkan user untuk memilih pilihan yang tidak ada dalam item JComboBox.

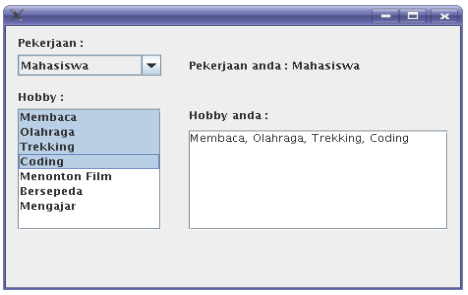


Contoh JComboBox

JList memungkinkan multiple selection dengan menekan tombol : SHIFT + Left Click atau CTRL + Left Click. Kemampuan ini membantu user jika harus melakukan multiple selection.

JComboBox dan JList sangat fleksibel, kita dapat menambah dan menghapus item di dalamnya dengan sangat mudah. Sehingga cocok digunakan untuk merepresentasikan pilihan yang item pilihannya bersifat dinamis.

Aplikasi di bawah ini adalah contoh penggunaan JComboBox dan JList.



Contoh program menggunakan JComboBox dan JList

Bagian pertama program ini terdapat sebuah JComboBox dan JLabel, setiap kali item di dalam JComboBox dipilih, JLabel di sebelahnya akan menampilkan item yang dipilih tersebut.

penting

JComboBox dan JList digunakan jika item pilihan bersifat dinamis.

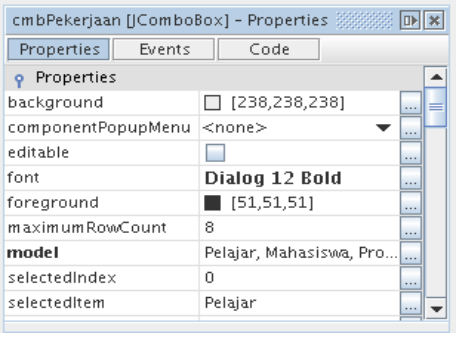
JComboBox dapat mempunyai bentuk yang dapat diedit sehingga user dapat memasukkan pilihan yang tidak ada dalam daftar.

JList dapat menerima pilihan lebih dari satu.

Bagian kedua program ini terdapat sebuah JList dan JTextArea. Setiap kali item-item di dalam JList dipilih, JTextArea akan menampilkan item-item yang dipilih tersebut dipisahkan dengan koma (,).

Ikuti langkah-langkah berikut ini untuk membuat program di atas:

1. Buatlah class JFrame Form baru dan beri nama ListAndCombo.java.
2. Buat tampilan program diatas menggunakan Matisse, kemudian tambahkan komponen-komponen:
   a) Empat buah JLabel : lblPekerjaan, lblPilihanPekerjaan, lblHobby, lblPilihanHobby.
   b) Satu buah JComboBox : cmbPekerjaan
   c) Satu buah JList : lstHobby
   d) Satu buah JtextArea : txtPilihanHobby
3. Merubah isi JComboBox. Untuk merubah isi dari JComboBox dan JList kita akan menggunakan Jendela Properties, Jendela ini letaknya di sebelah kanan bawah, dibawah Jendela Pallete dan akan muncul hanya jika jendela Design yang dipilih.



Jendela Properties

Pilih komponen JComboBox di Jendela Design, Jendela Properties akan menampilkan properties dari JComboBox.

Pada bagian model di dalam Jendela Properties masukkan item Pelajar, Mahasiswa, Programmer, Technical Writer dan Tester. Setiap item dipisahkan dengan koma (,).

4. Merubah isi JList. Pilih JList di Jendela Design maka Jendela Properties untuk JList akan muncul. Di bagian model isikan item : Membaca, Olahraga, Trekking, Coding, Menonton Film, Bersepeda dan Mengajar. Setiap item dipisahkan dengan koma (,).

penting

Jendela Properties tidak hanya berisi properties dari komponen swing yang sedang aktif tetapi juga berisi Tab Events dan Tab Code.

Tab Events digunakan untuk memasukkan kode yang akan dieksekusi ketika event tertentu dikenakan terhadap komponen swing.

Tab Code digunakan untuk mendefinisikan kode apa yang harus dieksekusi dalam kondisi tertentu, misalnya setelah object komponen swing diinisialisasi.

5. Menangani pemilihan JComboBox. Klik kanan JComboBox di Jendela Design, kemudian pilih menu :

```
Events > Action > actionPerformed
```

Jendela Code akan terbuka, tambahkan code seperti di bawah ini :

```
private void cmbPekerjaanActionPerformed(
        java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
        lblPilihanPekerjaan.setText(
                "Pekerjaan anda : " +
                cmbPekerjaan.getSelectedItem());
    }
```

method getSelectedItem dari JComboBox digunakan untuk memperoleh item yang sedang di pilih dalam JComboBox.

6. Menangani event pemilihan dari JList. Event yang digunakan untuk menangani pemilihan item dari JList berbeda dengan JComboBox. JList akan mengaktifkan ListSelection event ketika user memilih item dalam JList. Untuk menangani event ini, lakukan langkah-langkah berikut :

   a) Klik kanan pada JList di dalam Jendela Design, kemudian pilih menu :

```
Events > ListSelection > valueChanged
```

   b) Dalam jendela kode yang ketik kode seperti berikut ini :

```
private void lstHobbyValueChanged(
       javax.swing.event.ListSelectionEvent evt) {
       // TODO add your handling code here:
       Object[] selectedItems =
              lstHobby.getSelectedValues();
       if(selectedItems  == null ||
              selectedItems.length == 0)
             txtPilihanHobby.setText("");
       else{
              StringBuffer strValues = new
StringBuffer();
              for(Object item : selectedItems){
                 strValues.append(item.toString() + ", ");
            }
       txtPilihanHobby.setText(
              strValues.substring(0, strValues.length() -
              2));
       }
}
```

Catatan :

- Method getSelectedValues dari JList mengembalikan item-item yang terpilih.

penting

JMenuBar dan JToolBar hanya bisa ditambahkan ke dalam JFrame.

JMenuItem adalah struktur terluar dari Menu yang tidak bisa mempunyai child.

JToolBar pada umumnya menampung JButton yang diberi Icon dan mempunyai background transparan.

Bekerja dengan Menu, Popup Menu dan Toolbar

Menu, Popup menu dan Toolbar digunakan untuk melakukan navigasi dalam aplikasi. dengan ketiga komponen itu navigasi dalam aplikasi menjadi lebih fleksibel dan mudah digunakan oleh user. Menu dan Toolbar pada umumnya diletakkan di bagian atas dari aplikasi agar mudah ditemukan oleh user. Sedangkan Popup Menu bisa muncul di mana saja sesuai dengan konteks aplikasi.

JMenuBar adalah class yang digunakan untuk menampung JMenu. JMenu dapat menampung satu atau lebih JMenuItem. JMenuItem merupakan bagian terluar dari struktur menu yang tidak bisa mempunyai child. JSeparatordigunakan untuk memisahkan antara satu menu item dan menu item yang lain. Jika didalam menu terdapat sub menu, gunakan JMenu untuk menampung sub menu tersebut. Selain JMenuItem, JMenu juga dapat menerima class JCheckBoxMenuItem dan JRadioButtonMenuItem.

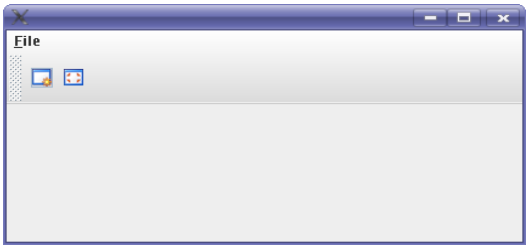
JPopupMenu mempunyai banyak kesamaan dibandingkan dengan JMenuBar. Perbedaan utamanya adalah : JMenuBar hanya bisa

berada di atas sebuah jendela JFrame. Sedangkan JPopupMenu bisa muncul di mana saja sesuai dengan konteks dari aplikasi.

Perbedaan lainnya terletak di dalam penggunaan umum keduanya. JMenuBar berisikan menu/instruksi yang bersifat umum dan berlaku untuk semua keadaan dalam aplikasi. Sedangkan JPopupMenu akan mempunyai menu/instruksi yang berbeda-beda berdasarkan dari konteks aplikasi. Oleh karena itu JPopupMenu terkadang disebut juga sebagai konteks menu.

Toolbar memberikan cara yang lebih praktis dibandingkan menu, bahkan bisa dikatakan bahwa toolbar adalah cara cepat untuk mengakses menu. Oleh karena itu, setiap item dalam toolbarbiasanya juga tersedia dalam menu. Pada umumnya toolbar diwakili hanya dengan gambar/icon yang melambangkan perintah dari toolbarnya. Di internet banyak tersedia toolbar icon gratis yang dapat kita gunakan.

Berbeda dengan JMenuBar dan JPopupMenu yang hanya bisa menerima menu item, JToolBar dapat menampung JButton atau control lainya. Seperti contohnya : JCheckBox, JRadioButton, JtoggleButton dan lainya. Normalnya, JToolBar akan diisi dengan JButton yang dihilangkan text-nya dan diganti dengan icon. Kita juga perlu merubah dekorasi JButton agar tampilannya terlihat cantik dan menarik.



Contoh program dengan Menu, Popup Menu dan Toolbar

Untuk membuat program seperti di atas, ada beberapa tahap yang perlu dilakukan. Tahap pertama adalah membuat Menu, yang kedua adalah membuat Popup Menu dan yang ketiga adalah membuat Toolbar.

Membuat Menu

Bekerja dengan Menu dalam Java melibatkan enam komponen swing, antara lain :

1. JMenuBar : Class yang menampung semua menu, hanya bisa menampung JMenu sebagai child.
2. JMenu : Class yang mempunyai child menu item. Biasanya JMenu ini yang jadi child langsung dengan JMenuBar

3. JMenuItem : Ujung dari menu, disinilah object Action diletakkan, sehingga ketika kita memilih JMenuItem ada action tertentu yang dijalankan aplikasi.
4. JCheckBoxMenuItem : Ujung dari menu, namun bentuknya lebih mirip JCheckBox.
5. JRadioButtonMenuItem : Ujung dari menu, namun bentuknya lebih mirip JButton.
6. JSeparator : pemisah antar JMenuItem atau antar JMenu

Setiap komponen menu mempunyai fungsi dan kegunaan masing-masing. Jika kita perhatikan, menu dalam aplikasi umumnya mempunyai shortcut untuk mengakses menu tanpa menggunakan bantuan mouse. Misalnya menu File biasanya dapat diakses menggunakan tombol ALT + F, menu Format dapat diakses dengan ALT + O. Fasilitas shortcut menu ini disebut sebagai Keyboard Mnemonic. File mempunyai mnemonic F, Format mempunyai mnemonic o, dan seterusnya. Pada umumnya tampilan mnemonic dari sebuah menu diwakili dengan huruf yang bergaris bawah.

penting

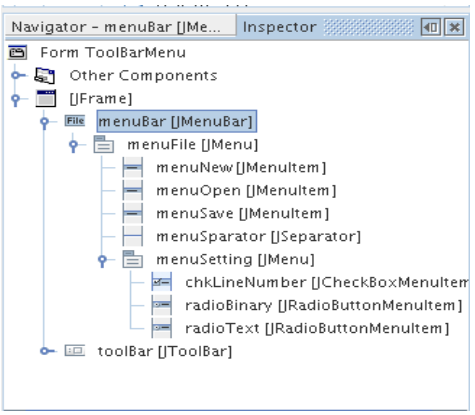
Jendela Inspector akan memperlihatkan semua komponen swing baik yang terlihat atau tidak terlihat dalam jendela design.

Jendela Inspector sangat berguna ketika kita bekerja dengan Menu.

Proses penambahan, pengurangan dan pengaturan posisi menu semua dilaksanakan dari Jendela Inspector

Matisse mempunyai fasilitas yang sangat OK untuk bekerja dengan Menu, fasilitas drag-and-dropnya membantu banyak pekerjaan membangun aplikasi barbasis GUI secara signifikan.

Dalam program diatas kita akan membuat struktur menu sebagai berikut:



Struktur menu dari aplikasi

Ikuti langkah-langkah berikut ini untuk membuat struktur menu seperti diatas:

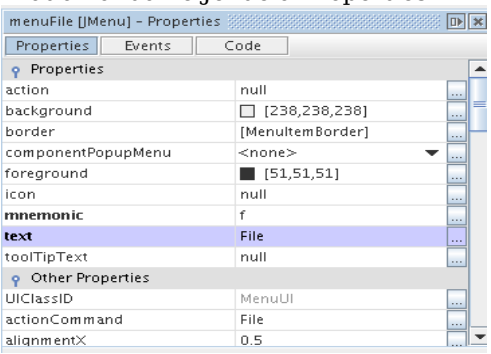
1. Buat sebuah class JFrame dan beri nama ToolbarMenu.java
2. Menambahkan JMenuBar ke dalam JFrame. Pilih komponen

Menu Bar dari Jendela Pallete kemudian klik JFrame di Jendela Design. Sebuah class JMenuBar akan ditambahkan di dalam JFrame. Ganti namanya menjadi menuBar.

3. Menambahkan JMenu ke dalam JMenuBar. Klik kanan JMenuBar yang baru saja kita buat di Jendela Inspector, kemudian pilih menu :

```
Add > JMenu
```

Ganti nama JMenu tersebut menjadi menuFile. Kemudian alihkan perhatian anda ke Jendela Properties



Jendela Properties dari class JMenu

Isi properti text dengan string “File”. Kemudian set isi properti mnemonic dengan string “f”, hal ini akan menyebabkan tampilanya menuFile menjadi File dan user dapat menekan tombol ALT + F untuk mengaktifkan menu menuFile.

4. Menambahkan JMenuItem. Langkah berikutnya adalah menambahkan JMenuItem ke dalam JMenu menuFile yang telah dibuat di langkah sebelumnya. caranya, klik kanan di JMenu menuFile di Jendela Inspector, kemudian pilih menu :

```
Add > JMenuItem
```

Tambahkan berturut-turut menuNew, menuOpen dan menuSave. Pilih JMenuItem dari Jendela Inspector, kemudian untuk masing-masing JMenuItem set text dan mnemonic yang sesuai dari Jendela Properties.

5. Menambahkan JSeparator. Dalam struktur menu yang bagus, menu yang mempunyai fungsi serupa diletakkan dalam urutan berderdekatan dan dipisahkan dengan separator (pemisah). Langkah menambahkan JSeparatortidak berbeda dengan langkah menambahkan JMenuItem, klik kanan di JMenu menuFile kemudian pilih menu:

```
Add > JSeparator
```

6. Menambahkan JMenu. Berikutnya kita akan menambahkan JMenu baru ke dalam JMenu menuFile. JMenu yang baru ini akan bertindak sebagai sub menu. Caranya juga sama : klik kanan di JMenu menuFile kemudian pilih menu :

```
Add > JMenu
```

   Beri nama menuSetting, set text dan mnemonic yang sesuai pada Jendela Properties.

7. Menambahkan JCheckBoxMenuItem. Perilaku JCheckBoxMenuItem tidak berbeda jauh dengan JCheckBox biasa, bedanya hanyalah JCheckBoxMenuItem berada dalam struktur menu. Cara menambahkan JCheckBoxMenuItem sama dengan komponen lain : klik kanan JMenu menuSetting kemudian pilih menu :

```
Add > JCheckBoxMenuItem
```

   Beri nama chkLineNumber, set text dan mnemonic yang sesuai pada Jendela Properties.

   JCheckBoxMenuItem sedikit sepesial dibandingkan dengan JMenuItem, karena JCheckBoxMenuItem memiliki properties selected. Properties selected ini digunakan untuk menentukan apakah JCheckBoxMenuItem dalam keadaan terpilih atau tidak.

8. Menambahkan JRadioButtonMenuItem. Dalam contoh ini kita akan mempunyai dua buah JRadioButtonMenuItem, radioBinary dan radioText. Keduanya dibuat dengan langkah yang sama dengan komponen lain, klik kanan di JMenu menuSetting, kemudian pilih menu :

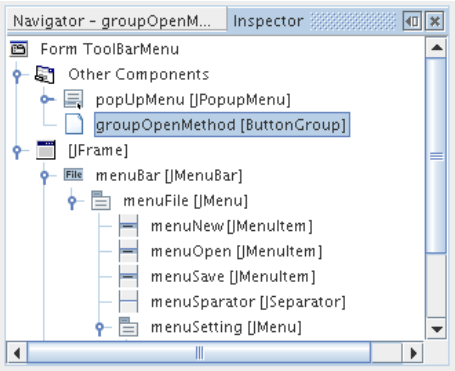
```
Add > JRadioButtonMenuItem
```

   Set text dan mnemonic yang sesuai dari Jendela Properties.

9. Menambahkan ButtonGroup. Seperti halnya JRadioButton, JRadioButtonMenuItem juga memerlukan ButtonGroup agar hanya satu buah JRadioButtonMenuItem yang bisa dipilih. Cara menambahkan ButtonGroup sangat mudah, klik item ButtonGroup dari Jendela Pallete kemudian klik Jendela Design, maka otomatis ButtonGroup akan ditambahkan. Ganti namanya menjadi groupOpenMethod.

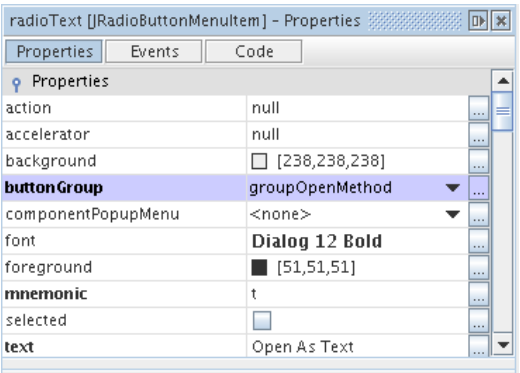
   Dalam Jendela Inspector, ButtonGroup yang baru dibuat tadi akan berada dalam kategori Other Components, seperti terlihat dalam gambar di bawah ini :

\



ButtonGroup berada dalam kategori Other Components

10. Menambahkan JRadioButtonMenuItem ke dalam ButtonGroup. Pilih masing-masing JRadioButtonMenuItem dari Jendela Inspector, kemudian perahatikan Jendela Properties dari JRadioButtonMenuItem tersebut, pada bagian groupButton pilih item groupOpenMethod, seperti terlihat dalam gambar di bawah ini :



Properties dari JRadioButtonMenuItem

11. Compile dan jalankan class ToolbarMenu.java. Klik kanan class ToolbarMenu dari Jendela Design kemudaian pilih menu Run File atau tekan tombol SHIFT + F6.

Bekerja dengan Menu menggunakan Matisse sangatlah menyenangkan dan produktif. Hal ini berbeda sekali jika harus mengetik satu demi satu kode untuk menyusun struktur menu seperti contoh program diatas.

Membuat Popup Menu menggunakan Matisse juga sama mudahnya. Hanya saja kita harus menentukan dimana dan dengan

cara apa popup menu itu muncul, apakah dengan penekanan tombol tertentu dari keyboard atau ketika tombol mouse ditekan.

Membuat Popup Menu

Popup menu pada dasarnya tidak jauh berbeda dibandingkan dengan menu biasa, hanya saja popup menu dapat muncul di mana saja, tidak hanya di bagian atas JFrame seperti halnya JMenuBar. Selain itu kita harus menentukan kapan popup muncul, pada umumnya popup akan muncul ketika user melakukan klik kanan terhadap suatu komponen swing. Misalnya, ketika suatu table di klik kanan terdapat popup yang muncul, dan sebagainya.

Popup menu terutama digunakan sebagai “context sensitive menu”, dimana menu yang ditampilkan oleh popup menu tergantung konteks dari aplikasi, semisal : komponen apa yang dikenai aksi klik kanan, bagaimana keadaan data dalam komponen tersebut dan sebagainya.

Aplikasi yang memerlukan interaksi yang sangat intens dengan user sebaiknya menggunakan popup menu untuk memudahkan user mengakses action tertentu. Hal ini jauh lebih praktis dibanding user harus mengakses menu dalam JMenuBar di bagian atas JFrame.

Popup menu dalam contoh program diatas akan muncul ketika user melakukan klik kanan terhadap JFrame. menu yang ditampilkanya pun hanya ada tiga buah: cut, copy dan paste.

Ikuti langkah-langkah beritkut ini untuk membuat Popup menu :

1. Buka class ToolbarMenu.java, yang telah dibuat dalam langkah sebelumnya, dalam Jendela Design.
2. Klik Jendela Pallete dan pilih JPopupMenu, kemudian klik Jendela Design. Secara otomatis JPopupMenu akan ditambahkan dalam class ToolbarMenu.java. JPopupMenu tidak terlihat dalam Jendela Design, namun anda bisa mengkasesnya melalui Jendela Inspector.
3. Menambahkan JMenuItem. Seperti halnya JMenuBar, JPopupMenu dapat memiliki child berupa JMenu, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem dan JSeparator. Menambahkan JMenuItem ke dalam JPopupMenu sangat sederhana, caranya : klik kanan pada JPopupMenu di Jendela Design, kemudian pilih menu :

```
Add > JMenuitem
```

   Ganti nama objectnya menjadi menuCut, beralihlah ke Jendela Properties kemudian set text dan mnemonic yang sesuai.

   Lakukan langkah ini untuk JMenuItem yang lain, menuCopy

dan menuPaste.

4. Memunculkan JPopupMenu. Ketika tombol kanan mouse di klik diatas JFrame, JPopupMenu akan tampil. Agar behavior tersebut berjalan, kita perlu menangani event mouseClick terhadap JFrame. Caranya :

   a) Klik kanan JFrame di Jendela Design, kemudian pilih menu :

```
Events > Mouse > mouseClicked
```

   b) Di dalam jendela source yang terbuka masukkan kode berikut ini :

```
private void formMouseClicked(
       java.awt.event.MouseEvent evt) {
       // TODO add your handling code here:
       if(evt.getButton() == MouseEvent.BUTTON3){
            popUpMenu.show(
                    (Component)evt.getSource(),
                    evt.getX(),evt.getY());
        }
 }
```

   Kondisi if diatas digunakan apakah tombol yang diklik mouse adalah tombol sebelah kanan, jika nilai kembalian method getButton sama dengan nilai BUTTON3 maka benar tombol kanan yang ditekan.

   Method show digunakan untuk memunculkan popup menu, parameter pertama diisi dengan Component dimana nantinya popup menu akan ditampilkan, sedangkan parameter kedua dan ketiga diisi dengan letak koordinat popup menu akan ditampilkan.

5. Simpan file ToolbarMenu.java, compile dan jalankan. Kemudian coba munculkan popup menu dengan mengklik kanan JFrame.

Popup menu sangat berguna jika aplikasi yang kita kembangkan membutuhkan interaksi yang intensif dengan user. Popup menu menyediakan cara mudah untuk mengakses menu/action yang sesuai dengan konteks dari aplikasi.

Membuat Toolbar

Toolbar memberikan dimensi lain dalam mengakses menu dbandingkan menu ataupun popup menu. Pada umumnya Toolbar merupakan cara singkat untuk mengakses menu. Menu yang diwakili toolbar adalah menu yang bersifat umum dan tidak terikat pada konteks tertentu.

Kegunaan lain dari toolbar adalah mempercantik tampilan aplikasi, karena toolbar biasanya adalah tombol yang didekorasi dengan

icon yang menarik. Selain itu toolbar juga memberikan kesempatan kepada user untuk mengkustomisasi tampilan dari aplikasi. Karena layout toolbar sangat fleksibel, user bisa memindah-mindahkan letak toolbar di dalam aplikasi, di atas, dibawah atau disamping, atau bahkan mengambang (floating) diatas jendela yang sedang aktif.

Dalam contoh program diatas kita akan membuat sebuah JToolBar dengan dua buah JButton yang telah didekorasi dengan icon cantik. Icon yang digunakan banyak tersedia di internet, format file yang dipilih adalah .png, karena format file ini paling bagus dalam menangani transparasi komponen.

Sebelum mulai membuat JToolBar, kita perlu mempersiapkan terlebih dahulu icon yang akan digunakan sebagai dekorasi JButton. Ikuti langkah-langkah berikut ini :

1. Buatlah sebuah java package baru untuk menampung semua icon yang akan digunakan. caranya klik kanan di jendela Projects bagian nama project, pilih menu :

```
New > Java Package
```

   Beri nama images untuk java package yang baru saja kita buka.

2. Memasukkan Icon ke dalam package. Untuk memasukkan image ke dalam package kita perlu tahu dimana project disimpan, misalkan project disimpan dalam folder :

```
c:\javaswing
```

   Buka file explorer, kemudian navigasi ke folder

```
c:\javaswing\src\images
```

   Copy semua icon yang diperlukan ke dalam folder diatas.

3. Build project. Langkah ini diperlukan untuk mengcompile semua file .java menjadi file .class. Selain itu proses ini juga akan mengkopi file selain file .java (termasuk file icon) ke dalam folder build\classes. Jika proses ini tidak dilaksanakan, maka ketika program dijalankan, file icon tidak akan ditemukan dan program menjadi error .

penting

Build Project akan berhasil jika tidak ada satupun error dalam kode program.

Sebelum melakukan build project pastikan terlebih dahulu tidak ada error dalam kode.

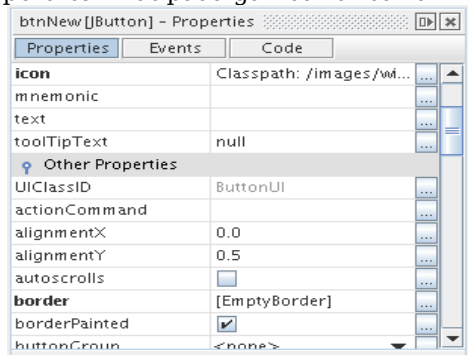
Lakukan build setiap kali menambahkan file non-java ke dalam folder source file. Agar file tersebut ikut tercopy ke dalam folder build\classes

Setelah proses persiapan selesai, lakukan langkah-langkah berikut ini untuk membuat Toolbar :

1. Buka class ToolbarMenu.java yang sudah dibuat di langkah sebelumnya.
2. Buat sebuah object JToolBar, caranya : klik item JToolBar dari Jendela Pallete, kemudian klik JFrame di Jendela Design. Secara otomatis sebuah object JToolBar akan dimasukkan ke dalam JFrame. Ganti namanya menjadi toolBar.

3. Menambahkan JButton dalam JToolBar. Klik item JButton dalam Jendela Pallete kemudian klik komponen JToolBar yang baru saja kita buat tadi. JButton baru akan diletakkan diatas JToolBar, ganti nama JButton tersebut menjadi btnNew. Letakkan lagi satu buah JButton diatas JToolBar dan beri nama btnMaximize.

4. Mendekorasi Tampilan JButton. Agar tampilan JButton terlihat cantik, kita perlu mengeset beberapa nilai dari properti JButton, seperti terlihat pada gambar di bawah ini :

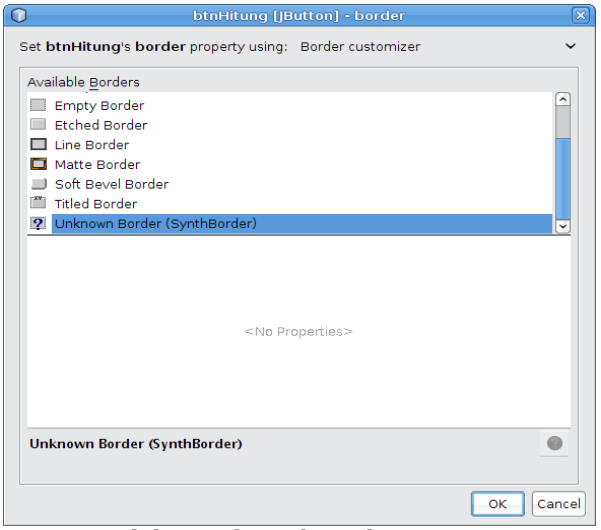


Jendela Properties JButton

a) Text, hapus nilai textnya.

b) Border, pilih bordernya menjadi empty border dan set nilai bordernya menjadi [5,5,5,5]. Tujuan pemberian empty border ini agar tombol berukuran lebih besar dibandingkan dengan icon yang akan digunakan nanti, dan setiap mouse melewati JButton, ada efek transisi yang cantik.

   Untuk mengedit border dari JButton, Matisse menyediakan Jendela Border untuk memilih border yang kita inginkan untuk Jbutton. Border yang dipilih bisa single border, atau composite border yang terdiri dari beberapa border.



Jendela Border Editor dari JButton

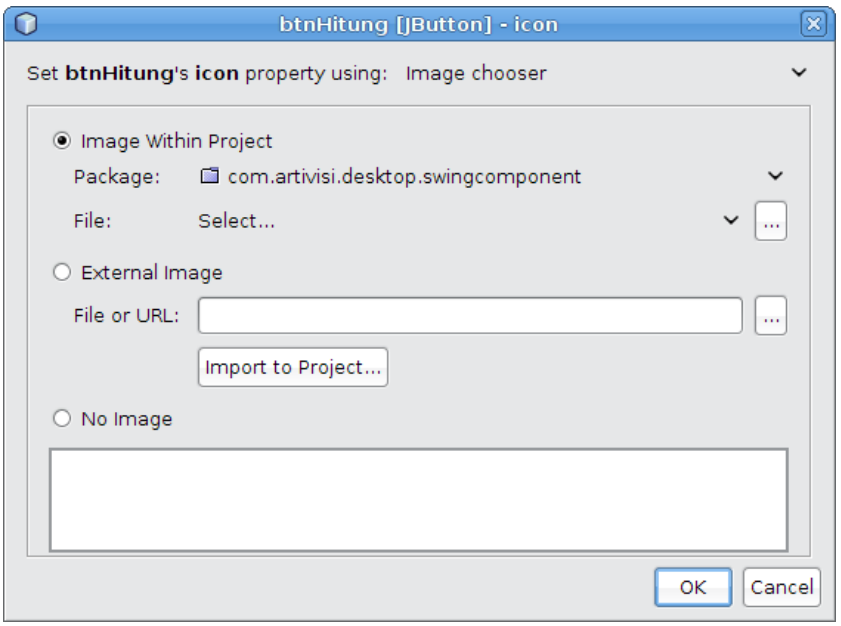
c) Opaque, uncheck nilai opaque. Bertujuan agar tombolnya berwarna transparan, sehingga mempunyai warna background yang sama dengan background JToolBar.

d) Icon, ganti iconya dengan icon yang telah disiapkan. Untuk memasukkan icon ke dalam JButton, tekan tombol di samping pilihan Icon di dalam Jendela Properties, kemudian akan muncul Dialog Icon Editor seperti di bawah ini :

Jendela icon editor

Pilih radio button Classpath, kemudian tekan tombol Select File dan pilih salah satu icon yang telah disiapkan. Tekan OK. Lakukan langkah-langkah yang sama terhadap JButton yang lain.

5. Compile dan jalankan class ToolbarMenu untuk melihat hasilnya.



Membuat Dialog dan JFileChooser

Dialog memerankan peran yang penting dalam aplikasi berbasis desktop. Interaksi antara user dengan aplikasi terkadang tidak berjalan dengan baik karena user memberikan aksi yang tidak valid kepada aplikasi. Ketika hal tersebut terjadi, aplikasi harus memberitahukan kepada user apa yang telah terjadi dan bagaimana seharusnya user memperbaikinya. Model interaksi seperti ini tepat dilaksanakan menggunakan dialog.

Skenario lain adalah ketika aplikasi memerlukan input dari user agar aplikasi bisa terus melaksanakan tugasnya, misalnya meminta konfirmasi apakah user yakin akan melaksanakan sebuah aksi penting terhadap aplikasi seperti delete, update atau add data.

Dialog juga memberikan pembatasan kepada user, sebelum dialog selesai diproses, user tidak akan bisa berinteraksi dengan bagian aplikasi lainya. Dialog mencegah hal ini terjadi dengan memastikan bahwa jendela yang bisa diaktifkan hanyalah jendela dialog, sedangkan jendela aplikasi yang lain tidak dapat diaktifkan selama jendela dialog masih aktif.

Aplikasi sangat sering menggunakan dialog untuk berinteraksi dengan user, tetapi jenis interaksinya selalu seragam dan berulang-

ulang. Swing menyediakan dialog yang didesign untuk keperluan yang sering muncul dalam aplikasi, seperti JOptionPane dan JFileChooser. Swing juga menyediakan class JDialog jika kita ingin membuat dialog custom sesuai keinginan kita.

Membuat pre-defined dialog dengan JOptionPane

JOptionPane menyediakan beberapa dialog yang siap pakai dan sering digunakan dalam aplikasi. JOptionPane sangat memudahkan kita dalam meminta user suatu input tertentu atau memberitahu user apa yang terjadi dalam aplikasi.

JOptionPane mempunyai banyak static method untuk menampilkan popup dialog dengan mudah. Terdapat empat method utama yang dapat kita gunakan sebagai landasan membuat dialog. Keempat method tersebut secara rinci digambarkan dalam table berikut ini:

Method	Deskripsi
showConfirmDialog	Meminta konfirmasi daru user, seperti yes/no/cancel
showInputDialog	Meminta input dari user, baik berupa input text menggunakan JTextField maupun pilihan menggunakan JComboBox
showMessageDialog	Memberitahukan user tentang apa yang baru saja terjadi
showOptionDialog	Gabungan dari ketiga jenis dialog diatas

Table method JOptionPane

Swing juga menyediakan method showInternalXXX yang digunakan jika kita bekerja dengan JInternalFrame.

Parameter dari keempat method tersebut mengikuti pola yang konsisten. Terurut dari kiri ke kanan, berikut ini parameter-parameter yang bisa diterima oleh method-method dalam class JOptionPane:

1. parentComponent

   Mendefisikan komponen yang akan menjadi parent dari dialog box ini. Frame dari parent component tersebut akan menjadi frame dari dialog dan dialog akan ditampilkan di tengah-tengah parent component. Jika nilai dari parentComponent diset null, maka dialog akan menggunakan frame default dan dialog akan diletakkan ditengah-tengah layar monitor (tergantung L&F).

2. message

   Pesan yang deskriptif menerangkan perihal dialog yang muncul. Pada umumnya message berupa pesan String yang akan diletakkan dalam dialog, namun jenis object lain juga diijinkan digunakan sebagai message. Object-object yang

diijinkan akan diperlakukan berbeda, object-object tersebut antara lain

a) Object[]

   Setiap object akan ditampilkan dalam dialog berurut dari atas ke bawah. Aturan ini berlaku rekursif untuk semua object didalam array.

b) Component

   Jika object yang dimasukkan sebagai message bertipe Component, maka Component tersebut akan ditampilkan ditengah-tengah dialog.

c) Icon

   Icon akan dimasukkan ke dalam sebuah JLabel kemudian ditampilkan di sebelah kiri dari dialog.

d) others

   Object lainya akan ditampilkan dalam dialog dengan mengambil nilai kembalian dari method toString dari setiap object.

3. messageType

   Mendefisikan jenis dari pesan. Pada umumnya memberikan custom icon untuk setiap jenis pesan. Setiap L&F manager akan memperlakukan setiap jenis pesan dengan berbeda, namun perbedaanya tidak akan terlalu mencolok. Pilihan yang mungkin dan icon yang mewakilinya adalah:

   a) ERROR_MESSAGE 
   b) INFORMATION_MESSAGE 
   c) WARNING_MESSAGE 
   d) PLAIN_MESSAGE (tanpa icon)

4. optionType

   Mendefisikan tombol-tombol yang akan ditampilkan di bagian bawah dari dialog.

   a) DEFAULT_OPTION
   b) YES_NO_OPTION
   c) YES_NO_CANCEL_OPTION
   d) OK_CANCEL_OPTION

   Namun kita tidak dibatasi untuk hanya menggunakan empat jenis set tombol diatas, kita dapat mendefisikan tombol-tombol

yang akan muncul sesuai kebutuhan.

5. options

   Deskripsi yang lebih detail dari set tombol yang digunakan dialog. Nilai yang lazim adalah sebuah array String berisi text yang akan ditampilkan di setiap tombol. Namun Object lain juga dapat diterima, antara lain:

   a) Component

      Component akan diletakkan dalam baris tombol secara langsung.

   b) Icon

      Sebuah JButton akan dibuat dan didekorasi dengan icon ini.

   c) other

      Object dengan tipe selainnya akan dirubah ke dalam bentuk String dengan mengambil nilai kembalian dari method toString dari object tersebut.

6. icon

   Icon yang digunakan untuk mendekorasi dialog. Jika icon ini didefinisikan maka akan menimpa icon default yang didefinisikan oleh messageType.

7. title

   Judul dari dialog yang diletakkan di bagian paling atas dari dialog.

8. initialValue

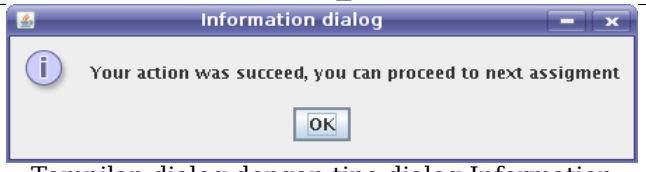
   Nilai default dari pilihan yang mungkin ada dalam dialog.

Untuk lebih jelasnya, berikut ini beberapa contoh kode penggunaan JOptionPane beserta hasil tampilanya :

```
JOptionPane.showMessageDialog(null,
        "Simple plain dialog","Plain dialig",
        JOptionPane.PLAIN_MESSAGE);
```

Tampilan dialog sederhana

```
JOptionPane.showMessageDialog(null,
       "Your action was succeed, " +
       "you can proceed to next assigment",
       "Information dialog",
       JOptionPane.INFORMATION_MESSAGE);
```

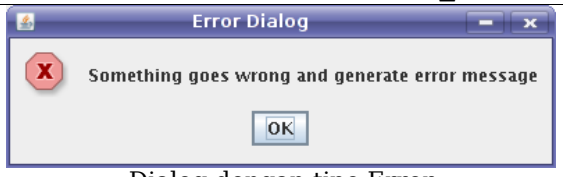


Tampilan dialog dengan tipe dialog Information

```
JOptionPane.showMessageDialog(null,
       "You neet to be sure to do this action!",
       "Dialog Peringatan", JOptionPane.WARNING_MESSAGE);
```

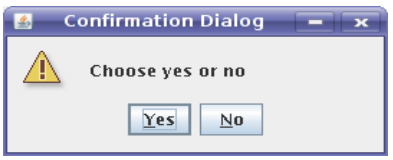
Dialog dengan tipe Warning

```
JOptionPane.showMessageDialog(null,
       "Something goes wrong and generate error message",
       "Error Dialog", JOptionPane.ERROR_MESSAGE);
```



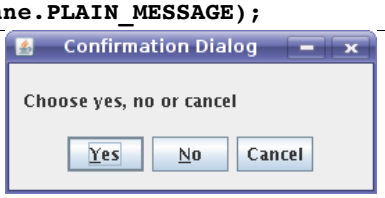
Dialog dengan tipe Error

```
JOptionPane.showConfirmDialog(null,
       "Choose yes or no","Confirmation Dialog",
       JOptionPane.YES_NO_OPTION,
       JoptionPane.WARNING_MESSAGE);
```



Option dialog dengan tipe Information dan pilihan YES_NO

```
JOptionPane.showConfirmDialog(null,
        "Choose yes, no or cancel","Confirmation Dialog",
        JOptionPane.YES_NO_CANCEL_OPTION,
        JOptionPane.PLAIN_MESSAGE);
```

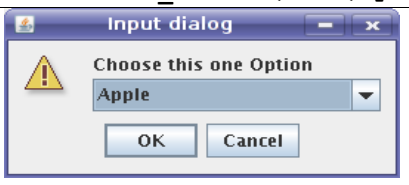


OptionDialog

```
JOptionPane.showInputDialog(null,
        "Input your name here","Input Dialog",
        JOptionPane.INFORMATION_MESSAGE);
```

InputDialog dengan tipe message Information

```
String[] options = {"Apple","Mango","Grape","Guava"};
JOptionPane.showInputDialog(null,
        "Choose this one Option","Input dialog",
        JOptionPane.WARNING_MESSAGE,null,options,"Apple");
```



InputDialog dialog dengan tipe Warning, Options berupa array of String dan initialValue = 'Apple'

Membuat JFileChooser

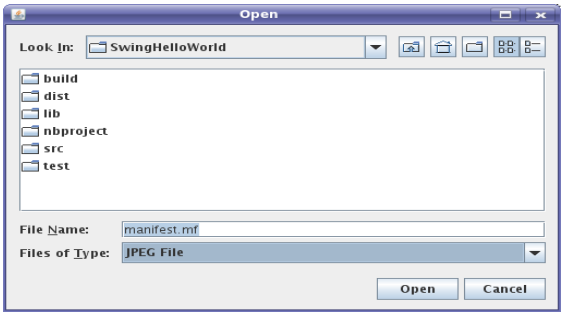
JFileChooser digunakan untuk bernavigasi dalam file system, kemudian memilih satu atau lebih file atau folder dari list file dan folder. JFileChooser pada dasarnya adalah pengembangan dari dialog yang dapat digunakan untuk memilih file. JFileChooser dapat digunakan sebagai dialog untuk menyimpan file atau untuk membuka file.

JFileChooser hanya memberikan fasilitas untuk memilih file atau folder, sedangkan mekanisme untuk menyimpan atau membuka file dilakukan sendiri menggunakan library I/O.

Aplikasi berikut ini adalah contoh penggunaan JFileChooser untuk membuka dan menyimpan file.

Contoh program menggunakan JFileChooser

Tampilan JFileChooser ketika tombol open ditekan adalah seperti di bawah ini :



Tampilan JFileChooser

Untuk membuat aplikasi diatas lakukan langkah-langkah berikut ini :

1. Buat class JFrame Form baru, beri nama Chooser.java
2. Masukkan dua buah JTextField : txtOpen dan txtSave, dua buah

Jbutton : btnOpen dan btn save, sebuah JLabel : lblStatus. Sesuaikan penataan komponen sesuai dengan gambar diatas.

3. Tambahkan sebuah object JFileChooser sebagai field dari class Chooser, beri nama chooser.

```
public class Chooser{
    JFileChooser chooser = new JFileChooser();
    //kode lain di sini
    }
```

4. FileNameExtentionFilter digunakan sebagai file filter dalam JFileChooser. Metode filteringnya adalah mencocokkan ekstensi file dalam file system dengan ekstensi yang ada dalam FileNameExtentionFilter. Contoh kode di bawah ini akan menyebabkan JFileChooser mempunyai pilihan "JPEG File", dan jika pilihan tersebut dipilih, maka file dengan ekstensi "jpg", "jpeg","JPG" atau"JPEG" saja yang akan ditampilkan oleh JFileChooser.

```
FileNameExtensionFilter JPEGFilter =
       new FileNameExtensionFilter(
              "JPEG File","jpg","jpeg","JPG","JPEG");
chooser.addChoosableFileFilter(JPEGFilter);
```

5. Set direktori yang akan dituju oleh JFileChooser. Untuk mengetahui dimana direktori aktif aplikasi, kita bisa menggunakan system property "user.dir". Kode berikut ini akan menyebabkan JFileChooser dibuka pada direktori aktif aplikasi :

```
String dir = System.getProperty("user.dir");
chooser.setCurrentDirectory(new File(dir));
```

6. Menghandle event penekanan tombol btnSave. Ketika tombol btnSave ditekan, chooser akan menampilkan dialog save file, kemudian mengambil nama file yang dipilih dan menampilkannya dalam txtSave, serta menampilkanya dalam lblStatus. Berikut ini kodenya :

```
private void btnSaveActionPerformed(ActionEvent evt) {
       // TODO add your handling code here:
       int ret = chooser.showSaveDialog(this);
       if(ret == JFileChooser.APPROVE_OPTION){
              File f = chooser.getSelectedFile();
              lblStatus.setText("Status : saving file" +
                     f.getAbsolutePath());
             txtSave.setText(f.getAbsolutePath());
       }
}
```

7. Menghandle penekanan tombol btnOpen. Kode untuk menangani penekanan tombol btnOpen mirip dengan kode untuk menangani penenakan tombol btnSave, perbedaanya adalah btnOpen akan menampilkan dialog open file, berikit ini kodenya :

```
private void btnBrowseActionPerformed(ActionEvent evt){
       // TODO add your handling code here:
       int ret = chooser.showOpenDialog(this);
       if(ret == JFileChooser.APPROVE_OPTION){
              File f = chooser.getSelectedFile();
              lblStatus.setText("Status : opening file" +
                     f.getAbsolutePath());
             txtOpen.setText(f.getAbsolutePath());
        }
}
```

8. Compile dan jalankan aplikasinya dengan menekan tombol SHIFT + F6

Bekerja dengan JOptionPane dan dengan JFileChooser sangat sederhana. Keduanya menggunakan modal dialog untuk mengambil input dari user. Modal dialog akan mencegah user mengakses bagian aplikasi lain sebelum dialog ditutup, atau dalam hal ini memutuskan pilihan apa yang diambil oleh user.

Masih banyak lagi komponen swing yang disediakan oleh JDK, anda tinggal melanjutkan membaca dari referensi yang diberikan modul ini pada bagian akhir untuk melanjutkan pembelajaran anda tentang Java desktop.

Konsep MVC

MVC adalah arsitektur aplikasi yang memisahkan kode-kode aplikasi dalam tiga lapisan, Model, View dan Control. MVC termasuk dalam arsitektural design pattern yang menghendaki organisasi kode yang terstruktur dan tidak bercampur aduk. Ketika aplikasi sudah sangat besar dan menangani struktur data yang kompleks, harus ada pemisahan yang jelas antara domain model, komponen view dan kontroler yang mengatur penampilan model dalam view.

Arsitektur MVC ini memungkinkan adanya perubahan dalam domain model tanpa harus mengubah code untuk menampilkan domain model tersebut. Hal ini sangat bermanfaat ketika aplikasi mempunyai domain model dan view komponen sangat besar dan kompleks.

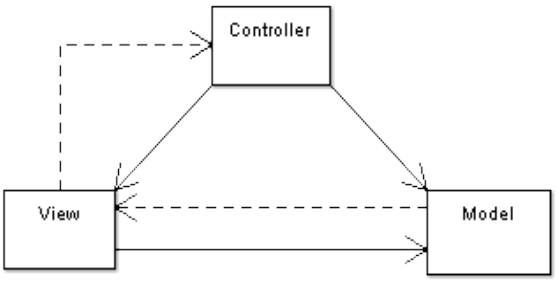


Diagram interaksi antar komponen dalam arsitektur MVC (Wikipedia.org)

Model adalah representasi dari object yang sedang diolah oleh aplikasi, dalam Java, model ini biasanya direpresesentasikan sebagai Java Bean. Java Bean adalah class Java biasa atau POJO (Plain Old Java Object). Syarat sebuah POJO dianggap sebagai Java Bean adalah :

1. Mempunyai constructor default, constructor yang tidak mempunyai parameter.
2. Semua field-field yang bisa diakses dilengkapi dengan getter dan setter method.

Lebih jelasnya lihat kode dari class Person di bawah ini :

```
public class Person {
    private int id;
    private String name;
    private String email;
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setNama(String name) { this.name = name;}
//getter dan setter method untuk field lainnya di sini
}
```

Kode diatas adalah representasi Model dalam Java untuk Entity Person. Beberapa orang terkadang salah mengartikan model ini sebagai data akses domain. Dimana data dari sumber data, misalnya database, diambil dan diolah. Pada hakekatnya Model adalah representasi data dari object sebenarnya, bukan kumpulan kode untuk mengakses data dari database.

Pendekatan terbaik adalah memisahkan kode untuk melakukan akses sumber data ke dalam lapisan tersendiri, lapisan ini biasanya disebut sebagai service. Service diimplementasikan dalam bentuk class-class yang disebut sebagai manager, misalnya SQLManager, PrintManager, ReportManager, XMLManager, WebServiceManager dan seterusnya. Dengan begitu kode akan menjadi lebih rapi dan terstruktur. Manfaat paling terasa adalah kemudahan pencarian kesalahan dan penambahan modul-modul baru tidak harus merombak seluruh struktur aplikasi.

View adalah komponen untuk merepresentasikan Model dalam bentuk visual. Semisal komponen swing, seperti : JTable, JList, JComboBox dan sebagainya. View juga bertanggung jawab untuk menangkap interaksi user terhadap sistem, semisal : klik mouse, penekanan tombol keyboard, barcode scanning dan sebagainya.

Controller sebenarnya hanya sekumpulan kode-kode untuk mensinkronisasi keadaan Model dan View. Jika ada perubahan data dari Model, Controller harus mengupdate tampilan View. Dan sebaliknya jika user memberikan event terhadap View, Controller harus mengupdate Model sesuai dengan hasil interaksi user terhadap View.

Model dalam Komponen Swing

Sebagaian besar komponen swing mempunyai model. JButton mempunyai model yaitu ButtonModel yang memegang 'state' dari JButton – apa keyboard mnemonicnya, apakah JButton tersebut sedang dipilih atau tidak dan seterusnya. Ada pula komponen swing yang mempunyai lebih dari satu model. JList mempunyai

ListModel yang memegang isi dari JList dan ListSelectionModel untuk mencatat item JList yang sedang dipilih.

Pada banyak kasus normal kita tidak perlu pusing memikirkan model ini. Semisal kita tidak perlu memikirkan model dari JButton karena pada kasus umum kita tidak perlu memodifikasi model dari JButton.

Lalu, kenapa model komponen swing dibuat? Alasan utamanya adalah fleksibilitas untuk menentukan bagaimana data disimpan dan diambil dari komponen swing. Misalnya kita mempunyai aplikasi spreadsheet yang menggunakan komponen JTable, karakteristik utama spreadsheet adalah banyak cell yang kosong, dengan begitu kita bisa memilih model data yang sesuai dengan karakteristik tersebut.

Contoh lainnya adalah JTable yang digunakan untuk menampilkan data dari database dengan jumlah baris luar biasa banyak. Kita bisa mengatur agar tampilan JTable dibuat halaman-per-halaman dalam menampilkan baris data, tidak semua data ditampilkan dalam satu halaman, hal ini ditujukan untuk efisiensi dan mempertahankan agar aplikasi tetap responsif walau bekerja dengan data yang besar.

Model dalam komponen swing juga mempunyai keuntungan lain, yaitu tidak perlu ada dua data terpisah, untuk struktur data aplikasi dan untuk komponen swing.

Kegunaan Model yang cukup penting juga adalah adanya konsep event-listener, dimana jika terjadi event perubahan data dalam model, semua listener yang terdaftar dalam model tersebut akan diberitahu dan tindakan yang tepat dapat diambil untuk menangani event yang muncul. Sebagai contoh, untuk menambahkan item dalam JList kita bisa memanggil method addItem dari JLIst. Penambahan item dalam JList ini akan mengakibatkan ListModel memicu event dalam JList dan listener lainnya. Komponen swing—dalam hal ini JList—akan diupdate tampilanya untuk merefleksikan perubahan item dalam ListModel.

Walaupun terkadang banyak yang menyebut arsetektur komponen swing sebagai MVC, tetapi pada dasarnya arsitektur komponen swing tidak sepenuhnya MVC. Komponen swing secara umum dibuat agar View dan Controller diletakkan dalam satu tempat (class) yaitu class UI yang disediakan oleh Look-and-Feel. Arsitektur komponen swing lebih tepat disebut sebagai “Arsitektur dengan Model yang terpisah”.

Selanjutnya kita akan membahas beberapa model yang seringkali harus kita kustomisasi sesuai dengan kebutuhan. Sedangkan model yang nyaris tidak pernah kita rubah—ButtonModel—tidak dibahas dalam bagian ini.

TableModel

TableModel adalah class model yang paling sering dikustomisasi. Karakteristik data dari JTable yang berbentuk koleksi data dua dimensi membutuhkan perhatian khusus agar efisien digunakan dalam aplikasi. Jika kita tidak hati-hati, maka aplikasi kita bisa menjadi sangat lambat dan tidak efisien.

TableModel adalah interface yang digunakan oleh JTable untuk mendefinisikan ciri-ciri dari data tabular yang akan ditampilkan oleh JTable. Misalnya : jumlah kolom, nama kolom, class dari object dalam kolom, jumlah baris dan nilai setiap cell. Dengan adanya data-data ini JTable dapat secara efisien menentukan bagaimana menampilkan data tersebut.

Berikut ini adalah kode untuk menampilkan koleksi object Person. Class ArrayList<Person> adalah implementasi dari generics, konsep dalam Java yang digunakan untuk mendefinisikan isi dari koleksi. ArrayList<Person> artinya adalah membuat sebuah object koleksi ArrayList yang harus diisi oleh object Person dan tidak bisa diisi oleh object lainya, misalnya String.

```
public class PersonTableModel extends AbstractTableModel{
private List<Person> persons;
    public PersonTableModel(List<Person> persons) {
        this.persons = persons;
    }
    public int getRowCount() {
        return persons.size();
    }
    public int getColumnCount() {
        return 3;
    }
    public Object getValueAt(int rowIndex, int
       columnIndex) {
        Person p = persons.get(rowIndex);
        switch(columnIndex){
            case 0 : return p.getId();
            case 1 : return p.getName();
            case 2 : return p.getEmail();
            default : return "";
        }
    }
```

```
   @Override
    public String getColumnName(int column) {
        switch(column){
            case 0 : return "ID";
            case 1 : return "NAME";
            case 2 : return "EMAIL";
            default : return "";
        }
    }
}
```

Yang perlu diperhatikan bahwa dalam AbstracTableModel, method isCellEditable selalu mengembalikan nilai false, artinya semua cell tidak dapat diedit. Kemudian method setValueAt adalah method kosong belaka, artinya jika kita memanggil method ini tidak akan terjadi apa-apa.

Class kedua adalah DefaultTableModel yang telah mengimplementasi semua method abstract dari interface TableModel. Representasi data DefaultTableModel menggunakan dua jenis data tabular, yaitu array dua dimensi, Object[][], dan Vector dari Vector, Vector<Vector<Object>>. Jika kita mempunyai struktur data selain kedua jenis tersebut kita harus melakukan konversi data ke dalam salah satu bentuk struktur data tersebut. Cara yang lebih cerdas adalah mendefinisikan sendiri class yang mengimplement interface TableModel seperti class CustomerTableModel diatas.

Setelah TableModel selesai didefinisikan kita tinggal memanggil method setTableModel dari object JTable, atau membuat object JTable baru menggunakan constructor yang menerima argumen TableModel. Contohnya seperti potongan kode di bawah ini :

```
JTable table = new JTable(new DefaultTableModel());
JTable table1 = new JTable();
table1.setModel(new DefaultTableModel());
```

ListModel

JList adalah komponen swing yang mempunyai dua model sekaligus, ListModel dan ListSelectionModel. ListModel digunakan untuk mendefinisikan item/element yang dikandung oleh JList. Sedangkan ListSelectionModel digunakan untuk mendefinisikan bagaimana representasi data jika terjadi proses pemilihan di JList.

Seperti halnya TableModel, ListModel mempunyai dua class yang mengimplement ListModel, AbstractListModel dan DefaultListModel. Kita bisa menggunakan salah satu dari tiga tipe

tersebut untuk membuat object ListModel. Cara pertama dengan membuat class baru yang mengimplement ListModel. Cara kedua dengan membuat class baru yang menextends AbstractListModel dan cara ketiga dengan langsung menggunakan DefaultListModel.

Struktur data JList tidak terlalu rumit seperti JTable, dan pada umumnya, cukup hanya dengan menggunakan DefaultListModel sudah memenuhi sebagaian besar kebutuhan penggunaan JList.

Berikut ini contoh bagaimana membuat ListModel untuk data customer, contoh ini menggunakan cara kedua untuk membuat obejct ListModel, yaitu dengan cara membuat class baru yang mengextends AbstractListModel :

```
public class CustomerListModel extends AbstractListModel{
	private ArrayList<Person> customer =
		new ArrayList<Person>();
	public CustomerListModel(List<Person> cust){
		customers.addAll(cust);
	}
	public Object getValueAt(int index) {
		return customers.get(index);
	}
	public int getSize() { return customers.size(); }
}
```

Implementasi ListModel sangat mudah dan tidak serumit TableModel, namun implementasi dari ListSelectionModel sangat rumit, karena kita harus mengimplementasi dua puluh buah method. Lebih baik menggunakan implementasi standard dari ListSelectionModel yaitu DefaultListSelectionModel.

Menangani Event

Event dan Listener adalah implementasi dari pattern Observer dalam Java. Pattern Observer sangat berguna digunakan untuk mendesign komunikasi yang konsisten antara object yang berdiri sendiri dan object-object yang bergantung padanya.

Observer design pattern melibatkan dua object utama, object pertama berlaku sebgai Subject dan object lainya berlaku sebagai Observer. Object Subject merupakan pusat perhatian dari object Observer, perubahan keadaan dari object Subject selalu dipantau oleh Observer.

Observer dapat melakukan register-unregister terhadap Subject. Jika Observer tertarik dengan perilaku dan keadaan dari Subject, Observer dapat meregister dirinya kepada Subject. Begitu juga sebaliknya jika Observer tidak tertarik terhadap keadaan atau perilaku Subject, Observer tidak perlu melakukan resgistrasi atau kalau sudah terlanjur reguister dapat melakukan unregister.

Subject mempunyai banyak aspek perilaku dan keadaan yang dapat dipantau oleh Observer. Untuk setiap aspek, Subject menyediakan method untuk register-unregister dan menyediakan interface yang harus diimplement oleh Observer yang ingin memantau aspek tersebut.

Pada satu titik tertentu, Subject akan memberitahu (notify) Observer tentang perilaku atau keadaanya. Subject akan mengumpulkan informasi tentang keadaan atau perilakunya kemudian mengirimkan pesan kepada Observer lewat interface yang telah disepakati keduanya, pola ini dikenal juga sebgai Event-Passing.

Pattern Observer dimaksudkan untuk mengurangi ketergantungan satu object terhadap object lain, istilah kerennya adalah Decoupling. Dengan mekanisme register-unregister, Observer dapat secara lebih leluasa memutuskan untuk memantau Subject tertentu atau tidak. Mekanisme notify memudahkan Subject memberitahu keadaan dan perilakunya kepada Observer yang sedang memantaunya.

Di bagian berikutnya kita akan melihat bagaimana pattern Observer diimplementasikan dalam swing. Akan dijelaskan pula bagaimana swing mengimplementasikan mekanisme register-unregister dan notify dalam menangani interaksi user terhadap komponen swing.

Event Listener dalam Swing

Pattern Observer melibatkan dua object Subject dan Observer, dalam swing Observer dikenal sebagai Listener. Kemudian, ketika Subject akan memberitahu (notify) Observer tentang apa yang sedang terjadi dalam object Subject, ada satu informasi yang akan di-passing oleh Subject ke Observer, informasi ini disebut sebagai Event object. Sedangkan kejadian ketika Subject melakukan notify kepada Observer disebut sebagai Event triggering.

Agar penjelasan diatas mudah dipahami, kita akan membuat aplikasi sederhana yang mengimplementasikan pattern Observer. Aplikasi sederhana ini terdiri dari dua class utama yaitu Subject dan Observer.

Class Subject akan menjalankan sebuah loop tanpa batas, di dalam loop tersebut Subject akan meminta input dari user berupa sebuah kata yang diakhiri dengan penekanan enter. Ketika user menekan enter, Subject akan menotify Observer. Dalam proses notifikasi tersebut, Subject mengumpulkan informasi tentang event pemasukan kata oleh user, informasi tersebut berupa : kata apa yang dimaksukkan dan object subject dimana event pemasukan kata tersebut terjadi (source). Kemudian Observer akan menerima informasi dari Subject dan mencetak informasi tersebut ke standard output. Berikut ini tampilan dari aplikasi sederhana ini :

```
type a word  : ifnu
print from observer : first observer
        event from : subject observed
        key presed is ifnu
```

Subject akan mencetak string “type a word :” dan menunggu user untuk memasukkan satu kata dan menekan enter. Misalnya dalam contoh diatas “ifnu”. Kemudian Subject akan menghimpun informasi tentang sumber event (Subject itu sendiri) dan kata yang diketikkan user (ifnu). Setelah itu, Subject memberitahu (notify) Observer bahwa telah terjadi event pemasukan kata dengan menyertakan informasi yang telah dihimpun Subject.

Observer menerima informasi dari Subject bahwa telah terjadi event pemasukan kata oleh user, selanjutnya Observer akan menjalankan tindakan-tindakan untuk menangani event tersebut. Tindakan tersebut adalah : mencetak informasi Observer, source, dan kata yang dimasukkan oleh user.
Di dalam class Subject terdapat field

1. Koleksi Observer (listeners)
2. Nama (name)
3. Kata yang dimasukkan user (wordEntered)

Kemudian ada juga method :

1. Constructor yang menerima parameter String, parameter ini digunakan sebagai pengenal (name) dari object Subject.
2. Register-unregister Observer (registerListener, removeListener)
3. Method private untuk menotify Observer (triggerListener)
4. Method untuk menerima input kata dari user (runProgram)

Berikut ini kode lengkapnya :

```
public class Subject {
    private Set<KeyboardPressedListener> listeners =
       new HashSet<KeyboardPressedListener>();
    private String wordEntered;
    private String name;
    public Subject(String subjectName){
        name = subjectName;
    }
    public void runProgram(){
        while(true){
            Console c = System.console();
            wordEntered = c.readLine("type a word  : ");
            if(wordEntered.equals("exit"))
                break;
            else
                triggerListener();
        }
    }
    private void triggerListener(){
        KeyboardPressedEvent event =
             new KeyboardPressedEvent();
        event.setSource(this);
        event.setWord(wordEntered);
        for(KeyboardPressedListener l : listeners){
            l.keyPressed(event);
         }
    }
    public void registerObserver(
       KeyboardPressedListener l){ listeners.add(l); }
    public void removeObserver(
       KeyboardPressedListener l){
        listeners.remove(l);
    }
    public String toString(){ return name; }
}
```

Interface KeyboardPressedListener digunakan sebagai “kontrak” antara Subject dan Observer. Interface ini menjamin bahwa Observer yang akan memantau event pemasukan kata dari user dalam Subject mempunyai method keyPressed. Method keyPressed

ini nanti yang akan dipanggil oleh Subject ketika event pemasukan kata oleh user terjadi di Subject.

```
public interface KeyboardPressedListener {
    public void keyPressed(KeyboardPressedEvent e);
}
```

Class Observer mengimplement interface KeyboardPressedListener dan nantinya akan didaftarkan ke subject sebagai Observer. Method keyPressed diimplementasikan dengan mencetak informasi yang diperoleh dari Subject ke standard output.

```
public class Observer implements KeyboardPressedListener{
    private String name;
    public Observer(String name){ this.name = name; }
    public void keyPressed(KeyboardPressedEvent e) {
        System.out.println("print from observer : " +
             name + "\n\tevent from : " + e.getSource()
            + "\n\tkey presed is " + e.getWord() );}}
```

Class KeyboardPressedEvent adalah Java Bean biasa yang menyimpan informasi kejadian pemasukan kata oleh user, didalamnya hanya ada field source dan word serta getter-setter method untuk kedua field tersebut.

```
public class KeyboardPressedEvent  {
    private Object source;
    private String word;
    public Object getSource() { return source; }
    public void setSource(Object src) { source = src;}
    public String getWord() { return word; }
    public void setWord(String wrd) { word = wrd; }
}
```

Sampai disini, class-class diatas masih berdiri sendiri dan belum ada class yang mempunyai method main. Nah, disinilah kode class MainClass untuk menyatukan semua object diatas menjadi aplikasi utuh.

```
public class MainClass {
    public static void main(String[] str){
        Subject subject =
             new Subject("subject observed");
        Observer observer =
             new Observer("first observer");
        subject.registerObserver(observer);
        subject.runProgram();
    }
}
```

Langkah-langkah dalam menggunakan pattern Observer ini adalah :

1. Membuat object subject dari class Subject

```
Subject subject = new Subject("subject observed");
```

2. Membuat object observer dari class Observer

```
Observer observer = new Observer("first observer");
```

3. Daftarkan object observer ke object subject

```
subject.registerObserver(observer);
```

4. Jalankan program utamanya

```
subject.runProgram();
```

Pattern Observer ini digunakan secara intensif dalam komponen swing. Terutama untuk menangangi event dari input peripheral—keyboard, mouse, barcode reader—yang terjadi di komponen swing —JTextField, JButton, JTable—. Dalam bagian-bagian selanjutnya kita akan belajar bagaimana menangani event pada komponen swing.

ActionListener

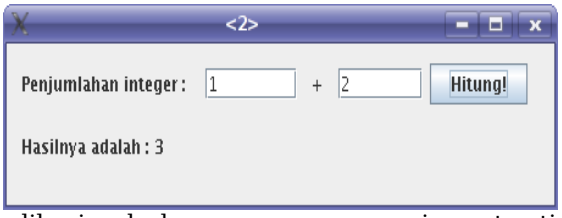
ActionListener digunakan untuk mendengarkan action dari event :

- Klik terhadap JButton
- Pemilihan menu item
- Penekanan tombol enter dalam JTextField

Method dalam ActionListener hanya satu yaitu actionPerformed yang menerima argumen object ActionEvent. ActionEvent berisi informasi-informasi penting ketika Action event terjadi, termasuk tombol modifiers apa yang sedang ditekan. Tombol modifiers antara lain : CTRL, ALT, META dan SHIFT. Method untuk menentukan tombol modifiers apa yang sedang aktif adalah

getModifiers. Method getActionCommand digunakan untuk mengambil command string yang didefinisikan oleh JButton.

Di bagian sebelumnya kita telah bekerja menggunakan komponen swing, dan sudah berlatih bagaimana menangani event klik mouse terhadap JButton. Mari kita lihat lagi aplikasi sederhana berikut :



Contoh aplikasi sederhana yang menangani event actionEvent pada JButton

Menangani klik mouse pada JButton dalam Netbeans cukup dengan memilih JButton di Jendela Design kemudian klik kanan dan pilih menu :

```
Events > Action > ActionPerformed
```

Setelah itu anda akan dibawa ke jendela kode dan baris berikut ini akan dibuat secara otomatis oleh Netbeans :

```
private void btnHitungActionPerformed(
        java.awt.event.ActionEvent evt) {
}
```

Kemudian kita akan menempatkan kode untuk menangani penekanan tombol di bagian bawah baris //TODO. Sebenarnya Netbeans men-generate beberapa lagi kode di bagian yang tidak dapat diedit, berikut ini cuplikanya :

```
btnHitung.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt){
                btnHitungActionPerformed(evt);
        }
});
```

Method addActionListerner ini mempunyai fungsi yang sama dengan method registerListener pada class Subject yang kita bahas di bagian sebelumnya. addActionListerner berfungsi untuk mendaftarkan Observer ke Subject. Perbedaan terbesar disini adalah Netbeans tidak membuat public class Observer baru untuk mengimplementasi interface ActionListener. Tetapi Netbeans membuat anonymous innerclass yang mengimplement interface ActionListener.

Perhatikan petikan kode berikut ini :

```
new ActionListener() {
      public void actionPerformed(ActionEvent evt){
            btnHitungActionPerformed(evt);
      }
```

Yang dilakukan oleh kode diatas sebenarnya adalah:
1. Membuat Class baru yang tidak punya nama (anonymous)
2. Class baru tersebut turunan dari Object dan mengimplement interface ActionListener
3. Mengimplementasi method abstrac actionPerformed
4. Method actionPerformed dari class tak bernama ini akan memanggil method btnHitungActionPerformed dari class parentnya.

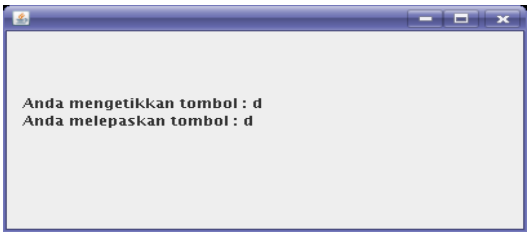
Secara umum Netbeans akan membuat sebuah anonymous innerclass seperti diatas untuk setiap penanganan event. Dari sisi kerapian kode metode ini sangat bagus, karena Netbeans menyembunyikan kerumitan kode untuk menangani event seperti diatas. Kita tidak perlu susah-susah membuat sebuah public class Observer tersendiri untuk menangani event. Cukup dengan anonymous innerclass.

KeyListener

KeyListener akan mendengarkan penekanan tombol oleh komponen yang berada dalam keadaan fokus. Semua komponen swing dapat menerima KeyListener sebagai Observer. KeyListener dapat mendengarkan tiga event berbeda : penekanan tombol, pelepasan tombol dan pengetikan tombol. Ketiganya ditangani oleh method yang berbeda-beda, yaitu :
- keyPressed : dipanggil ketika terjadi penekanan tombol keyboard.
- keyReleased : dipanggil ketika tombol keyboard dilepaskan.
- keyTyped : dipanggil ketika tombol diketikkan, alias ditekan kemudian dilepaskan. Method ini dipanggil jika tombol yang ditekan mempunyai representasi karakter unicode, seperti tombol angka dan tombol huruf. Sedangkan penekanan tombol modifiers seperti ALT, CTRL, ARROW, CAPSLOCK, NUMLOCK, INS dan lainnya tidak akan mengakibatkan method ini dipanggil.

Ketiga method diatas menerima parameter KeyEvent. Untuk mengetes tombol apakah yang ditekan oleh user, digunakan method getKeyCode. Kemudian hasil kembalian method getKeyCode dibandingkan dengan field static kepunyaan class KeyEvent yang diawali dengan huruf VK, seperti : VK_ENTER, VK_A, VK_B, VK_1, VK_LEFT_ARROW dan seterusnya.

Method getKeyChar digunakan untuk menentukan karakter apa yang diwakili oleh tombol yang ditekan. Jika tombol yang ditekan adalah tombol modifiers maka method getKeyChar akan mengembalikan karakter KeyEvent.CHAR_UNDEFINED.



Aplikasi sederhana yang menangani penekanan tombol keyboard

Untuk membuat aplikasi yang mendengarkan penekanan tombol keyboard seperti diatas lakukan langkah-langkah berikut ini :

1. Buat class JFrame baru, beri nama FrameKeyPressed.
2. Tambahkan dua buah JLabel, beri nama lblStatus dan lblKeyTyped.
3. lblStatus digunakan untuk menandakan adanya event keyPressed dan keyReleased yang terjadi berurutan
4. lblKeyTyped digunakan untuk menandakan adanya tombol yang diketik.
5. Pilih JFrame di Jendela Design, klik kanan, dan pilih menu :

```
Events > Key > keyPressed
```

6. Jendela Code akan terbuka, modifikasi method formKeyPressed menjadi seperti berikut ini :

```
private void formKeyPressed(KeyEvent evt) {
  // TODO add your handling code here:
  if(evt.getKeyChar() ==  KeyEvent.CHAR_UNDEFINED)
       lblStatus.setText(
              "Anda menekan tombol : CHAR_UNDEFINED");
       else
            lblStatus.setText("Anda menekan tombol : " +

                evt.getKeyChar());
}
```

7. Pilih JFrame di Jendela Design, klik kanan, dan pilih menu :

```
Events > Key > keyReleased
```

8. Jendela Code akan terbuka, modfikasi method formKeyReleased menjadi seperti berikut ini :

```
 private void formKeyReleased(KeyEvent evt) {
       // TODO add your handling code here:
       if(evt.getKeyChar() ==  KeyEvent.CHAR_UNDEFINED)
         lblStatus.setText(
           "Anda melepaskan tombol : CHAR_UNDEFINED");
       else
         lblStatus.setText("Anda melepaskan tombol : " +
           evt.getKeyChar());
}
```

9. Pilih JFrame di Jendela Design, klik kanan, dan pilih menu :

```
Events > Key > keyTyped
```

10. Jendela Code akan terbuka, modfikasi method formKeyTyped menjadi seperti berikut ini :

```
private void formKeyTyped(KeyEvent evt) {
       // TODO add your handling code here:
       lblKeyType.setText("Anda mengetikkan tombol : " +
              evt.getKeyChar());
}
```

MouseListener dan MouseMotionListener

MouseListener mendengarkan interaksi mouse terhadap komponen swing. MouseListener dapat didaftarkan pada semua komponen swing. MouseListener mendengarkan event :

- mousePressed : event ini terjadi ketika user menekan salah satu tombol mouse diatas komponen swing.
- mouseReleased : setelah tombol ditekan, komponen swing akan menerima pelepasan tombol mouse. Tetapi jika tombol mouse dilepaskan pada saat pointer mouse tidak berada diatas komponen swing, maka event ini tidak akan terjadi.
- mouseClicked : event ini muncul ketika user melakukan click tombol mouse diatas komponen swing.
- mouseEntered : ketika mouse memasuki area diatas komponen swing, event ini akan dipicu.
- mouseExited : muncul ketika mouse akan meninggalkan area diatas komponen swing

Ketika user menekan tombol mouse (click), event mousePressed dibuat, kemudian mouseReleased dan akhirnya mouseClicked

muncul terakhir.

MouseMotionListener juga dapat didaftarkan sebagai listener pada semua komponen swing. MouseMotionListener dipisahkan dari MouseListener karena penanganan event mouseMove yang lebih berat dan intensif. MouseMotionListener mendengarkan dua envent:

- mouseMoved : terjadi ketika user menggerakkan mouse diatas komponen swing
- mouseDragged : terjadi ketika user menekan tombol mouse sekaligus menggerakkanya diatas komponen swing

Semua method diatas menerima argumen berupa class MouseEvent. Method getClickCount digunakan untuk menentukan jumlah click yang terjadi dalam waktu yang berdekatan. Method getClickCount juga digunakan untuk menentukan apakah klik yang terjadi adalah single klik atau double klik.

Method getButton digunakan untuk menentukan tombol mana yang ditekan oleh user. Pada umumnya mouse yang tersedia di pasaran mempunyai tiga timbol yang dapat di klik, tombol kiri, tombol tengah dan tombol kanan. Method getButton akan mengembalikan nilai MouseEvent.BUTTON1 jika tombol kiri ditekan, MouseEvent.BUTTON2 jika tombol tengah ditekan dan MouseEvent.BUTTON3 jika tombol kanan ditekan.

Method getX dan getY akan mengembalikan koordinat dimana MouseEvent terjadi. Koordinat yang digunakan adalah koordinat relatif. Koordinat (0,0) berada di pojok kiri atas dari komponen swing, semakin kebawah nilai Y semakin besar dan semakin ke kanan nilai X semakin besar. Nilai koordinatnya dinyatakan dalam satuan pixel.

Aplikasi di bawah ini adalah sebuah JFrame yang mempunyai JLabel di dalamnya. Ketika terjadi event mouseClick dan mouseMove, JLabel akan menampilkan dimana event tersebut terjadi. Jika event klik yang muncul, maka text dari JLabel akan berisi “clicked at (x,y)”, sedangkan event move hanya akan memunculkan koordinat “(x,y)” saja.

Contoh aplikasi sederhana yang mengani MouseEvent

Lakukan langkah-langkah berikut ini untuk membuat aplikasi seperti diatas :

1. Buat class JFrame baru, beri nama FrameMouseMotion.
2. Letakkan sebuah JLabel baru, beri nama lblStatus.
3. Pilih JFrame di Jendela Designer, klik kanan dan pilih menu :

```
Set Layout > Null Layout
```

   Langkah ini bertujuan untuk membuat agar JFrame menggunakan null layout. Kalau tidak menggunakan null layout kita tidak bisa meletakkan JLabel pada sembarang posisi.

4. Pilih kembali JFrame, klik kanan dan pilih menu

```
Events > Mouse > mouseClicked
```

5. modifikasi kode pada Jendela Code yang tampil menjadi seperti di bawah ini :

```
private void formMouseClicked(MouseEvent evt) {
       lblStatus.setText("clicked at (" + evt.getX() +
              "," + evt.getY() + ")");
       lblStatus.setLocation(evt.getX(),evt.getY());
}
```

   Kode diatas menangani penekanan tombol mouse, kemudian mengubah text JLabel dan memindahkan JLabel ke posisi dimana event mouseClicked terjadi.

6. Pilih JFrame lagi di Jendela Design, klik kanan dan pilih menu :

```
Events > MouseMotion > mouseMoved
```

7. Modifikasi kode yang muncul pada Jendela Code menjadi seperti di bawah ini :

```
private void formMouseMoved(MouseEvent evt) {
       lblStatus.setText("(" + evt.getX() + "," +
              evt.getY() + ")");
       lblStatus.setLocation(evt.getX(),evt.getY());
    }
```

8. Compile dan jalankan aplikasi di atas.

Masih banyak lagi Event-Listener yang disediakan oleh JDK. Dari bab di atas kita sudah dapat mengerti dengan baik konsep Event-Listener dan pattern Observer yang mendasarinya. Dengan kemampuan ini kita bisa dengan mudah mengerti bagaimana event-listener yang lain bekerja.

Event-Listener juga dapat dijalankan terhadap Java Bean menggunakan PropertyChangeListener dan PropertyEvent. Konsep ini dapat digunakan untuk mengamati perubahan pada field Java Bean.

Koneksi Database Dengan JDBC

Mengenal JDBC

Java Database Connectivity adalah API yang digunakan Java untuk melakukan koneksi dengan aplikasi lain atau dengan berbagai macam database. JDBC memungkinkan kita untuk membuat aplikasi Java yang melakukan tiga hal: konek ke sumber data, mengirimkan query dan statement ke database, menerima dan mengolah resultset yang diperoleh dari database.

JDBC mempunyai empat komponen :

1. JDBC API

   JDBC API menyediakan metode akses yang sederhana ke sumber data relational (RDBMS) menggunakan pemrograman Java. dengan menggunakan JDBC API, kita bisa membuat program yang dapat mengeksekusi SQL, menerima hasil ResultSet, dan mengubah data dalam database. JDBC API juga mempunyai kemampuan untuk berinteraksi dengan lingkungan terdistribusi dari jenis sumber data yang berbeda-beda.

   JDBC API adalah bagian dari Java Platform yang disertakan dalam library JDK maupun JRE. JDBC API sekarang ini sudah mencapai versi 4.0 yang disertakan dalan JDK 6.0. JDBC API 4.0 dibagi dalam dua package yaitu : java.sql dan javax.sql.

2. JDBC Driver Manager

   Class DriverManager dari JDBC bertugas untuk mendefisikan object-object yang dapat digunakan untuk melakukan koneksi ke sebuah sumber data. Secara tradisional DriverManager telah menjadi tulang punggung arsitektur JDBC.

3. JDBC Test Suite

   JDBC Test Suite membantu kita untuk mencara driver mana yang cocok digunakan untuk melakukan sebuah koneksi ke sumber data tertentu. Tes yang dilakukan tidak memerlukan resource besar ataupun tes yang komprehensif, namun cukup tes-tes sederhana yang memastikan fitur-fitur penting JDBC dapat berjalan dengan lancar.

4. JDBC-ODBC Bridge

Brige ini menyediakan fasilitas JDBC untuk melakukan koneksi ke sumber data menggunakan ODBC (Open DataBase Connectivity) driver. Sebagai catatan, anda perlu meload driver ODBC di setiap komputer client untuk dapat menggunakan bridge ini. Sebagai konsekuensinya, cara ini hanya cocok dilakukan di lingkungan intranet dimana isu instalasi tidak menjadi masalah.

Dengan keempat komponen yang dipunyainya, JDBC menjadi tools yang dapat diandalkan untuk melakukan koneksi, mengambil data dan merubah data dari berbagai macam sumber data. Modul ini hanya akan membahas dua komponen pertama dari keempat komponen yang dipunyai oleh JDBC, yaitu JDBC API dan DriverManager. Sumber data yang digunakan adalah Relational Database.

Database Driver

JDBC memerlukan database driver untuk melakukan koneksi ke suatu sumber data. Database driver ini bersifat spesifik untuk setiap jenis sumber data. Database driver biasanya dibuat oleh pihak pembuat sumber datanya, namun tidak jarang juga komunitas atau pihak ketiga menyediakan database driver untuk sebuah sumber data tertentu.

Perlu dipahami sekali lagi bahwa database driver bersifat spesifik untuk setiap jenis sumber data. Misalnya, Database Driver MySql hanya bisa digunakan untuk melakukan koneksi ke database MySql dan begitu juga database driver untuk Postgre SQL juga hanya bisa digunakan untuk melakukan koneksi ke database Postgre SQL.

Database driver untuk setiap DBMS pada umumnya dapat didownload dari website pembuat DBMS tersebut. Beberapa vendor DBMS menyebut Database driver ini dengan sebutan Java Connector (J/Connector). Database driver biasanya dibungkus dalam file yang berekstensi jar. Setiap database driver harus mengimplement interface java.sql.Driver.

Membuat Koneksi

Melakukan koneksi ke database melibatkan dua langkah: Meload driver dan membuat koneksi itu sendiri. Cara meload driver sangat mudah, pertama letakkan file jar database driver ke dalam classpath. Kemudian load driver dengan menambahkan kode berikut ini:

```
Class.forName("com.mysql.jdbc.Driver");
```

Nama class database driver untuk setiap DBMS berbeda, anda bisa menemukan nama class tersebut dalam dokumentasi driver database yang anda gunakan. Dalam contoh ini, nama class database driver dari MySql adalah com.mysql.jdbc.Driver.

Memanggil method Class.forName secara otomatis membuat instance dari database driver, class DriverManager secara otomatis juga dipanggil untuk mengelola class database driver ini. Jadi anda tidak perlu menggunakan statement new untuk membuat instance dari class database driver tersebut.

Langkah berikutnya adalah membuat koneksi ke database menggunakan database driver yang sudah diload tadi. Class DriverManager bekerja sama dengan interface Driver untuk mengelola driver-driver yang diload oleh aplikasi, jadi dalam satu sesi anda bisa meload beberapa database driver yang berbeda.

Ketika kita benar-benar melakukan koneksi, JDBC Test Suite akan melakukan serangkaian tes untuk menentukan driver mana yang akan digunakan. Parameter yang digunakan untuk menentukan driver yang sesuai adalah URL. Aplikasi yang akan melakukan koneksi ke database menyediakan URL pengenal dari server databse tersebut. Sebagai contoh adalah URL yang digunakan untuk melakukan koneksi ke MySql :

```
jdbc:mysql://[host]:[port]/[schema]
```

contoh konkritnya :

```
jdbc:mysql://localhost:3306/latihan
```

Setiap vendor DBMS akan menyertakan cara untuk menentukan URL ini di dalam dokumentasi. Anda tinggal membaca dokumentasi tersebut tanpa harus khawatir tidak menemukan informasi yang anda perlukan.

Method DriverManager.getConnection bertugas untuk membuat koneksi:

```
Connection conn =
      DriverManager.getConnection(
       "jdbc:mysql://localhost:3306/latihan");
```

Dalam kebanyakan kasus anda juga harus memasukkan parameter username dan password untuk dapat melakukan koneksi ke dalam database. Method getConnection menerima Username sebagai parameter kedua dan pasword sebagai parameter ketiga, sehingga kode diatas dapat dirubah menjadi :

```
Connection conn =
      DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/latihan",
            "root","");
```

Jika salah satu dari driver yang diload berhasil digunakan untuk melakukan koneksi dengan URL tersebut, maka koneksi ke database berhasil dilaksanakan. Class Connection akan memegang informasi koneksi ke database yang didefinisikan oleh URL tersebut.

Setelah sukses melakukan koneksi ke database, kita dapat mengambil data dari database menggunakan perintah query ataupun melakukan perubahan terhadap database. bagian berikut ini akan menerangkan bagaimana cara mengambil dan memanipulasi data dari database.

Mengambil dan Memanipulasi Data dari Database

Proses pengambilan data dari database memerlukan suatu class untuk menampung data yang berhasil diambil, class tersebut harus mengimplement interface ResultSet.

Object yang bertipe ResultSet dapat mempunyai level fungsionalitas yang berbeda, hal ini tergantung dari tipe dari result set. Level fungsionalitas dari setiap tipe result set dibedakan berdasarkan dua area:

- Dengan cara bagaimana result set itu dapat dimanipulasi
- Bagaimana result set itu menangani perubahan data yang dilakukan oleh proses lain secara bersamaan (concurrent).

JDBC menyediakan tiga tipe result set untuk tujuan berbeda:

1. TYPE_FORWARD_ONLY : result set tersebut tidak bisa berjalan mundur, reslut set hanya bisa berjalan maju dari baris pertama hingga baris terakhir. result set hanya menggambarkan keadaan data ketika query dijalankan atau ketika data diterima oleh resul set. Jika setelah itu ada perubahan data dalam database, result set tidak akan diupdate alias tidak ada perubahan dalam result set tipe ini.
2. TYPE_SCROLL_INSENSITIVE : result set dapat berjalan maju mundur. result set dapat berjalan maju dari row pertama hingga terakhir atau bergerak bebas berdasarkan posisi relatif atau absolute.

3. TYPE_SCROLL_SENSITIVE : result set dapat berjalan maju mundur. result set dapat berjalan maju dari row pertama hingga terakhir atau bergerak bebas berdasarkan posisi relatif atau absolute.

Instance dari object bertipe ResultSet diperlukan untuk menampung hasil kembalian data dari database. Sebelum kita bisa memperoleh instance dari ResultSet, kita harus membuat instance dari class Statement. Class Statement mempunyai method executeQuery yang digunakan untuk menjalankan perintah query dalam database kemudian mengembalikan data hasil eksekusi query ke dalam object ResultSet.

Berikut ini adalah contoh kode untuk membuat instance class Statement, kemudian menjalankan query untuk mengambil data dari database yang hasilnya dipegang oleh ResultSet :

```
Statement statement =
       conn.createStatement(
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_READ_ONLY);
ResulSet rs =
       statement.executeQuery("select * from Customers");
```

ResultSet akan meletakkan kursornya (posisi pembacaan baris) di sebuah posisi sebelum baris pertama. Untuk menggerakkan kursor maju, mundur, ke suatu posisi relatif atau ke suatu posisi absolute tertentu, gunakan method-method dari ResultSet:

- next() -- mengarahkan kursor maju satu baris.
- previous() -- mengarahkan kursor mundur satu baris.
- first() -- mengarahkan kursor ke baris pertama.
- last() -- mengarahkan kursor ke baris terakhir.
- beforeFirst() -- mengarahkan kursor ke sebelum baris pertama.
- afterLast() -- mengarahkan kursor ke setelah baris terakhir.
- relative(int rows) -- mengarahkan kursor relatif dari posisinya yang sekarang. Set nilai rows dengan nilai positif untuk maju, dan nilai negatif untuk mundur.
- absolute(int rowNumber) – mengarahkan kursor ke posisi tertentu sesuai dengan nilai rowNumber, dan tentu saja nilainya harus positif.

Interface ResultSet menyediakan method getter untuk mengakses nilai dari setiap kolom dalam baris yang sedang aktif. Parameter fungsi getter bisa menerima nilai index dari kolom ataupun nama kolomnya. Namun begitu, penggunaan nilai index lebih efisien dibanding menggunakan nama kolom.

Nilai index dimulai dengan satu hingga banyaknya kolom. Penggunaan nama kolom adalah case insensitive, artinya huruf kecil atau huruf besar tidak menjadi masalah.

getString digunakan untuk mengambil kolom dengan tiper data char, varchar atau tipe data string lainya. getInt digunakan untuk mengambil kolom dengan tipe data integer.

Berikut ini dalah contoh program lengkap dari melakukan koneksi hingga mengambil data dari database.

```
Class.forName("com.mysql.jdbc.Driver);
Connection conn =
       DriverManager.getConnection(
              "jdbc:mysql://localhost:3306/latihan",
              "root","");
Statement statement =
       conn.createStatement(
              ResultSet.TYPE_SCROLL_SENSITIVE,
              ResultSet.CONCUR_READ_ONLY);
ResulSet rs =
       statement.executeQuery("select * from Customers");
while(rs.next()){
       System.out.println(rs.getInt("id"));
       System.out.println(rs.getString("Nama"));
}
```

Method executeQuery hanya dapat menjalankan perintah SQL select, gunakan method executeUpdate untuk menjalankan perintah insert, update dan delete. Hasil dari eksekusi insert, update dan delete tidak mengembalikan result set, tetapi mengembalikan sebuah nilai integer yang merepresentasikan status hasil eksekusi method executeUpdate.

Berikut ini contoh insert, update dan delete :

```
result = statement.executeUpdate(
       "update Customers set nama ='roby' where
       nama='andy'");
result = statement.executeUpdate(
       "delete Customers where nama='andy'");
```

Menggunakan executeQuery dan executeUpdate sangat mudah dan fleksible, namun sangat tidak efisien, PreparedStatement menawarkan keunggulan dalam bentuk efisiensi.

Menggunakan PreparedStatement

Memanggil method executeUpdate berulang-ulang, misalnya melakukan insert ratusan atau ribuan baris, sangat tidak efisien.

Hal ini disebabkan karena DBMS harus memproses setiap query yang dikirimkan dalam beberapa langkah: memparsing query, mengcompile query dan kemudian baru mengeksekusi query tersebut.

PreparedStatement menawarkan solusi yang lebih baik dalam menangani keadaan tersebut. PreparedStatement menyaratkan query yang akan dieksekusi didefinisikan terlebih dahulu ketika PreparedStatement dibuat. Kemudian query tersebut dikirimkan ke dalam database untuk dicompile terlebih dahulu sebelum digunakan. Konsekuensinya, PreparedStatement bukan hanya mempunyai query, tetapi mempunyai query yang sudah dicompile. Ketika PreparedStatement dijalankan, DBMS tidak perlu melakukan kompilasi ulang terhadap query yang dijalankan PreparedStatement. Hal inilah yang menyebabkan PreparedStatement jauh lebih efisien dibandingkan menggunakan method Statement.executeUpdate.

Berikut ini contoh pembuatan PreparedStatement menggunakan class Connection yang telah dibuat sebelumnya :

```
PreparedStatement ps = conn.prepareStatement(
       "update T_PERSON set name = ? where name = ?");
```

Perhatikan tanda ? yang ada dalam query diatas, tanda ? disebut sebagai parameter. Kita bisa memberikan nilai yang berbeda ke dalam parameter dalam setiap pemanggilan PreparedStatement.

Method setString, setFloat, setInt dan beberapa method lain digunakan untuk memasukkan nilai dari setiap parameter. Method tersebut mempunyai dua parameter, parameter pertama adalah int yang digunakan untuk menentukan parameter PreparedStatement mana yang akan diberi nilai. Parameter kedua adalah nilai yang akan dimasukkan ke dalam PreparedStatement, tipe data dari parameter kedua tergantung dari method yang digunakan. Berdasarkan kode diatas, berikut ini contoh penggunaan method PreparedStatement.setString :

```
ps.setString(1,"andy");
ps.setString(2,"rizal");
```

Kode diatas memberikan contoh bagaimana memasukkan nilai ke dalam parameter PreparedStatement. Baris pertama memasukkan String "andy" ke dalam parameter pertama dan baris kedua memasukkan String "rizal" ke parameter kedua. Sehingga pemanggilan query oleh PreparedStatement berdasarkan kode diatas sama dengan query statement di bawah ini :

```
"update T_PERSON set name = 'andy' where name = 'rizal'"
```

Berikut ini contoh lengkap penggunaan PreparedStatement untuk melakukan update dan insert data :

```
PreparedStatement pInsert = conn.prepareStatement(
       "insert into Person(name) values(?)");
pInsert.setString(1,"dian");
pInsert.executeUpdate();
PreparedStatement pUpdate = conn.prepareStatement(
       "update Person set name=? where name=?");
pUpdate.setString(1,"andry");
pUpdate.setString(2,"andri");
pUpdate.executeUpdate();
```

Dalam contoh diatas, insert dan update data hanya dilaksanakan sekali saja, hal ini tidak memberikan gambaran yang tepat untuk melihat keunggulan PreparedStatement dibandingkan Statement.executeUpdate.

Batch Execution

Misalnya kita ingin meng-insert seratus baris data dalam sebuah loop, kita bisa menggunakan fasilitas batc execution dari PreparedStatement. batch execution mengumpulkan semua eksekusi program yang akan dilaksanakan, setelah semuanya terkumpul batch execution kemudian mengirimkan kumpulan eksekusi program secara bersamaan ke DBMS dalam satu kesatuan. Metode ini sangat efisien karena mengurangi overhead yang diperlukan program untuk berkomunikasi dengan DBMS.

Dalam contoh di bawah ini kita akan menggunakan batch execution untuk melakukan insert data sebanyak seratus kali.

```
PreparedStatement pInsert = conn.prepareStatement(
       "insert into Person(nama) values(?)");
for(int i=0;i<100;i++){
       pInsert.setString(1,"user ke " + i);
       pInsert.addBatch();
}
pInsert.executeBatch();
```

Setiap kali iterasi, method setString dipanggil untuk mengisikan sebuah string ke dalam PreparedStatement, kemudian method addBatch dipanggil untuk mengumpulkan batch dalam satu wadah. Setelah iterasi selesai, method executeBatch dipanggil untuk melaksanakan semua keseratus instruksi insert secara berurut dengan sekali saja melaksanakan koneksi ke database.

Menangani Transaction

Dukungan transaction oleh JDBC tergantung dengan Databasenya,

karena ada database yang mendukung transaction dan ada pula database yang tidak mendukung transaction. MySQL mendukung transaction jika kita menggunakan InnoDB sebagai sistem tablenya, kalau kita menggunakan MyISAM maka transaction tidak didukung.

Transaction merupakan konsep penting dari database. Transaction memastikan perubahan data dilaksanakan dengan kaidah ACID (Atomicity, Consistency, Isolation, Durability). Kaidah ini memastikan semua proses perubahan data berjalan secara benar, jika ada yang salah maka semua perubahan dalam satu kesatuan logika harus dibatalkan (rollback).

Mari kita evaluasi kode diatas agar menggunakan transaction, sehingga jika satu proses insert gagal, maka semua insert yang dilaksanakan sebelumnya akan dibatalkan :

```
try{
connection.setAutoCommit(false);
PreparedStatement pInsert = conn.prepareStatement(
       "insert into Person(nama) values(?)");
for(int i=0;i<100;i++){
       pInsert.setString(1,"user ke " + i);
       pInsert.addBatch();
}
pInsert.executeBatch();
connection.commit();
connection.setAutoCommit(true);
} catch (SQLException ex) {
       try{
              connection.rollback();
       }catch(SQLException e){
       }
}
```

DAO dan Service Pattern

Akses terhadap database merupakan bagian yang sangat penting dari aplikasi database. Penggunaan pattern yang sesuai dapat memberikan manfaat sangat besar. Pattern yang sering digunakan dalam akses database adalah DAO (Data Access Object) dan Service/Facade pattern.

Kedua pattern ini digunakan untuk menerapkan “separation of concern” atau pemisahan kode proram berdasarkan fungsi kode program. Semua kode untuk akses data harus dipisahkan dengan kode untuk pengaturan user inteface. Hal ini memungkinkan kode akses data yang dibuat untuk aplikasi desktop, dengan mudah

digunakan untuk aplikasi web.

Penerapan konsep separation of concern secara disiplin, dapat menghasilkan kode program yang dapat dites secara otomatis menggunakan JUnit atau DBUnit. Unit testing merupakan paramater utama dalam menentukan apakah kode program yang kita hasilkan mempunyai mutu yang tinggi atau tidak. Coverage unit testing yang tinggi mencerminkan kode program yang berkualitas tinggi pula.

Dao pattern berisi semua kode untuk mengakses data, seperti query. Semua kode yang sepesifik terhadap implementasi akses data berhenti di sini, lapisan lebih atas tidak boleh tahu bagaimana akses data diterapkan, apakah menggunakan JDBC murni atau Hibernate atau JPA. Lapisan lainya hanya perlu tahu fungsionalitas dari suatu method di dalam DAO class, tidak perlu tahu bagimana method tersebut diimplementasikan. Class DAO akan mempunyai method seperti save, delete, getById atau getAll. Praktek yang lazim digunakan adalah satu buah Entity/Table akan mempunyai satu buah class DAO.

Di bawah ini adalah contoh Class DAO menggunakan JDBC untuk table T_CATEGORY :

```
public class CategoryDaoJdbc {

    private Connection connection;
    private PreparedStatement insertStatement;
    private PreparedStatement updateStatement;
    private PreparedStatement deleteStatement;
    private PreparedStatement getByIdStatement;
    private PreparedStatement getAllStatement;
    private final String insertQuery =
       "insert into T_CATEGORY(name,description)" +
       " values(?,?)";
    private final String updateQuery =
       "update T_CATEGORY set name=?, description=? " +
       " where id=?";
    private final String deleteQuery =
       "delete from T_CATEGORY where id=?";
    private final String getByIdQuery =
       "select * from T_CATEGORY where id =?";
    private final String getAllQuery =
       "select * from T_CATEGORY";
    public void setConnection(Connection connection) {
        try {
            this.connection = connection;
       insertStatement =
         this.connection.prepareStatement(
       insertQuery,Statement.RETURN_GENERATED_KEYS);
```

```
      updateStatement =
        this.connection.prepareStatement(
        updateQuery);
      deleteStatement =
        this.connection.prepareStatement(
        deleteQuery);
      getByIdStatement =
        this.connection.prepareStatement(
        getByIdQuery);
      getAllStatement =
        this.connection.prepareStatement(
        getAllQuery);
       } catch (SQLException ex) {
       }
   }

   public void save(Category category) {
       try {
      if (category.getId() == null) {
        insertStatement.setString(1,
```

```
            category.getName());
        insertStatement.setString(2,
            category.getDescription());
        int id = insertStatement.executeUpdate();
            category.setId(id);
      } else {
        updateStatement.setString(1,
            category.getName());
        updateStatement.setString(2,
            category.getDescription());
        updateStatement.setInt(3, category.getId());
              updateStatement.executeUpdate();
      }
      } catch (SQLException ex) {
      }
   }

   public void delete(Category category) {
      try {
      deleteStatement.setInt(1, category.getId());
      deleteStatement.executeUpdate();
      } catch (SQLException ex) {
      }
   }

   public Category getById(Long id) {
      try {
      getByIdStatement.setLong(1, id);
      ResultSet rs = getByIdStatement.executeQuery();
      //proses mapping dari relational ke object
      if (rs.next()) {
        Category category = new Category();
        category.setId(rs.getInt("id"));
        category.setName(rs.getString("name"));
        category.setDescription(
            rs.getString("description"));
        return category;
      }
      } catch (SQLException ex) {
      }
      return null;
   }
   public List<Category> getAll() {
      try {
      List<Category> categories =
            new ArrayList<Category>();
```

```
        ResultSet rs = getAllStatement.executeQuery();
        while(rs.next()){
          Category category = new Category();
          category.setId(rs.getInt("id"));
          category.setName(rs.getString("name"));
          category.setDescription(
               rs.getString("description"));
          categories.add(category);
        }
        return categories;
        } catch (SQLException ex) {
        }
        return null;
    }
}
```

Service pattern digunakan utamanya untuk menyederhanakan class-class DAO yang ada, misalnya kita mempunyai 50 buah table maka lazimnya akan ada 50 buah class DAO. Class DAO tersebut perlu disederhanakan, caranya adalah dengan mengelompokkan class-class DAO dalam satu modul aplikasi ke class Service. Misalnya DAO yang berhubungan dengan user management ke dalam class UserService.

Transaction diatur dalam class Service, praktek yang lazim digunakan adalah satu method dalam class service adalah satu scoop transaction. Jadi ketika method dalam service mulai dieksekusi transaction akan dimulai (begin), ketika method akan berakhir, transaction akan dicommit.

Berikut ini adalah contoh class Service :

```
public class ServiceJdbc {

    private CategoryDaoJdbc categoryDao;
    private Connection connection;

    public void setDataSource(DataSource dataSource){
        try {
            connection = dataSource.getConnection();
            categoryDao = new CategoryDaoJdbc();
            categoryDao.setConnection(connection);
        } catch (SQLException ex) {
        }
    }

    public void save(Category category){
        try {
            connection.setAutoCommit(false);
            categoryDao.save(category);
            connection.commit();
            connection.setAutoCommit(true);
        } catch (SQLException ex) {
            try{
                connection.rollback();
            }catch(SQLException e){
            }
        }
    }
    public void delete(Category category){
        try {
            connection.setAutoCommit(false);
            categoryDao.save(category);
            connection.commit();
            connection.setAutoCommit(true);
        } catch (SQLException ex) {
            try{
                connection.rollback();
            }catch(SQLException e){
            }
        }
   }

    public Category getGroup(Long id){
        return categoryDao.getById(id);
    }

    public List<Category> getGroups(){
```

```
        return categoryDao.getAll();
    }
}
```

Setelah class DAO dan service berhasil dibuat, mari kita lihat bagaimana cara menggunakannya :

```
public class MainJdbc {

    public static void main(String[] args) {

        MysqlDataSource dataSource =
             new MysqlDataSource();
        dataSource.setUser("root");
        dataSource.setPassword("admin");
        dataSource.setDatabaseName("latihan");
        dataSource.setServerName("localhost");
        dataSource.setPortNumber(1527);

        ServiceJdbc service = new ServiceJdbc();
        service.setDataSource(dataSource);

        Category category = new Category();
        category.setName("administrator");
        service.save(category);

        System.out.println("id : " + category.getId());
        System.out.println("name : " +
             category.getName());

        try {
            dataSource.getConnection().close();
        } catch (SQLException ex) {
        }
    }

}
```

Setelah mengenal JDBC, kita akan membahas cara mengakses database yang lebih baik dengan manggunakan JPA. Dengan menggunakan JPA kode program yang akan dibuat akan lebih ringkas dan terlihat konsep OOP dibanding dengan JDBC murni. JPA adalah framework ORM untuk memetakan tabel dalam database dan class dalam konsep OOP. Dengan menggunakan JPA kode program kita akan lebih rapi dan terlihat OOP.

Java Persistence API (JPA)

Hampir semua aplikasi bisnis menggunakan database. Konsep database relasional sudah populer jauh sebelum konsep object-oriented digunakan orang. Oleh karena itu, apapun bahasa pemrograman yang kita gunakan, menggunakan database adalah suatu keniscayaan.

Dalam interaksi kita dengan database, ada beberapa masalah yang umum kita temukan, diantaranya

- perbedaan dialek SQL antar merek database
- perbedaan konsep relasional dan object-oriented
- peningkatan kinerja (performance)

Pada buku ini kita akan membahas tentang framework yang disebut Object/Relational Mapping (ORM). JPA termasuk framework dalam kategori ORM. Dengan kita menggunakan ORM, maka banyak masalah umum di atas yang dapat diselesaikan tanpa kita harus membuat solusi sendiri.

Mari kita elaborasi.

Perbedaan dialek SQL

Masing-masing vendor database berusaha mempertahankan pelanggannya dengan cara membuat nilai tambah di produk mereka, yang membedakan produknya dengan produk pesaing. Nilai tambah ini tentunya diberikan dalam bentuk fitur-fitur canggih yang dimiliki database. Karena antarmuka utama database dan user (dalam hal ini programmer) adalah perintah SQL, maka kita sekarang melihat banyak sekali perbedaan SQL antar merek database. Bahkan untuk kasus sederhana saja, mengambil 10 baris pertama dari hasil query, para vendor database ini tidak bisa sepakat.

Perbedaan ini sangat menyulitkan bila kita ingin membuat produk aplikasi. Untuk menjangkau pasar yang luas, produk kita harus bisa digunakan di berbagai kondisi dan konfigurasi. Kita harus bisa mendukung pelanggan yang menggunakan PostgreSQL maupun Oracle. Perbedaan dialek SQL menyebabkan kita harus mengeluarkan biaya tambahan untuk mendukung berbagai jenis database.

Framework ORM menjembatani antara kode program dengan SQL. Kita tidak lagi menjalankan SQL sesuai merek database, tapi kita menggunakan Query Language yang disediakan ORM. Bila suatu

saat kita ingin mengganti database, kita cukup mengkonfigurasi ORM untuk berganti dialek.

Perbedaan konsep relasional dan object-oriented

Cara berpikir kita dalam mendesain struktur tabel database sangat berbeda dengan paradigma yang kita gunakan pada waktu mendesain struktur objek aplikasi kita. Mari kita lihat beberapa konsep berikut:

- equality
- inheritance
- relationship

Konsep *equality* membahas bagaimana kita menyatakan bahwa object `a` sama dengan object `b`. Di dalam bahasa Java, kedua object dinyatakan sama jika berada dalam lokasi memori yang sama, sehingga bisa dibandingkan dengan operator ==. Selain itu, kita juga bisa membuat definisi kita sendiri tentang kesamaan dengan cara meng-override method equals.

Di database, suatu record disebut sama dengan record lainnya bila kedua record memiliki nilai primary key yang sama dan tersimpan dalam tabel yang sama.

Bahasa pemrograman berorientasi objek seperti Java mengenal konsep inheritance. Dengan konsep ini, banyak teknik pemrograman canggih yang bisa dilakukan. Konsep ini tidak ada di database.

Relasi di kode Java ditentukan oleh method yang kita sediakan untuk navigasi. Sebagai contoh, bila satu Category memiliki banyak Article di dalamnya, ada beberapa kemungkinan yang bisa dilakukan:

- class Category bisa melihat semua Article dan sebaliknya, Article tahu dalam Category apa dia berada

```
public class Category {
  public List<Article> articles;
}
public class Article {
  private Category category;
}
```

- class Article tahu tentang Category, tapi Category bisa melihat Article yang ada di dalamnya

```
public class Category {
}
public class Article {
  private Category category;
}
```

- class Category bisa melihat isi Article, tapi Article tidak tahu Category tempat dia disimpan.

```
public class Category {
  public List<Article> articles;
}
public class Article {
}
```

Database tidak mengenal konsep navigasi. Bila kita punya struktur database seperti ini:

```
create table Category (
  id INT,
  name VARCHAR
);
create table Article (
  id INT,
  category_id INT,
  title VARCHAR
);
```

Maka kita bisa mendapatkan baik Article maupun Category dengan query seperti ini

```
select * from Article where category_id = ?
```

ORM menjembatani perbedaan ini, sehingga kita bisa membuat kode program dengan lebih rapi.

Peningkatan kinerja

Ada banyak hal yang bisa kita lakukan untuk mengoptimasi akses database, diantaranya:

- database connection pooling
- lazy-loading
- deferred SQL
- mengurangi hit ke database
- query result cache

Urusan connect/disconnect dari database merupakan urusan yang rumit. Banyak langkah yang terlibat di sana, misalnya inisialisasi koneksi, proses otentikasi, dan sebagainya.

Aplikasi besar biasanya mengoptimasi urusan connect/disconnect ini dengan menggunakan teknik connection pooling. Pada waktu dinyalakan, aplikasi langsung membuat banyak koneksi sekaligus (pool). Bila ada kode program yang ingin menggunakan koneksi untuk menjalankan SQL, maka koneksi diambil dari pool dan diberikan. Setelah selesai menjalankan SQL, koneksi tidak ditutup, melainkan dikembalikan ke pool. Dengan cara ini, overhead proses connect/disconnect dapat dikurangi.

Dengan menggunakan ORM, semua koneksi database diatur di satu tempat, sehingga mengubah konfigurasi dari koneksi biasa menjadi connection-pooling tidak berpengaruh besar terhadap keseluruhan kode program.

Tidak semua data yang kita ambil dari database akan digunakan. Kadangkala sebagian besar data yang diambil akan dibuang. Tentunya ini memboroskan kerja CPU, penggunaan memori, dan juga bandwidth. Beberapa ORM mendukung fitur lazy-loading, yaitu mengambil data kalau benar-benar diperlukan.

Bila aplikasi kita menggunakan ORM, perintah SQL ke database dieksekusi oleh ORM, bukan oleh kode program kita. Karena itu, ORM memiliki kebebasan untuk menjalankan SQL kapan saja, asal tidak mengacaukan aplikasi kita. Kebebasan ini digunakan oleh ORM untuk menunda dan mengumpulkan eksekusi SQL, sehingga bisa dijalankan secara efisien.

Pada aplikasi besar, umumnya application server dan database server diinstal di mesin yang berbeda. Untuk mengaksesnya digunakan akses melalui jaringan (network). Oleh karena itu, akses ke database harus diperhitungkan dengan teliti, agar tidak terlalu banyak perintah atau data yang “menyeberangi” jaringan. ORM senantiasa berusaha mengurangi komunikasinya dengan database, sehingga kinerja aplikasi dapat ditingkatkan.

Mengambil data dari memori selalu jauh lebih cepat daripada mengambil data dari disk atau jaringan. Karena kontrol penuh yang dimilikinya atas hubungan ke database, ORM dapat menggunakan cache dengan leluasa. ORM dapat mengamati data mana yang jarang berubah, dan kemudian akan menyimpannya di memori (cache). Bila kode program kita meminta data tersebut, ORM tidak perlu mengakses database, melainkan langsung mengambilnya dari cache. Ini akan sangat mempercepat eksekusi aplikasi.

Persiapan

Instalasi

Seperti library Java lainnya, JPA tidak membutuhkan instalasi

khusus. Cukup masukkan file `*.jar` yang dibutuhkan ke dalam CLASSPATH. NetBeans 6.1 sudah disertai dengan library implementasi JPA, yaitu toplink essensial. Kita tinggal menambahkan saja library toplink essential ke dalam project yang sudah kita buat.

Aplikasi Blog

Menjelaskan konsep saja terasa kurang menggigit apabila tidak disertai aplikasi. Dalam buku ini, kita akan membuat aplikasi blog, yaitu aplikasi untuk memuat tulisan atau artikel di website.

Fungsionalitas Aplikasi

Fitur dari aplikasi ini adalah:

- Menulis artikel
- Mengelompokkan artikel ke dalam kategori
- Memberikan komentar untuk masing-masing artikel
- Artikel memiliki informasi tentang penulis (author)
- Author dapat login ke dalam aplikasi

Dasar-dasar ORM

Mapping Sederhana

Untuk contoh awal, kita akan melakukan mapping untuk class `Category`. Class ini hanya memiliki tiga properti saja, `id`, `name`, dan `description`. Berikut adalah kode Javanya.

```
public class Category {
  private Integer id;
  private String name;
  private String description;

  // getter dan setter
}
```

Class ini akan kita simpan di database. Berikut struktur tabelnya, ditulis dalam dialek MySQL.

```
create table Category (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255),
  description VARCHAR(255)
);
```

Untuk kesederhanaan, kita samakan dulu nama tabel dan nama class. Demikian juga dengan nama property di kode Java dan nama kolom di tabel database. Pada bagian selanjutnya, kita akan lihat

bagaimana cara memetakan nama yang berbeda.

JPA mendukung dua cara untuk membuat pemetaan, dengan XML dan Annotation. Kita akan mengambil pendekatan Annotation, karena lebih mudah dan sederhana. Dengan pendekatan ini, kita tidak perlu memelihara dua file (Java dan XML). Berikut class `Category` yang sudah dianotasi.

```
@Entity
public class Category {
  @Id @GeneratedValue
  private Integer id;
  private String name;
  private String description;
}
```

Konfigurasi JPA

Setelah kita membuat tabel dan class Java, sekarang kita akan membuat konfigurasi JPA. NetBeans 6.1 mempunyai fasilitas yang memudahkan konfigurasi JPA.

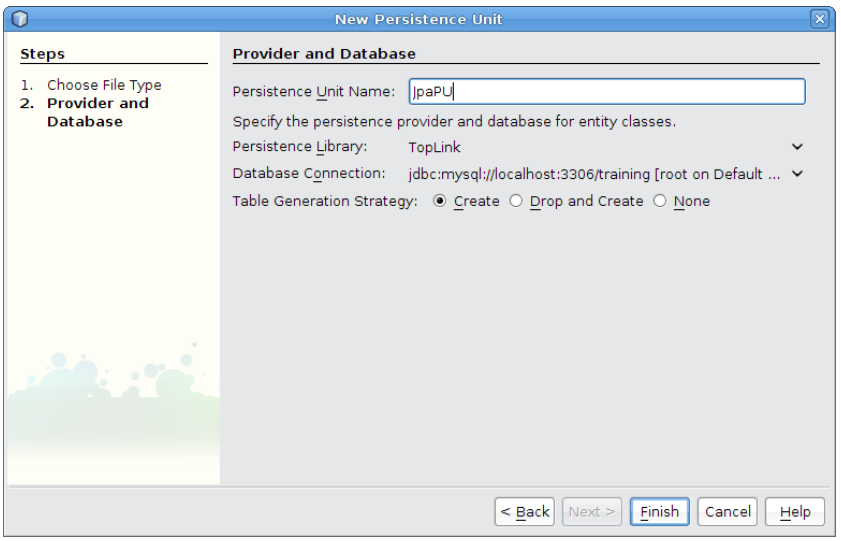
Langkah pertama adalah membuat koneksi ke dalam database MySQL dengan menggunakan database tools netbeans. Buka tab service, kemudian expand bagian database. Kalau kita belum mempunyai database, kita bisa membuat database terlebih dahulu dengan melakukan klik kanan di node mysql dan pilih menu create database, beri nama training. Kalau sudah ada database (schema) yang dibuat sebelumnya, node mysql dapat diexpand untuk melihat database tersebut.

Langkah berikutnya adalah membuat koneksi ke dalam database yang sudah dibuat. Expand node mysql, klik kanan di node database training, pilih menu connect. Jangan lupa untuk mencek pilihan “Remember Password”. Koneksi ini nanti akan kita gunakan untuk membuat Persistence Unit.

Persistence Unit adalah konfigurasi yang digunakan oleh JPA untuk meletakkan parameter username, password, url dan dialect yang digunakan untuk melakukan koneksi ke database.

Proses membuat Persistence Unit sangat mudah, pilih menu :

file → new file → persistence → persistence unit



Masukkan koneksi yang sudah dibuat pada langkah sebelumnya di bagian “Database Connection”. Kemudian pilih “Table Generation Strategy” agar ketika kita akan menjalankan aplikasi, semua table akan digenerate dari Entity. Kita akan membahas lebih lanjut tentang Entity di bab berikutnya, tidak perlu bingung dahulu, tenang saja.

Hasil dari Persistence Unit adalah file XML dengan mana persistence.xml dan terletak di package META-INF, isinya seperti berikut ini :

```
<?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/persistenc
e
http://java.sun.com/xml/ns/persistence/persistence_1_0.xs
d">
  <persistence-unit name="latihanPU" transaction-
type="RESOURCE_LOCAL">
    <provider>
       oracle.toplink.essentials.PersistenceProvider
    </provider>
    <class>com.artivisi.desktop.model.Group</class>
    <class>com.artivisi.desktop.model.Person</class>
    <properties>
      <property name="toplink.jdbc.user" value="root"/>
      <property name="toplink.jdbc.password" value=""/>
      <property name="toplink.jdbc.url"
       value="jdbc:mysql://localhost:3306/training"/>
      <property name="toplink.jdbc.driver"
          value="com.mysql.jdbc.Driver"/>
      <property name="toplink.ddl-generation"
          value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

file persistence.xml diatas sebaiknya dibuat dengan otomatis menggunakan tools dari netbeans, dan jangan dibuat secara manual dengan diketik, karena file xml tersebut case sensitif dan harus benar secara sintask, sehingga kalau diketik manual akan beresiko menghasilkan kesalahan ketik.

Menyimpan dan Mengambil Object

Sekarang, konfigurasi sudah siap. Kita akan mencoba untuk menyimpan dan mengambil object dari database dengan menggunakan JPA.

Demonstrasi sederhana dapat dibuat dengan class sederhana yang memiliki method `main`, sehingga bisa dijalankan. Berikut kerangkanya.

```
public class DemoJpa {
  public static void main(String[] xx){

  }
}
```

Selanjutnya, kode program kita akan dibuat dalam method `main`.

EntityManagerFactory

Pertama, kita harus membaca konfigurasi yang sudah kita buat tadi. Konfigurasi yang telah diaktifkan akan menghasilkan object `EntityManagerFactory`. `EntityManagerFactory` ini hanya dibuat sekali saja sepanjang aplikasi berjalan.

```
EntityManagerFactory entityManagerFactory=
Persistence.createEntityManagerFactory("JpaPU");
```

penting

Jangan terlalu lama menyimpan object EntityManager.

Biasanya, session dibuka di awal method, dan segera ditutup di akhir method.

EntityManager JPA tidak sama dengan koneksi database. Koneksi database dikelola oleh EntityManagerFactory.

Secara default, JPA akan mencari file yang bernama persistence.`xml` dalam META-INF.

Selanjutnya, kita instankan dulu object `Category` yang akan disimpan.

```
// buat object category yang mau disave
Category category = new Category();
category.setName("berita");
category.setDescription("berita hangat");
```

Menyimpan object ke database

Kita bisa langsung menyimpan object tersebut dengan menggunakan `EntityManager` JPA. Jangan lupa menggunakan

Transaction.

```
EntityManager entityManager =
entityManagerFactory.createEntityManager();
entityManager.getTransaction().begin();

entityManager.persist(category);

entityManager.getTransaction().commit();
entityManager.close();
```

EntityManager adalah object sekali pakai. Kita ambil EntityManager dari EntityManagerFactory, gunakan untuk operasi persistence, dan kemudian langsung ditutup.

Siklus hidup EntityManager berbeda dengan EntityManagerFactory. EntityManagerFactory umurnya sama dengan aplikasi, dibuat satu buah, pada saat aplikasi mulai dijalankan, dan digunakan terus sepanjang aplikasi berjalan.

Kita bisa membagi-pakai (share) EntityManagerFactory ke berbagai object yang berjalan berbarengan (concurrent), karena EntityManagerFactory dibuat secara *thread*-safe, sehingga aman untuk pemakaian *multi-threading*. Tapi kita tidak bisa menggunakan EntityManager di banyak objek sekaligus.

Setelah dijalankan, kode di atas akan memasukkan satu record ke tabel Category dalam database.

Mengambil data dari database

Object yang ada dalam database juga bisa diambil dengan cara yang tidak jauh berbeda. Bila kita sudah tahu ID object tersebut, kita bisa langsung mengambilnya dari database.

Kode untuk membuka EntityManager, menjalankan transaction, menyelesaikan transaction (commit), dan menutup EntityManager tidak ditulis, karena sama dengan di atas.

```
Category c = (Category) entityManager.createQuery(“select
c from Category where c.id=:id”).setParameter("id",
1).getSingleResult();

System.out.println("ID: "+c.getId());
System.out.println("Name: "+c.getName());
```

Tidak sulit bukan? Kita tidak perlu menulis SQL lagi. Tapi ini tidak berarti kita boleh tidak mengerti SQL. Seperti kita akan lihat pada bagian selanjutnya, pemahaman SQL sangat penting agar kita bisa menggunakan JPA dengan optimal.

Membuat skema database

Pada contoh sebelumnya, kita membuat tabel di database dengan menggunakan perintah SQL langsung di database. Sebetulnya, JPA juga bisa membuatkan tabel untuk kita. Tambahkan baris berikut dalam konfigurasi persistence.xml.

```
<property name="toplink.ddl-generation" value="create-
tables"/>
```

Ada beberapa opsi yang bisa digunakan untuk property toplink.ddl-generation ini, yaitu:

- drop-and-create-tables: dengan opsi ini, JPA akan membuat semua tabel pada saat inisialisasi EntityManagerFactory. Begitu EntityManagerFactory ditutup, semua tabel akan di-drop.
- create:-tables opsi ini hanya akan membuat t

  abel pada saat EntityManagerFactory diinisialisasi. JPA tidak men-drop tabel pada saat EntityManagerFactory ditutup.

Selanjutnya kita akan membuat class-class untuk data access object (DAO).

Membuat CategoryDao

Class Category akan memiliki beberapa method, diantaranya:

- save : untuk menyimpan object ke database
- findById : untuk mengambil object dari database berdasarkan ID yang diberikan

Nantinya CategoryDao tidak hanya memiliki dua method saja. Tapi untuk awal, cukup kita pindahkan apa yang ada di class DemoJpa di atas ke dalam CategoryDao.

Berikut kerangka CategoryDao.

```
public class CategoryDao {
  public void save(Category cat){}
  public Category findById(Integer id){}
}
```

Selanjutnya, CategoryDao butuh EntityManagerFactory untuk melakukan interaksi dengan database melalui JPA. Kita berikan EntityManagerFactory melalui *setter-injection*.

```
public class CategoryDao {
  private EntityManager entityManager;

  public void setEntityManager(
       EntityManager em) {
    this.entityManager = em;
  }
}
```

Menggunakan EntityManager

Selanjutnya, mari kita isi method `save`. Di dalam method ini harus ada logika untuk menentukan apakah operasi save yang akan dilakukan adalah insert atau update. Konvensi yang digunakan adalah: jika id dari entity masih null maka entity tersebut belum ada di dalam database, sehingga operasi insert yang akan dilaksanakan. Sebaliknya, jika idnya tidak null maka operasi update yang akan dilaksanakan:

```
public void save(Category cat) {
   if(cat.getId()==null){
       entityManager.persist(cat);
   } else {
       entityManager.merge(cat);
   }
}
```

method `findById`.

```
public Category findById(Integer id){
   return (Category) entityManager.createQuery(
       "select g from Group g where g.id=:id")
       .setParameter("id", id)
       .getSingleResult();
}
```

Client Code

Kita dapat membuat kode sederhana untuk mengetes kinerja `CategoryDao`. Buat class DemoDao yang memiliki method `main`. Isikan kode program berikut dalam method `main`.

```
EntityManagerFactory factory =
Persistence.createEntityManagerFactory(“JpaPU”);

CategoryDao categoryDao = new CategoryDao();
categoryDao.setEntityManager(factory.createEntityManager(
));

Category cat = new Category();
cat.setName("Tutorial JPA");
cat.setDescription("Belajar JPA");

categoryDao.save(cat);
```

Setelah dijalankan, coba periksa isi database.

Untuk mengambil data, kita juga bisa menggunakan method `findById`. Buat class `JpaDemo` lengkap dengan method `main`. Isi method `main` adalah sebagai berikut.

```
EntityManagerFactory factory =
Persistence.createEntityManagerFactory(“JpaPU”);

CategoryDao categoryDao = new CategoryDao();
categoryDao.setEntityManager(factory.createEntityManager(
));

Category cat = categoryDao.findById(1);
System.out.println("ID: " + cat.getId());
System.out.println("Name: " + cat.getName());
System.out.println("Description: "+cat.getDescription());
```

Kode di atas bila dijalankan akan menampilkan tulisan sebagai berikut:

```
ID: 1
Name: Tutorial JPA
Description: Belajar JPA
```

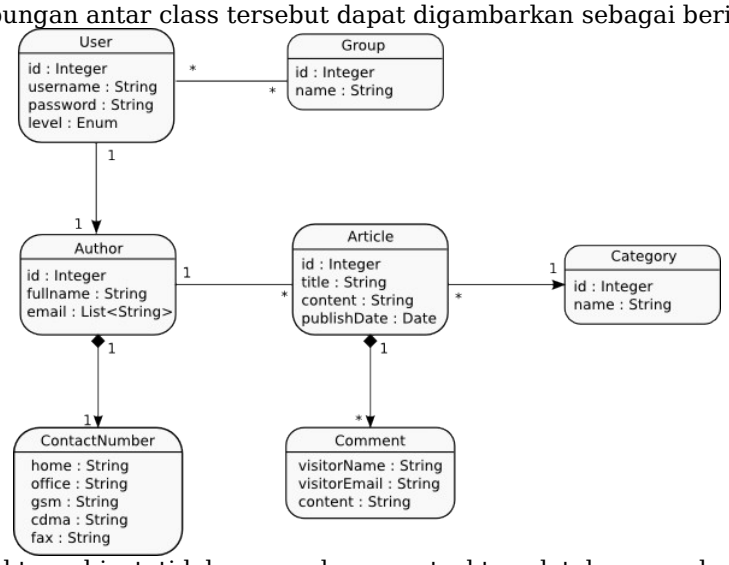
Pemetaan Relasi

Selanjutnya, setelah kita dapat mengkonfigurasi dan melakukan test terhadap kerja JPA, sekarang kita bisa melangkah ke topik yang lebih bermanfaat, yaitu membuat mapping yang lebih beragam.

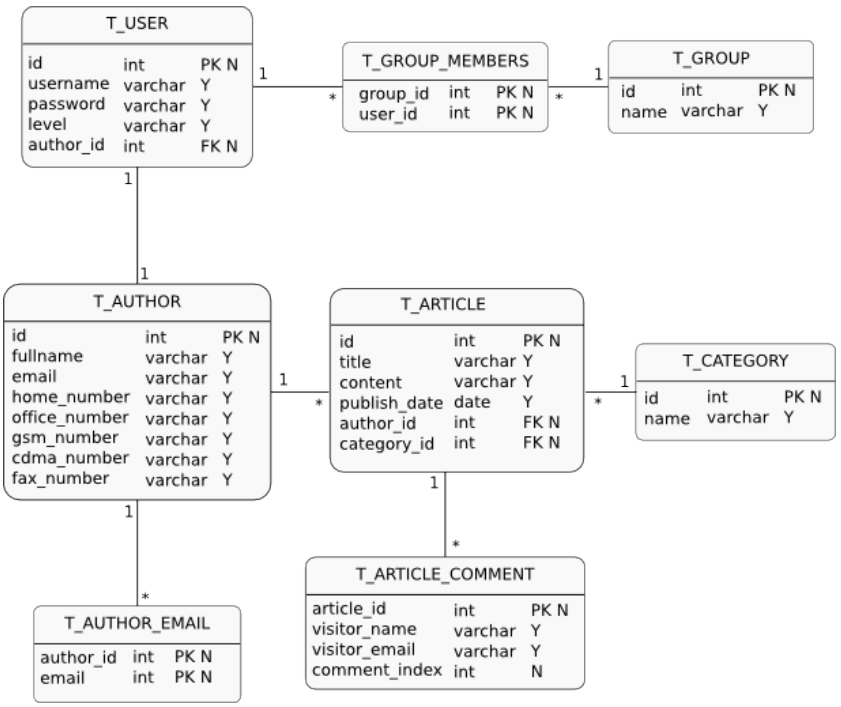
Pada bab ini, kita akan memetakan seluruh struktur database aplikasi blog kita. Berikut class-class yang akan terlibat:

- `User` : mencerminkan semua account yang dapat login ke aplikasi.
- `Author` : mencerminkan penulis yang mengarang artikel atau menulis berita. Satu `author` memiliki satu `user`, dan sebaliknya. Kita memisahkan class `User` dan `Author`, karena `User` berisi informasi tentang security, ijin akses, password, dan informasi keamanan lainnya. Sedangkan class `Author` menyimpan informasi berkaitan tentang informasi personal, seperti nama lengkap, tanggal lahir, dan sebagainya.
- `Group` : berisi banyak `User`, gunanya nanti untuk memudahkan pengaturan ijin akses.
- `Category` : mencerminkan kelompok artikel. Ada kategori berita, gosip, opini, dan sebagainya
- `Article`
- `Comment` : komentar atas artikel tertentu

Hubungan antar class tersebut dapat digambarkan sebagai berikut:



Struktur object tidak sama dengan struktur database, walaupun agak mirip. Berikut adalah struktur tabel dalam database.



Sekarang, setelah kita mengetahui desain objek dan desain database, mari mulai melakukan pemetaan.

Pertimbangan dalam pemetaan

Seperti sudah disinggung sekilas dalam bagian sebelumnya, ada beberapa ketidak sesuaian (mismatch) antara desain objek dan desain database. Hubungan antar tabel di database jauh lebih sederhana daripada hubungan antar class di aplikasi berorientasi objek.

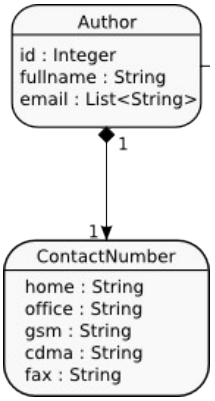
Dalam kaitannya dengan hubungan antar class, ada beberapa hal yang harus kita perhatikan, yaitu:

- kasta class
- navigasi
- jenis collection
- optionality
- transitive persistence

Mari kita bahas satu persatu.

Kasta

JPA mengenal tiga kasta object: entity, component, dan value type. Mari kita lihat contohnya.



Pada domain model di atas, kita lihat seorang `Author` memiliki informasi nomer yang bisa dihubungi. Karena jaman sudah semakin canggih, satu orang bisa dihubungi di banyak nomer, yaitu nomer telepon rumah, kantor, ponsel GSM, ponsel CDMA, dan juga nomer fax. Untuk mengelompokkan semua nomer ini, kita membuat satu class khusus yaitu `ContactNumber`.

Pemetaan objek seperti ini disebut dengan istilah *fine-grained*. Artinya, informasi dikelompokkan secara detail dan spesifik. Tiap class memiliki informasi yang terinci. Karena class adalah tipe data, bisa dikatakan bahwa kita telah membuat tipe data baru, yaitu `ContactNumber`.

Pertanyaannya, bagaimana kita akan menyimpan data ini ke database? Database tidak mengenal tipe data khusus. Kita hanya bisa menggunakan tipe data yang telah disediakan seperti INT, atau VARCHAR. Kita tidak bisa membuat tipe data baru `ContactNumber` dalam database.

Ada tiga alternatif yang bisa kita gunakan untuk memetakan `ContactNumber` ke database. Tiga alternatif ini mencerminkan kasta object dalam JPA.

1. Kasta tertinggi, adalah Entity object. Sebuah Entity memiliki tabel sendiri, primary key sendiri, dan kehidupan sendiri. Class `Author` adalah sebuah Entity.
2. Kasta di bawah Entity disebut Component. Dia memiliki

tabel sendiri dan kadang-kadang memiliki primary key sendiri. Tetapi, Component hidup menempel pada Entity dan tidak bisa berdiri sendiri. Kalau inang Entitynya dihapus, maka Component akan ikut dihapus dari database. Class `ContactNumber` adalah contoh Component.

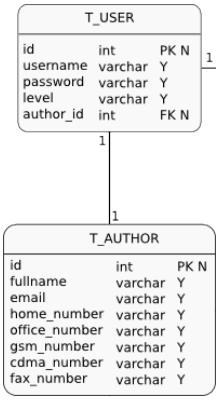
3. Kasta paling rendah disebut Value Type. Sama dengan Component, hidupnya menempel pada Entity. Value Type tidak memiliki tabel sendiri. Informasi yang dimilikinya disimpan pada tabel induknya. Daftar email yang dimiliki `Author` (`List<String> emails`) adalah contoh Value Type.

Ada beberapa kriteria yang bisa digunakan untuk menentukan kasta suatu object. Dengan menggunakan kriteria ini, kita dapat memetakan desain object ke desain relasional dengan baik. Berikut kriterianya:

1. Shared Reference.

   Apakah object tertentu mempunyai referensi ke object lain? Pada contoh kita di atas, class `Author` memiliki hubungan dengan class `User` dan class `Article`. Oleh karena itu, jelas bahwa class `Author` akan menjadi Entity.

   Di database, tabel `Author` dan `User` akan terlihat seperti ini.



   Class `ContactNumber` hanya digunakan oleh class `Author`. Class ini bahkan tidak memiliki referensi ke class induknya, yaitu class `Author`. Ini ditunjukkan oleh hubungan composition antara class `Author` dan `ContactNumber`.

   Oleh karena itu, data `ContactNumber` dimasukkan ke dalam tabel yang sama dengan data `Author`, seperti dapat dilihat pada gambar di atas.

2. Kehidupan.

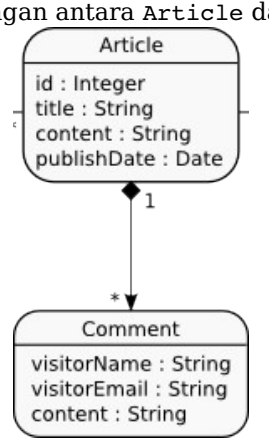
Apakah suatu object memiliki kehidupan yang terpisah dari object yang berhubungan dengannya? Data `Author` masih akan tetap tersimpan walaupun `Article` yang ditulisnya dihapus.

Tidak demikian halnya dengan `ContactNumber`. Begitu data `Author` dihapus, data `ContactNumber` tidak memiliki relevansi lagi.

Object yang memiliki kehidupan sendiri merupakan kandidat Entity.

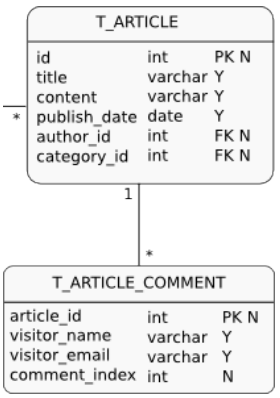
3. Jumlah

   Untuk memahami pentingnya jumlah dalam pemetaan, mari kita lihat hubungan antara `Article` dan `Comment`.



   Satu `Article` memiliki beberapa `Comment`. Sama dengan `ContactNumber`, kehidupan `Comment` juga sangat tergantung pada `Article`. Kalau `Article` dihapus, `Comment` tidak lagi relevan. Apa yang mau dikomentari kalau artikelnya tidak ada?

   Satu `Author` hanya memiliki satu `ContactNumber`, tapi satu `Article` memiliki banyak `Comment`. Perbedaan jumlah ini akan menentukan susunan tabelnya. Berikut susunan tabel `Article` dan `Comment`.



Dengan mempertimbangkan ketiga hal di atas, berikut adalah kasta untuk masing-masing object dalam desain domain model kita.

Class	***Kasta***
Article	Entity
Author	Entity
Category	Entity
Comment	Component
ContactNumber	Value Type
Group	Entity
User	Entity
UserLevel	Value Type

Navigasi

Kita harus memikirkan urusan arah navigasi (direction) dalam merancang pemetaan. Dua class dikatakan memiliki navigasi dua arah (bidirectional) apabila kita dapat mengakses class yang satu dari class yang lain, dan sebaliknya. Sebagai contoh, `User` dan `Group` mempunyai hubungan bidirectional, jadi kode Javanya kira-kira seperti ini.

```
public class User {
  private Set<Group> groups = new HashSet<Group>();
}
```

sedangkan class `Group` berisi seperti ini

```
public class Group {
  private Set<User> members = new HashSet<User>();
}
```

`User` dan `Author` unidirectional, artinya dari class `User` kita dapat mengakses class `Author`. Tapi dari class `Author` kita tidak dapat mengakses class `User`. Berikut kode Javanya.

```
public class User {
  private Author author;
}
```

dalam class `Author`, tidak ada property `User`.

```
public class Author {
  // tidak ada property User
}
```

Urusan navigasi ini hanya ada di kode Java. Apapun desain navigasi yang kita buat, tidak mempengaruhi struktur tabel kita.

Jenis Collection

Dalam desain kita di atas, ada beberapa hubungan one-to-many dan many-to-many. Urusan many ini direpresentasikan dalam Java dengan menggunakan Collections Framework.

Kita mengenal beberapa interface dalam Collections Framework dengan sifatnya masing-masing:

- `Set` : Objek yang disimpan dalam `Set` tidak boleh ada yang sama. `Set` tidak bisa mengingat urutan. Jadi kita tidak bisa mengharapkan objek yang dimasukkan pertama akan muncul pertama juga. `Set` biasanya diinisialisasi dengan menggunakan implementasi `HashSet`.
- `List` : Berbeda dengan `Set`, `List` mampu mengingat urutan. Perilakunya mirip dengan array. Kita dapat mengatur urutan objek yang disimpan di dalamnya. Objek `List` bisa diurutkan melalui perintah `order by` dalam SQL, atau dengan adanya nilai index, yang menunjukkan urutan objek di dalam `List`. Biasanya `List` diinisialisasi dengan implementasi `ArrayList`.
- `SortedSet` : Interface ini digunakan bila kita ingin menyimpan objek yang tidak boleh sama, tapi menginginkan ada urutan tertentu. Kita dapat memasang `Comparator` ke dalam `SortedSet` untuk mengatur urutan penyimpanan objek. Biasanya `SortedSet` diinisialisasi menggunakan `TreeSet`.
- `Map` : Interface ini menyimpan pasangan key dan value. Biasanya diinisialisasi menggunakan `HashMap`.

Kita memiliki beberapa hubungan many dalam domain model kita. Berikut adalah pemilihan jenis collection beserta pertimbangannya.

Relasi	***Collection***	***Pertimbangan***
Author – Email	Set<String>	alamat email tidak boleh terduplikasi. Business requirement tidak mengharuskan adanya prioritas tertentu antar email
Author – Article	List<Article>	kita ingin menampilkan daftar artikel berdasarkan tanggal terbitnya.
Article – Comment	List<Comment>	urutan masuknya komentar sangat penting, karena terjadi diskusi antar pengunjung. Untuk memastikan urutan, kita menggunakan List dengan nilai index.
User – Group	Set<Group>	Satu user tidak boleh tercatat dua kali sebagai member group yang sama
Group – User	Set<User>	Satu group tidak boleh diikuti dua kali oleh satu user yang sama

Optionality

Masalah optionality ini hanya ada di sisi database. Dari sudut pandang desain object, optionality tidak menjadi pertimbangan.

Mari kita lihat lagi relasi antara `User` dan `Author`. Pada kasus kita, satu `User` pasti memiliki satu `Author`, dan sebaliknya. Desain seperti ini direalisasikan di database dengan DDL berikut:

```
create table T_AUTHOR (
  id INT PRIMARY KEY AUTO_INCREMENT
);

create table T_USER (
  id INT PRIMARY KEY AUTO_INCREMENT,
  author_id INT NOT NULL
);
```

Seperti kita lihat, tabel `T_USER` di atas memiliki foreign key `author_id`. Kolom `author_id` tersebut diberikan atribut `NOT NULL` karena dia adalah foreign key, sehingga harus diisi.

Desain seperti ini akan menjadi masalah jika business requirement menentukan bahwa `User` belum tentu punya informasi `Author` (*optional*). Jika menggunakan desain di atas, maka kita terpaksa

menghilangkan atribut `NOT NULL` pada `author_id`.

Cara ini kurang elegan, karena foreign key tidak diperuntukkan agar bisa diisi `NULL`. Di beberapa merek database, foreign key dilarang `NULL`. Cara yang lebih baik adalah dengan menggunakan join table. Skema database dengan join table menjadi seperti ini.

```
create table T_AUTHOR (
  id INT PRIMARY KEY AUTO_INCREMENT
);

create table T_USER (
  id INT PRIMARY KEY AUTO_INCREMENT
);

create table T_USER_AUTHOR (
  user_id INT,
  author_id INT,
  PRIMARY KEY(user_id, author_id)
);
```

Kesimpulannya, kita harus mempertimbangkan faktor optionality dalam mendesain struktur tabel database.

Transitive Persistence

Hal lainnya yang harus dipertimbangkan dalam melakukan pemetaan adalah efek suatu operasi entity (save, update, delete) terhadap entity lain yang terhubung dengannya.

Misalnya, author Endy menulis artikel baru. Berikut kode program untuk menyimpannya ke database.

```
Author endy = entityManager.createQuery("select a from
       Article a where a.id=:id")
       .setParameter("id",100).getSingleResult();

Article tutorial = new Article();
tutorial.setTitle("Belajar JPA");

tutorial.setAuthor(endy);
endy.getArticles().add(tutorial);

entityManager.persist(endy);
```

Pada kasus di atas, class `Article` adalah Entity, jadi JPA tidak otomatis melakukan insert. Jika dijalankan, kode di atas akan menghasilkan error, yang menyebutkan bahwa object `tutorial` belum ada di database, sehingga tidak bisa dihubungkan dengan object `endy`.

Agar tidak error, kita harus menjalankan save terhadap object `tutorial` terlebih dulu.

```
entityManager.persist(tutorial);
entityManager.persist(endy);
```

Kita dapat menghilangkan perintah save terhadap object `tutorial` dengan mengaktifkan transitive persistence, yaitu `cascade-save`. Dengan opsi ini, begitu object `endy` di-save, JPA akan melakukan save terhadap semua object yang terhubung dengan object `endy`.

Transitive persistence juga dapat diaplikasikan untuk update dan delete.

Setelah kita memahami pertimbangan dalam membuat mapping, mari sekarang kita buat deklarasi pemetaannya.

Enum Type

Kita mulai dari yang paling sederhana, yaitu tipe data enum. Dalam domain model kita, enum digunakan untuk menentukan ijin akses user. Mari kita lihat class `User`.

```
public class User {
  private Integer id;
  private String username;
  private String password;
  private UserLevel level;
}
```

Class `User` ini menggunakan fitur baru dalam Java 5, yaitu Typesafe Enum. Adapun enum `UserLevel` tidak sulit, kodenya seperti ini.

```
public enum UserLevel {
  CONTRIBUTOR, EDITOR, ADMINISTRATOR
}
```

Kita memiliki dua pilihan bagaimana `UserLevel` ini akan disimpan dalam tabel, yaitu:

- ordinal : menyimpan nilai index dari enum tersebut. CONTRIBUTOR akan disimpan dengan nilai 1, EDITOR dengan nilai 2, dan ADMINISTRATOR dengan nilai 3. Ordinal adalah modus default.
- string : menyimpan nilai teks enum. Di database, nilainya akan disimpan sebagai tipe data `enumeration` bila tersedia, atau `varchar`.

Untuk memetakan enum, cukup berikan anotasi `@Enumerated`.

```
public class User {
  private Integer id;
  private String username;
  private String password;

  @Enumerated(value = EnumType.STRING)
  private UserLevel level;
}
```

Value Type

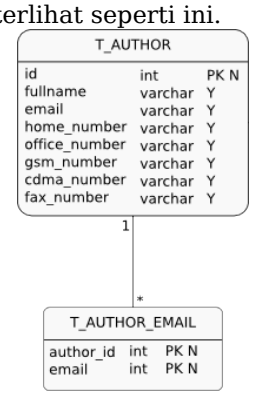
Kita memiliki satu value type, yaitu `List<String> emails` pada class `Author`. Berikut pemetaannya.

```
public class Author {

  @CollectionOfElements(targetElement = String.class)
  @JoinTable(
    name = "T_AUTHOR_EMAIL",
    joinColumns = {@JoinColumn(name = "author_id")}
  )
  @Column(name = "email")
  private Set<String> emails = new HashSet<String>();

}
```

Dengan informasi mapping di atas, tabel `T_AUTHOR` dan `T_AUTHOR_EMAIL` akan terlihat seperti ini.



Pada `T_AUTHOR_EMAIL` terlihat bahwa kedua kolom (`author_id` dan `email`) dibuat menjadi primary key. Ini adalah konsekuensi dari tipe collection `Set`, yang tidak mengijinkan duplikasi di dalamnya.

One to One

One to One menghubungkan dua entity. Kita menggunakan hubungan ini untuk `Author` dan `User`. Yang perlu diperhatikan hanyalah class `User`, karena navigasinya cuma satu arah. Berikut class `User`.

```
public class User {
    @OneToOne
    @JoinColumn(name = "author_id", nullable = false)
    private Author author;
}
```

Kita mengatur kolom foreign key `author_id` agar tidak boleh `null`. Ini berkaitan dengan masalah optionality yang sudah dijelaskan di atas.

Many to One

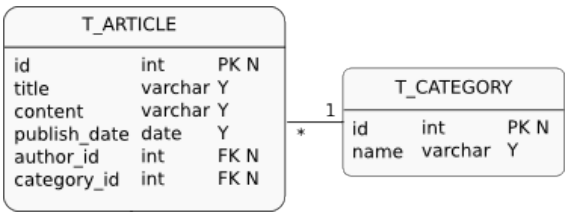
Pada contoh aplikasi kita, mapping Many to One ada pada hubungan antara class `Article` dan `Category`. Navigasinya satu arah, dari `Article` ke `Category`. Dengan demikian, pada class `Category`, tidak ada property `Article`.

```
public class Category {
    @Id @GeneratedValue
    private Integer id;
    private String name;
    private String description;
}
```

Pada class `Article`, kita mendeklarasikan property `Category`, dengan mapping `@ManyToOne` sebagai berikut.

```
public class Article {
    @ManyToOne
    @JoinColumn(name = "category_id", nullable = false)
    private Category category;
}
```

Deklarasinya tidak jauh berbeda dengan `@OneToOne`. Pemetaan di atas mencerminkan skema database sebagai berikut.

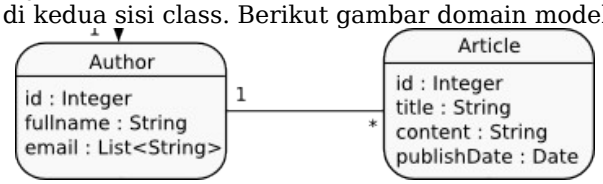


One to Many

List dengan order by

Kita memiliki beberapa kasus One To Many. Pertama, kita lihat dulu tipe collection `List` yang menggunakan sorting di sisi database melalui perintah `order by`.

Hubungan ini ada antara `Author` dan `Article`. Karena hubungannya dua arah (bidirectional), maka kita akan melihat mapping di kedua sisi class. Berikut gambar domain modelnya.



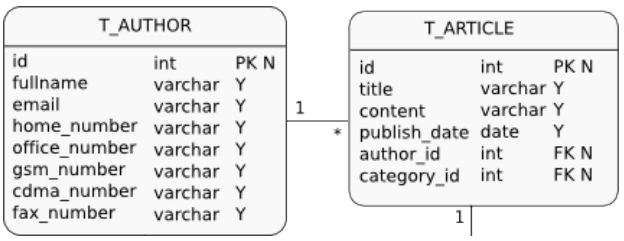
Di sisi `Article`, mapping Many-to-One `Author` sama dengan `Category` yang kita sudah buat sebelumnya.

```
public class Article {
    @ManyToOne
    @JoinColumn(name = "author_id", nullable = false)
    private Author author;
}
```

Di sisi `Author`,

```
public class Author {
    @OneToMany(mappedBy = "author")
    @OrderBy("publishDate")
    private List<Article> articles
         = new ArrayList<Article>();
}
```

Pemetaan di atas akan menghasilkan struktur tabel database sebagai berikut.



Kita menggunakan anotasi `@OrderBy` untuk menentukan urutan munculnya `Article`. Jika anotasi tersebut kita hilangkan, JPA akan mengurutkannya berdasarkan `id`.

Pada mapping di atas, kita menggunakan keyword `mappedBy`. Keyword ini dikenal juga dengan atribut `inverse=true` pada mapping XML.

Keyword `inverse` atau `mappedBy` memberitahukan JPA bahwa isi `List` adalah Entity object, bukan Component. Dengan demikian, kalau object `Author` dihapus, JPA tidak akan otomatis menghapus juga `Article`.

penting

JPA mampu membuat pemetaan satu arah dari Author ke Article, bahkan tanpa adanya property author di sisi Article.

Pemetaan satu arah ini tetap dilakukan dengan adanya foreign key author_id di T_ARTICLE

Tapi konsekuensinya, JPA tidak tahu apakah foreign key author_id tersebut sudah dipetakan atau belum.

Ini sebabnya mengapa kita harus menggunakan keyword inverse.

Selain mengatur masalah ketergantungan, keyword `inverse` ini juga mengatur eksekusi SQL. Mari kita bahas internal JPA sedikit.

Asosiasi dua arah antara `Author` dan `Article` sebetulnya terdiri dari dua asosiasi satu arah, yaitu dari `Author` ke `Article`, dan sebaliknya. Di dalam database, asosiasi ini diimplementasikan dengan foreign key `author_id` yang ada dalam tabel `T_ARTICLE`. Dengan demikian, satu hubungan foreign key dipetakan dua kali oleh JPA.

Konsekuensinya, dua mapping ini akan menghasilkan dua kali eksekusi SQL. Coba perhatikan skenario berikut.

```
Author endy = entityManager.createQuery("select a from
       Article a where a.id=:id")
       .setParameter("id",100).getSingleResult();


Article tutorial = new Article();
tutorial.setTitle("Belajar JPA");

tutorial.setAuthor(endy);
endy.getArticles().add(tutorial);

entityManager.persist(tutorial);
entityManager.persist(endy);
```

Rangkaian kode program di atas akan menghasilkan eksekusi SQL sebagai berikut:

```
select * from T_AUTHOR where id=100;

insert into T_ARTICLE (title, author_id)
values ('Belajar JPA', 100);

update T_ARTICLE set author_id=100
where title='Belajar JPA';
```

Kita lihat di atas, JPA menjalankan SQL update, padahal sebelumnya `author_id` sudah diset menurut `id` authornya. Ini disebabkan karena foreign key `author_id` dipetakan dua kali.

JPA tidak bisa mendeteksi ini secara otomatis, karena tidak memungkinkan secara logika. Jadi, kita harus memberi tahu JPA bahwa `author_id` sudah dipetakan oleh class Article.

Setelah kita menggunakan keyword `mappedBy`, kode program di atas akan menghasilkan eksekusi SQL sebagai berikut.

```
select * from T_AUTHOR where id=100;

insert into T_ARTICLE (title, author_id)
values ('Belajar JPA', 100);
```

Keyword `mappedBy` menyebabkan JPA mengabaikan pemetaan di sisi Author. Yang dipantau oleh JPA hanyalah sisi Article.

Dengan demikian, bila kita save seperti ini:

penting

Inverse tidak berhubungan dengan cascade.

Cascade hanya menyuruh JPA untuk memeriksa apakah entity lain yang terhubung sudah di-save/update/delete atau belum

```
Article tutorial = new Article();
endy.getArticles().add(tutorial);
entityManager.persist(endy);
```

Object `tutorial` tidak akan tersimpan ke database. Untuk memperbaikinya, kita lakukan seperti ini:

```
Article tutorial = new Article();
tutorial.setAuthor(endy)
entityManager.persist(tutorial);
```

Bila kita sudah mengaktifkan transitive persistence, maka kita bisa melakukannya seperti ini:

```
Article tutorial = new Article();
tutorial.setAuthor(endy)
entityManager.persist(endy);
```

Bagaimana jika kita ingin membalik sisi inverse? Misalnya, kita ingin agar JPA mengabaikan sisi `Article`.

Keyword `mappedBy` atau `inverse` tidak dapat digunakan di sisi `@ManyToOne`. Untuk melakukan hal ini, kita gunakan keyword `updateble` dan `insertable`. Di sisi `Article`, kita tambahkan `updatable` dan `insertable`.

```
public class Article {
    @ManyToOne
    @JoinColumn(
      name = "author_id", nullable = false
      updatable = false, insertable = false
    )
    private Author author;
}
```

Di sisi `Author`, kita hilangkan saja `mappedBy`.

```
public class Author {
    @OneToMany
    @OrderBy("publishDate")
    private List<Article> articles
         = new ArrayList<Article>();
}
```

Many to Many

Bila kita sudah menguasai pemetaan One-to-Many, maka Many-to-Many tidak sulit. Perbedaannya hanya terletak di anotasinya. Yang tadinya `@OneToMany`, diganti menjadi `@ManyToMany`.

Sama dengan One-to-Many, kita harus menentukan sisi mana yang

inverse. Pada contoh kali ini, sisi inverse terletak pada class `User`. Berikut isi class `User`.

```
public class User {
    @ManyToMany(mappedBy = "members")
    private Set<Group> groups = new HashSet<Group>();
}
```

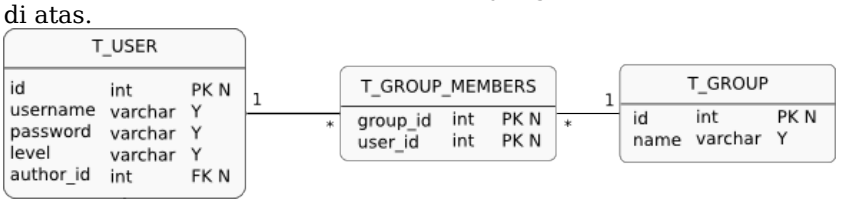
Berikut pemetaan di class `Group`.

```
public class Group {

  @ManyToMany
  @JoinTable(
     name = "T_GROUP_MEMBERS",
     joinColumns = {@JoinColumn(name = "group_id")},
     inverseJoinColumns = {@JoinColumn(name = "user_id")}
  )
  private Set<User> members = new HashSet<User>();
}
```

Pemetaan many-to-many membutuhkan tabel perantara. Anotasi `@JoinTable` digunakan untuk mengatur nama tabel dan kolom dalam tabel perantara. Jika kita menghilangkan anotasinya, maka JPA akan mencari tabel bernama `T_GROUP_T_USER` dengan nama kolom `groups_id` dan `members_id`.

Berikut adalah struktur tabel database yang dipetakan oleh anotasi di atas.



Parent-Child

Terakhir, kita akan membuat pemetaan parent-child. Ini merupakan pemetaannya yang sering kita gunakan. Pemetaan ini biasanya berbentuk one-to-many. Sisi one menentukan kondisi sisi many. Bila terjadi save, update, atau delete pada sisi one, maka sisi many akan terpengaruh.

Sebagai contoh, kita akan menggunakan relasi parent-child ini pada hubungan antara `Author` dan `Article`. Bila `Author` dihapus, maka seluruh `Article` yang pernah ditulisnya juga akan ikut terhapus.

Untuk mengaktifkan parent-child ini, kita ubah pemetaan di sisi `Author` sebagai berikut.

```
public class Author {
    @OneToMany(
      mappedBy = ”author”,
      cascade = CascadeType.ALL
    )
    @OrderBy("publishDate")
    private List<Article> articles
         = new ArrayList<Article>();
}
```

Pada contoh di atas, kita mengaktifkan fitur cascade untuk semua jenis operasi, yaitu save, update, dan delete. Bila kita hanya ingin cascade untuk operasi save dan delete, anotasinya menjadi seperti ini.

```
public class Author {
    @OneToMany(
      mappedBy = ”author”,
      cascade = {CascadeType.PERSIST, CascadeType.REMOVE}
    )
    @OrderBy("publishDate")
    private List<Article> articles
         = new ArrayList<Article>();
}
```

Mengambil Objek dari Database

JPA menyediakan banyak cara untuk mengambil data dari database. Berbagai pilihan cara ini disesuaikan dengan preferensi programmer. Ada orang yang lebih menyukai gaya SQL karena lebih ringkas dan mudah. Ada juga yang lebih menyukai gaya programmatic berorientasi objek karena type-safe. Beberapa programmer yang merupakan alumni Microsoft Access juga bisa ikut menikmati JPA dengan Query By Example.

JPA Query Language (JPAQL)

JPAQL adalah bahasa query JPA yang mirip dengan SQL. Bedanya, dalam JPAQL kita menggunakan nama object dan property, bukan nama tabel dan kolom.

Query Sederhana

Untuk contoh pertama, mari kita implementasikan

ArticleDao.findAll().

```
public class ArticleDao {
  public List<Article> findAll() {
    return entityManager.createQuery(
       “selct a from Article a”).getResultList();
  }
}
```

Query di atas akan menghasilkan semua object `Article` yang ada dalam database.

Pagination

Bila aplikasi kita menyimpan ribuan artikel, tentunya kita ingin mengambil data secara bertahap, halaman per halaman (paging). SQL yang digunakan untuk pagination ini biasanya berbeda antar database. Dengan JPA, kita bisa menggunakan satu JPAQL saja. JPA nantinya akan menerjemahkan sesuai dengan merek database yang kita gunakan.

Untuk mengambil artikel per halaman, kita overload method `findAll` agar menerima parameter `startIndex` dan `pageSize`. Parameter `startIndex` adalah index yang menunjukkan record pertama dalam halaman, sedangkan `pageSize` menunjukkan jumlah record per halaman.

Berikut overloading method `findAll`.

```
public List<Article> findAll
   (int startIndex, int pageSize){
        return entityManager
                .createQuery("select a from Article a")
                .setFirstResult(startIndex)
                .setMaxResults(pageSize)
                .getResultList();
    }
```

Parameter Binding

Bagaimana kalau kita mau mengirim parameter ke query? JPA mendukung parameter binding, mirip dengan `PreparedStatement` di JDBC.

Bila kita ingin mengambil artikel yang ditulis pengarang tertentu, berikut kodenya.

```
public List<Article> findByAuthor(Author author) {
  String query = "from Article a ";
  query += " where a.author.id = :authorId ";

  return entityManager
        .createQuery(query)
        .setParameter("authorId", author.getId())
        .getResultList();

}
```

Selain itu, kita juga bisa menggunakan parameter bertipe `java.util.Date`. Contohnya bila kita ingin mendapatkan artikel yang diterbitkan setelah tanggal tertentu. Buat method `articlesPublishedAfter` dalam `ArticleDao`.

```
public List<Article> articlesPublishedAfter
                              (Date publishDate) {
  String query = "from Article a ";
  query += "where a.publishDate > :publishDate";

  return entityManager.createQuery(query)
         .setParameter("publishDate", publishDate)
         .getResultList();
}
```

Named Query

Kita bisa mengeluarkan JPAQL dari dalam objek `DAO` dan memindahkannya ke class Entity. Dari class `DAO`, kita cukup memanggil nama query tersebut. Fitur ini disebut Named Query.

Misalnya, kita ingin mengambil artikel berdasarkan kategori. Berikut JPAQL-nya.

```
from Article a where a.category.id = :categoryId
```

JPAQL ini bisa kita letakkan dalam class `Article`. Berikut deklarasi class `Article`.

```
@Entity @Table(name = "T_ARTICLE")
@NamedQueries({
  @NamedQuery(
     name = "articleByCategory",
     query = "from Article a "+
             "where a.category.id = :categoryId"
  )
})
public class Article { }
```

Named Query ini dipanggil dari `ArticleDao` dalam method `findByCategory` seperti ini.

```
public List<Article> findByCategory(Category category) {
  return entityManager
        .createNamedQuery("articleByCategory")
        .setParameter("categoryId", category.getId())
        .getResultList();
}
```

Restriction

Kita menggunakan restriction untuk membatasi jumlah record yang dihasilkan. Mirip dengan SQL, kita menggunakan keyword `where` untuk membatasi hasil query.

Contohnya, kita mengambil artikel yang diterbitkan setelah tanggal tertentu. JPAQLnya tampak seperti ini:

```
from Article a where a.publishDate > :publishDate
```

Hampir semua operator yang biasa kita gunakan di Java dapat digunakan. Berikut tabel operator yang didukung, diurutkan berdasarkan prioritas eksekusinya:

Operator	***Penjelasan***
.	navigasi object
+ -	untuk membuat menjadi positif/negatif
* /	perkalian dan pembagian
+ -	penjumlahan dan pengurangan
= <>	sama dengan atau tidak sama dengan
<, >, <=, >=	perbandingan
[NOT] BETWEEN, [NOT] LIKE, [NOT] IN, IS [NOT] NULL	perbandingan, mirip dengan sintaks SQL
IS [NOT] EMPTY, [NOT] MEMBER [OF]	pengecekan collection

NOT, AND, OR	operator logika

Misalnya, bila kita ingin mengambil artikel yang bukan kategori Berita, query-nya seperti ini:

```
String jpaql = “from Article a “;
jpaql += “where a.category <> :category”;

List<Article> hasil =
entityManager.createQuery(jpaql).getResultList();
```

Kita menggunakan karakter `%` untuk menggantikan karakter apa saja dalam jumlah berapa saja. Misalnya, kita ingin mencari artikel yang diawali dengan huruf A, maka kita gunakan JPAQL seperti ini:

```
from Article a where a.title like 'A%'
```

Lalu bagaimana kalau kita ingin mencari artikel yang diawali dengan karakter `%`?

Gunakan escape character `\`, sehingga JPAQLnya menjadi seperti ini:

```
from Article a where a.title like '\%'
```

Bila kita menggunakan lebih dari satu persyaratan, gunakan keyword `and` atau `or` sesuai kebutuhan. Contohnya, kita ingin mencari artikel yang judulnya diawali huruf A, dan nama pengarangnya juga diawali dengan huruf A.

```
from Article a where a.title like 'A%'
and a.author.fullname like 'A%'
```

Contoh lain, bila kita ingin mencari artikel yang diawali huruf A dan juga huruf B, JPAQLnya seperti ini.

```
from Article a where a.title like 'A%'
or a.title like 'B%'
```

Kita bisa mengurutkan hasil pencarian dengan menggunakan keyword `order by`, sama seperti SQL.

Berikut contoh pengurutan artikel berdasarkan judulnya.

```
String jpaql = “from Article a “;
jpaql += “order by a.title asc ”;

List<Article> hasil =
entityManager.createQuery(jpaql).getResultList();
```

Kita menggunakan keyword `asc`, yang artinya ascending untuk mengurutkan dari kecil ke besar. Untuk urutan sebaliknya, gunakan keyword `desc`, yang artinya descending.

Query untuk Collection

Ada beberapa keyword yang dapat digunakan untuk memanipulasi collection. Misalnya, kita ingin mencari group yang tidak punya member.

```
from Group g where g.members is empty
```

Atau mencari user yang tergabung dalam grup dengan nama tertentu.

```
from User u, Group g where g.name=? and u member of
g.members
```

Projection

Bila tadi kita sudah menentukan sumber data dengan keyword `from` dan batasan hasil dengan keyword `where`, sekarang kita bisa menentukan format hasil yang dikeluarkan dengan keyword `select`.

Mari kita lihat lagi query user yang menjadi anggota grup.

```
from User u, Group g where g.name=? and u member of
g.members
```

Pada query di atas, kita bisa memilih untuk hanya mengambil data user saja dengan menggunakan `select`. Query di atas bisa ditulis menjadi seperti ini:

```
select u
from User u, Group g
where g.name=? and u member of g.members
```

Dengan query seperti itu, kita bisa membuat method di class `GroupDao` sebagai berikut:

```
public List<User> membersOf(String groupName) {
  String jpaql = "select u ";
  jpaql += "from User u, Group g ";
  jpaql += "where g.name = :groupName ";
  jpaql += "and u member of g.members';

  return entityManager
         .createQuery(jpaql)
         .setString("groupName", groupName)
         .getResultList();
}
```

Join

JPA memiliki kemampuan untuk menggabungkan beberapa objek sekaligus. Sebagai contoh, kita ingin menampilkan nama penulis

yang pernah mengisi artikel dengan kategori Headline News. Query ini melibatkan tiga objek, yaitu `Author`, `Article`, dan `Category`.

```
select au.fullname
from Article ar join ar.author as au
where ar.category .name like 'Headline%'
```

Query di atas akan menghasilkan beberapa record dengan nama yang sama. Untuk membuang hasil yang duplikat, gunakan keyword `distinct`.

```
select distinct au.fullname
from Article ar join ar.author as au
where ar.category.name like 'Headline%'
```

Report Query

Report query adalah query yang mengandalkan kemampuan database untuk melakukan agregasi terhadap data yang jumlahnya banyak. Biasanya report query banyak memanfaatkan aggregate function seperti `sum`, `count`, `group by`, dan `having`.

Sebagai contoh, bila kita ingin menghitung jumlah artikel yang sudah ditulis seorang author, kita bisa menggunakan query berikut.

```
select count (a)
from Article a
where a.author.fullname = 'endy'
```

Atau kita bisa juga menghitung berapa artikel yang tidak dikomentari pengunjung.

```
select count (a)
from Article a
where a.comments is empty
```

Subquery

JPA juga mendukung query di dalam query. Sebagai contoh, kita ingin menampilkan penulis mana yang pernah membuat artikel dengan nama kategori berakhiran News.

```
select distinct (au)
from Author au, Article ar
where ar.category.id in (
  select c.id from Category c where c.name like '%News'
)
```

Native SQL

Jika JPAQL di atas masih juga tidak mencukupi, kita bisa langsung mengeksekusi SQL ke database melalui JPA.

Sejauh ini, JPA sudah mencukupi untuk aplikasi blog kita. Tapi sekedar untuk menambah pengetahuan, mari kita ambil 10 artikel terbaru dengan menggunakan native SQL MySQL.

SQLnya sebagai berikut

```
select * from T_ARTICLE
order by publishDate desc limit 0,10
```

SQL di atas kita gunakan dalam ArticleDao sebagai berikut.

```
public List<Article> topTenNative() {
   String sql = "select * from T_ARTICLE ";
   sql += "order by publishDate desc limit 0,10";

   return entityManager
          .createNativeQuery(sql,Article.class)
          .getResultList();
}
```

Untuk menggunakannya, kita panggil methodnya seperti biasa. Client code tidak perlu tahu implementasi internalnya. Dari sisi client code, sama saja seperti mengakses JPAQL.

Jasper Report

Pengenalan

Menampilkan report atau laporan merupakan hal yang penting dan umum digunakan oleh aplikasi manapun, terutama yang bersifat sistem informasi. Oleh karena itu kita akan membahas bagaimana membuat desain report, mengisi report dengan data, lalu memberikan hasilnya kepada pengguna. Dalam Java, terdapat berbagai teknologi untuk membuat report diantaranya adalah,

- JasperReport, http://jasperreport.sourceforge.net/
- BIRT, http://www.birt-exchange.com/
- Data Vision, http://datavision.sourceforge.net/

JasperReport merupakan reporting tools yang paling populer di Java dan bahkan beberapa aplikasi PHP menggunakan JasperReport sebagai solusi untuk menampilkan report.

Instalasi

Persiapan yang perlu kita lakukan adalah:
1. Menginstall Java SDK, versi yang digunakan adalah JDK 1.6.0. Anda dapat mendownload JDK tersebut dari website http://java.sun.com
2. Menginstall NetBeans 6.0
3. Menginstall iReport
4. Menginstall MySQL

Java SDK diperlukan sebagai toolkit untuk mengembangkan aplikasi Java. Di dalamnya terdapat Java compiler, java virtual machine dan java library standard. Tanpa SDK kita tidak akan bisa mengembangkan aplikasi Java.

NetBeans adalah IDE (Integrated Development Environment) yang dikembangkan oleh Sun Microsystems. NetBeans difokuskan untuk mengembangkan aplikasi java secara produktif, didalamnya terdapat banyak wizard yang mengotomatisasi pengembangan aplikasi, menghindarkan developer dari low level configuration yang rumit.

iReport adalah visual designer untuk membuat reporting menggunakan library jasperreport. Tanpa iReport development

report menjadi sangat susah dan tidak efisien

MySQL digunakan sebagai basis data yang akan dibuatkan reportnya.

Alur Proses JasperReport

Untuk membuat dan menampilkan sebuah report ada beberapa hal yang harus dilakukan,

1. Merancang report menggunakan visual report designer, seperti iReport. File dihasilkan ber-ekstensi *.jrxml.
2. Mengkompilasi desain report agar siap diisi data. Hasil rancangan report yang sudah dikompilasi berekstensi *.jasper.

   Ada berbagai cara untuk mengkompilasi desain report,

   - Di dalam kode java atau,
   - Menggunakan ANT task.

   Cara yang kedua merupakan pendekatan yang paling umum digunakan karena kita tidak mungkin mengubah desain report pada saat run-time atau setelah aplikasi di-deploy.
3. Mengisi report dengan data. Data yang diberikan dapat berbagai jenis sesuai dengan kebutuhan, hal ini akan dibahas pada bab selanjutnya. Report yang sudah diisi data berekstensi *.jrprint.
4. Menentukan jenis report yang ingin ditampilkan. Saat ini JasperReport mendukung berbagai jenis format seperti, PDF, XLS, RTF, ODT, HTML, CSV, dan plain text.

Istilah Umum dalam JasperReport

Sebelum membahas lebih jauh mengenai JasperReport perlu diketahui beberapa istilah umum yang biasa digunakan,

- Reporting data

  Mengisi data ke dalam report dapat dilakukan dengan menggunakan,

  - Field, untuk menampilkan data yang berasal dari query database atau datasource.
  - Paramater, untuk menampilkan parameter report. Bentuk dari parameter adalah Map dengan tipe key dan

value berupa String.

- Variable, sebagai data yang dapat dimanipulasi dan digunakan berkali-kali dalam report.

- Layout report

  Biasanya desain report terdiri dari beberapa bagian yaitu,

  - Background.
  - Title, judul dari report.
  - Page header, bagian di setiap awal halaman.
  - Column header, bagian di setiap awal tabel.
  - Detail, bagian yang menampilkan data yang berasal dari query atau data source. Setiap data dapat ditampilkan secara vertikal dari atas ke bawah atau horizontal dari kiri ke kanan.
  - Column footer, bagian di setiap akhir tabel.
  - Page footer, bagian di bagian footer setiap halaman.
  - Last page footer, sama seperti page footer namun hanya muncul pada halaman terakhir.
  - Summary, bagian yang hanya ditampilkan sekali di akhir report.

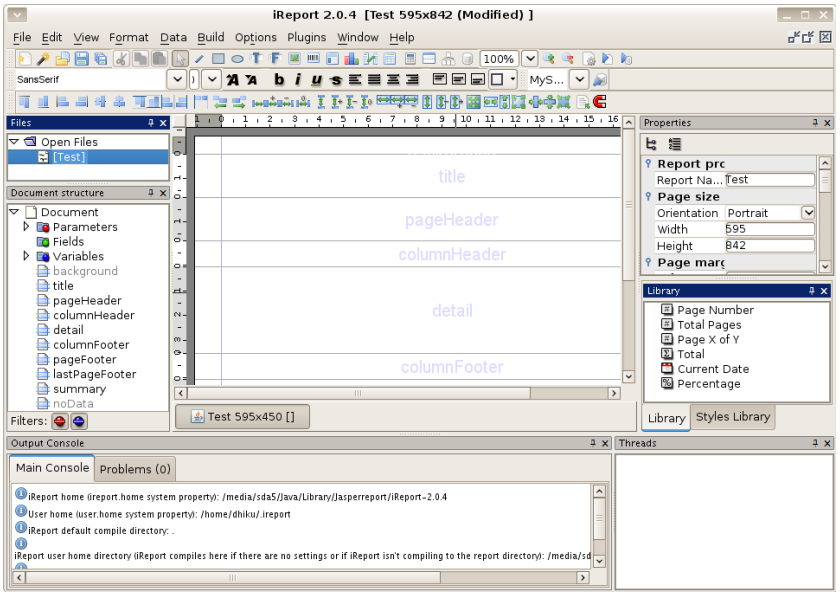
iReport sebagai Visual Designer

Untuk mendesain report kita akan menggunakan iReport, instalasinya sangat mudah.

- Pengguna Windows bisa mengunduh file installer iReport, menginstall dan langsung menjalankannya.
- Bagi pengguna sistem operasi *Nix bisa mengunduh file arsipnya berupa tar.gz dan mengekstraknya. Setelah itu menjalankannya dengan,

```
cd /home/user/iReport-2.0.4
sh iReport.sh
```

Berikut tampilan muka dari iReport,



Konfigurasi Koneksi Database dalam iReport

Sebelum menulis SQL dalam report, kita perlu mengkonfigurasi koneksi database. Caranya dengan,

```
Data > Connection / Data Sources
New > Pilih Database JDBC Connection > Next
```

Isikan form konfigurasi seperti berikut,

```
Name: MySQLConnection
JDBC Driver: com.mysql.jdbc.Driver
JDBC URL: jdbc:mysql://localhost/ejb3
Username: belajar
Password: java
```

Konfigurasi di atas mengasumsikan kita sudah mempunyai database yang bernama belajar dengan username dan password belajar / java. Berikut script untuk membuat schema berdasarkan konfigurasi di atas,

```
CREATE DATABASE ejb3;
CREATE TABLE T_PERSON
GRANT ALL ON belajar.* TO belajar@localhost IDENTIFIED BY
'java';
FLUSH PRIVILEGES;
```

Menampilkan Data dalam Report

Ada dua pendekatan umum untuk menampilkan atau mengisi data ke dalam report,

- Inline SQL, menampilkan data melalui SQL yang disimpan pada file desain report.
- DataSource, menampilkan data melalui interface DataSource yang disediakan oleh JasperReport. DataSource ini didefinisikan di dalam kode Java sehingga report hanya berfungsi untuk menampilkan data saja.

Pendekatan pertama merupakan pendekatan yang paling sederhana. Oleh karena itu kita akan membahas menampilkan data melalui inline SQL. Untuk mendefinisikan SQL dalam report dapat dilakukan dengan,

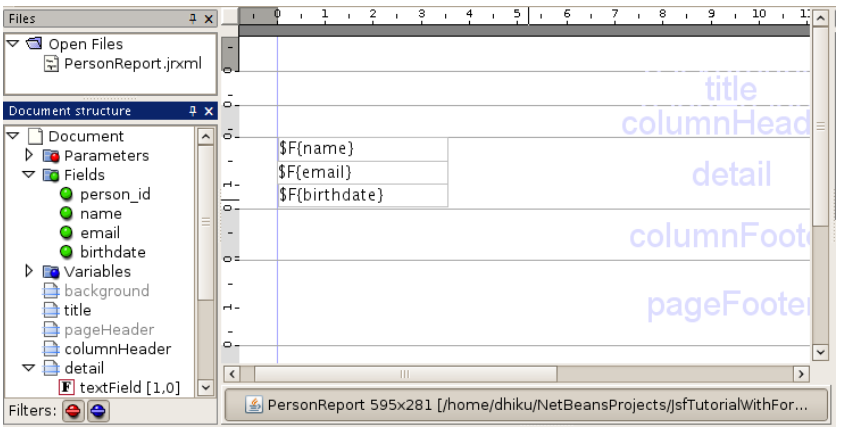
```
File > New Document
Beri nama report dengan PersonReport > OK
Edit > Report Query
```

Tulis SQL dalam query editor,

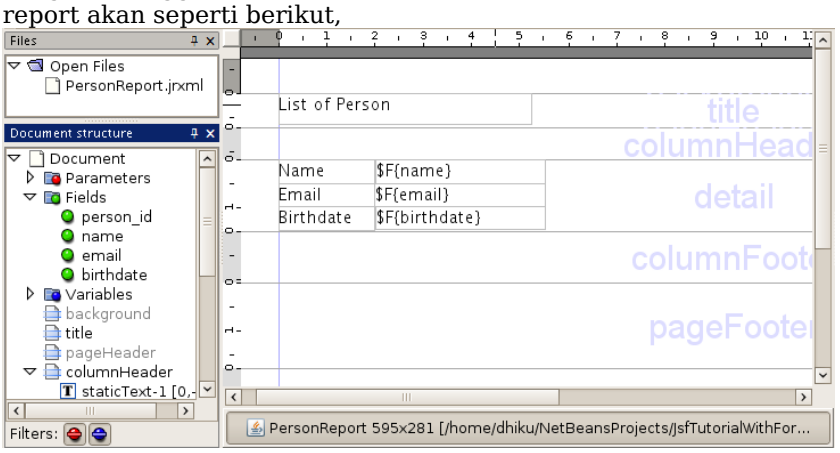
```
select * from T_PERSON
```

Kemudian tekan tombol “Read Fields” untuk memapping kolom tabel dengan field lalu tekan tombol “OK”.

Ada beberapa field baru sesuai dengan kolom di tabel T_PERSON yaitu, person_id, name, email, dan birthdate. Lalu pindahkan field di atas ke bagian “detail” dalam report. Hasilnya seperti berikut,



Kemudian tambahkan judul dan label untuk setiap field data dengan menggunakan menu toolbar "Static Text". Desain akhir report akan seperti berikut,



Tampilkan hasil report dengan cara,

```
Build > Execute (with active connection)
```

Jangan lupa untuk menyimpan report yang sudah dibuat dengan nama **PersonReport.jrxml**,

```
File > Save
```

Kompilasi Report dengan Ant

Setelah desain sudah jadi maka kita perlu membuat ant target

untuk mengkompilasi seluruh desain report. Caranya,

```
Klik tab Files > Buka file build.xml
```

Setelah itu kita definisikan terlebih dahulu dimana lokasi kode java dan report berada dan mengatur classpath untuk library yang dibutuhkan dalam kompilasi report.

```
<!-- definisi lokasi -->
<property name="source.java" value="src/java" />
<property name="compile.report.dir"
value="src/java/report"/>

<!-- setting classpath -->
<path id="compile.classpath">
        <pathelement location="${source.java}" />
        <pathelement location="${compile.report.dir}" />
        <fileset dir="lib" includes="**/*jar" />
</path>
```

Setelah lokasi dan classpath ditentukan, maka langkah terakhir adalah mendefinisikan ant target untuk mengkompilasi seluruh file report dengan ekstensi *.jrxml.

```
<taskdef name="jrc"
   classname="net.sf.jasperreports.ant.JRAntCompileTask">
  <classpath refid="compile.classpath" />
</taskdef>

<target name="compile-report" depends="compile">
   <jrc srcdir="${compile.report.dir}"
        destdir="${compile.report.dir}"
        tempdir="${compile.report.dir}"
        xmlvalidation="true">
        <classpath refid="compile.classpath" />
        <include name="**/*.jrxml" />
   </jrc>
</target>
```

Jalankan ant target dengan nama “compile report” dengan cara,

```
Klik kanan build.xml > Run Target > Other Target
Klik compile-report
```

Akan muncul output seperti ini,

```
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-report:
Compiling 1 report design files.
File :
/home/ifnu/NetBeansProjects/jasper/jasper/report/PersonRe
port.jrxml ... OK.
BUILD SUCCESSFUL (total time: 4 seconds)
```

Integrasi JasperReport dengan Swing

Integrasi jasperreport dengan Swing sebelumnya dapat dilakukan dengan mudah. Ada berbagai skenario untuk menampilkan halaman atau data dalam Swing yaitu,

- Menampilkan dalam JPanel menggunakan JRViewer
- Menampilkan dalam JFrame menggunakan JasperViewer
- Mengeksport langsung hasil report ke dalam format file seperti PDF, Excell atau Word

Pendekatan pertama dan kedua sama dari sisi kode, hanya saja ditampilkan dengan menggunakan komponen yang berbeda. Cara yang paling lazim adalah dengan menggunakan JRViewer, dimana kita bisa meletakkan viewer report di manapun kita inginkan, karena JRViewer adalah JPanel.

```
ResultSet rs = stmn.executeQuery();
String jasper =
       JasperCompileManager.compileReportToFile(
       System.getProperty("user.dir") +
       "/src/jasper/report/person.jrxml");
JRResultSetDataSource resource = new
       JRResultSetDataSource(rs);
String result =
       JasperFillManager.fillReportToFile(jasper,
       new HashMap(), resource);
JRViewer viewer = new JRViewer(result, false);
```

Menampilkan Report Menggunakan DataSource

Sebelumnya kita telah mencoba pendekatan pertama untuk menampilkan report yaitu dengan menyimpan kode SQL di dalam report. Kekurangan dari pendekatan ini adalah dari sisi code maintability, karena kode SQL terpisah dari source code aplikasi. Akibatnya akan terjadi banyak duplikasi, misalnya kita sudah punya kode SQL di source code aplikasi untuk menampilkan seluruh daftar Person dan kemudian kita ingin menampilkan report daftar Person dalam bentuk PDF. Maka yang selanjutnya dilakukan adalah copy-paste kode SQL dari source code aplikasi ke dalam report. Bayangkan jika kita mempunyai 10 report yang menampilkan daftar Person dengan kode SQL yang sama dengan source code aplikasi lalu kita mengubah salah satu nama kolom dari tabel T_PERSON, maka tentu seluruh report harus diubah dan dicompile ulang.

JasperReport menyediakan solusi untuk hal ini dengan menggunakan DataSource. DataSource berfungsi untuk menyimpan data yang ingin ditampilkan, misalkan dari database, kemudian dikirim ke dalam report untuk selanjutnya ditampilkan. Ada berbagai jenis DataSource yang dapat kita gunakan berdasarkan tipe data yang ingin disimpan,

- JRMapArrayDataSource untuk menyimpan tipe data array of map
- JRMapCollectionDataSource untuk menyimpan tipe data collection of map
- JRBeanArrayDataSource untuk menyimpan tipe data array of bean
- JRBeanCollectionDataSource untuk menyimpan tipe data collection of bean
- JRTableModelDataSource untuk menyimpan tipe data tablemodel di swing
- JRXmlDataSource untuk menyimpan tipe data xml

DataSource yang umum digunakan adalah JRBeanCollectionDataSource. Kita akan menampilkan daftar Person, dengan membuat Collection dari Person bean yang kemudian akan diberikan ke JRBeanCollectionDataSource. Pertama dibuat terlebih dahulu **Person.java**,

```
public class Person {
   private int id;
   private String name;
   private String email;

   // Generate getter and setter
}
```

Buka file **PersonReport.jrxml** yang dibuat sebelumnya melalui iReport dengan text editor misalnya, notepad. Kurang lebih hasilnya seperti ini,

```
<jasperReport name="PersonReport">

  <queryString>
      <![CDATA[select * from T_PERSON]]>
  </queryString>
  <field name="id" class="java.lang.Integer"/>
  <field name="name" class="java.lang.String"/>
  <field name="email" class="java.lang.String"/>

  <!-- desain report -->

</jasperReport>
```

Kita dapat langsung menggunakan file **PersonReport.jrxml** di atas agar dapat menampilkan DataSource. Yang perlu dilakukan adalah,

- Hilangkan tag queryString
- Sesuaikan nama field dan tipenya dengan property pada class Person , sehingga PersonReport.jrxml menjadi,

Berikut file PersonReport.jrxml setelah diubah,

```
<jasperReport name="PersonReport">

  <field name="id" class="java.lang.Integer"/>
  <field name="name" class="java.lang.String"/>
  <field name="email" class="java.lang.String"/>

  <!-- desain report -->

</jasperReport>
```

Selanjutnya adalah membuat **PersonReport.java** untuk men-generate report,

```
public class PersonReport {
  public static void main(String[] args) throws
     JRException {
       // Siapkan datasource
       JRBeanCollectionDataSource dataSource = new
          JRBeanCollectionDataSource(prepareDataSource());

       // Compile JRXML menjadi Jasper
       JasperReport jasperReport = JasperCompileManager
         .compileReport(
              "/home/dhiku/jasper/PersonReport.jrxml");
       // Fill report dengan datasource
       JasperPrint jasperPrint =
      JasperFillManager.fillReport(jasperReport,
          new HashMap(), dataSource);

       // Export report
      JasperExportManager.exportReportToPdfFile(
         jasperPrint,
         "/home/dhiku/jasper/PersonReport.pdf");
 }

 public static List prepareDataSource() {
       List users = new ArrayList();
       users.add(new Person("1", "Hadikusuma Wahab",
                    "dh1ku.ilkom@gmail.com"));
       users.add(new Person("2", "Endy Muhardin",
                    "emuhardin@gmail.com"));
       users.add(new Person("3", "Ifnu Bima",
                    "ifnubima@gmail.com"));
       return users;
 }
}
```

Biasanya untuk kasus nyata, daftar Person pada method prepareDataSource() dipanggil dengan menggunakan SQL query. Setelah kelas PersonReport dijalankan maka hasil report dapat dilihat di /home/dhiku/jasper/PersonReport.pdf.

Mengirim Parameter

Selain DataSource, adapula parameter yang dapat dikirimkan ke dalam report. Misalkan kita ingin mencetak judul report secara dinamis yang dikirim dari kode Java. Berikut langkah-langkah untuk mengirimkan parameter,

1. Siapkan parameter di report

```
View > Parameter
Klik New
Isi parameter name, “judul”
Klik OK
```

2. Berikan parameter ke report

```
// Compile Report dan siapkan DataSource

HashMap<String, String> param =
                  new HashMap<String, String>();

// Fill Report
JasperPrint jasperPrint =
      JasperFillManager.fillReport(jasperReport,
          param, dataSource);

// Export Report
```

Menggunakan Scriptlet

Scriptlet adalah kode Java yang diselipkan pada saat proses fill report. Proses fill report terdiri dari beberapa tahap, scriptlet dapat menjalankan kode Java pada sebelum dan sesudah tahap-tahap tersebut misalnya,

- Sebelum atau sesudah report diinisialisasi
- Sebelum atau sesudah setiap halaman report diinisialisasi
- Sebelum atau sesudah setiap kolom diinisialisasi
- Sebelum atau sesudah setiap group diinisalisasi
- Sebelum atau sesudah setiap baris atau record diinisialisasi

Untuk membuat scriptlet maka kita harus membuat sebuah class yang meng-extend kelas JRAbstractScriptlet. Berikut contoh scriptlet yang akan dijalankan setiap sebelum baris diinisialisasi,

```
public class PersonScriplet extends JRAbstractScriptlet {
  public int number = 0;
  public void beforeDetailEval() throws
      JRScriptletException
  {
   String name = (java.lang.String)getFieldValue("name");
   System.out.println("number" + ". "  + name);
   number++
  }

}
```

Untuk menggunakan kelas PersonScriptlet di dalam PersonReport caranya adalah sebagai berikut,

```
Edit > Report Properties
Klik tab “Scriptlet Class”
Pilih “Use this scriptlet class ...”
Ketik PersonScriptlet
```

Jika menggunakan package, maka masukkan full path untuk kelas scriptlet, misalkan com.artivisi.report.PersonScriptlet. Jalankan kembali kelas PersonReport yang sudah dibuat sebelumnya, seharusnya selain menghasilkan daftar Person, juga akan memberikan output ke console berupa,

```
1. Hadikusuma Wahab
2. Endy Muhardin
3. Ifnu Bima
```

Penutup

Dengan ilmu yang sudah diperoleh dalam pelatihan ini, anda sudah bisa mulai untuk membuat program Java desktop sederhana. Pada awalnya pasti terasa sulit, sikapi dengan pantang menyerah dan selalu cari cara yang lebih baik dalam membuat aplikasi.

Langkah selanjutnya anda bisa mulai aktif bertanya atau menjawab hal-hal yang berhubungan dengan Java. Media yang bisa digunakan banyak sekali, bisa forum milis atau diskusi dengan teman. Ini cara terbaik untuk mengetes apakah pemahaman anda mengenai Java sudah cukup lengkap atau anda masih perlu belajar lebih banyak lagi.

Setelah yakin dengan kemampuan anda, berfikirlah untuk mengambil sertifikasi profesional Java. Pelatihan untuk persiapan sertifikasi Java banyak tersedia di lembaga pelatihan. Kalau anda merasa terlalu berat mengambil kursus persiapan sertifikasi, berlatihlah sendiri menggunakan materi yang banyak tersedia di internet, misalnya javaranch.com.

Cara belajar Java yang paling efektif adalah dengan melibatkan diri dalam project berbasis Java. Jika di perusahaan anda tidak memungkinkan, di luar sana banyak sekali project opensource yang memerlukan bantuan anda. Berkunjunglah ke website-website open source project hosting seperti sourceforge.net atau dev.java.net

Learn, Try dan Teach adalah formula untuk meningkatkan pengetahuan anda tentang Java. Jika sudah belajar dan sukses mencoba, tularkan ilmu anda pada orang disekeliling anda, dijamin ilmunya bakalan bertambah berlipat-lipat.

Referensi dan Bacaan Lebih Lanjut

1. Java Tutorial : http://java.sun.com/docs/books/tutorial/
2. Java Documentation : http://java.sun.com/javase/6/docs/
3. Netbeans Indonesia : http://groups.yahoo.com/group/Netbeans-indonesia/
4. Java User Group Indonesia : http://groups.yahoo.com/group/jug-indonesia/
5. Java Design Pattern : http://www.javacamp.org/designPattern/
6. Java Desktop : http://www.javadesktop.org
7. JGoodies : http://www.jgoodies.com
8. SwingX : http://www.swinglabs.org