Model-View-Controller (MVC) Design Pattern Untuk Aplikasi Perangkat Bergerak Berbasis Java

Panji Wisnu Wirawan

Program Studi Teknik Informatika, Universitas Diponegoro
maspanji@undip.ac.id

Abstrak

Design pattern merupakan salah satu teknik mendesain komponen perangkat lunak yang bisa digunakan kembali. MVC design pattern merupakan design pattern untuk komponen perangkat lunak yang memisahkan secara tegas antara model data, tampilan, dan kendali perangkat lunak. Artikel ini akan membahas MVC design pattern disertai kesesuaiannya dengan aplikasi perangkat bergerak berbasis Java. Bagian akhir artikel ini menunjukkan bagaimana MVC design pattern untuk aplikasi bergerak berbasis Java.

Kata Kunci : MVC, *design pattern*, aplikasi perangkat bergerak berbasis Java

1. Pendahuluan

Komponen perangkat lunak sudah semestinya didesain dan diimplementasikan sehingga bisa digunakan ulang (*reused*) pada perangkat lunak yang lain[6]. Sebagai contoh penggunaan komponen perangkat lunak yang dapat digunakan ulang adalah penggunaan *text box*, *drop down menu* dan sebagainya.

Contoh lain dari komponen perangkat lunak adalah pola-pola tertentu untuk menyelesaikan masalah yang disebut sebagai *design pattern*. Pola-pola tersebut akan tetap sama untuk aplikasi yang berbeda. Sebagai contoh, ada pola tertentu untuk memisahkan tampilan, kendali program, dan model data yang kemudian disebut sebagai *model-view-controller* (MVC) *design pattern*. Biasanya *design pattern* ini digunakan pada aplikasi perangkat lunak yang menggunakan antarmuka grafis / graphical user interface (GUI)

Perkembangan aplikasi bergerak yang begitu pesat sudah selayaknya menggunakan komponen-komponen perangkat lunak yang bisa digunakan ulang sehingga dapat meminimalkan usaha pengembangan aplikasi perangkat bergerak. Pada umumnya, aplikasi perangkat bergerak menggunakan GUI. Penggunaan MVC diharapkan bermanfaat untuk pembuatan komponen perangkat lunak untuk aplikasi bergerak yang bisa digunakan ulang.

Salah satu penelitian mengenai *design pattern* untuk aplikasi perangkat bergerak berbasis Java telah dilakukan oleh Narsoo dan Mohamudally (2008). Narsoo dan Mohamudally (2008) melakukan identifikasi *design pattern* untuk *mobile services* menggunakan *Java* 2 *Mobile Edition* (J2ME). *Mobile services* tersebut termasuk dalam *behavioral pattern* yaitu *add*, *edit*, *erase* dan *search*. Berbeda dengan artikel tersebut, artikel ini membahas penggunaan MVC *design pattern* pada J2ME.

2. *Design Pattern*

Design pattern merupakan deskripsi kelas-kelas dan objek-objek yang berkomunikasi yang disusun untuk memecahkan masalah perancangan secara umum pada konteks tertentu[3]. *Sebuah design pattern* memiliki nama, membuat abstraksi, dan mengidentifikasi aspek-aspek desain yang umum

untuk membuat desain berorientasi objek yang bisa digunakan ulang.

2.1. Kategori

Design pattern memiliki 3 kategori [7], di mana masing-masing kategori memiliki tujuan tertentu. Kategori tersebut adalah :

1. *Structural pattern*
Structural pattern berfokus pada bagaimana beberapa kelas dan objek disusun untuk membentuk struktur yang lebih besar[3]. *Façade pattern* merupakan satu contoh dari *structural pattern*. *Façade pattern* menunjukkan bagaimana membuat sebuah objek yang mencerminkan keseluruhan sub sistem[2].

2. *Behavioral pattern*
Behavioral pattern berhubungan dengan algoritma dan tanggung jawab antar objek-objek[3]. *Strategy pattern* merupakan satu contoh dari *behavioral pattern* yang menunjukkan bagaimana algoritma dienkapsulasi[2].

3. *Creational pattern*
Creational pattern bertujuan untuk membuat abstraksi proses instansiasi[3]. *Creational pattern* membantu sistem bagaimana objek dibuat secara independen. *Singleton pattern* merupakan satu contoh dari *creational pattern* yang menjamin bahwa sebuah kelas hanya memiliki sebuah instan dan memberikan satu akses global[2].

2.2. MVC Design Pattern

MVC terdiri dari 3 macam objek[3]. *Model* merupakan objek aplikasi, *View* adalah presentasi sedangkan *Controller* mendefinisikan bagaimana antarmuka pengguna bereaksi terhadap input. Tanpa MVC, antarmuka pengguna cenderung menyatukan objek-objek tersebut. Dengan MVC, diharapkan menambah fleksibilitas dan pemakaian kembali.

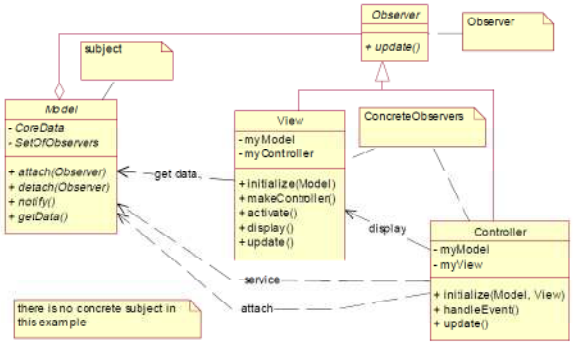
MVC digunakan utamanya ketika mengembangkan perangkat lunak dengan antarmuka grafis (*Graphical User Interface,* GUI)[3][8]. Namun, secara umum, MVC dapat digunakan untuk mengembangkan sistem yang interaktif. Kegunaan tersebut adalah untuk memisahkan data (*Model*), menyajikan data (*View*), dan logika untuk menangani kejadian (*event*) dari data ke penyajian maupun sebaliknya (*Controller*).

MVC merupakan *design pattern* yang terdiri dari *design pattern* yang lain disebut sebagai *compound pattern* [2]. *Compound pattern* mengkombinasikan dua atau lebih *design pattern* yang memecahkan masalah tertentu. MVC meliputi 3 *design pattern* yaitu :

1. *Composite pattern*
Composite pattern bertindak sebagai komponen View. Composite pattern bertugas menampilkan komponen antarmuka grafis dan memperbarui tampilan ketika diperlukan. Notifikasi perubahan tampilan dilakukan oleh *Controller*.

2. *Strategy pattern*
Strategy pattern bertindak sebagai komponen Model. Model diorganisasikan dalam *strategy pattern*.

3. *Observer pattern*
Observer pattern bertindak sebagai komponen Controller, yang memungkinkan penggunaan *view* yang berbeda dengan model yang sama. Atau bahkan menggunakan banyak *view* sekaligus.



Gambar 1. MVC *design pattern*
(Shalloway, 2001)

Gambar 1 menunjukkan *class diagram* yang menggambarkan MVC *design pattern*. Gambar tersebut menjelaskan bagaimana masing-masing

komponen (design pattern sebagai *model*, *view*, dan controller) berkomunikasi.

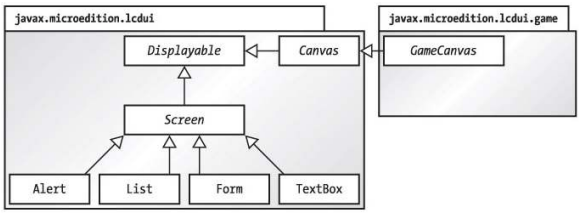
Pada gambar 1, jika diamati lebih lanjut, MVC merupakan *Observer Pattern* di mana *Controller* dan *View* merupakan *concrete observer* dan *Model* berperan sebagai *subject*.

3. Aplikasi Perangkat Bergerak Berbasis Java

Aplikasi perangkat bergerak berbasis Java didukung oleh Java 2 Micro Edition (J2ME). J2ME merupakan keluarga Java untuk telepon selular dan perangkat bergerak yang lain[1]. J2ME didesain untuk perangkat elektronik yang tidak memiliki cukup memori untuk menjalankan Java edisi standar (J2SE).

Aplikasi perangkat bergerak yang dibangun di atas J2ME disebut MIDlet [4]. MIDlet dibuat dengan pustaka J2ME yang berbeda dengan J2SE. Dalam kaitannya dengan MVC, pustaka java.util.Observer pada J2SE yang pada umumnya digunakan sebagai pustaka untuk membuat komponen Controller MVC, tidak tersedia pada J2ME. Jadi, diperlukan hal khusus untuk mengatasi ketiadaan java.util.Observer, jika akan menggunakan MVC oada J2ME.

Dalam J2ME dikenal 2 pustaka, yaitu *high-level Application Programming Interface* (API) dan *low-level* API. Perbedaan keduanya adalah pada pengaksesan kendali tampilan dan masukan pengguna. *High-level* API menggunakan kelas Screen, sedangkan Low-level API menggunakan kelas Canvas. Gambar 2 menjelaskan hirarki dari masing-masing API tersebut pada pustaka J2ME.



Gambar 2. Hirarki *High-level* API dan *Low-level* API (Knudsen, 2003)

4. MVC Design Pattern Pada J2ME

Pada bagian ini, kesesuaian MVC *design pattern* dengan J2ME akan dibahas untuk setiap komponen MVC. Fokus pada artikel ini adalah pada *High-level* API.

4.1. *View*

J2ME telah memberikan kemudahan dalam view, karena pustaka untuk menangani peletakan komponen pada tampilan telah disediakan. Yang perlu diperhatikan adalah fungsi masing-masing kelas seperti Form, List, dan lain-lain, karena masing-masing memiliki karakteristik dalam mengorganisasikan tampilan.

Komponen view pada MVC *design pattern* J2ME menurunkan (*inherit*) kelas pustaka J2ME untuk tampilan javax.microedition.lcdui.Form. Hal ini dilakukan karena pengaturan tampilan telah dilakukan secara *default* oleh pustaka tersebut. Kelas lain yang dalam hirarki gambar 2 sejajar dengan javax.microedition.lcdui.Form, bisa digunakan semisal akan dibuat *view* yang lain.

Pada komponen *View* akan terdapat metode *update*, yang akan memperbarui tampilan berdasarkan notifikasi dari komponen *Model*.

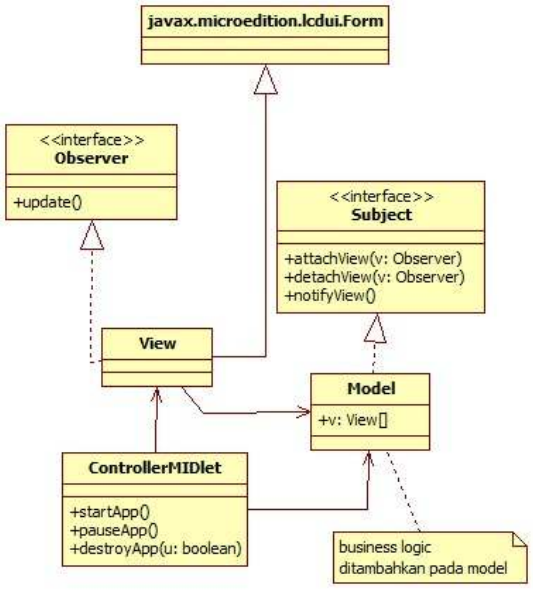
4.2. *Model*

Pada gambar 1, komponen *Model* merupakan implementasi dari *subject interface*, di mana *interface* tersebut berisi kontrak untuk memasukkan dan mengeluarkan *view* yang akan diperbarui tampilannya ketika ada perubahan pada model data. Logika bisnis (*business logic*) juga diisikan pada kontrak *interface* tersebut.

Pada model terdapat metode *notifyView* yang akan melakukan notifikasi ketika ada perubahan pada komponen *Model*. Notifikasi ini dilakukan kepada semua View yang sudah dimasukkan/terdaftar pada komponen *Model*.

4.3. *Controller*

Pada J2ME, komponen *Controller* bertugas untuk meminta komponen *Model* untuk melakukan logika bisnis tertentu sekaligus sebagai *entry point* aplikasi. Jadi, Komponen *Controller* adalah MIDlet itu sendiri.



Gambar 3. MVC *design pattern* J2ME

Dari pemaparan masing-masing komponen MVC yang disesuaikan dengan J2ME di atas, dapat disusun diagram kelas pada gambar 3 yang menunjukkan MVC *design pattern* untuk J2ME.

5. Kesimpulan

MVC *design pattern* bisa disusun untuk aplikasi perangkat bergerak berbasis Java / J2ME. Masing-masing komponen (*Model*, *View* dan *Controller*) bisa disusun dalam kelas-kelas tersendiri.

Komponen *View* memiliki komponen tambahan khusus dari pustaka J2ME untuk mengorganisasikan tampilan / View. Komponen Controller merupakan MIDlet yang akan meneruskan pesan ke Komponen Model. Komponen Model berisi daftar tampilan/*view* yang akan diperbarui karena memiliki metode untuk melakukan notifikasi ketika ada perubahan data. Komponen Model juga berisi logika bisnis aplikasi.

Dengan pemisahan tersebut diharapkan terbentuk komponen perangkat lunak yang bisa dipakai kembali (*reusable*).

6. Referensi

[1] Barbagallo, R., *Wireless Game Development In Java With MIDP 2.0*, Wordware Publishing, 2004.

[2] Freeman, E., et.al, *Head First Design Pattern* , California: O'Reilly Media Inc, 2004.

[3] Gamma, E., et.al., *Design Patterns : Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1994.

[4] Knudsen, J, *Wireless Java-Developing with J2ME*, Apress, 2003.

[5] Narsoo, J. dan Mohamudally,N., Journal of Issues in Informing Science Information Technology, 42(5), *Identification of Design Pattern for Mobile Services with J2ME*, 621-643.

[6] Pressman, R.S, *Software Engineering: A Practitioner's Approach 6th ed*, Mc Graw Hill, 2005.

[7] Shalloway,A. dan Trott J.R., *Design Pattern Explained : A New Perspective On Object Oriented Design*, Addison-Wesley Professional ,2001.

[8] Yacoub S.M, dan Ammar H.H, *Pattern-Oriented Analysis and Design: Composing Patterns to Design Software Systems*, Addison-Wesley Professional, 2003.