PENERAPAN JAVA SERVER FACES UNTUK DESIGN PATTERN WEB

Yanto [1)]

Abstrak:

J2EE Pattern adalah kumpulan pola-pola yang digunakan dalam menyelesaikan masalah yang umumnya dihadapi oleh setiap programmer Java yang menerapkan aplikasi J2EE. Penerapan JSF (*Java Server Faces*) pada dasarnya menggunakan bahasa pemograman Java, sehingga memiliki sifat sama dengan aplikasi Java yang lainnya terutama aplikasi pada platform J2EE. Karena itu pattern-pattern yang pada aplikasi J2EE sebelumnya mungkin dapat diterapkan pada JSF. Maka pada penelitian ini, dibangun sebuah website yang menggunakan JSF dengan menerapkan J2EE Design Pattern pada pemogramannya. Untuk perbandingan akan dilihat perbedaan cara penerapan pattern pada JSF, dengan cara penerapan pattern yang umumnya digunakan pada aplikasi JSP.

Kata Kunci : *design pattern, java server faces, j2ee*

1. Pendahuluan

Java Server Faces (JSF) merupakan suatu lingkungan pemograman komponen antarmuka pemakai (*UI/user interface*) yang menyediakan komponen antarmuka pemakai baku yang lebih variatif, dan dijalankan pada sisi server. *Java server faces* menyediakan komponen antarmuka pemakai dapat dihubungkan langsung ke server melalui servlet. Dengan komunikasi langsung dari komponen antarmuka pemakai ke servlet, class pada servlet dan *javabean* yang mirip tidak perlu ditulis ulang untuk setiap komponen antarmuka pemakai dan setiap halaman web dengan melihat pola tersebut pada web. Web dapat dibuat lebih sederhana tapi kaya akan variasi. Pengecekan terhadap kesalahan juga lebih mudah dilakukan karena pengecekan hanya perlu dilakukan pada UI component atau class pada servlet yang terdapat kesalahan.

Dengan memanfaatkan teknologi JSF pada penelitian ini, akan diterapkan pola-pola (*patterns*) J2EE pada teknologi JSF, dan dicari perbedaan cara penerapan pemograman JSF dengan aplikasi JSP biasa. Agar penilitian ini dapat lebih fokus, untuk penerapannya akan digunakan studi kasus Toko Handphone On-line. Web sepenuhnya akan dibuat dengan teknologi *Java Server Faces*, *Java Server Pages*, dan *Servlet*. Aplikasi web yang dibuat akan tampak seperti website umumnya, namun tampilan akan dibuat dari JSF dengan bantuan *backing bean*, *deployment descript file*, dan *application configuration resource file*.

Batasan aplikasi yang dibuat hanya pada pemanfaatan JSF untuk membangun aplikasi web, *backing bean* untuk aplikasi, membuat deployment descript file sesuai servlet yang digunakan dan membuat application configuration resource file yang mengatur konfigurasi dari aplikasi. Aplikasi JSP tidak akan dibuat pada penelitian ini. Untuk membandingkan JSF dengan JSP hanya akan dibabarkan pada saat analisa, tanpa contoh aplikasi JSP.

2. Tinjauan Pustaka

2.1 Tekonologi *Java Server Faces*

Java Server Faces (JSF) adalah teknologi java untuk aplikasi web yang dibuat oleh Sun Microsystems. Tujuannya adalah untuk menciptakan standard framework komponen User Interface untuk aplikasi web. JSF membangun suatu aplikasi web yang jalan pada sisi server dan kemudian menampilkan User Interface kembali ke Client. JSF menyediakan komponen antarmuka pemakai yang umumnya digunakan di web, dan memiliki standard yang sama sehingga dapat dijalankan di berbagai platform dan device.

[1)] *Yanto, Alumni Program Studi Teknik Informatika F. Teknik Universitas Kristen Duta Wacana.*

JSF juga menyediakan kemudahan dalam penulisan script pada halaman statisnya. Berikut cara penulisan script tabel untuk JSF.

```
<h:dataTable value="#{MyDAO.queryData}" var="data">

<h:column>

<h:outputText value="#{data.item}"/>

</h:column>

</h:dataTable>
```

Kelebihan JSF dari JSP adalah pemisahan yang jelas antara logika bisnis dan tampilan web, dengan memanfaatkan komponen antarmuka pemakai untuk tampilan web dan backing bean sebagai logika bisnisnya. Pemisahan terlihat pada cara penulisan script.
Dengan pemisahan tersebut logika bisnis pada web yang biasanya harus ditulis beberapa kali pada halaman JSP atau servlet yang hanya dapat diakses tiap halaman, sekarang dapat diakses untuk tiap komponen antarmuka pemakai. Dan tampilan komponen antarmuka pemakai dapat diubah dengan mengubah attributes-nya, sehingga komponen antarmuka pemakai yang class-nya sama dapat menggunakan komponen antarmuka pemakai yang sama.

2.2 Backing Bean

Backing bean adalah *java bean component* yang berfungsi untuk mengatur sifat (properties) dan fungsi (method) untuk tiap komponen antarmuka pemakai. *Backing bean* juga melakukan berbagai fungsi untuk menjalankan aplikasi, seperti pengesahan, perhitungan, dan sebagainya.

Karena *backing bean* adalah *java bean component*, maka *backing bean* memiliki format yang sama dengan java bean lainnya. *Backing bean* harus memiliki *zero-argument constructor*, tidak memiliki *public instance variables*, dan untuk mengakses *bean property* (nilai yang tersimpan pada bean) dilakukan dengan mengakses method “getNamaproperty” dan “setNamaproperty”.

2.3 Application Configuration Resource File

Application Configuration Resource File disimpan dengan format XML, dengan nama “faces-config.xml”, fungsinya antara lain :

- Mencatat semua objek yang akan dipakai aplikasi sehingga tiap bagian memiliki akses terhadapnya
- Mengatur backing bean dan model bean sehingga memiliki nilai awal tepat ketika diakses oleh halaman web
- Menentukan aturan perpindahan antar halaman web, objek, dan file lainnya sehingga aplikasi memiliki perpindahan yang lancar.
- Membuat paket aplikasi dengan mengikutkan semua halaman, objek dan file lainnya, agar aplikasi dapat di deploy pada container yang berbeda.

Implementasi untuk Application Configuration Resource File harus sesuai dengan ketentuan, dan harus mengikutkan header awalnya. Dan untuk tiap konfigurasinya harus berada dalam tag awal “<faces-config>”.

2.4 Design Pattern

2.4.1 Design with Reuse

Design proses yang didasarkan pada *component reuse* memiliki berbagai keuntungan, antara lain component yang telah digunakan berarti memiliki tingkat kepercayaan lebih tinggi, karena pernah digunakan pada berbagai system sebelumnya, memberikan suatu standard tertentu terutama untuk user interfaces, mempercepat pembangunan system, dan mengurangi pengeluaran karena pembangunan yang cepat, sehingga system dapat lebih cepat dipakai.

Rekayasa perangkat lunak berdasarkan pengunaan ulang (*reuse*) dilakukan dengan memaksimalkan pengunaan ulang dari perangkat lunak yang pernah ada. Ukuran penggunaan ulang dari unit perangkat lunak dapat berbeda-beda, antara lain :

- *Application system reuse*. Seluruh aplikasi dapat digunakan kembali tanpa perubahan kedalam system lain, atau system dapat digunakan seluruhnya tetapi dengan perubahan untuk menjalankannya pada platform yang berbeda, atau perubahan untuk spesialisasi untuk konsumen tertentu.
- *Component reuse*. Komponen dari aplikasi berukuran mulai dari sub-systems hingga object yang dapat digunakan kembali. Misal pola (pattern) system untuk text processing system yang memiliki kecocokan dapat digunakan untuk database management system.
- *Function reuse*. Komponen perangkat lunak yang digunakan kembali hanya berupa single function. Contoh fungsi untuk melakukan perhitungan matematika tertentu dapat digunakan untuk fungsi lain atau system tertentu.

Untuk penggunaan ulang suatu komponen diperlukan pengetahuan tentang komponen tersebut, pengguna juga harus yakin komponen harus bertindak sesuai dengan yang dinginkan, dan perlu dokumentasi atau pengetahuan tentang bagaimana kondisi, cara, kapan, dan masalah apa saja yang mungkin muncul dalam penggunaan.

2.4.2 Component-based Development

Component-based development muncul pada akhir tahun 1990 sebagai pendekatan untuk reused-based untuk pembangunan sistem perangkat lunak. *Component-based development* ditujukan untuk memperbaiki kurangnya pemanfaatan *reuse* pada pengembangan berbasis objek. Hal ini disebabkan sifat class-class objek yang terlalu detail dan spesifik dan harus terikat dengan aplikasi baik saat kompilsi atau saat sistem dihubungkan. Komponen lebih abstrak dibandingkan class-class objek dan dapat dianggap sebagai penyedia layanan yang dapat berjalan sendiri. Ketika sistem membutuhkan suatu layanan, sistem akan memanggil component tanpa perlu melihat bagaimana proses dijalankan. Melihat komponen sebagai penyedia layanan, maka dapat diambil dua karakteristik dari *reuseable-component* :

- Komponen adalah suatu entity yang berjalan sendiri.
- Komponen menyediakan antarmuka tertentu dan semua interaksi melalui antarmuka tersebut. Komponen antarmuka berupa parameter operasi dan keadaan proses didalam tidak pernah ditampilkan.

Dalam proses perancangan yang menggunakan *reuseable component*, rancangan dibuat dengan memperhatikan komponen yang tersedia. Sehingga perancangan dan kebutuhan system yang dibuat diubah mengikuti komponen yang tersedia. Perancangan ini kurang efisien dan interaktif dibanding perancangan untuk keperluan khusus, tetapi perancangan ini dapat mengurangi biaya yang diperlukan untuk sistem, dan tingkat kepercayaan dapat meningkat karena menggunakan komponen yang pernah digunakan untuk berbagai sistem sebelumnya.

2.4.3 Design Pattern

Penggunaan kembali suatu komponen dapat disimpulkan seperti penggunaan *abstract design* yang kemudian diterapkan disesuaikan dengan kebutuhan aplikasi. Pendekatan ini sering disebut *design pattern*. *Design pattern* dikemukakan pertama kali oleh E. Gamma, R. Helm, R. Johnson, dan J. Vlissides pada tahun 1995 lewat bukunya "*Design Patterns : Elements of Reuseable Object-Oriented Software*", ide teori ini didapat dari Christopher Alexander seorang arsitektur (1977) yang menyatakan tentang adanya *pattern* (pola) tertentu yang umum dan dapat

digunakan terus-menerus dan efektif pada design bangunan. Ide ini ternyata cocok diterapkan pada design software. Design pattern pertama ini lebih dikenal dengan sebutan *Gang of Four (GoF) pattern*.

Secara umum *pattern* dapat dijelaskan sebagai bagian penting yang diidentifikasi yang menangkap informasi penting untuk penyelesaian masalah yang seringkali terjadi. Hal ini cocok dengan sifat Object-oriented yang menggunakan classes dan object yang dapat digunakan beberapa kali untuk memecahkan masalah yang sama atau hampir sama cara pemecahannya.

Object-oriented design pattern umumnya menunjukkan relasi dan interaksi antara classes dan objects, tanpa menentukan terlebih dahulu classes atau objects yang terkait. Design pattern menangkap, mengidentifikasi, menggambarkan bagian penting dari suatu struktur design biasa dan menjadikannya berguna untuk dijadikan design Object-oriented yang dapat digunakan kembali.

2.4.4 Design Pattern untuk J2EE platform

J2EE merupakan lingkungan yang kompleks. J2EE mengabungkan berbagai spesifikasi yang didukung oleh berbagai variasi tools. Kompleksitas ini dapat diterjemahkan sebagai kaya akan seperangkat fitur dimana developer dapat membangun solusi yang bagus. Dalam prakteknya, kompleksitas sering menjadi kebingungan dalam menerapkan cara yang tepat untuk menerapkan fitur dan tools (alat) yang tersedia. Sun Microsystems menangkap kebingungan ini dan pada tahun 2001 meluncurkan daftar J2EE patterns.

Design pattern pada J2EE platform didefinisikan oleh Sun Microsystems dengan melihat masalah yang sering muncul dalam pembuatan aplikasi J2EE. J2EE design pattern dibuat berdasarkan pada GoF pattern dan disesuaikan dengan platform J2EE yang bersifat multi-tier.

2.4.5 Arsitektur Model-View-Controller

Arsitektur Model-View-Controller adalah pendekatan arsitektur yang digunakan secara luas untuk aplikasi interaktif. Model-View-Controller membagi fungsionalitas diantara objects yang berkaitan dengan menjaga dan menampilkan data untuk meminimalkan tingkat ketergantungan antara objects. Arsitektur itu memetakan tugas aplikasi tradisional seperti input, proses, dan output ke dalam model grafis interaksi user. Selain itu juga memetakan aplikasi perusahaan berbasis web yang multi-tier.

Arsitektur MVC memisahkan aplikasi pada tiga layer, yaitu model, view, dan controller dan memisahkan tanggung jawab masing-masing. Tiap layer menangani tugas khusus dan memiliki tanggung jawab khusus dibandingkan layer yang lain.

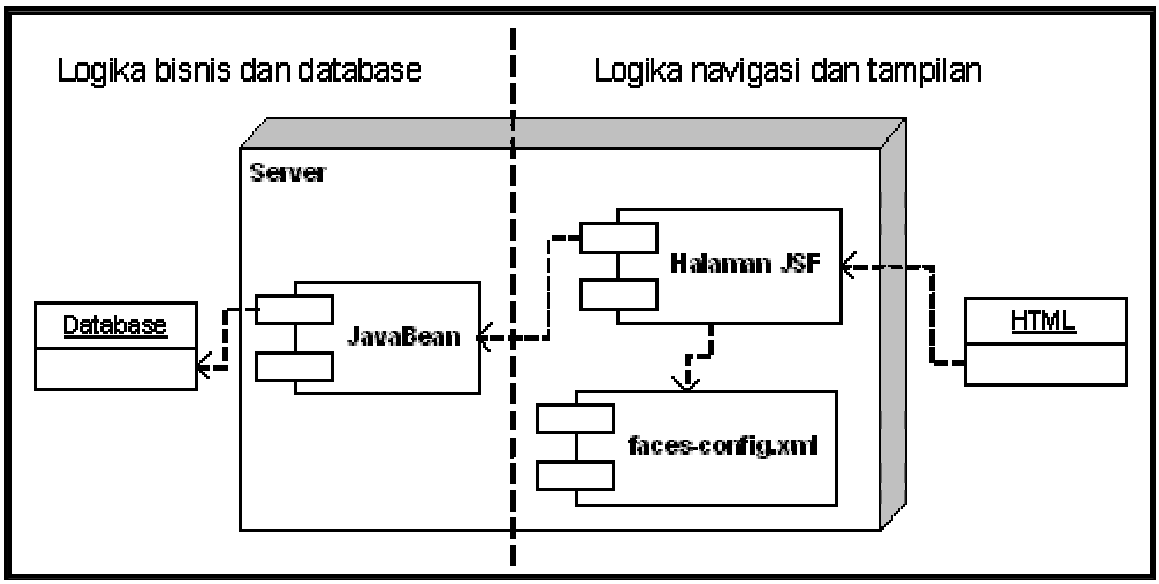
- *Model* menunjukkan data bisnis dan logika bisnis atau operasi yang menentukan akses dan modifikasi dari data bisnis itu. Seringkali model menjadi perkiraan software dengan fungsinya pada dunia nyata. Model memberitahukan views ketika ada perubahan dan menyediakan view kemampuan untuk menanyakan model tentang perubahannya. Selain itu juga menyediakan controller kemampuan untuk mengakses fungsi aplikasi yang terdapat dalam model.
- *View* menerjemahkan isi dari model. View mengakses data dari model dan menentukan bagaimana data harus ditampilkan. View juga memperbaharui tampilan data ketika model berubah. View juga meneruskan masukan (input) menuju ke controller.
- *Controller* menentukan bagaimana aplikasi bertingkahlaku. Controller mengirimkan user request dan memilih view yang akan ditampilkan, menerjemahkan user input dan memetakan ke tindakan yang harus dilaksanakan oleh model. Pada GUI (Graphical User Interface) client yang berdiri sendiri (stand alone), user input termasuk button click (klik pada tombol) dan pemilihan menu. Pada aplikasi web berupa request HTTP GET dan POST ke web tier. Controller memilih view selanjutnya yang akan ditampilkan berdasarkan interaksi user dan keluaran dari operasi model. Aplikasi biasanya memiliki satu contoller untuk tiap fungsi yang terhubung. Beberapa aplikasi menggunakan controller terpisah untuk tiap type client yang berbeda, karena interaksi dan seleksi view berbeda antara client yang berbeda.

Pemisahan tanggung jawab antara model, view dan controller mengurangi duplikasi code dan membuat aplikasi mudah dipelihara. Karena logika bisnis dipisahkan dari data, penanganan terhadap data menjadi lebih mudah baik untuk penambahan sumber data baru atau perubahan terhadap tampilan data. Dukungan untuk type client baru juga menjadi lebih mudah, karena tidak diperlukan perubahan pada logika bisnis untuk penambahan tiap type client yang baru.

3. Rancangan Sistem Toko Online Ponsel

3.1 Peran Java Server Faces pada Sistem Toko HP Online

Untuk membuat aplikasi yang digunakan adalah program JSF, maka perlu diketahui dimana peran JSF pada sistem. JSF merupakan salah satu teknologi pemograman yang disediakan oleh java untuk aplikasi web, dan JSF dibuat untuk mengantikan user interface yang selama ini disediakan HTML. JSF menjalankan render untuk halaman web pada server dan menampilkan hasil keluaran ke browser, yang pada JSP tag untuk HTML sama sekali tidak diubah dan hanya memroses logika script JSP-nya saja. Berikut dapat dilihat skema proses JSF untuk menampilkan halaman HTML.

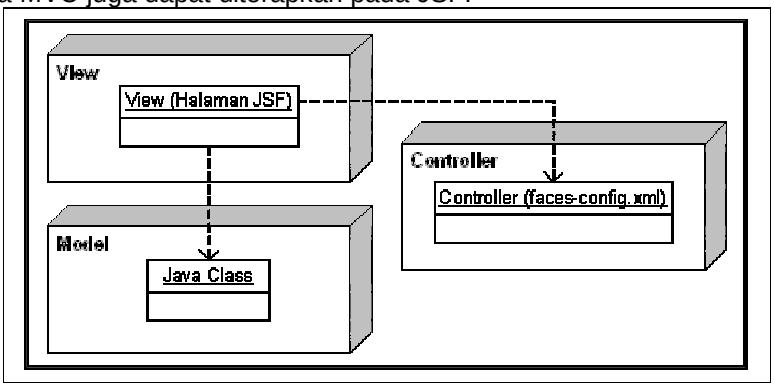


Gambar 1. Skema Proses JSF

Dengan memperhatikan skema dapat dilihat bahwa JSF berperan terutama pada layer presentasi. Dan untuk layer bisnis dan integrasi diatur oleh Javabean.

3.2 Gambaran MVC untuk Java Server Faces

Java Server Faces merupakan bagian dari spesifikasi yang disediakan oleh platform J2EE, maka pola MVC juga dapat diterapkan pada JSF.

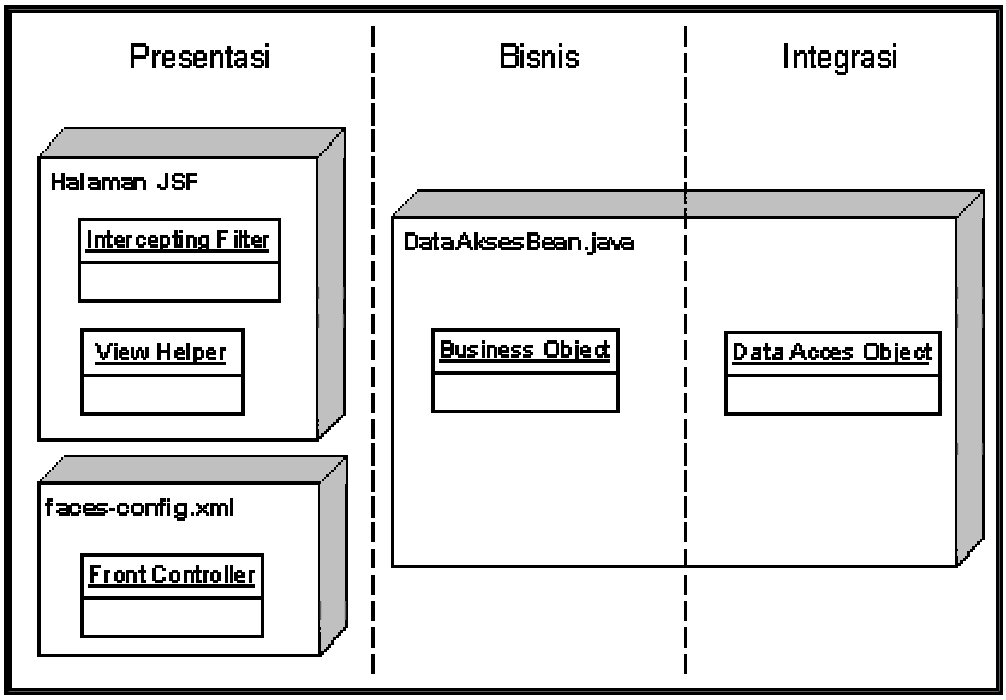


Gambar 2. Pola MVC

Pola MVC pada aplikasi JSF lebih memiliki pemisahan yang jelas dibanding aplikasi JSP. Umumnya pada aplikasi JSP, controller diletakkan pada halaman JSP. Sehingga antara view dan controller tidak terpisah secara fisik, tetapi hanya berdasarkan jenis script yang digunakan.

3.3 Pattern Pada Sistem

J2EE pattern yang akan diterapkan pada system terdapat pada tiga layer, berikut rancangan untuk penerapannya :



Gambar 3. Pattern Pada Sistem

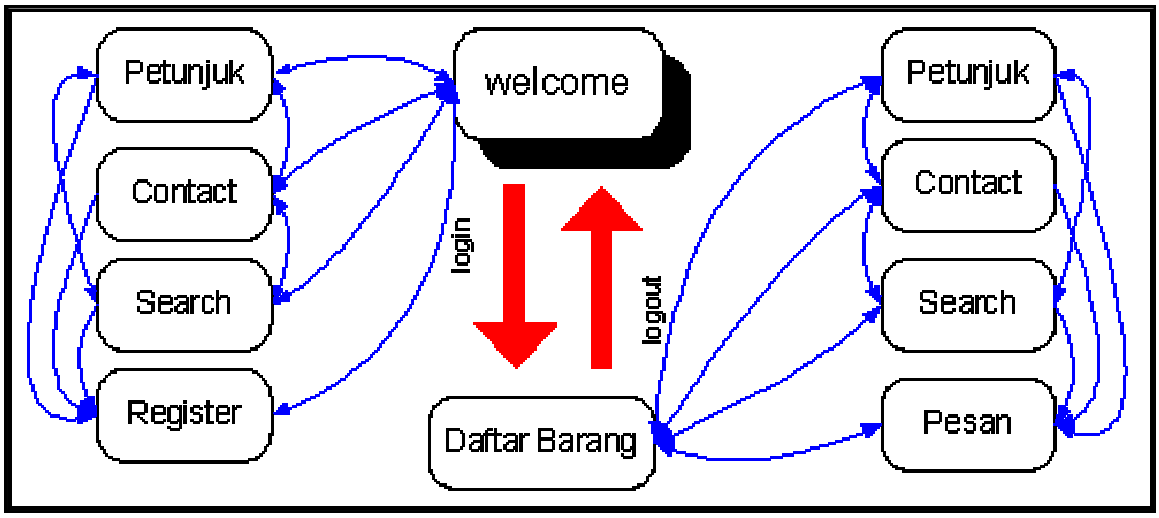
Pattern yang akan diterapkan pada aplikasi, yaitu intercepting filter, view helper, front controller untuk layer presentasi, business object untuk layer bisnis, dan data access object untuk layer integrasi.

Pattern yang digunakan adalah pattern yang umumnya muncul pada aplikasi web. Untuk alur implementasi, client pada aplikasi bisa pelanggan atau administrator yang melakukan akses ke web site, kemudian diterima oleh Front Controller yang berfungsi untuk mengatur data yang dikirim bersama request. Front Controller akan menentukan apakah request perlu di-filter sebelum dikirim ke view dan menentukan view yang akan ditampilkan. Intercepting Filter berfungsi untuk mem-filter data jika diperlukan sebelum diproses. View akan menerima request dan menggunakan View Helper atau Business Object untuk menampilkan bagian yang dinamisnya. Business Object diperlukan jika proses terkait dengan logika bisnis dan memerlukan data dari data source. Data Access Object menyediakan akses dari dan menuju data source. Untuk tampilan yang tidak memerlukan akses ke data source atau logika bisnis tertentu, View Helper berfungsi menampilkan tampilan halaman web yang sesuai dengan request dari client.

Aplikasi selain berfungsi memberikan tampilan halaman web yang sesuai request, juga berfungsi menyimpan data transaksi ke dalam data source. Business Object berfungsi untuk menyampaikan data yang akan disimpan, dan Data Access Object yang mengatur akses ke data source dan menyimpan data.

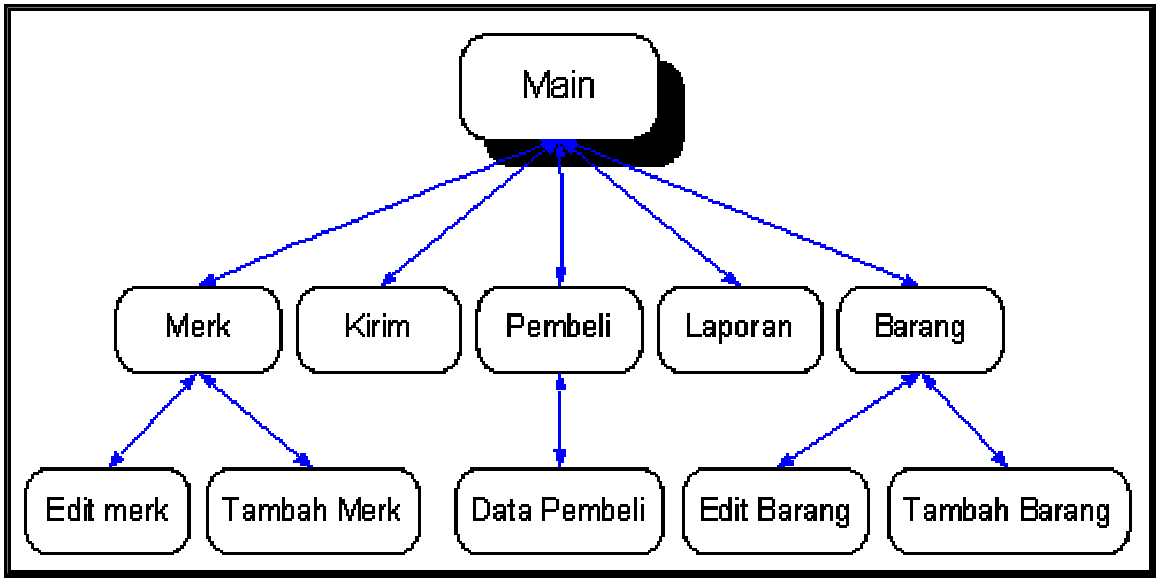
3.4 Rancangan Website

Web site yang akan dibuat terdiri dari 2 bagian yang berbeda, yaitu website untuk pelanggan dan administrator. Berikut rancangan struktur web site yang akan dibuat :



Gambar 4. Rancangan Struktur Website Pelanggan

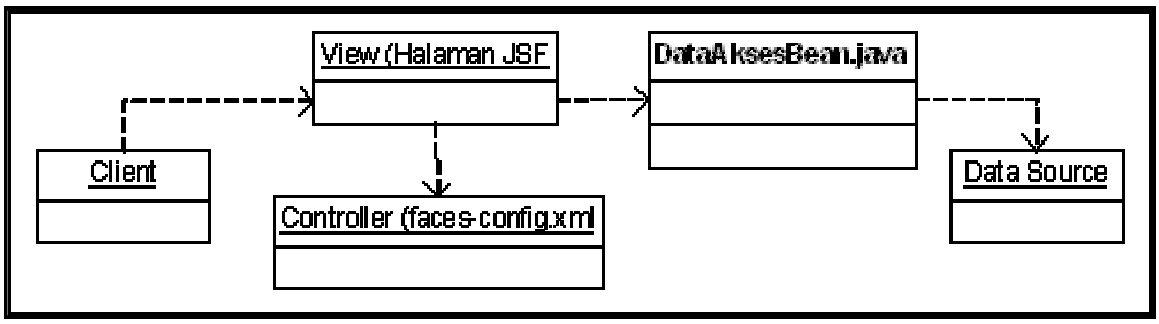
Rancangan web site berpusat pada halaman welcome untuk pelanggan yang belum melakukan login. Untuk pelanggan yang telah login, halaman daftar barang merupakan halaman pertama yang ditampilkan. Halaman register hanya dapat diakses oleh pelanggan yang belum melakukan login. Halaman pesan barang hanya dapat diakses oleh pelanggan yang telah melakukan login. Untuk halaman petunjuk dan contact dapat diakses sesudah atau sebelum login, hanya tampilan menu yang berubah untuk membedakan pelanggan yang sudah atau belum login. Pada halaman search walau dapat diakses semua pelanggan, tetapi ada fitur tambahan yang disediakan untuk pelanggan yang sudah login.



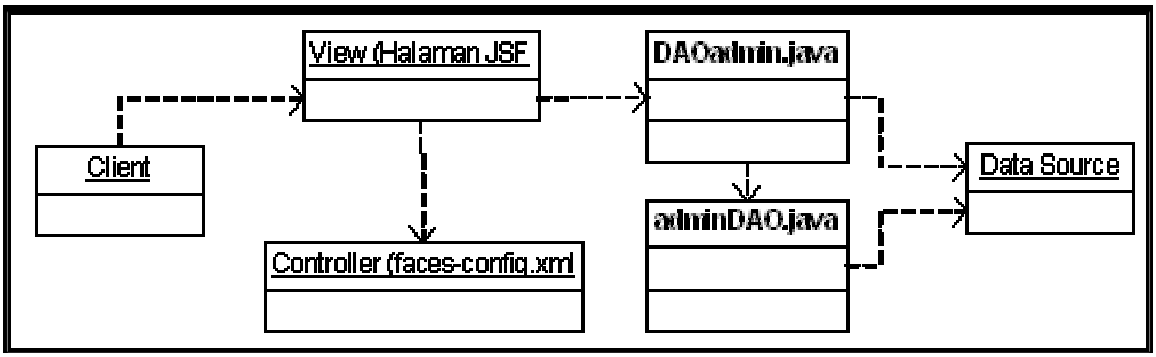
Gambar 5. Rancangan Struktur Website Administrator

4. Implementasi Sistem

Pada rancangan class yang mengatur layer bisnis dan integrasi adalah class yang sama. Dengan demikian pemisahan antara layer hanya terlihat jelas pada layer presentasi. Karena itu pada dicoba untuk membuat implementasi lain dari JSF untuk memisahkan lagi layer bisnis dengan layer integrasi. Untuk web site admin akan ditambahkan pattern yang disebut application service. Pattern ini menambahkan coding yang harus dilakukan untuk memisahkan antara layer bisnis dan integrasi. Pemisahan tersebut memudahkan untuk penggunaan kembali oleh web site yang sama atau web site lain yang memiliki fungsi sama. Gambar 4.xxx dan 4.xxxx menunjukkan perbedaan pengunaan business object pada web pelanggan, dibandingkan dengan pengunaan application service dan business object pada web admin.



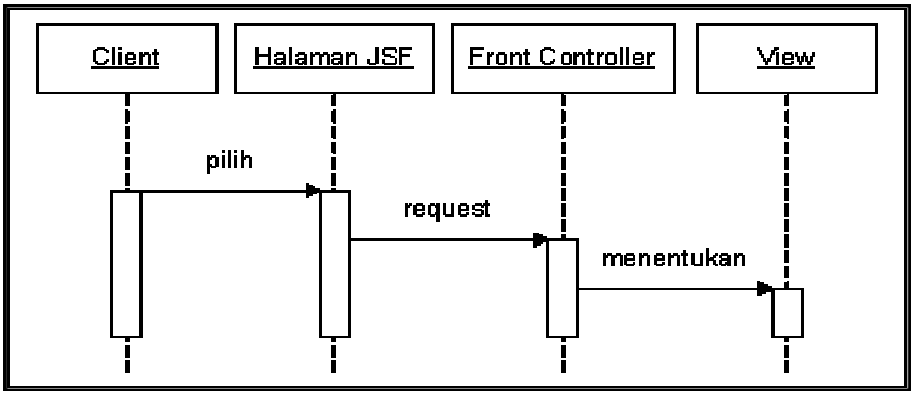
Gambar 6. Class diagram website pelanggan



Gambar 7. Class diagram website admin

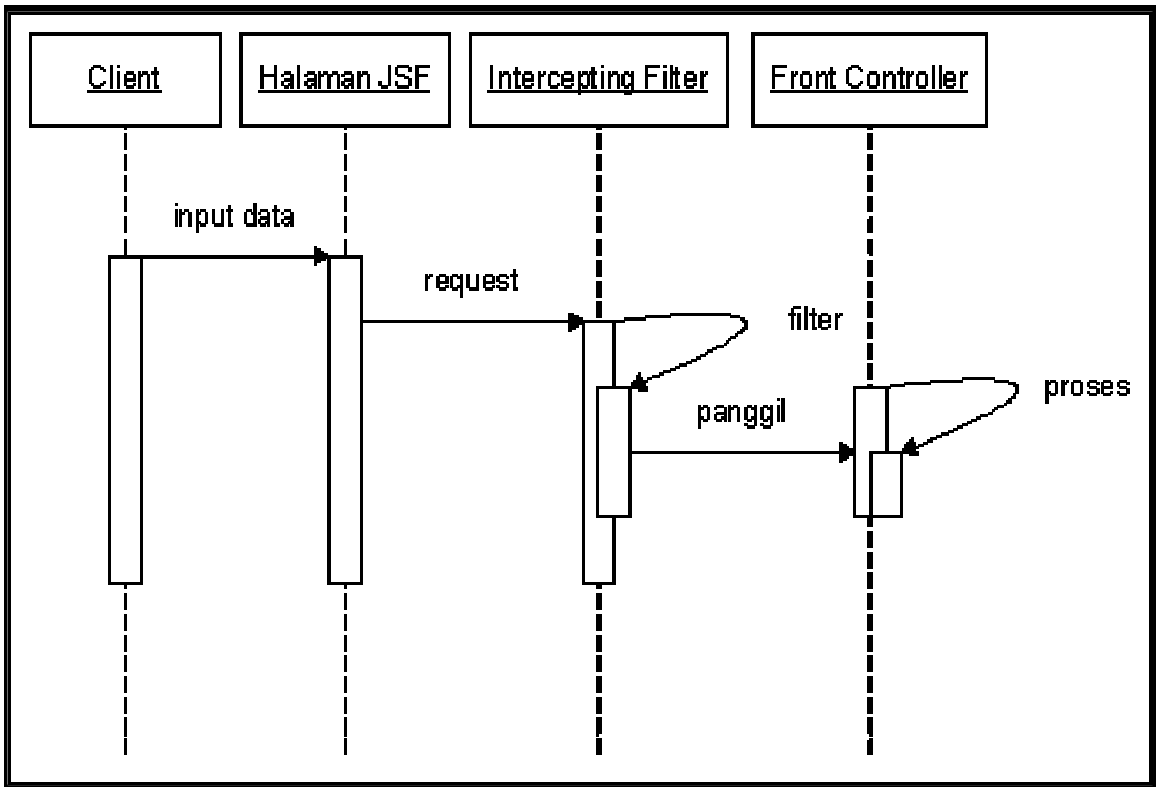
4.1 Implementasi Pattern Pada Web

Pattern yang digunakan pada web secara keseluruhan ada 6 pattern. Dari 6 pattern tersebut 3 diantaranya dapat diimplementasikan pada JSF tanpa menggunakan *coding* java sama sekali. Hal ini dikarenakan JSF telah menyediakannya pada spesifikasi JSF. Berikut sequence diagram dari 3 pattern tersebut, dan penjelasan implementasinya pada JSF.



Gambar 8. Sequence diagram Front Controller

Gambar 4.5 merupakan implementasi dari “faces-config.xml” pada aplikasi JSF. Client mengakses server melalui halaman JSF dengan memilih tampilan web selanjutnya. Kemudian oleh “faces-config.xml” yang berfungsi sebagai Front Controller akan menerima request dan menentukan view yang ditampilkan ke client. Pattern ini terletak pada layer presentasi.

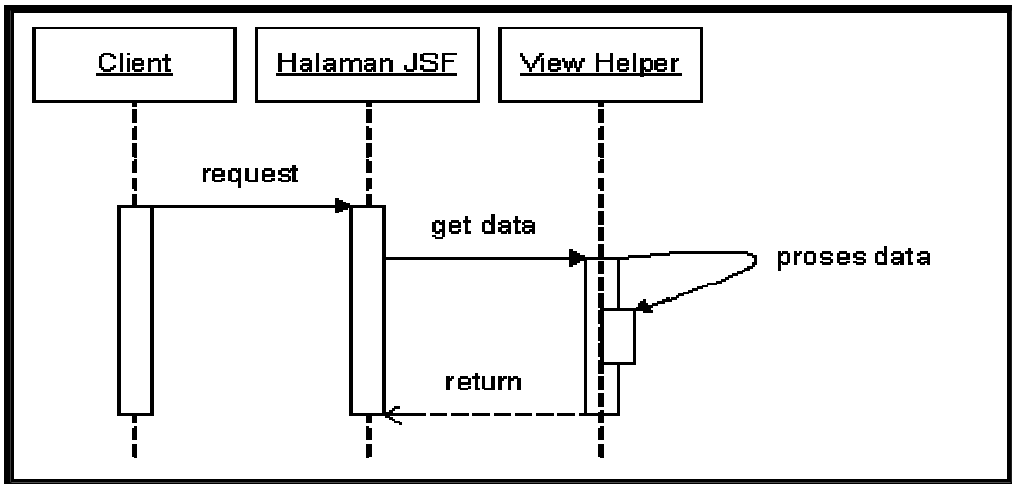


Gambar 9. Sequence diagram Intercepting Filter

Gambar 4.6 menunjukkan implementasi dari pattern Intercepting Filter. Sequence diagramnya mirip dengan sequence diagram Front Controller karena implementasinya memang terjadi sebelumnya dan berhubungan dengan Front Controller.

Intercepting Filter berfungsi memastikan data yang dikirimkan valid, dan mengkonversi data yang dikirimkan ke view agar sesuai dengan yang diinginkan. Pattern ini juga terletak pada layer presentasi dan implementasinya sudah terdapat dalam spesifikasi JSF. Untuk melakukan validasi dan konversi script diletakkan pada Halaman JSFnya.

Selanjutnya akan dijelaskan pattern lain yang terletak pada layer presentasi, yaitu View Helper. Pattern ini berfungsi untuk membantu Front Controller untuk memberikan tampilan yang sesuai bagi user. View Helper bisa berupa class javabean atau script yang diletakkan pada Halaman JSF.

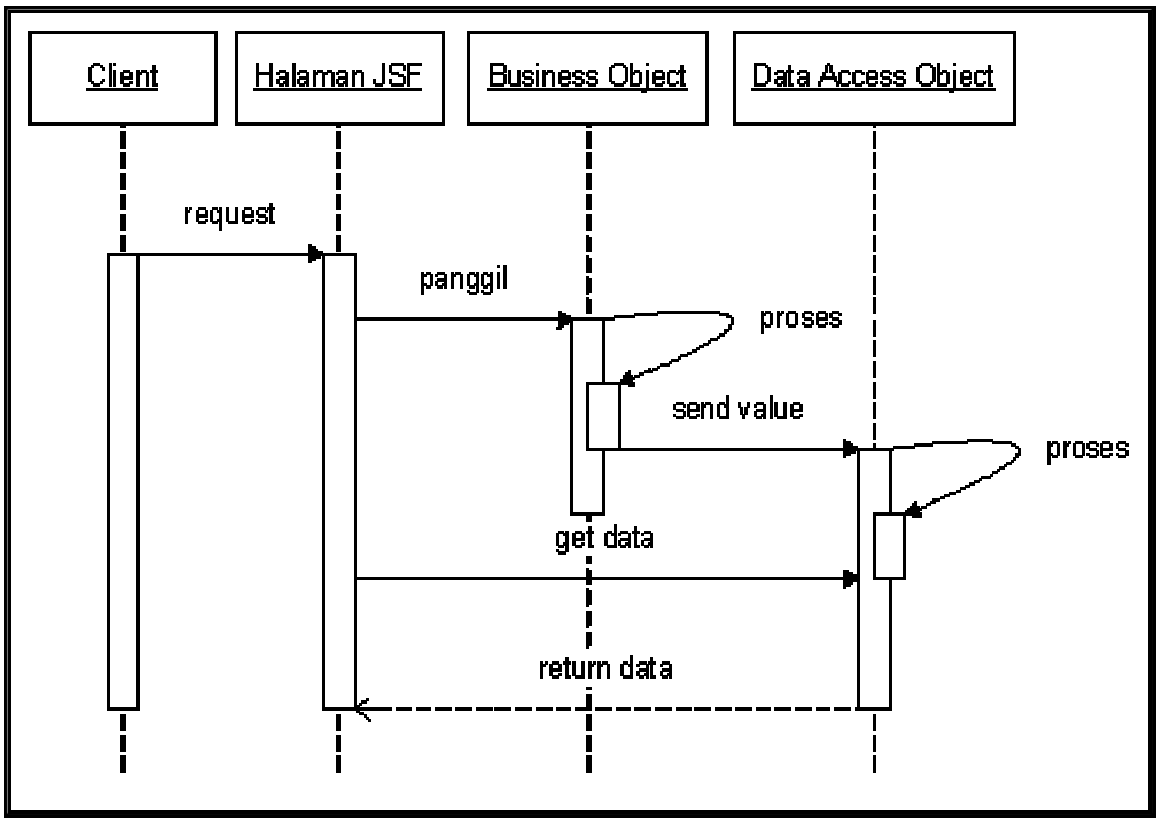


Gambar 10. Sequence diagram View Helper

Sequence diagram gambar 4.7 menunjukkan bahwa View Helper hanya berfungsi membantu Halaman JSF.

Untuk 3 pattern yang lain, 2 diantaranya adalah pattern yang berada pada layer bisnis, dan salah satunya adalah application service yang ditambahkan hanya pada aplikasi admin. Penjelasan 3 pattern yang tersisa akan dibuat berdasarkan 2 metode pemanfaatannya pada web site, karena ketiga berkaitan pada implementasinya.

Implementasi pada web site pelanggan tidak menggunakan application service sepeti yang dilakukan pada web site admin. Hal ini sesuai dengan rancangan dan lebih sederhana dibanding dengan web site admin. Implementasi ini lebih tepat jika tidak ingin mengunakan kembali class pada aplikasi lain.



Gambar 11. Sequence diagram Business Object

4.1 Analisa Hasil Implementasi

Aplikasi yang dibuat terbagi atas 3 bagian, yaitu implementasi web site pelanggan yang tidak memperhatikan pengunaan kembali class, implementasi web site admin yang menambahkan coding untuk pengunaan kembali class pada JSF, dan implementasi web site sample yang memberikan contoh pengunaan kembali suatu component dari JSF.

Implementasi web site yang dibuat menunjukkan perbandingan antara aplikasi JSF dengan aplikasi JSP, perbedaan pengunaan pattern tertentu pada JSF dan kemampuan pengunaan kembali pada aplikasi.

Dari implementasi web site JSF yang dibuat, terlihat bahwa JSF memiliki sifat yang berbeda dengan JSP. JSF merupakan pengembangan dari aplikasi JSP dengan memisahkan layer presentasi dari layer lainnya. Hal ini terlihat pada pemanfaatan pattern-pattern pada layer presentasi telah disediakan pada spesifikasi JSF, sedangkan pada JSF umumnya harus dilakukan dengan mengabungkannya dengan pemograman Servlet atau dengan menempatkan *coding* pada halaman JSP itu sendiri.

J2EE Pattern merupakan pattern yang berasal dari implementasi aplikasi java yang terdahulu. Dan ternyata dapat digunakan pada aplikasi JSF, hal ini disebabkan karena JSF menggunakan JavaBean untuk menampung semua logika bisnis dan integrasinya.

Pada JSP logika yang ditempatkan pada halaman JSPnya tidak dapat digunakan kembali pada aplikasi yang lain. JSF memisahkan logikanya ke dalam JavaBean sehingga pengunaan kembali dapat dilakukan jika aturan untuk menggunakan logikanya dipenuhi. Proses pengunaan kembali ini seperti pada implementasi JSP yang menggunakan custom tags.

Tabel 4.1 yang menunjukkan perbedaan antara JSF dan JSP berdasarkan layer pattern-pattern J2EE diterapkan :

Tabel 4.1 Tabel perbandingan JSP dan JSF

Layer	JSP	JSF
Presentasi	• Untuk menerapkan Pattern Front Controller, Intercepting Filter, dan View Helper harus dilakukan programming rumit yang diletakkan pada halaman JSP sebagai bagian dari logika.	• Front controller digantikan oleh file faces-config.xml. • Intercepting filter untuk inputan data dapat dilakukan tanpa programming yang rumit. • View Helper untuk UI component dapat dilakukan oleh JSF tanpa programming yang rumit.
Bisnis	• Business Object diterapkan dengan membuat class Java untuk akses ke file yang dikirim saat proses request. • Application service diterapkan dengan menbuat class Java Bean yang membantu mengakses class yang lain.	• Business Object cara penerapan sama pada aplikasi JSF. Tetapi berperan lebih penting karena harus ada sebagai penghubung class java dengan Halaman JSF. • Application service tidak memiliki perbedaan cara penerapan dengan JSP.
Integrasi	• Data Access Bean adalah class yang berisi programming yang digunakan untuk akses database.	• Data Access Bean pada aplikasi JSF tidak berbeda karena cara menggunakan class dan programming yang digunakan sama.

5. Penutup

JSF tidak memiliki perbedaan pada layer bisnis karena programming yang dilakukan sama, yaitu menggunakan Java bean sebagai dasar programmingnya. JSF tidak memiliki perbedaan pada layer integrasi karena cara untuk akses database dan programming untuk memanggil databasenya sama. J2EE Pattern dapat diterapkan pada JSF seperti yang diterapkan pada aplikasi J2EE lainnya. Bahkan beberapa fitur dari JSF merupakan implementasi dari pattern, seperti faces-config.xml yang merupakan implementasi front controller.