Teknologi JavaServer Faces

Frederic Constantianus Bokau
Program Studi D3 Teknologi Informasi, Fakultas Teknologi Informasi
Universitas Kristen Maranatha
Jl. Suria Sumantri 65 Bandung
frederick_constantianus@yahoo.com

Abstract

Following the rapid development of web-based programming concept, JavaServer Faces was released to become one of the most important features offered.Lining up from the root of Java family, and was developed as an improvement for Java Server Page or JSP, JSF as it is commonly pronounced, is a new server side User interface component for Java technology. Java developer would find new benefits and a more enhanced development, as well as deployment process using JSF. With minimal effort, building a web application based on Java technology would be streamlined as its most effective way. This article will explain globally about the new JSF technology, and the basic understanding about it.

Keyword : *Java, JavaServer Faces, Java Server Page, JSF, JSP*

1. Pendahuluan

JavaServer Faces adalah sebuah teknologi baru di dalam bahasa pemrograman Java. Pembangunan aplikasi berbasiskan web menggunakan teknologi Java, dapat dilakukan dengan Java Server Page. Kini Java Server Page dilengkapi dengan adanya teknologi JavaServer Faces. JavaServer Faces adalah teknologi baru yang merupakan framework untuk berbagai komponen User interface dalam pembuatan aplikasi web. Teknologi server-side ini, memberikan kemudahan dan mempercepat proses pengembangan aplikasi web berbasis Java.

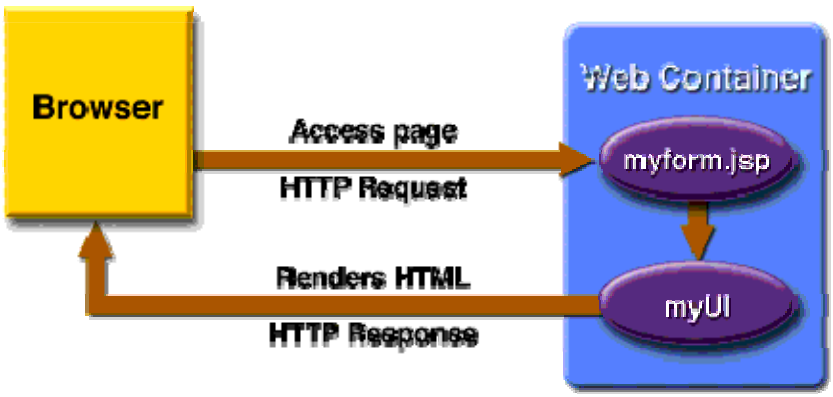
Komponen utama dari JavaServer Faces adalah:

- Kumpulan API (Application Programming Interface) untuk merepresentasikan komponen *User interface* (*UI*) dan menangani berbagai proses didalamnya

- Dua *custom tag libraries Java Server Page* (*JSP*), untuk mendeklarasikan komponen *UI* dalam halaman *JSP* dan menyambungkannya dengan objek-objek *server side*
- Model *server side event*
- *State Management*
- *Managed Beans*, *JavaBeans* yang dikembangkan dengan metode *dependency injection*
- *Unified Expression Language*, untuk teknologi *JSP 2.0* dan *JSF 1.2*

Kumpulan *tag libraries* dan model pemrograman yang ada dalam *JavaServer Faces*, mempermudah pengembangan dan pengelolaan aplikasi *web* yang menggunakan *UI server-side*. Dengan pengembangan yang sederhana, seorang *web developer* dapat melakukan beberapa hal berikut:

- Menghubungkan event berbasis client-side ke dalam kode aplikasi yang server-side
- Melakukan binding data dengan komponen UI dalam halaman web
- Membangun UI menggunakan komponen yang sudah ada dan dapat dikembangkan lagi
- Menyimpan dan mengembalikan state UI melebihi batas penggunaan server request.



Gambar 1. Bagan Kerja UI pada Server

Sumber: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSFIntro.HTML

User interface yang dikembangkan menggunakan JavaServer Faces (dilambangkan dengan myUI pada gambar.1.), berjalan di server dan akan direnderkembali pada client saat sebuah halaman JSP (myform.jsp) menerima HTTP Request. Halaman JSP tersebut, adalah halaman JavaServer Faces, yaitu halaman JSP yang didalamnya terdapat tag-tag JavaServer Faces. Didalamnya, berbagai komponen UI dideklarasikan menggunakan teknologi JavaServer Faces. UI untuk aplikasi web tersebut, akan mengatur objek-objek yang direferensikan oleh halaman JSP. Objek tersebut antara lain:

- Komponen UI yang dipetakan dari tag-tag pada halaman JSP.
- Event Listener, Validator dan Converter yang terdaftar pada komponen.
- Objek yang mengenkapsulasi data dan fungsi pada tiap komponen.

2. Perkembangan Java Server Faces

JavaServer Faces versi 1.0 pertama kali diluncurkan pada 11 Maret 2004. Spesifikasi JSF ini dikembangkan sebagai JSR 127 dalam Java Community Process. Perbaikan dari versi 1.0, adalah versi 1.1 dimana banyak bug yang diperbaiki, kendati tidak ada perubahan pada spesifikasi maupun renderkit untuk HTML. Versi 1.1 dirilis pada tanggal 27 Mei 2004 dan masih berada dalam JSR 127.

Rilis terbaru yaitu JSF 1.2, diluncurkan pada 11 Mei 2006 dibawah JSR 252. Didalamnya terdapat beberapa perbaikan dan fitur baru antara lain:

- Pengembangan yang menyediakan solusi interim terhadap permasalahan content-interweaving.
- Adanya skema XML untuk dokumen config, tidak hanya berbasiskan DTD saja.
- Pengembangan yang memungkinkan aplikasi ditangani dalam multi-frame atau dalam desain UI multi-window.
- Pengembangan untuk tag library f: untuk memperbaiki cakupan TCK, masa berlaku dari f:view, dan berbagai fitur tambahan.
- Pengembangan untuk dukungan decorator objek-objek API.
- Pengembangan sekuritas untuk penyimpanan state berbasis client-side.
- Re-organisasi spesifikasi menjadi bentuk normatif dan non-normatif, untuk mempermudah implementasi.
- Perbaikan terhadap bug-bug portlet.
- Perbaikan terhadap bug yang membutuhkan perubahan spesifikasi minimum.

3. Teknologi JavaServer Faces

Secara umum, JavaServer Faces memiliki karakteristik yang sama dengan aplikasi web Java lainnya. Berjalan dalam container servlet dan umumnya memiliki komponen JavaBeans untuk fungsionalitas dan data, Event Listeners, Halaman web seperti JSP, serta Class-Class untuk pengolahan server-side.

Sebagai tambahan, sebuah aplikasi JavaServer Faces akan memiliki:

- Tag library untuk merender komponen UI dalam halaman
- Tag library untuk menjalankan event handler, validator dan aksi lain
- Komponen UI yang merupakan objek ber-state dalam server
- Backing Beans, yang menyediakan properti dan fungsi untuk komponen UI
- Validator, Converter, Event Listener dan Event Handlers
- Dokumen konfigurasi untuk mengatur resource aplikasi

Aplikasi JavaServer Faces yang menggunakan JSP untuk merender HTML, harus menyertakan tag library yang akan mendeklarasikan komponen UI. Tag library ini menghilangkan kebutuhan digunakannya komponen UI tingkat tinggi dalam HTML, atau bahasa markup lain sehingga komponen UI tersebut dapat didaur ulang. Berikut adalah bagan tag library utama (core) yang ada dalam JSF.

Tabel 1. Kumpulan Core Tag Libraries JavaServer Faces

Source: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSFPage4.HTML

Tag Categories	*Tags*	*Functions*
Event-handling tags	*actionListener*	*Registers an action listener on a parent component*
	valueChangeListener	*Registers a value-change listener on a parent component*
Attribute configuration tag	*attribute*	*Adds configurable attributes to a parent component*
Data conversion tags	*converter*	*Registers an arbitrary converter on the parent component*
	convertDateTime	*Registers a DateTime converter instance on the parent component*
	convertNumber	*Registers a Number converter instance on the parent component*

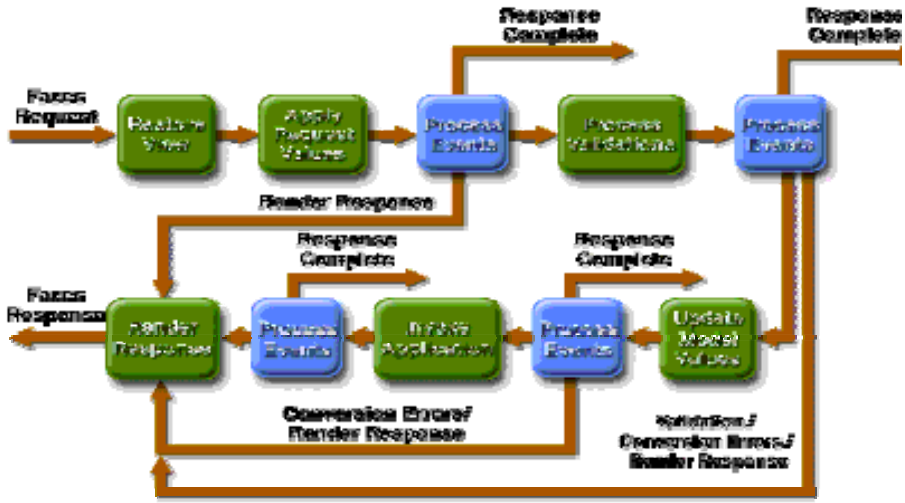
Tag Categories	Tags	Functions
Facet tag	facet	Signifies a nested component that has a special relationship to its enclosing tag
Localization tag	loadBundle	Specifies a ResourceBundle that is exposed as a Map
Parameter substitution tag	param	Substitutes parameters into a MessageFormat instance and adds query string name-value pairs to a URL
Tags for representing items in a list	selectItem	Represents one item in a list of items in a UISelectOne or UISelectMany component
	selectItems	Represents a set of items in a UISelectOne or UISelectMany component
Container tag	subview	Contains all JavaServer Faces tags in a page that is included in another JSP page containing JavaServer Faces tags
Validator tags	validateDoubleRange	Registers a DoubleRangeValidator on a component
	validateLength	Registers a LengthValidator on a component
	validateLongRange	Registers a LongRangeValidator on a component
	validator	Registers a custom validator on a component
Output tag	verbatim	Generates a UIOutput component that gets its content from the body of this tag
Container for form tags	view	Encloses all JavaServer Faces tags on the page

Daur hidup (*lifecycle*) sebuah halaman *JavaServer Faces* serupa dengan sebuah halaman *JSP*. *Client* mengirimkan *HTTP Request* untuk meminta halaman, dan *server* akan merespon dengan *HTTP Response* berupa halaman *web* yang sudah ditranslasikan ke dalam bentuk *HTML*. Namun, dalam prosesnya *JavaServer Faces* menambahkan beberapa layanan tambahan.

Sebuah halaman *JavaServer Faces*, digambarkan sebagai bentuk *tree* dari komponen *UI*. Hal ini disebut *View*. Saat *client* melakukan *request* untuk

halaman tertentu, daur hidup dimulai. Sepanjang masa daur hidup tersebut, *JavaServer Faces* akan membangun *view* dengan mengacu pada kondisi *state* yang dikirimkan halaman sebelumnya. Saat *client* mengirim sebuah halaman, *JavaServer Faces* akan melakukan beberapa langkah, seperti melakukan validasi *input* dalam view tersebut, dan melakukan konversi menjadi tipe data yang dikenali *server*. Setiap implementasi *JavaServer Faces*, akan menjalankan proses-proses ini dalam daur hidupnya.

Ada dua macam *request* yang dikenali *JavaServer Faces*, *Faces Request* dan *Non Faces Request*. Sebaliknya pun ada dua macam *response* yang akan dijalankan yaitu *Faces Response* dan *Non Faces Response*. Dalam prakteknya, sistem akan menangani dua jenis *request*, yaitu *initial request* dan *postbacks*. *Initial request* adalah *request* terhadap sebuah halaman, yang dilakukan untuk pertama kali oleh *user*. *Postbacks*, adalah *request* yang dijalankan kembali terhadap sebuah halaman untuk mengolah data, sebagai kelanjutan dari *initial request*. Saat daur hidup berjalan pada pemrosesan *initial request*, hanya fase *Restore View* dan *Render Response* yang dijalankan. Sebaliknya, untuk menangani *postbacks*, system akan menjalankan semua fase, seperti yang terlihat pada gambar 2.



Gambar 2. Daur Hidup Request Response Standar

Source: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSFIntro10.HTML

4. Komponen UI Java Server Faces

Fitur utama yang ditawarkan *JavaServer Faces* adalah kumpulan komponen *UI* terintegrasi, yang siap digunakan dan dikembangkan dalam pembangunan sebuah aplikasi *web*. Komponen *UI* tersebut merupakan elemen yang dapat dikonfigurasi, dan dapat digunakan kembali bilamana dibutuhkan. Komponen yang dimaksud mencakup bentuk sederhana seperti *button* biasa, sampai kepada yang kompleks seperti struktur tabel yang terdiri dari banyak komponen.

Arsitektur komponen *JavaServer Faces* terdiri dari:

- Kumpulan *Class UI*Component untuk menentukan *state* dan *behavior* komponen *UI.*
- Model *rendering* yang menentukan berbagai cara untuk merender komponen.
- Model *event* dan *listener* untuk menangani *event* dalam tiap komponen.
- Model konversi yang menentukan bagaimana menggunakan *converter* data dalam komponen.
- Model validasi yang menentukan bagaimana *validator* digunakan dalam komponen.

Berikut pembahasan lebih lanjut mengenai komponen tersebut:

- ***Class UI Component***
  Semua komponen *UI* dalam *JavaServer Faces* merupakan turunan dari *UIComponentBase*, yang menentukan *state* dan *behavior* dasar untuk sebuah komponen *UI*. Berikut adalah kumpulan *Class* komponen *UI* yang disertakan dalam *JavaServer Faces*.

  - *UIColumn*
    Mengacu pada satu kolom data dalam komponen *UIData*.
  - *UICommand*
    Mengacu pada kontrol terhadap aksi tertentu saat dijalankan.
  - *UIData*
    Mengacu pada proses data *binding* untuk koleksi data yang dihasilkan oleh *instance DataModel*.
  - *UIForm*
    Mengacu pada kumpulan kontrol untuk mengirimkan data ke aplikasi. Komponen ini diibaratkan seperti *FORM* pada HTML.

- *UIGraphic*
  Menampilkan gambar.
- *UIInput*
  Menerima *input* dari *user,* merupakan *sub-Class* dari *UI*Output.
- *UIMessage*
  Menampilkan pesan lokal.
- *UIMessages*
  Menampilkan kumpulan pesan lokal.
- *UIOutput*
  Menampilkan keluaran keluaran pada halaman.
- *UIPanel*
  Mengatur *layout* dari komponen dibawahnya.
- *UIParameter*
  Mengacu pada parameter *sub*stitusi.
- *UISelectBoolean*
  Memungkinkan *user* menghasilkan nilai *boolean* pada sebuah kontrol dengan cara melakukan seleksi. Merupakan *subClass* dari *UIInput*.
  - *UISelectItem*
    Mengacu pada satu buah item dari kumpulan item.
  - *UISelectItems*
    Mengacu pada kesatuan atau satu set item.
  - *UISelectMany*
    Memungkinkan *user* memilih beberapa item dari grup item. Merupakan *subClass* dari *UIInput*.
  - *UI*SelectOne
    Memungkinkan *user* memilih satu item dari grup item. Merupakan *subClass* dari *UIInput*.
  - *UIViewRoot*
    Mengacu pada poin awal dari hierarkis *view* sistem.

Sebagai tambahan, semua *Class* komponen juga mengimplementasikan satu atau lebih *behavioral interface*, antara lain:

- *ActionSource*
  Menunjukkan bahwa komponen tersebut dapat menjalankan event tertentu.
- *EditableValueHolder*
  Turunan dari ValueHolder dan menentukan fitur tambahan untuk komponen yang dapat diedit, seperti event validasi dan perubahan nilai.
- *NamingContaine*
  Mengharuskan bahwa ID unik diberikan pada tiap komponen.
- *StateHolder*
  Menunjukkan bahwa komponen tersebut memiliki state yang harus disimpan diantara request.
- *ValueHolder*
  Menunjukkan bahwa komponen tersebut mengelola nilai lokal serta akses data.

UICommand menerapkan *ActionSource* dan *StateHolder*. *UIOutput* dan turunannya menerapkan *StateHolder* dan *ValueHolder*. *UIInput* dan turunannya menerapkan *EditableValueHolder*, *StateHolder* dan *ValueHolder*. *UIComponentBase* menerapkan *StateHolder*.

- ***Component Rendering Model***
  Sebuah komponen akan dirender sesuai dengan bentuk tampilannya. Berikut kumpulan *tag-tag* umum yang digunakan untuk merender data menjadi bentuk *HTML* tertentu di halaman *web*.

Tabel 2. Kumpulan Tag UI Component JavaServer Faces

Sumber: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/*JSF*Page4.*HTML*

Tag	*Functions*	*Rendered As*	*Appearance*
column	*Represents a column of data in a UIData component.*	*A column of data in an HTML table*	*A column in a table*
commandButton	*Submits a form to the application.*	*An HTML <input type=type> element, where the type value can be submit, reset, or image*	*A button*
commandLink	*Links to another page or location on a page.*	*An HTML <a href> element*	*A hyperlink*
dataTable	*Represents a data wrapper.*	*An HTML <table> element*	*A table that can be updated dynamically*
form	*Represents an input form. The inner tags of the form receive the data that will be submitted with the form.*	*An HTML <form> element*	*No appearance*
graphicImage	*Displays an image.*	*An HTML <img> element*	*An image*
inputHidden	*Allows a page author to include a hidden variable in a page.*	*An HTML <input type=hidden> element*	*No appearance*
inputSecret	*Allows a user to input a string without the actual string appearing in the field.*	*An HTML <input type=password> element*	*A text field, which displays a row of characters instead of the actual string entered*

Tag	Functions	Rendered As	Appearance
inputText	*Allows a user to input a string.*	*An HTML <input type=text> element*	*A text field*
inputTextarea	*Allows a user to enter a multiline string.*	*An HTML <textarea> element*	*A multirow text field*
message	*Displays a localized message.*	*An HTML tag if styles are used*	*A text string*
messages	*Displays localized messages.*	*A set of HTML tags if styles are used*	*A text string*
outputLabel	*Displays a nested component as a label for a specified input field.*	*An HTML <label> element*	*Plain text*
outputLink	*Links to another page or location on a page without generating an action event.*	*An HTML <a> element*	*A hyperlink*
outputFormat	*Displays a localized message.*	*Plain text*	*Plain text*
outputText	*Displays a line of text.*	*Plain text*	*Plain text*
panelGrid	*Displays a table.*	*An HTML <table> element with <tr> and <td> elements*	*A table*
panelGroup	*Groups a set of components under one parent.*		*A row in a table*
selectBoolean Checkbox	*Allows a user to change the value of a Boolean choice.*	*An HTML <input type=checkbox> element.*	*A checkbox*
selectItem	*Represents one item in a list of*	*An HTML <option> element*	*No appearance*

Tag	Functions	Rendered As	Appearance
	items in a UISelectOne component.		
selectItems	Represents a list of items in a UISelectOne component.	A list of HTML <option> elements	No appearance
selectMany Checkbox	Displays a set of checkboxes from which the user can select multiple values.	A set of HTML <input> elements of type checkbox	A set of checkboxes
selectMany Listbox	Allows a user to select multiple items from a set of items, all displayed at once.	An HTML <select> element	A list box
selectManyMenu	Allows a user to select multiple items from a set of items.	An HTML <select> element	A scrollable combo box
selectOne Listbox	Allows a user to select one item from a set of items, all displayed at once.	An HTML <select> element	A list box
selectOneMenu	Allows a user to select one item from a set of items.	An HTML <select> element	A scrollable combo box
selectOneRadio	Allows a user to select one item from a set of items.	An HTML <input type=radio> element	A set of radio buttons

- ***Conversion Model***
  Aplikasi *JavaServer Faces* dapat berasosiasi dengan komponen data objek *server-side*. Objek yang dimaksud adalah *Java*Beans seperti contohnya *Backing Bean*. Aplikasi akan menerima dan membuat data objek untuk

komponen dengan jalan memanggil properti yang bersesuaian dengannya.

Saat komponen diikat pada sebuah objek, aplikasi akan melihat data komponen dari dua sisi:

- *Model View, dimana data ditunjukkan dengan tipe data seperti Int atau Long.*
- *Presentation View, dimana data ditunjukkan dalam cara yang dapat dikenali user. Misalnya java.util.date akan ditampilkan sebagai teks string dengan format mm/dd/yy.*

JavaServer Faces akan secara otomatis menkonversi data komponen antara dua *view* berbeda diatas, bilamana bean yang diasosiasikan memiliki tipe yang didukung oleh komponen data. Namun ada kalanya, data butuh dikonversi menjadi format lain. *JavaServer Faces* memungkinkan implementasi *Converter* pada komponen *UIOutput*, dan *Class* turunannya. *Converter* akan melakukan konversi data antara dua *view* secara otomatis.

- ***Event & Listener Model***
  Event Object menentukan komponen yang akan menjalankan *event* tertentu dan menyimpan informasi tentangnya. Aplikasi harus mengimplementasikan *Class Listener* pada komponen yang dimaksud. Saat *user* mengaktifkan komponen tersebut, misal dengan mengklik *button* tertentu, maka *event* akan dijalankan. Saat hal ini terjadi, *JavaServer Faces* akan memanggil *listener* yang akan memproses *event* tersebut.

  Ada tiga jenis *event* yang dikenali *JavaServer Faces*:

  - *Action Events*
    Terjadi saat *user* menjalankan komponen yang menerapkan *ActionSource*. Komponen yang dimaksud antara lain *button* dan *hyperlink*.
  - *Value-Change Events*
    Terjadi saat *user* melakukan perubahan nilai, pada komponen *UIInput* dan turunannya. Contoh komponen ini adalah *Checkbox*. Tipe komponen yang bisa menjalankan *event* ini antara lain komponen *UIInput*, *UISelectOne*, *UISelectMany* dan *UISelectBoolean*.
  - *Data-Model Events*
    Terjadi bila sebuah baris dari komponen *UIData* dipilih.

- ***Validation Model***
  JavaServer Faces menyediakan mekanisme untuk validasi data lokal dari komponen yang dapat diedit. Serupa dengan *Conversion Model*, *Validation Model* menyediakan kumpulan *tag* yang berhubungan dengan *Validator*. *Tag* tersebut, sebagian besar memiliki atribut untuk konfigurasi properti *validator*.

  Seperti yang sudah disebutkan sebelumnya, *tag libraries* dalam *JavaServer Faces* dapat dikembangkan dan diolah sesuai kebutuhannya. Komponen yang ada dapat dimodifikasi menjadi komponen baru, hal ini diistilahkan dengan Custom *UI* Component. Fitur ini memungkinkan seorang developer untuk menghasilkan komponen spesifik untuk aksi atau *event* yang berbeda, dan unik didalam halaman *web* nya.

5. Peran Dalam Pengembangan Teknologi *JavaServer Faces*

Teknologi *JavaServer Faces* memungkinan pembagian tugas untuk desain, pengembangan aplikasi dan pengelolaan selanjutnya. Pembuatan *JavaServer Faces* dapat terdiri dari tim dengan ketentuan:

- *Page Authors*
  Menggunakan bahasa markup, seperti *HTML* untuk merancang halaman-halaman sebuah aplikasi *web* dan biasanya memiliki pengalaman dalam desain grafis. *Page Authors* adalah pihak yang akan terlibat langsung dalam penggunaan *tag libraries JavaServer Faces*
- *Application Developers*
  Melakukan pemrogram untuk membuat objek, *event handlers*, *converter* dan *validator*.
- *Component Writers*
  Memiliki pengalaman dalam pemrograman *UI* dan mampu membuat komponen *UI* baru menggunakan bahasa pemrograman. *Component Writers* dapat membuat komponen baru menggunakan *Class-Class* komponen *UI* yang ada, atau mengembangkan komponen standar yang disediakan *JavaServer Faces*
- *Application Architect*
  Mendesain aplikasi *web*, memastikan tingkat skalabilitasnya, mengatur navigasi antar halaman, melakukan konfigurasi *beans* dan mendaftarkan objek dengan aplikasi.

- *Tools Vendors*
  Menyediakan perangkat pengembangan aplikasi yang memiliki dukungan terhadap *JavaServer Faces*, sehingga pengembangan sebuah *UI* berbasis *server-side* akan menjadi lebih mudah.

6. Teknologi Pembanding *Java Server Faces*

Didalam lingkungan teknologi pemrograman *web* dinamis, *JavaServer Faces* memiliki beberapa teknologi pembanding. Berbagai teknologi yang serupa dengan *JavaServer Faces* dan memiliki fitur yang sebanding dengan *JavaServer Faces* antara lain:

- ***ASP.NET***
  Teknologi yang dikembangkan oleh *Microsoft* ini memiliki fokus pada pengembangan komponen aplikasi *web*, serupa dengan *JavaServer Faces*. Dalam versi awal *ASP.NET*, kode untuk merender komponen disatukan dalam komponen *UI* itu sendiri. Hal ini berbeda dengan *JavaServer Faces* yang mampu memisahkan komponen *UI* dan proses *render*. *ASP.NET* terintegrasi dengan aplikasi pengembangan perangkat lunak buatan *Microsoft* yaitu *Microsoft Visual Studio*, memungkinkan pengembangan *UI* dengan teknik *Drag & Drop*.

  Kode dan logika program dihubungkan dengan komponen *UI*, dengan meletakkannya sebagai *event* yang akan dijalankan komponen, dan dapat dipisahkan dalam dokumen terpisah (teknologi *Code Behind*). Halaman *ASP.NET* hanya akan terdiri dari *HTML* yang membentuk desain layout, dan sebuah dokumen terpisah untuk logika halaman tersebut. Keduanya dianggap sebagai satu entitas. Hal ini berbeda dengan *JavaServer Faces* dimana logika program merupakan bagian yang benar-benar terpisah.

- **Web Objects/Wotonomy**
  Web Object adalah contoh *framework* aplikasi *web* paling pertama yang pernah ada, dan dikembangkan oleh *NeXT Software*, yang kemudian diambil oleh *Apple Computer. Web Object* memiliki arsitektur yang serupa dengan *JavaServer Faces*, baik untuk model *event* maupun komponennya. *Web* Object dapat dideploy tanpa *J2EE*, atau dapat disertakan dalam *container Servlet*.

Kendati demikian, tidak seperti *JavaServer Faces*, komponen *Web* Object tidak dideklarasikan dalam halaman *JSP*, tapi dalam direktori *.woc* yang didalamnya terdiri dari dokumen template *HTML*, *wml*, atau *xml*, dokumen yang memetakan baris kode dan aksi tertentu pada kode *Java*, dan sebuah *Class Java*. *Web Object* juga menyertakan arsitektur berlayer dengan salah satu *framework Object Relational Mapping* terdahulu yaitu *Enterprise Object Framework*.

Wotonomy adalah hasil re-implementasi dari *Web Object*. Sebagai bentuk lain dari *Web* Object, perbandingannya dengan teknologi *JavaServer Faces* pun serupa. *Wotonomy* bersifat *Open-Source*.

- ***Apache Tapestry***
  Tapestry adalah *framework open source* untuk pembuatan aplikasi dinamis, kompleks dan memiliki skalabilitas tinggi menggunakan *Java*. *Tapestry* dibangun berdasarkan *API Servlet Java* standar, dan dapat berjalan dalam *container servlet* atau *application server*. Tidak seperti *JSF*, *Tapestry* tidak menggunakan *JSP* untuk menampilkan data. *Tapestry* menggunakan sistem *template* untuk memungkinan manipulasi *HTML* secara lebih mudah dan sederhana.

  Tapestry membagi sebuah aplikasi *web* menjadi kumpulan halaman, yang masing-masing dibangun dari komponen tertentu. *Tapestry* secara spesifik didesain untuk mempermudah pembuatan komponen baru yang merupakan langkah rutin saat mengembangkan aplikasi. Didalamnya terdapat lebih dari 50 komponen, dari komponen sederhana yang digunakan untuk menampilkan keluaran tertentu, sampai bentuk *dat grid* kompleks.

- ***Apache Struts***
  Struts adalah *framework* populer dari *Apache Software Foundation*. *Struts* tidak menyediakan model komponen seperti gaya *JavaServer Faces*. Halaman akan dipetakan menggunakan sebuah *dispatch servlet* (*controller*), yang akan melempar *input* menjadi sebuah aksi. Sebuah aksi *Struts*, akan bertanggung jawab untuk memahami dan mengidentifikasi *parameter request*. *Struts* menggunakan *JSP* untuk tampilan, dan dapat mempadupadankan penggunaan *tag libraries JSP* untuk melaksanakannya.

7. Keunggulan Penggunaan *Java Server Faces*

Java Server Face menawarkan pemisahan kode program antara method dan tampilan. Aplikasi *web* yang dibangun menggunakan teknologi *JSP* sudah menerapkan metode pemisahan ini. Namun aplikasi berbasiskan *JSP* biasa tidak dapat memetakan *HTTP Request* untuk menjalankan *event handling* yang *component specific* atau menangai berbagai elemen *UI* sebagai object dalam *server*. Hal ini bisa ditangani oleh *Java Server Faces*. *JSF* memungkinkan pengembang untuk membangun aplikasi *web* yang mengimplementasikan metode pemisahan antara *behavior* dan *layer presentation* seperti yang ada pada arsitektur *UI Client-Side*.

Pemisahan logika program dari tampilan atau layer presentation, memungkinkan sebuah aplikasi dikembangkan oleh tim, dimana tiap anggota akan bertanggung jawab terhadap unit bagiannya tersendiri. *JSF* kemudian menyediakan model pemrograman sederhana untuk menyatukan unit-unit tersebut. Seorang pembuat halaman *web* dapat menggunakan berbagai *tag JavaServer Faces* melakukan koneksi ke objek dalam *server*, tanpa harus menulis baris-baris script tertentu.

Tujuan lain dari penggunaan teknologi *JavaServer Faces* adalah untuk meningkatkan penggunaan komponen *UI* yang umum dan konsep *web-tier* tanpa memberikan batasan mengenai teknologi *scripting* atau bahasa *markup* tertentu. Walaupun *JavaServer Faces* menyertakan *tag library JSP* untuk berbagai komponen *JSP*, teknologi *JavaServer Faces* diletakkan di bagian *Servlet API* paling atas. Penempatan *layer API* ini memungkinkan dijalankannya *use case* tertentu, seperti menggunakan teknologi *presentation layer* lain diluar *JSP*, pembuatan komponen *custom* dari *Class-Class* yang ada, dan menghasilkan keluaran untuk berbagai jenis media *client*. *JavaServer Faces* menyediakan arsitektur yang lengkap untuk menangani *state* komponen, untuk memproses data komponen, melakukan validasi *user input* dan menangani berbagai *event*.

8. Kesimpulan

Pengembangan aplikasi *web* menggunakan teknologi *Java*, kini dipermudah dengan hadirnya teknologi *JavaServer Faces*. *JavaServer Faces* menyediakan komponen siap pakai, dalam bentuk *tag libraries* khusus yang bilamana digunakan bersama komponen *JSP* dapat menghasilkan sebuah halaman *web* yang dinamis. *JavaServer Faces* merupakan solusi untuk pengembangan *UI* dinamis berbasiskan *Java*. Komponen *UI* yang ada dapat dikembangkan

lebih luas lagi, sehingga mampu menangani berbagai proses tertentu, baik yang sederhana maupun yang kompleks.

Secara global, *JavaServer Faces* bersanding dengan teknologi *web* lainnya yang juga menawarkan kemudahan pengembangan aplikasi *web* yang dinamis. Kendati demikian, fitur yang didukung oleh *JavaServer Faces* memiliki keunggulan tersendiri dibanding teknologi lain. Para programmer yang tergabung dalam komunitas *Java* akan merasakan bahwa pengembangan aplikasi *web* menjadi semakin cepat dan efektif, dengan dukungan komponen-komponen pada *JavaServer Faces*. *JavaServer Faces* adalah teknologi yang dipastikan akan berkembang terus, seiring pesatnya perkembangan teknologi aplikasi *web* dan kebutuhan *user* akan lingkungan pengembangan aplikasi yang baik.

Daftar Pustaka

Schalk, C., 2003, 'Developing *Web* Inter*Faces* with *JSF*', Fawcette Technical Publication, JavaPro, available at: http://www.fawcette.com/javapro/2004_01/magazine/features/cschalk.

J2EE Tutorial, 2005, '*Java Server Faces* Technology', The J2EE 1.4 Tutorial, available at: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSFIntro.htm.

Wikipedia, 2006, '*Java Server Faces*', Wikipedia The Free Encyclopedia, available at: http://en.wikipedia.org/wiki/JavaServer_Faces.

Schalk, C., 2005, 'Building Custom *Java Server Faces UI* Components', The Server Side.Com, available at:

http://www.theserverside.com/tt/articles/content/BuildingCustomJSF/article.HTML.

Sun Microsystem, 2006, '*Java Server Faces* Technology', Sun Microsystem, available at: https://java.sun.com/javaee/javaserverfaces/.

Hall, M, 2006, '*JSF* Tutorials An Introduction to *Java Server Faces*', CoreServlets, available at: http://www.coreservlets.com/JSF-Tutorial/.