ISSN: 1978 - 8282

Model Pengelolaan Situs Web secara Otomatis (WEBMATIC)

Iping Supriana[1]
email: iping@informatika.org

Abstraksi

Tulisan ini berisi kajian model pengelolaan situs secara otomatis. Tujuan utama dari kajian ini adalah membangun sistem yang dapat melakukan pengelolaan isi situs berdasarkan suatu model kaidah pengelolaan, sehingga isi dari situs mencerminkan aktualitas yang lebih baik. Model kaidah pengelolaan dibangun dalam dua kelompok: kaidah pengelolaan layout situs, dan kaidah pengelolaan isi layout (content) situs.

Kata kunci : layout situs, isi layout, kaidah pengelolaan. webmatic, web generator, layout generator.

PENDAHULUAN

Selama ini yang menjadi kendala dalam perawatan situs *web* adalah dibutuhkan biaya, waktu dan usaha yang besar untuk dapat membangun dan kemudian merawat situs tersebut. Di lain pihak, arus informasi semakin lama, semakin cepat dan berlimpah. Situasi seperti ini mengakibatkan macetnya jalur informasi dari sumber-sumber informasi menuju situs-situs web yang berfungsi sebagai kanal penyampaian informasi ke seluruh dunia.

Dengan semakin tingginya kebutuhan pengembangan situs, maka dibutuhkan pula *tools* yang mampu membantu pengguna hingga tahap automatisasi. Salah satu unsur utama dalam pengembangan situs web yang dapat di-automatisasi adalah penyusunan *layout* lembar yang mencakup penyusunan area dan penyusunan *content* lembar tersebut.

Secara manual, penyusunan area dilakukan dengan berbagai pertimbangan dan perhitungan pengguna. Begitu pula halnya jika dilakukan perubahan area. Perubahan area diantaranya adalah penambahan ruang artikel dalam *layout* lembar atau penghilangan ruang artikel dalam *layout* lembar. Jika pengguna menggunakan alat bantu pembuatan situs seperti Macromedia atau Frontpage, maka pengubahan area *layout* secara manual berarti bahwa pengguna harus mengubah tampilan situs melalui alat bantu tersebut dengan mempertimbangkan ruang artikel lainnya yang terpengaruh

1. **Dosen Institut Teknologi Bandung**
Jl. Ganesha 10 Bandung 40/32 Telp. 022-2500935, Fax. 022-2500935

oleh perubahan tersebut. Hal ini berarti bahwa kerja yang dilakukan pengguna untuk menangani pengubahan area *layout* sama dengan kerja yang dilakukannya untuk membuat *layout* situs pertama kali. Hal ini juga dapat berlaku jika ada perubahan *content* situs. Perubahan *content* situs berupa isi data dapat ditangani dengan penggunaan berbagai bahasa pemrograman seperti PHP, ASP, Perl, dan lain lain. Tetapi perubahan *content* yang mengakibatkan ruang kosong atau membutuhkan ruang yang lebih banyak dari sebelumnya harus ditangani dengan cara yang sama seperti perubahan area *layout*.

Oleh karena itu dalam kajian ini, *webmatic* diimplementasikan untuk menjawab tantangan pembuatan dan perawatan situs *web* dengan biaya yang murah dan mudah. *Webmatic* menyediakan kakas yang memungkinkan orang dengan keahlian minim untuk dapat ikut berpartisipasi untuk meremajakan konten dari situs *web*. Kemudian orang-orang dengan keahlian khusus seperti desainer web tetap dapat bekerja tanpa perlu terlalu peduli dengan konten-konten yang ada. Masing-masing orang akan bekerja sesuai dengan keahliannya masing-masing dalam membangun dan merawat situs *webmatic* ini.

MODEL STRATEGI

Definisi permasalahan utama pada kajian ini adalah bagaimana menyusun (atau menerapkan aturan tertentu yang terdapat pada *layout script*) dan membagi (berdasarkan pada berbagai parameter) sekumpulan blok data (teks, gambar, animasi, suara) terpisah yang berasal dari berbagai sumber menjadi suatu rangkaian data yang memenuhi batasan tertentu – contohnya adalah batasan tempat dan waktu – secara dinamis dan efisien.

Strategi penyusunan *layout* lembar (layar) situs terdiri atas pemilihan artikel yang *up-to-date*, pemecahan artikel untuk ditampilkan pada berbagai lembar yang berbeda, dan penataan letak artikel pada lembar tertentu. Setiap strategi tersebut akan dijelaskan pada point-point berikut ini.

1. Strategi pengambilan data dari berbagai sumber yang bervariasi

   Strategi ini berkaitan dengan komponen antar muka antara pengguna dengan sistem. Komponen antar muka terbagi atas dua jenis yaitu editor manual dan supplier. Editor manual adalah komponen yang menerima masukan dari pengguna secara langsung dimana pengguna mengetikkan masukan bagi sistem. Supplier adalah komponen yang menghubungkan berbagai sumber masukan yang tidak berupa masukan langsung dari pengguna, misalnya file teks, data dari DBMS, dan lain lain. Setiap jenis sumber masukan membutuhkan sejenis supplier tertentu.

Semakin banyak jenis sumber masukan maka semakin banyak supplier yang dikembangkan dan berarti sistem menjadi semakin fleksibel.

2. Strategi Pemilihan Artikel yang Up-To-Date

Pemilihan artikel yang *up-to-date* dapat diselesaikan dengan menerapkan sebuah fungsi tertentu pada data yang akan ditampilkan. Fungsi yang diterapkan di sini adalah fungsi berdasar waktu atau *scheduler*.

3. Strategi Pembagian Artikel

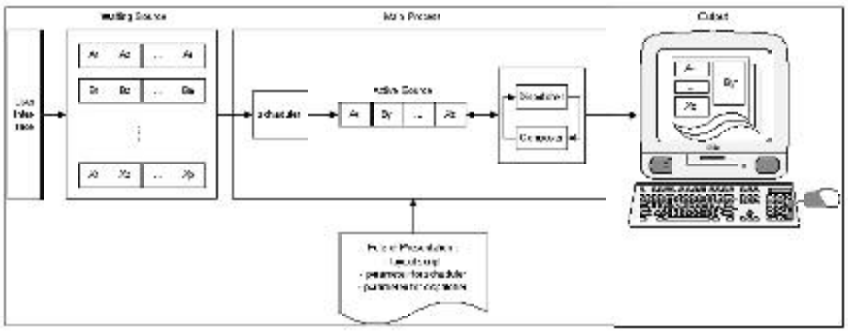
Pemecahan artikel dilakukan dengan berdasar atas berbagai parameter seperti ukuran ruang untuk artikel, keterhubungan antar bagian dari artikel, dan sebagainya. Setiap pemecahan artikel dapat dilakukan dengan parameter yang berbeda.

4. Strategi Penataan Letak Artikel

Penataan letak artikel dapat dilakukan dengan melihat pada acuan sketsa naskah yang diinginkan pengguna. Acuan sketsa naskah ini selanjutnya disebut *layout script*. Penulisan *layout script* memiliki format berupa list blok data yang dapat bersarang. List blok data ini menggambarkan komposisi artikel pada lembar. Contoh *layout script* adalah (A, (B,C)), yang berarti bahwa artikel A akan menempati setengah lembar dan setengah lembar lainnya akan dibagi dua untuk artikel B dan C.

MODEL DESKRIPSI

Arsitektur model sistem perangkat lunak secara keseluruhan dapat dilihat pada gambar berikut ini.



Gambar 1 Model Arsitektur Webmatic.

Gambar 1 menunjukkan bahwa terdapat berbagai komponen perangkat lunak yang perlu dibangun untuk memenuhi fungsi penyusunan *layout* situs, yaitu:

1. *user interface* yaitu antarmuka antara input pengguna dengan penyimpan blok data. Antarmuka ini dapat dibagi menjadi dua jenis komponen yaitu manual editor dan supplier. Manual editor adalah antarmuka yang menerima masukan dari pengguna secara langsung. Supplier adalah antarmuka yang menghubungkan antara berbagai *external source* dengan kumpulan *waiting source*. Supplier merupakan salah satu bagian penting dalam perangkat lunak karena supplier harus mampu menangani berbagai *external source*s yang bervariasi.
2. *scheduler* yaitu komponen yang berfungsi memilih blok data pada *waiting source* menjadi *active source*. Untuk melakukan pemilihan blok data ini, *scheduler* menggunakan parameter seperti waktu.
3. *dispatcher* yaitu komponen yang berfungsi memilah *active source* ke dalam berbagai lembar yang berkaitan, dengan berdasar pada parameter tertentu seperti marker, spasi, dan lain lain.
4. *composer* yaitu komponen yang berfungsi menyusun pilahan *active source* (hasil *dispatcher*) ke dalam berbagai lembar dengan mengacu pada *layout script*

Gambar 1 juga menunjukkan berbagai struktur data yang terlibat yaitu:

1. *Waiting Source*, terdiri atas berbagai kumpulan blok data yang terpisah. Dapat digambarkan sebagai berbagai himpunan A, B, ..., X, dimana setiap himpunan memiliki anggota yang berbeda tapi mungkin berkaitan dengan jumlah yang berbeda. $A=\{i \text{ å } 0..n \mid A_i\}$, $B = \{j \text{ å } 0..m \mid B_j\}$,..., $X = \{k \text{ å } 0..o \mid x_k\}$
2. *Active Source*, terdiri atas sekumpulan blok data dengan jumlah yang bervariasi dalam satuan waktu. Jumlah blok data yang terdapat pada *active source* tergantung pada *scheduler*. $AS = f(A,B,...X) = \{a_i,b_j,...,x_k\}$. Active *source* juga mengandung informasi hasil *dispatcher* yaitu bagian-bagian setiap blok data yang ditempatkan pada lembar-lembar situs. $AS' = \{disp(a_i), disp(b_j),..., disp(x_k)\} = \{\{a_{i1},a_{i2},...,a_{iy}\},\{b_{j1},b_{j2},...,b_{jy}\},...,\{x_{k1},x_{k2},...,x_{ky}\}\}$ dimana y menunjukkan jumlah lembar.
3. *Rule of Presentation*, yang terbagi atas:
   a. *Layout Script*, mencakup sekumpulan aturan penyusunan pada lembar. Setiap lembar dapat memiliki format penyusunan yang berbeda. Format penyusunan ini sama dengan yang dijelaskan pada bagian strategi penataan letak artikel.
   b. Parameter untuk *Scheduler*, yaitu variable penentu jenis *scheduler*, dapat berupa waktu atau yang lainnya.

c. Parameter untuk *Dispatcher*, terdiri atas sekumpulan variable penentu jenis *dispatcher* yang dijalankan. Variabel ini menentukan rumus pembagian *source* yang dibutuhkan. Contoh parameter adalah marker berupa simbol penghubung seperti titik (.), koma (,), titik koma (;) atau spasi (" "). Dengan parameter simbol penghubung, maka pemisahan artikel didasarkan pada simbol penghubung terdekat yang dapat digunakan. Contoh parameter lainnya adalah pemisahan kata terakhir dengan menggunakan "-" atau spasi (" ").

MODEL IMPLEMENTASI

Model Animasi Dan *Web.*

Pergantian isi layout berdasarkan waktu dapat dipandang sebagai model animasi. Walaupun konsep animasi dan *web* merupakan dua konsep yang berbeda. Namun dalam kajian ini akan dibuat suatu konsepsi baru yang menjembatani antara konsep animasi dan *web* tersebut.

Pertama-tama kita harus memahami sifat dasar dari animasi. Sifat dari animasi adalah:

a. Merupakan koleksi gambar siap saji yang ditampilkan secara beruntun dan bergantian dengan waktu jeda yang konstan.
b. Merupakan konsep aliran gambar (*stream* atau *flow*) dengan debit yang konstan.
c. Merupakan konsep *PUSH* dalam arti gambar ditampilkan tanpa menunggu diminta pengguna.
d. Setiap gambar memiliki *time limited passport* yang relatif terhadap waktu animasi mulai dimainkan.

Kemudian kita juga harus memahami karakteristik dan sifat dari *web*. Sifat dan karakteristik dari *web* adalah:

a. *State-less*, yaitu setiap pasangan *request/respond* adalah saling lepas dan saling bebas satu sama lain.
b. *Connection-less*, yaitu tidak dikenal konsep *stream/flow* dengan debit yang konstan.
c. Konsep *PULL*, yaitu informasi disajikan hanya ketika diminta oleh pengguna.

Karena animasi dan *web* masing-masing memiliki sifatnya yang berbeda, maka dibutuhkan strategi untuk mengadopsi konsep animasi ke dalam lingkungan *web*. Namun adopsi konsep animasi tersebut tidaklah mungkin menghasilkan animasi yang sejati, melainkan hanya simulasi dari animasi. Oleh karena itu dibutuhkan suatu konsepsi baru, yang dalam kajian ini disebut sebagai *webmatic*.

Konsep Implementasi *Webmatic*

Webmatic mencoba menyatukan keindahan animasi dengan kepopuleran *web* dengan cara mengadopsi konsep animasi sehingga membuat halaman web menjadi suatu "jendela hidup". *Webmatic* mencoba menjembatani perbedaan konsep antara animasi dan *web* kedalam suatu konsep baru yang memperlakukan halaman-halaman web sebagai sesuatu yang bergantian untuk ditampilkan seperti halnya dalam animasi.
Webmatic dikendalikan oleh *presentation rule* yang penulisannya mengikuti suatu himpuran aturan gramatika tertentu. Dengan demikian, *presentation rule* ini membentuk suatu bahasa tersendiri. Bahasa ini harus lebih ekspresif dibanding dengan bahasa-bahasa script ataupun sarana untuk produksi HTML seperti PHP, Perl, dan sebagainya. [MATIC03]. *Presentation rule* yang akan dibuat dalam kajian ini adalah dalam bentuk *file template* berformat XML [XML01] [XHTML05] [XSLT01] [CSS05] .

Terdapat dua poin yang harus disimulasikan oleh *webmatic* untuk dapat mengadopsi konsep animasi ke dalam lingkungan *web*:

a. Koleksi gambar siap saji

Webmatic tidak memiliki koleksi halaman siap saji. Namun webmatic memiliki komponen-komponen yang siap di-*render* untuk menghasilkan halaman siap saji. Proses *rendering* halaman dilakukan dengan cepat ketika ada permintaan dari pengguna sehingga pengguna akan merasa bahwa informasi disajikan dengan seketika seakan-akan pengelola situs memiliki koleksi halaman siap saji. Strategi ini dipilih karena untuk menjamin kesegaran data, bahwa halaman yang disajikan baru diolah ketika ada permintaan. Seperti halnya juga di dunia nyata, untuk menjamin kesegaran makanan, maka masakan hanya dimasak ketika ada permintaan dari pembeli.

Untuk mempercepat penyajian halaman, dapat diterapkan strategi *caching* sehingga proses *rendering* halaman yang repetitif tidak terjadi. Strategi *caching* ini sangat efektif apabila diterapkan pada halaman-halaman yang sering dikunjungi.

b. Konsep aliran gambar dengan debit konstan

Dalam *web* tidak ada yang namanya konsep aliran karena konsep tersebut hanya ada jika kita memiliki konsep *PUSH*, yaitu kita dapat memaksakan informasi ke sisi pengguna. Namun karena web hanya memiliki konsep *PULL*, yaitu informasi tidak dapat dipaksakan ke pengguna, informasi hanya disajikan ketika ada permintaan dari pengguna, maka konsep aliran dengan debit konstan didekati dengan memberikan *absolute time limited passport* pada setiap konten yang akan ditampilkan dalam halaman *web*.

Absolute time limited passport merupakan paspor yang dilekatkan kepada setiap konten *web* sehingga *webmatic* hanya akan menyajikan konten yang memiliki paspor yang valid. Kevalidan suatu paspor dibatasi dalam rentang waktu mutlak tertentu. Dengan menggunakan paspor ini maka, suatu konten hanya akan tampil ketika ada permintaan pengguna dan apabila memang sudah waktunya tampil (paspornya sah). Akibatnya adalah setiap konten memiliki waktunya masing-masing untuk tampil secara bergantian sehingga dengan melakukan request halaman yang sama berulang-ulang, pengguna akan merasakan pengalaman menyaksikan suatu animasi gerak lambat dari halaman-halaman *web* yang berubah-ubah sesuai fungsi waktu. Pendekatan dengan cara seperti mengakibatkan debit yang terjadi adalah variabel, tergantung dari waktu-waktu pengguna meminta informasi, hal ini diakibatkan karena tidak adanya konsep *PUSH* melainkan *PULL* dalam lingkungan *web*.

Diharapkan konsep ini sudah cukup memadai untuk membuat suatu ilusi animasi "jendela hidup" dalam lingkungan *web*.

Perbedaan antara konsep animasi dan *web* dihampiri dengan konsepsi baru yaitu *webmatic*. Tabel di bawah ini menampilkan perbedaan antara konsep sejati dan konsep hampirannya.

Tabel 1 Perbedaan konsep Animasi dan Webmatic

Animasi	Webmatic
1. Koleksi gambar siap saji.	1. Koleksi konten mentah dan template yang siap di-*render* untuk menghasilkan halaman siap saji.
2. *Stream/flow* sejati dengan debit konstan.	2. *Stream/flow* yang disimulasikan dengan debit variabel.
3. Konsep *PUSH*.	3. Konsep *PULL*.
4. *State-full*	4. *State-less*
5. Setiap gambar memiliki *relative time limited passport*.	5. Setiap konten memiliki *absolute time limited passport*.

HASIL

Hasil kajian adalah sebuah sistem perangkat lunak yang mampu menampilkan web secara dinamis dan otomatis serta mampu mengelola daftar data dokumen dalam berbagai bentuk (teks, gambar, animasi).

Kemampuan yang telah dicapai sampai saat ini adalah sebagai berikut:

1. mampu menerima masukan berupa rangkaian data dari berbagai sumber baik berupa teks, gambar, animasi ataupun suara.
2. mampu memilih data yang akan ditampilkan berdasar fungsi waktu .
3. mampu memilah data sesuai parameter pemilahan data yang ditentukan pengguna
4. mampu menyusun berbagai pilahan data tersebut menjadi situs yang terdiri atas lembar-lembar yang saling berhubungan dengan *layout* per lembar yang mungkin berbeda.

PENUTUP

Telah disampaikan suatu model pengelolaan web secara otomatis (webmatic). *Webmatic* bekerja dengan cara menggabungkan *template* yang telah diberikan suatu penanda *tag* khusus dengan konten-konten yang tersedia untuk menghasilkan suatu halaman web siap saji melalui proses yang disebut dengan *rendering*. Proses ini bekerja dengan cepat untuk melayani pembuatan halaman *web* sehingga seolah-olah tampak bahwa halaman *web* tersebut sudah siap saji sebelumnya. Untuk lebih mempercepat lagi waktu respons dari *webmatic*, dapat digunakan strategi *cache* sehingga kinerja dapat meningkat hingga 15 kali lipat.

PUSTAKA

1. Key, Michael (2001). ***XSLT*** , 2nd Edition, Wrox Press Ltd.

2. Supriana Iping; Hariyanto, Bambang; Purwarianti Ayu; Leyla Khodra, Masayu; Maulidevi, Nur Ulfa (2003). Kajian Konsepsi Otomatisasi Pembentukan Komponen Presentasi Pada Aplikasi Berbasis We. Bandung: Departemen Teknik Informatika ITB.

3. Williamson, Heather (2001). *XML The Complete Reference*, Osborne/McGrawHill http://www.w3.org/TR/xhtml1http://www.w3.org/Style/CSS