OPTIMASI APLIKASI WEB BERBASIS FRAMEWORK SYMFONY DENGAN TWEAK VIEW DAN TWEAK CACHE

Hendriana Mayang Sari [1)], Kusuma Ayu Laksitowening [2)] Hetti Hidayati [3)]
Fakultas Informatika Institut Teknologi Telkom
Jalan Telekomunikasi No.1 Terusan Buah Batu Bandung 40257
(022) 7564108
E-mail :[1]myg243@gmail.com, [2]kal@ittelkom.ac.id, [3]htt@ittelkom.ac.id

ABSTRAKS
Seiring dengan pesatnya perkembangan aplikasi web saat ini, maka performansi aplikasi adalah isu yang sangat penting untuk diperhatikan oleh developer. Agar dapat terus memenuhi user expectation terhadap performa aplikasi tanpa menambah resource perangkat keras, maka solusi yang mungkin dilakukan adalah melakukan optimasi dari sisi scripting aplikasi. Symfony adalah sebuah framework berbasis PHP 5 yang memfasilitasi developer dengan langkah-langkah optimasi yang dinamakan Tweak Method. Tweak Method dapat dilakukan dengan 3 cara : Tweak Model, Tweak View, dan Tweak cache. Dalam studi kasus ini penulis mengimplementasikan Tweak View dan Tweak cache. Pada Tweak View, langkah optimasi yang dilakukan fokus pada bagaimana mengurangi byte transferred. Sedangkan pada Tweak cache, proses optimasi yang dilakukan adalah mengimplementasikan teknologi PHP Cache yang berfungsi untuk cache opcode dari response. Hasilnya, pengimplementasian Tweak Method pada aplikasi kasus dapat mempercepat Response Time hingga 99.26% dan byte transferred hingga 88.05% dari sebelumnya.

Kata kunci : performa, optimasi, Tweak Method, Tweak View, Tweak Cache

1. PENDAHULUAN

1.1 Latar Belakang

Pesatnya penggunaan aplikasi web saat ini membuat persaingan semakin ketat. Agar dapat menarik minat semakin banyak *user*, maka kinerja dan performansi adalah isu yang sangat penting untuk diperhatikan oleh *developer* aplikasi web. Saat ini, sebagian besar developer hanya *concern* pada hal-hal teknis ketika aplikasi webnya di-*launch*, misalnya: apakah semua fungsionalitas sudah berjalan dengan baik, apakah sudah dipastikan tidak ada error dan bug. Sebagian besar developer lupa, bahwa kecepatan akses adalah salah satu hal esensial yang paling user inginkan ketika mengakses sebuah halaman web.

Berdasarkan referensi [5], *Response Time* optimal untuk me-*load* sebuah halaman web adalah kurang dari 1 detik. Nilai *Response Time* maksimal yang dapat ditolerir oleh user adalah 8 detik per halaman (*8 seconds rule*), bila lebih dari 8 detik maka kemungkinan besar user akan beralih ke website lain.

Untuk tetap dapat memenuhi user expectation akan performa sebuah aplikasi web, maka penting dilakukan langkah-langkah yang dapat dilakukan untuk meningkatkan performa aplikasi web tanpa harus menambah *resource hardware*. Pada konteks framework symfony, langkah-langkah optimasi tersebut disebut dengan Tweak. Dan langkah tweak yang diimplementasikan pada penelitian ini adalah *Tweak View* dan *Tweak Cache*.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka permasalahan yang dibahas dalam penelitian ini adalah :

a. meningkatkan performa sebuah aplikasi web tanpa harus menambah *resource* perangkat keras.
b. bagaimana proses optimasi tersebut dapat meningkatkan performa aplikasi.
c. bagaimana pengaruh optimasi pada performa aplikasi jika dibandingkan dengan performa awalnya.

1.3 Tujuan Penelitian

Tujuan dari dilakukannya penelitian ini adalah melakukan *Performance Testing* sebelum dan sesudah diimplementasikan *Tweak Method* kemudian melakukan analisa terhadap hasil. Adapun parameter yang menjadi acuan adalah:

a. *Response Time total* (waktu yang dibutuhkan untuk menampilkan *response*).
b. *Response Time* sebelum dan sesudah implementasi *Tweak cache*.
c. *Byte transferred* (jumlah byte data yang di-load untuk menampilkan *response*).

2. FRAMEWORK SYMFONY DAN MVC

Framework adalah sebuah *tool* yang terdiri dari sekumpulan fungsi, kelas dan aturan yang berfungsi untuk membangun aplikasi dengan cepat, mudah dan *maintanable*. Membangun aplikasi dengan menggunakan *framework* relatif lebih cepat dibandingkan dengan cara konvensional. Karena dengan menggunakan *framework*, *user* hanya perlu fokus pada inti permasalahan saja, sedangkan hal-hal

penunjang lainnya seperti koneksi *database*, *caching*, validasi *form*, GUI, keamanan, umumnya sudah disediakan oleh *framework* [6].

Framework Symfony disebut juga dengan MVC Framework. MVC adalah *architectural layer* yang mengklaster suatu aplikasi web menjadi 3 layer kerja , yaitu layer Model, View, dan Controller (MVC). Layer Model berperan untuk merepresentasikan informasi/ data dari suatu aplikasi yang disimpan dalam *database* atau XML. Layer View berperan untuk merepresentasikan elemen dari *user* interface, seperti: teks, Gambar, form input, dll. Dan layer Controller berperan untuk menerima inputan data dan men-*generate content* ke dalam format html [9].

3. TWEAK METHOD

Tweak adalah sekumpulan langkah-langkah atau cara yang digunakan untuk meningkatkan performa aplikasi tanpa harus menambah *resource hardware*[6]. Dengan kata lain proses *upgrade* performa ini dilakukan dari sisi *scripting* aplikasi itu sendiri.

3.1 Tweak View

Tweak pada sisi *view* adalah proses optimasi yang dilakukan pada layer *View* atau hal-hal yang berhubungan dengan front-end aplikasi. Berikut penjelasan beberapa teknik *Tweak*/ optimasi yang berpengaruh pada layer *View* :

a. Meminimalkan Jumlah *Request*
Implementasi teknik ini adalah dengan menggabungkan file-file Javascript dan CSS external ke dalam 1 file saja untuk mereduksi jumlah HTTP Request..
HTTP Request adalah effort yang dibutuhkan untuk mengirim request sampai menerima response antara browser dan server. Dalam satu waktu, browser hanya mampu mendeliver max 2 request pada host yang sama. Semakin banyak file Javascript dan CSS yang dikirim, semakin banyak request yang harus di deliver dan semakin lama waktu yang dibutuhkan. Ide dari teknik ini adalah mengurangi jumlah request file untuk mempercepat waktu akses. Idealnya dalam 1 aplikasi web, hanya ada 1 *File* JS eksternal dan 1 *File* CSS eksternal

b. Meletakkan CSS pada Bagian Atas *Script*
CSS akan di load sesuai dengan posisi dimana *script* tersebut diletakkan. *Front-End Developer* yang meletakkan CSS pada bagian bawah *script*, membuat CSS di load belakangan setelah logika pemrograman dan akses *database* di lakukan. Akibatnya, user akan melihat halaman blank atau putih sebelum *response* di hasilkan. Selain memperlambat proses *rendering*, implementasi CSS pada bagian atas *script* juga tidak baik untuk *User Interface*.

c. Meletakkan *Javascript* pada Bagian Bawah *Script*
Javascript yang diletakkan pada bagian atas *script* akan membuat reponse time menjadi lebih lama dari yang seharusnya. Hal ini disebabkan, *Javascript* tersebut tidak langsung di *parsing* melainkan di *buffering* dan menunggu seluruh HTML Page di load seluruhnya. Proses *Buffering* ini akan menambah deretan proses eksekusi *script* dan menambah waktu akses. Dengan meletakkan *Javascript* pada bagian bawah, maka *Javascript* aan dieksekusi sesuai dengan urutannya tanpa melalui proses *buffering*.

d. Kompresi *Response*
Setiap hari, lebih dari 99 tahun waktu manusia terbuang karena *content* aplikasi web yang tidak di kompresi [4]. Sebagian besar *developer* tidak terlalu *concern* mengenai masalah ini, namun bagi *user* performa aplikasi merupakan salah satu faktor penting agar user ingin terus *keep-in-touch* dengan suatu aplikasi web. Agar suatu aplikasi web dapat memenuhi harapan *user* akan kecepatan akses, maka penting dilakukan teknik Kompresi. Kompresi berguna untuk memperkecil ukuran data yang dideliver agar lebih ringan dan lebih cepat waktu transfernya. Teknik kompresi *content* yang akan diimplementasikan adalah *Gzip compression*. Analoginya adalah seperti ketika mengompress *File* dengan menggunakan format rar atau zip. *Content* atau *response* yang akan dikirim, dikompresi terlebih dahulu di server kemudian di kirim, setelah sampai pada *klien* (browser) kemudian di dekompresi ke dalam bentuk awal. Tingkat ratio rata-rata teknik kompresi ini adalah 2:1 sampai 5:1.

e. Meminimalkan ukuran *File Javascript* dan CSS
Teknik ini mengacu pada menghilangkan kode byte yang tidak perlu seperti spasi tambahan, *line break*, dan *indentation* (baris *script* yang menjorok untuk mempermudah membaca baris *script* satu dengan lainnya). Tujuannya, untuk memperkecil ukuran file. Semakin kecil ukuran file, semakin kecil effort yang dibutuhkan untuk di deliver, dan semakin cepat waku aksesnya.

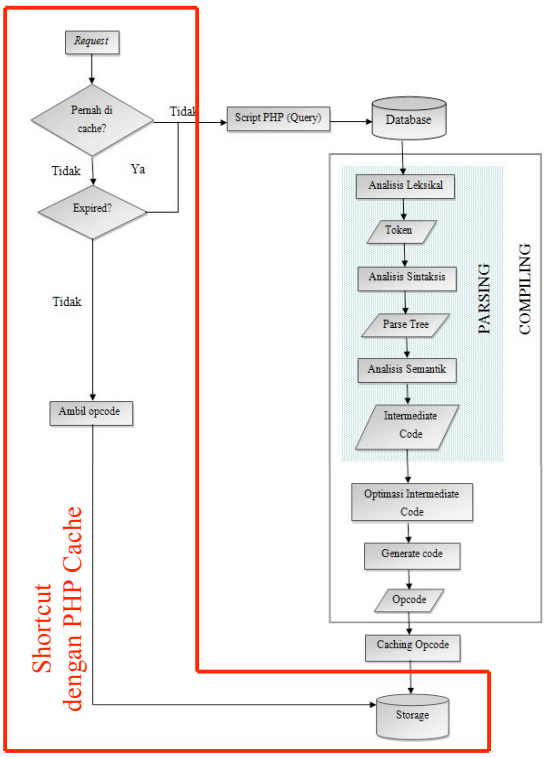
f. Menginisialisasi *Helper* yang Digunakan
Framework Symfony memiliki banyak *helper* yang secara otomatis ter-*load* pada setiap *request*. Apabila dalam melakukan *request*, *developer* tidak membutuhkan semua jenis *helper* maka *developer* bisa melakukan insialisasi *helper* apa saja yang dibutuhkan pada aplikasinya. Me-*load* semua *helper* secara *default* sebenarnya cukup menyita banyak *resource* karena ukurannya yang cukup besar [6].

 ISSN: 1907-5022

g. Implementasi Teknik *CSS Sprite*
CSS Sprite adalah sebuah teknik penggabungan beberapa gambar/ ikon pada sebuah halaman web menjadi sebuah gambar yang lebih besar. Tujuan dari teknik CSS Sprite adalah untuk mengurangi *HTTP Request* pada koneksi antara *klien* dan server [3]. Efeknya, *Response Time* pun menjadi lebih cepat. Misalkan dalam sebuah halaman web terdapat 20 gambar, maka dibutuhkan 20 *HTTP Request*. Ketika 20 gambar itu digabungkan menjadi 1 (implementasi CSS Sprite) maka *HTTP Request* yang dibutuhkan hanya 1. Mengurangi jumlah *HTTP Request* secara signifikan sangat efektif dalam mengurangi *Response Time*.

4. TWEAK CACHE

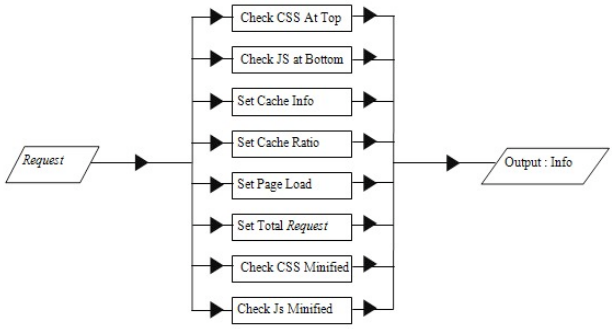
Implementasi *Tweak cache* yang akan digunakan adalah teknik *Opcode Cache*. Mekanisme *Opcode Cache* adalah melakukan *caching* opcode hasil *response query* yang di-*request*. Sehingga ketika ada *request* yang sama di lain waktu, web server tidak perlu melakukan *fetch* ke *database* dan melalui proses *parsing* dan *compiling* yang panjang. Request yang sudah pernah di *cache* dan tidak expired akan langsung di-*fetch* dari *cache storage* [4].



Gambar 1. Mekanisme Opcode Cache

5. PERANCANGAN PERFORMANCE TESTING TOOL

Di bawah ini adalah ilustrasi cara kerja Performance Testing Tool dalam sebuah diagram blok :



Gambar 5. Diagram Blok Performance Testing Tool

Pengujian performansi akan dilakukan dengan membandingkan aplikasi yang dibangun dengan Tweak Cache dan Tweak View (kondisi akhir) dengan aplikasi yang tidak menerapkan Tweak Cache dan Tweak View(kondisi awal). Pengujian dilakukan pada tiga modul dari masing-masing aplikasi.

6. SKENARIO UJI PERFORMA

Adapun prosedur pelaksanaan uji performa (awal dan akhir) aplikasi adalah sebagai berikut :

a. Uji performa akan disimulasikan pada 2 *bandwidth environment*: dial-up 56 Kbps (sampel *slow connection*), dan DSL/Cable 256 kbps (sampel *average connection*). Catatan : Simulasi *bandwidth environment* menggunakan addons Throttle.
b. Uji performa disimulasikan dalam keadaan *browser* bebas dari *cache* sehingga tidak mempengaruhi hasil uji performa.
c. Masing-masing menu sampel akan diuji performanya sebanyak 10 kali pada 2 jenis *bandwidth environment* untuk mengukur parameter *Response Time* Total dan *Response Time Cache*. Sedangkan untuk parameter *byte transferred* hanya akan diuji 1 kali untuk masing-masing menu sampel karena hasilnya statis.

7. HASIL UJI PERFORMA

Reduksi Response Time Total pada Dial Up 56Kbps

1. Modul 1 : ((74.73 - 0.55) / 74.73) detik x 100% = 99.26%
2. Modul 2 : ((12.53 – 0.22) / 12.53) detik x 100% = 98.24 %
3. Modul 3 : ((13.09 – 0.21) / 13.09) detik x 100% = 98.4 %

Reduksi Response Time Total pada Cable 256 Kbps

1. Modul 1 : ((4.3 – 0.614) / 4.3) detik x 100% = 85.72%
2. Modul 2 : ((0.97 – 0.22) / 0.97) detik x 100% = 77.32%
3. Modul 3 : ((0.37 – 0.23) / 0.37) detik x 100% = 37.84 %

Reduksi Response Time Cache pada DialUp 56Kbps

1. Modul 1 : ((0.4485 – 0.393) / 0.4485) x100% = 12.37%
2. Modul 2 : ((0.2163 – 0.2095) / 0.2163) x100% = 3.14%
3. Modul 3 : ((0.2542 – 0.2459) / 0.2542) x100% = 3.27%

Reduksi Response Time Cache pada Cable 256Kbps

1. Modul 1 : ((0.3848 – 0.3522) / 0.3848) x100% = 8.47%
2. Modul 2 : ((0.2233 – 0.215) / 0.2233) x100% = 3.71%
3. Modul 3 : ((0.2346 – 0.2245) / 0.2346) x100% = 4.31%

Reduksi Byte Transferred

1. Modul 1 : ((379.66 – 45.37) / 379.66) KB x 100% = 88.05%
2. Modul 2 : ((105.37 – 14.47) / 105.37) x 100% = 86.27%
3. Modul 3 : ((87.67 – 14.22) / 87.67) KB x 100% = 83.78%

8. ANALISA HASIL UJI

Reduksi *Response Time* sangat signifikan terjadi pada *bandwidth environment* 56kbps (*slow connection*). Hal ini ternyata sesuai dengan referensi [2] yang menyatakan bahwa proses optimasi akan bekerja optimal pada koneksi yang lambat. Pada kondisi tersebut pengurangan *Response Time* pada Modul 1 mencapai 99.26%.

Pada koneksi yang cepat, hampir semua halaman web dapat di-*load* dengan cepat (baik yang *byte*-nya besar maupun kecil). Pada koneksi yang cepat, *Response Time* aplikasi akan cenderung stabil, sehingga implementasi *Tweak Method* tidak berkontribusi terlalu banyak terhadap pengurangan *Response Time*. Sedangkan pada koneksi lambat, seseorang dapat lebih mudah menentukan performa sebuah aplikasi dengan lebih objektif.

Reduksi *byte transferred* terbesar juga terjadi pada Modul 1, dari 379.66 KB menjadi 45.37 KB, atau sebesar 88.05%.

Proses *Tweak View* akan berpengaruh secara signifikan pada performa aplikasi dengan file JS, CSS, dan *image* yang dominan. Alasannya, semakin besar prosentase ketiga file tersebut dalam sebuah aplikasi, semakin banyak metode *Tweak View* yang bisa diimplementasikan, dan semakin besar jumlah *byte* yang dapat direduksi. Modul Summary memiliki kriteria tersebut.

Proses caching cukup berkontribusi dalam peningkatan performa sebuah aplikasi web. Reduksi *Response Time* yang terjadi pada studi kasus antara 3.14% - 12.37%. Semakin besar reduksi *Response Time* yang terjadi, semakin banyak data yang di *cache*. Reduksi *Response Time* paling banyak pada Modul 1.

PHP Caching menyimpan opcode hasil *response query* dari *database*, dan modul summary memiliki jumlah akses *database* yang paling tinggi (jumlah query = 7) dari 2 modul sampel lainnya (jumlah query 2 dan 1). Sebelum fungsi *cache* di aktifkan, setiap *query* yang di *request* mengambil data dari *database* sehingga memperpanjang *Response Time*. Pada saat fungsi *cache* diaktifkan, response diambil langsung dari *cache storage* sehingga mempersingkat *Response Time*.

9. KESIMPULAN

Implementasi *Tweak View* dan *Tweak cache* dapat meningkatkan performa aplikasi berbasis framework Symfony.

Implementasi *Tweak Method* bekerja secara optimal pada *slow connection*.

Implementasi *Tweak View* bekerja secara optimal pada aplikasi atau modul-modul yang memiliki jumlah *file* JS, CSS, dan *image* yang dominan.

Implementasi *Tweak cache* (PHP *Cache*) bekerja secara optimal pada aplikasi atau modul-modul yang memiliki script PHP dengan jumlah *fetch* ke *database* dan tingkat kalkulasi yang tinggi

PUSTAKA

[1] Anonymous, *Page Speed Performance Best Practice*. 2010. Diakses pada : http://code.google.com/intl/id/speed/page-speed/docs/payload.html#GzipCompression. (15 Januari 2010)

[2] Anonymous, *Performance Tuning PHP*.2010. Diakses pada : http://www.scribd.com/doc/10633/Performance-Tuning-PHP (3 Februari 2010)

Shea, Dave.2004. *CSS Sprites : Images*

[3] *Slicing's Kiss of Death*. Diakses pada : http://www.alistapart.com/articles/sprites/ (24 Desember 2009)

[4] Slaughter, Harry. 2006. *Opcode Cache For Dummies*. Diakses pada : http://devbee.com/opcode_*cache*_for_dummies (19 Agustus 2009)

[5] Subraya, BM. 2006. *Integrated Approach to Web Performance Testing : A Practitioner's Guide*. India : IRM Press. Diakses pada : http://vista-server.com/upload*File*/6/2/18/15594993208.zip (6 Februari 2009)

[6] Anonymous, *The Definitive Guide to Symfony*. 2008. Diakses pada : http://symfony-project.com (15 November 2008).

[7] Anonymous, *Throttle*. 2010. Diakses pada : https://addons.mozilla.org/en-US/firefox/addon/5917 (26 Januari 2010)

[8] Anonymous, *YSlow Performance Rule*. 2009. Diakses pada : http://developer.yahoo.com/yslow/help/index.html (20 Desember 2009)