

# DESAIN DAN IMPLEMENTASI APLIKASI UNTUK VISUALISASI INFORMASI PADA *FILE OFFLINE LOG WEB SERVER*

Agus Kurniawan

Fakultas Ilmu Komputer, Universitas Indonesia, Depok, Indonesia  
agusk@cs.ui.ac.id

## Abstrak

Hampir sebagian besar *web server* mempunyai kemampuan untuk menghasilkan *file log* dalam format *file text* berbasis *delimiter*. *File log* ini dihasilkan setiap hari atau mingguan atau bulanan. Semakin banyak *file log* yang dihasilkan ini akan berdampak sulitnya dalam menganalisis *file log* tersebut sehingga diperlukan suatu *tool* berupa aplikasi yang dapat melakukan konsolidasi *file log* dan melakukan visualisasi informasi dari data yang terkandung didalam *file log* tersebut. Pada *paper* ini akan dipaparkan bagaimana mendesain dan mengimplementasi aplikasi yang dapat melakukan visualisasi pada *file log web server*.

**Kata kunci :** *Data Visualization, Administration System*

## 1. Pendahuluan

Perkembangan internet memicu pesatnya aplikasi *web* dengan berbagai corak dengan tujuan tertentu yang hendak dicapai. *Web server* yang berfungsi sebagai *host* dari semua aplikasi *web* mempunyai peran penting dalam hal ini.

Untuk memenuhi *audit* dan laporan, setiap *web server* umumnya menyediakan fitur yang dapat menulis *file log* setiap *request* dan *response* yang masuk kedalamnya. *File log* ini biasanya dibuat dalam interval harian, mingguan atau bulanan.

Seiring dengan lamanya *operational web server* maka *file log* yang dihasilkan juga semakin banyak. Hal ini akan menimbulkan masalah terutama dalam melakukan *auditing* atau membuat suatu laporan berdasarkan *file log web server* yang dihasilkan. Oleh karena itu, kita membutuhkan suatu *tool* yang dapat melakukan konsolidasi beberapa *file log* dan menampilkan visualisasi informasi berdasarkan data yang tersimpan pada *file log* tersebut.

### 1.1. Tujuan

Tujuan pembuatan aplikasi ini adalah mempermudah para administrator IT atau yang bertanggung jawab pada penanganan *web server* dalam menganalisa aktivitas *web server* pada *log file*.

### 1.2. Ruang Lingkup

Batasan pada desain dan implementasi aplikasi adalah

- Data log file berasal dari *web server* IIS (*Internet Information Service*) pada *Windows Server* 2003 dan *Windows Server* 2008
- Format data log file adalah W3C Extended
- Implementasi dengan memanfaatkan teknologi .NET

## 2. Landasan Teori

### 2.1. File log Web Server

*Web server* IIS mempunyai fitur untuk menulis data log file setiap ada *request* yang masuk ke *web server* yang dapat dijadwalkan sesuai dengan kebutuhan.

*Web server* IIS dapat menulis ke *file log* dengan tiga format yaitu [1]

- W3C Extended
- IIS log file format
- NCSA

Pada paper ini akan fokus ke format W3C Extended.

Format *file log* W3C Extended adalah format *file* berbasis ASCII yang mempunyai banyak fitur sesuai dengan kebutuhan. Data yang ditulis pada *file* ini merupakan *spaces delimiter*. Sedangkan waktu yang tercatat tersimpan sebagai waktu UTC.

Fitur data yang sediakan oleh format *file log* W3C Extended antara lain:

- **date.** Menunjukkan tanggal terjadinya aktivitas log ini
- **time.** Menunjukkan jam terjadinya aktivitas log ini
- **c-ip.** *IP address client* yang melakukan *request*

- **cs-username.** Nama yang melakukan autentikasi ke *server*
- **s-sitename.** Nama *internet service* dan nomor *instance* yang berjalan
- **s-computername.** Nama komputer *server* yang menulis *file log* ini
- **s-ip.** IP address *server*
- **s-port.** Port *server* yang digunakan
- **cs-method.** Metode akses yang dilakukan contohnya GET, POST
- **cs-uri-stem.** Target aksi
- **cs-uri-query.** *Query* yang dilakukan oleh *client* yang biasanya digunakan untuk *dynamic page*
- **sc-status.** Kode HTTP status
- **sc-substatus.** Kode *error* sub status yang telah terjadi
- **sc-win32-status.** Kode status Windows
- **sc-bytes.** Jumlah *bytes* yang dikirim oleh *server*
- **cs-bytes.** Jumlah *bytes* yang diterima oleh *server*
- **time-taken.** Lamanya waktu yang digunakan untuk mengeksekusi satu permintaan
- **cs-version.** Versi protokol HTTP atau FTP yang digunakan oleh *client*
- **cs-host.** Nama *host header*
- **cs(User-Agent).** Tipe *browser* yang digunakan oleh *client*
- **cs(Cookie).** Isi *cookie* yang dikirim dan diterima
- **cs(Referer).** *Site* yang terakhir dikunjungi sebelum mengakses *site* yang sedang diakses

Masing-masing fitur data diatas akan ditulis oleh IIS apabila kita memasukkan (mengaktifkan) ke dalam daftar fitur data yang akan ditulis.

## 2.2. Format File log

Setiap *file log* yang dihasilkan oleh *web server* IIS akan berisi informasi berupa *header* dan data. Kadangkalanya satu *file log* akan mempunyai *header* yang berbeda-beda seperti ilustrasinya pada Gambar 1.

*Header* pada log file IIS diawal dengan tanda # sedangkan datanya dengan *space delimiter* yang jumlahnya mengikuti *header*. Berikut ini contoh *header* dan data pada sebuah file IIS.

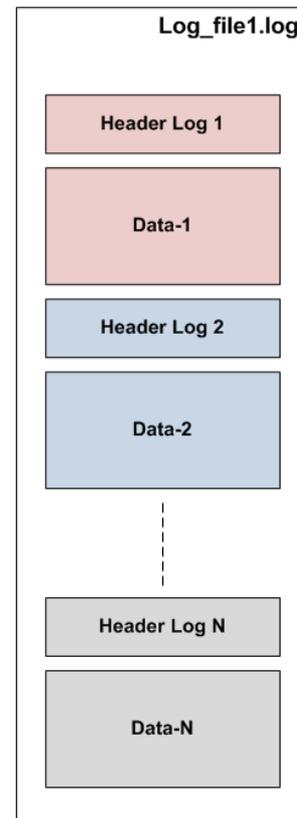
```
#Software: Internet Information Services 6.0
#Version: 1.0
#Date: 2001-05-02 17:42:15
#Fields: time c-ip cs-method cs-uri-stem sc-status
cs-version
17:42:15 172.16.255.255 GET /default.htm 200
HTTP/1.0
```

Pada umumnya *header* pada *file log* akan berisi informasi antara lain

- *Software*
- *Version*

- *Date*
- *Fields*

Informasi *field* yang digunakan dalam dilihat pada *header* dibagian #Fields dan data *file log* akan mengikuti berdasarkan *field* yang digunakan.



Gambar 1. Format *header* dan data log pada IIS

## 2.3. Teknologi .NET

.NET adalah *framework platform* yang dibuat oleh Microsoft. Tujuan utama dibuatnya *framework* ini memudahkan implementasi aplikasi baik *desktop* maupun *web* tanpa harus mengetahui banyak *library/module* pada OS Windows [2].

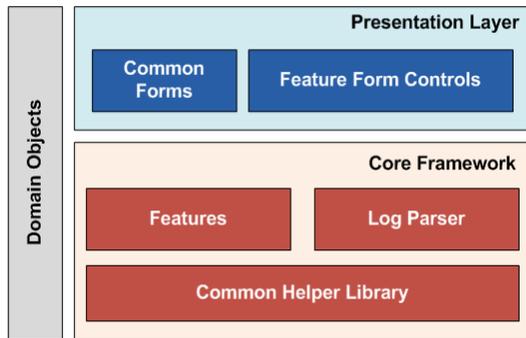
Di dalam .NET terdiri dari kumpulan *library* yang siap digunakan. Hasil kompilasi akan menghasilkan IL (*Intermediate Language*) sehingga dengan konsep ini maka .NET dapat *support* banyak bahasa pemrograman asalkan dapat menghasilkan IL sesuai standar.

## 3. Desain dan Implementasi

### 3.1. Desain Umum

Aplikasi ini didesain dengan pendekatan *layering* atau modular yang berbasis *object-oriented* sehingga diharapkan *library* atau modul dapat dipergunakan lagi (*reusable*) dan diperbarui sifatnya. Arsitektur

aplikasi ini dapat dilihat pada Gambar 2.



Gambar 2. Arsitektur umum aplikasi

Aplikasi ini secara arsitektur terbagi menjadi tiga bagian yaitu

- Core Framework
- Domain Objects
- Presentation Layers

Pembagian ini berdasarkan kesamaan fungsional masing-masing bagian.

### 3.1.1. Core Framework

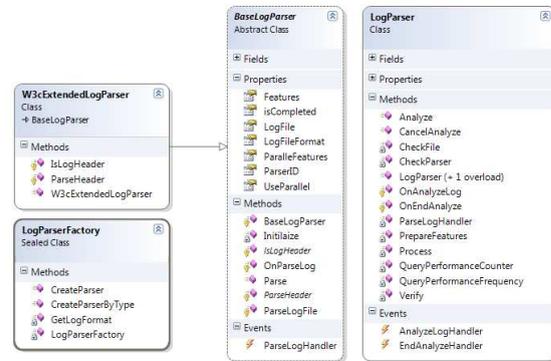
Core Framework merupakan bagian terpenting aplikasi yang berfungsi sebagai *parser* dari sumber data log file. Di dalam modul ini terdapat tiga bagian yang didasarkan pada fungsinya. Tiga bagian ini yaitu

- **Log Parser.** Ini berfungsi sebagai *parsing* pada data log dari input *log file*.
- **Features.** Ini merupakan fitur visualisasi yang akan ditampilkan. Setiap fitur mempunyai karakteristik berbeda-beda.
- **Common Helper Library.** Fungsi umum yang dapat digunakan untuk membantu proses terjadi pada aplikasi ini.

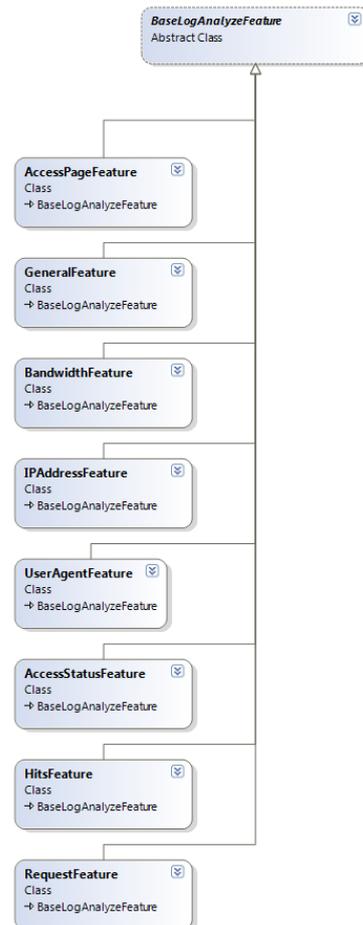
Implementasi *Log Parser* dengan memanfaatkan *Factory Method* dan *strategy patterns* [3]. Diagram kelas dapat dilihat pada Gambar 3. Ada 4 objek penting dalam realisasi bagian *Log Parser* yaitu

- **BaseLogParser.** Objek ini berfungsi sebagai *abstract object* dan dapat diimplementasi sesuai dengan kebutuhan.
- **W3cExtendedLogParser.** Objek ini merupakan turunan dari objek *BaseLogParser* dimana didalamnya mengimplementasi *parser* untuk *file log* berformat W3C Extended.
- **LogParserFactory.** Objek ini adalah realisasi *Factory Method* [2] yang melakukan instansiasi objek *BaseLogParser* berdasarkan kondisi format *file log*.
- **LogParser.** Objek ini adalah *composite object* yang mana didalamnya terdapat objek *BaseLogParser*. Objek *LogParser* akan

melakukan instansiasi objek *BaseLogParser* dengan instansiasi pada objek *W3cExtendedLogParser*.



Gambar 3. Diagram kelas untuk Log Parser



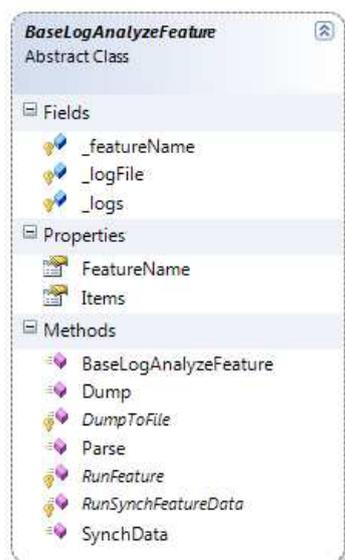
Gambar 4. Diagram kelas untuk feature

Teknik yang digunakan untuk mendeteksi apakah *file log web server* itu mempunyai format W3C Extended atau tidak adalah mendeteksi *header* yang dimiliki. Sesuai dengan *standard format log W3C*

Extended, *header* yang dimiliki suatu *file log* akan diawali dengan #. Oleh karena itu, algoritma paling mudah mendeteksi format *file log* W3C Extended adalah membaca *header* yang diawalnya adalah #.

Informasi apa yang akan divisualisasikan didasarkan pada *feature* yang disediakan. Diagram kelas *feature* dapat dilihat pada Gambar 4. Semua objek *feature* merupakan turunan objek *BaseLogAnalyzeFeature* yang merupakan *abstract object* yang dapat dilihat pada Gambar 5. Pada implementasi disini, dibuat 8 *feature* yang merupakan turunan objek *BaseLogAnalyzeFeature* antara lain:

- *AccessPageFeature*
- *AccessStatusFeature*
- *BandwidthFeature*
- *GeneralFeature*
- *HitsFeature*
- *IPAddressFeature*
- *RequestFeature*

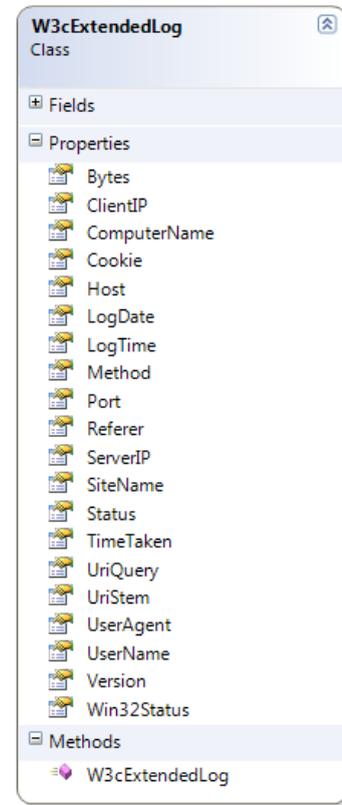


Gambar 5. Objek abstrak *BaseLogAnalyzeFeature*

### 3.1.2. Domain Objects

*Domain Objects* adalah *value object* yang menyimpan informasi dari data *file log*. Ini juga berfungsi sebagai *Data Transfer Object* (DTO).

Implementasi untuk aplikasi ini, objek domain akan menyimpan informasi yang berisi semua kolom sesuai format *file log* W3C Extended. Realisasi objek domain yaitu *W3cExtendedLog* yang dapat dilihat pada Gambar 6.



Gambar 6. Implementasi objek domain *W3cExtendedLog*

Objek *W3cExtendedLog* hanya mempunyai *property* (*Set/Get Methods*) dan tidak mempunyai *behaviors* atau *method*. Hal ini dikarenakan pada fungsinya sebagai *domain objects* yang hanya menyimpan data atau informasi.

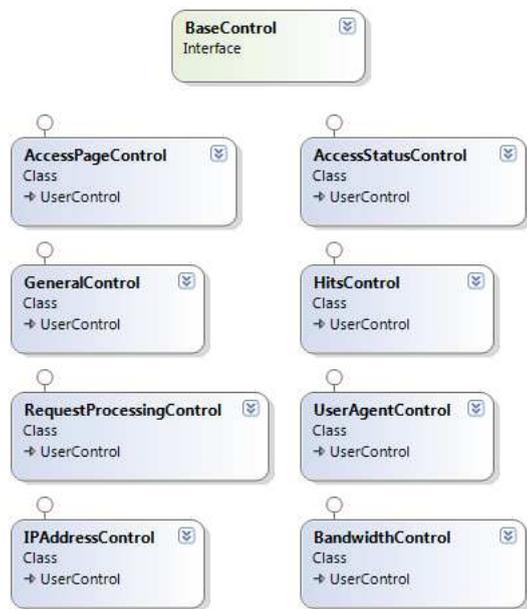
### 3.1.3. Presentation Layer

*Presentation Layer* adalah *Graphical User Interface* (GUI) aplikasi yang akan dibuat. Pada *layer* ini terbagi menjadi dua bagian besar *form* yaitu

- **Feature Form Control.** Ini merupakan GUI yang dipergunakan oleh setiap untuk dilakukan visualisasi informasi
- **Common Forms.** Ini adalah GUI umum yang dapat dipergunakan oleh setiap objek.

Untuk mempermudah kontrol terhadap *feature form control* maka semua GUI *form* akan mengimplementasi *Interface object* yaitu *BaseControl*. Objek ini berisi *property DataSource* yang digunakan untuk pertukaran data dan sebuah *method/function* yaitu *Populate()* yang berguna untuk melakukan populasi data ke GUI. Kelas diagram untuk *feature form control* dapat dilihat pada Gambar 7. Selain implementasi objek *interface BaseControl*, objek-objek *feature form control* juga merupakan

turunan objek *UserControl*.



**Gambar 7.** Diagram kelas untuk *feature form control*

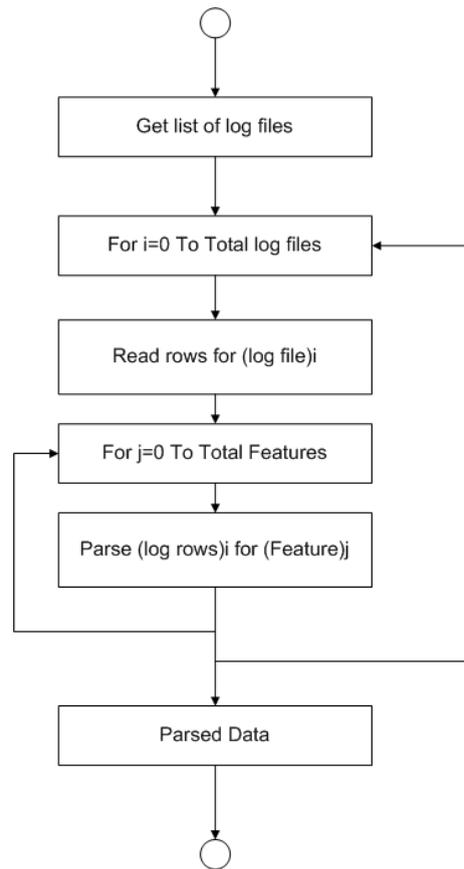
*Common form* disini digunakan sebagai *container* setiap konsolidasi *file log* yang akan telah diproses. *Container* yang dimaksud di sini yaitu objek *WebLogUserControl* yang merupakan turunan objek *UserControl*. Seperti yang diketahui, *feature form control* juga turunan objek *UserControl* sehingga cara menampilkannya cukup dengan menempelkan *feature form control* ke dalam *WebLogUserControl*.

### 3.2. Algoritma

Algoritma yang diterapkan untuk membaca *file log* dan selanjutnya melakukan *parsing* sesuai dengan kebutuhan *Feature* dapat dilihat pada Gambar 8.

Berikut ini algoritma yang diterapkan berdasarkan pada Gambar 8:

1. Mula-mula aplikasi mengidentifikasi jumlah *file log* dan menyimpan daftar nama *file log* tersebut
2. Masing-masing *file log* akan diterapkan proses seperti langkah 3 sampai 6
3. Isi data pada *file* dibaca setiap baris
4. Setiap data baris yang dibaca akan dilakukan deteksi apakah ini *header* atau data
5. Jika baris data itu adalah data maka akan dilakukan pemrosesan berdasarkan *features* yang ada
6. Jika selesai pembacaan datanya maka hasil pemrosesan akan dikonsolidasi
7. Apabila selesai semua proses pembacaan semua *file* maka akan diperoleh data konsolidasi berdasarkan *feature*-nya.



**Gambar 8.** Algoritma aplikasi visualisasi *file log*

8. Data konsolidasi ini yang akan ditampilkan pada GUI

### 3.3. Visualisasi Informasi

Visualisasi informasi pada aplikasi ini diwakili dengan *Feature*. Setiap *Feature* mempunyai cara memproses dan GUI untuk menampilkan data yang berbeda-beda.

Pada aplikasi ini, ada 5 *Feature* yang akan diimplementasi antara lain:

1. **Access Page.** Visualisasi yang menggambarkan jumlah akses pada setiap halaman portal
2. **Access Status.** Visualisasi status pada setiap akses yang terjadi
3. **Bandwidth.** Visualisasi pemakaian *bandwidth* keluar masuk setiap akses
4. **General.** Informasi umum mengenai data *file log* yang telah diproses
5. **Hits.** Visualisasi jumlah akses terhadap waktu
6. **IP Address.** Visualisasi jumlah akses terhadap *IP Address* pengakses
7. **Request.** Visualisasi lamanya waktu pemrosesan setiap akses yang terjadi

### 3.4. Background Process

Proses pembacaan beberapa file dan *parsing* data pada setiap file akan mempengaruhi interaktif aplikasi tersebut sehingga aplikasi terkesan seperti *hang*. Oleh karena itu, semua fase pemrosesan ini dilakukan dengan *background process* atau *asynchronous*.

Implementasi *background process* dilakukan dengan memanfaatkan sistem *threading*. Setiap pemrosesan akan dibuat satu *thread* dan apabila selesai maka *thread* ini akan memberitahukan ke aplikasi melalui pendekatan *event driven* [4].

### 3.5. File log pada Remote Web Server

Aplikasi ini tidak hanya dapat menerima input *file log* dari lokal *web server* tetapi juga dapat melakukan visualisasi *file log* pada *remote web server*.

Algoritma yang diimplementasi adalah *copy* semua *file log web server* dari *remote web server* ke lokal mesin dimana aplikasi ini dijalankan. Untuk masalah sekuriti ketika melakukan koneksi dan *copy file log* ke *remote web server* maka aplikasi melakukan *impersonate* dengan cara memberikan *user* dan *password* yang mempunyai hak pada *remote web server*.

### 3.6. Konsolidasi Log File

Aplikasi ini dapat melakukan konsolidasi beberapa *file log web server* untuk sekaligus dilakukan visualisasi informasi.

Pada Gambar 8 merupakan algoritma yang diterapkan pada aplikasi memungkinkan dalam melakukan konsolidasi karena aplikasi dapat menerima input beberapa *file log* sekaligus.

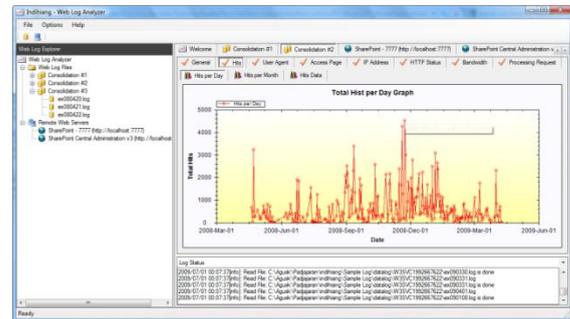
Acuan konsolidasi yang digunakan disini mengacu pada data *date*. Pada data baris yang mempunyai *date* yang sama akan dikelompokkan menjadi satu grup data.

### 3.7. Implementasi

Implementasi aplikasi ini memanfaatkan teknologi .NET Framework dan *development tool* yang digunakan adalah Visual Studio 2008. Sedangkan *source code* ditulis dengan menggunakan bahasa Visual C#. Hasil implementasi dapat dilihat pada Gambar 9.

Aplikasi ini juga dikompilasi pada dua target platform yaitu x86 dan x64. Ini dapat dilakukan melalui *tool* Visual Studio 2008.

Sumber kode aplikasi ini juga dipublikasikan sebagai *open source project* yang dapat diunduh di <http://indihiang.codeplex.com>.



Gambar 9. Hasil implementasi aplikasi visualisasi *file log*

## 4. Penutup

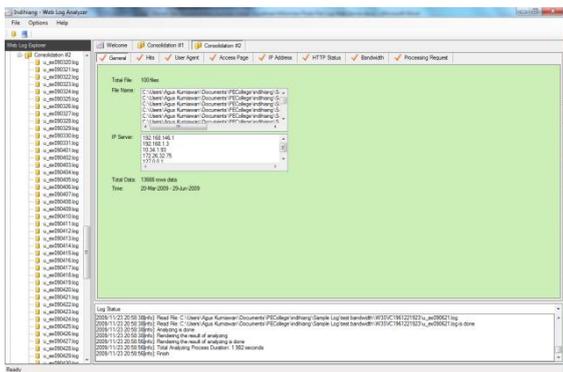
Pengujian aplikasi dengan menggunakan data *file log* yang diambil dari *web server* IIS dan diletakan di lokal mesin dimana aplikasi ini dijalankan. Selain itu, pengujian dilakukan dengan menggunakan *file log* pada *remote web server*.

Setelah memasukan satu *file log* atau beberapa kedalam aplikasi maka aplikasi ini akan bekerja untuk melakukan *parsing* dan menampilkan hasilnya dalam bentuk *reporting*. Contoh beberapa *reporting* yang dihasilkan aplikasi ini dapat dilihat pada Gambar 10a-10h.

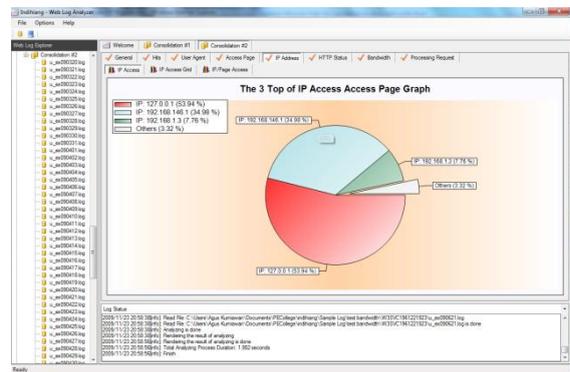
## REFERENSI

- [1] Microsoft, IIS Log File Formats, <http://www.microsoft.com/technet/prodtech/nol/WindowsServer2003/Library/IIS/be22e074-72f8-46da-bb7e-e27877c85bca.mspx?mfr=true>.
- [2] Microsoft .NET Framework, <http://www.microsoft.com/net>.
- [3] Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison-Wesley Professional, 1994.
- [4] Gregor Hohpe and Bobby Woolf, *Enterprise Integration Patterns*, Addison-Wesley, 2003.

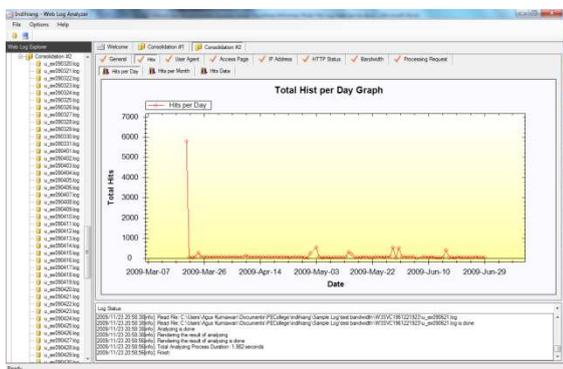
Desain dan Implementasi Aplikasi untuk Visualisasi Informasi pada File Offline Log Web Server



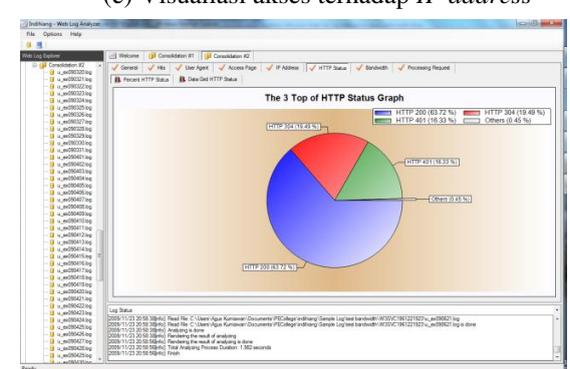
(a) Visualisasi general



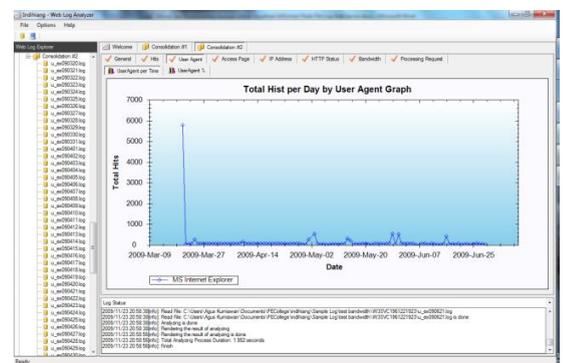
(e) Visualisasi akses IP terhadap IP address



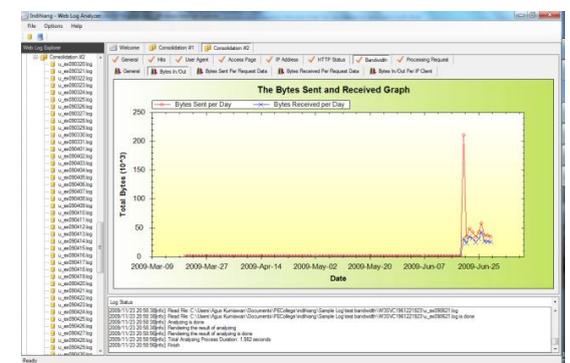
(b) Visualisasi hits



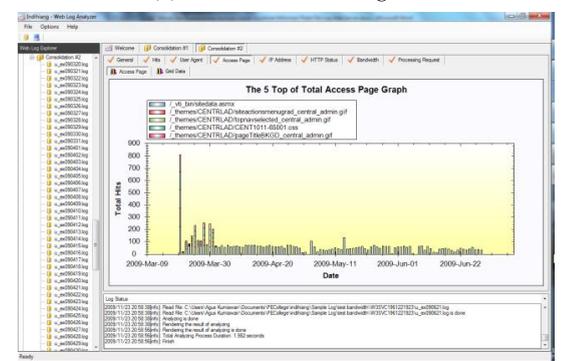
(f) Visualisasi HTTP status



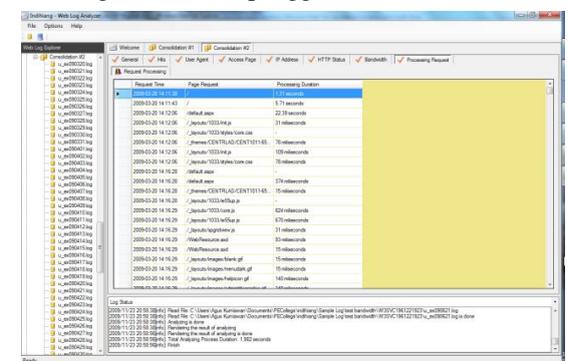
(c) Visualisasi user agent



(g) Visualisasi penggunaan bandwidth



(d) Visualisasi access page



(h) Visualisasi lamanya proses

Gambar 10a-10h. Contoh hasil visualisasi pada beberapa log file