

# Optimasi dan Performansi Aplikasi Web PHP

**Didik Dwi Prasetyo**

Didik\_rpl@yahoo.com

[http://groups.yahoo.com/group/didik\\_directory](http://groups.yahoo.com/group/didik_directory)

## ***Lisensi Dokumen:***

*Copyright © 2003-2006 IlmuKomputer.Com*

*Seluruh dokumen di IlmuKomputer.Com dapat digunakan, dimodifikasi dan disebarkan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut penulis dan pernyataan copyright yang disertakan dalam setiap dokumen. Tidak diperbolehkan melakukan penulisan ulang, kecuali mendapatkan ijin terlebih dahulu dari IlmuKomputer.Com.*

Setiap orang pasti mendambakan aplikasi yang memiliki performansi tinggi, baik itu Anda selaku pemrogram, ataupun user yang nantinya akan menggunakan program. Terlepas dari faktor spesifikasi perangkat keras, desain, algoritma, dan penulisan kode program juga sangat mempengaruhi hasil aplikasi, khususnya terkait dengan masalah performansi atau unjuk kerja.

Optimasi sendiri merupakan suatu proses *fine-tuning* program dalam upaya untuk meningkatkan kecepatan akses dan mengurangi penggunaan memori. Ada pun tujuan utamanya adalah menghasilkan waktu eksekusi kode program yang lebih lebih pendek (cepat). Ini tentu merupakan sesuatu yang sangat diharapkan sekali. Terkait dengan hal ini, sebagai pemrogram, apa yang harus kita lakukan untuk mengoptimasi kode program aplikasi PHP?

Artikel ini akan menjelaskan tip-tip praktis untuk mengoptimasi dan meningkatkan performansi aplikasi web PHP.

## **Optimasi Kode Program PHP**

Tidak bisa dipungkiri, menulis kode secara efektif dan efisien adalah kunci dasar yang penting untuk meningkatkan kecepatan eksekusi kode. Seperti diketahui, kode-kode PHP akan selalu dieksekusi setiap kali ia di-*request* oleh *client* (browser). Oleh karena itu, sangat menguntungkan sekali jika Anda bisa membuat kode yang efektif dan efisien.

Berikut ini adalah beberapa tip dan strategi untuk mengoptimasi kode program aplikasi PHP secara umum.

- **Hindari spasi untuk mengatur kode**  
Menghindari penggunaan spasi (atau &nbsp;) yang berlebihan adalah langkah efisien. Pertimbangkan, setiap spasi adalah 1 byte dan setiap tab (\t) juga 1 byte. Ketika Anda membuat empat spasi, Anda telah menghabiskan 4 byte data. Tentu akan lebih efisien jika hanya menggunakan sebuah tab.
- **Cara menggunakan tipe data boolean**  
PHP memungkinkan Anda menulis tipe data boolean dengan huruf kecil atau besar (*case-insensitive*). Meskipun demikian, penulisan dengan huruf kecil semua akan lebih cepat dibanding huruf besar. Ini dikarenakan, saat menemukan konstanta, PHP melakukan *lookup hash* nama konstanta.

```
if ($var = TRUE) {  
    .....  
}  
  
// akan dieksekusi lebih cepat  
if ($var = true) {  
    .....  
}
```

Sebagai tambahan, saat bekerja dengan nilai boolean, menggunakan nilai 1 dan 0 lebih cepat dibanding true dan false.

- **Hindari konkatenasi string yang besar**

Pada saat melakukan konkatenasi string, hindari penggabungan dengan string yang ukurannya besar. Ini bisa menghambat eksekusi kode yang sebenarnya bisa ditampilkan dengan cepat. Contohnya seperti berikut:

```
// Konkatenasi string yang besar  
$title = 'title';  
$body = '...a very large block...';  
echo "Subject: $title\n\n$body";  
  
// Menghindari konkatenasi string yang besar  
$title = 'title';  
$body = '...a very large block...';  
echo "Subject: $title\n\n";  
echo $body;
```

- **Mencetak output**

Tiga cara yang umum dilakukan untuk mencetak data ke output adalah: direct output, echo(), dan print(). Ketika kode program tidak mengandung kode-kode PHP, mencetak dengan direct output lebih efektif dan efisien. Contoh direct output seperti berikut:

```
<?php  
// kode php  
?>  
  
Direct output  
  
<?php  
// kode php  
?>
```

Apabila kondisi Anda memerlukan untuk mencetak output dengan menggunakan fungsi (konstruksi bahasa) PHP, gunakan echo(), bukannya print(). Meskipun secara garis besar print() dan echo() memiliki tujuan sama, akan tetapi ada beberapa perbedaan esensial yang perlu diperhatikan.

Fungsi print() berperilaku seperti fungsi pada umumnya, dan memiliki nilai kembalian (return value) berupa integer 1. Dengan demikian, print() dapat digunakan sebagai bagian dari ekspresi yang lebih kompleks. Sementara itu, echo() mampu menerima lebih dari satu parameter sekaligus, dan tidak memiliki nilai kembalian.

```
print 'String 1';  
echo 'String 1';  
  
// Menggunakan beberapa parameter  
echo 'String 1', "String 2", '...';
```

Fungsi string echo() akan dieksekusi lebih cepat dibanding dengan print(). Perbedaan ini disebabkan karena fungsi print() akan mengembalikan status (integer) yang menyatakan apakah proses berhasil dilaksanakan atau tidak.

Di sisi lain, echo() hanya menampilkan output saja dan tidak mengerjakan apa-apa lagi. Ada pun dalam implementasinya, status nilai kembalian dari penggunaan fungsi string hampir tidak pernah diperlukan.

- **Memeriksa panjang string**

Cara umum untuk memeriksa panjang karakter adalah dengan menggunakan fungsi `strlen()`. Untuk tujuan yang sama, sebenarnya ada cara yang lebih cepat, yaitu menggunakan `isset()`. Contohnya seperti berikut:

```
if (strlen($str) < 5) {  
    echo 'String must be at least 5 chars';  
}
```

```
if (!isset($str{5})) {  
    echo 'String must be at least 5 chars';  
}
```

Seperti halnya kasus `echo()` dan `print()`, `isset()` memerlukan waktu eksekusi lebih pendek karena ia merupakan konstruksi bahasa.

Konstruksi bahasa `echo()` juga mengizinkan Anda untuk memberikan lebih dari satu string sebagai parameter. Menggunakan beberapa parameter akan lebih cepat dibanding mencampur variabel ke dalam sebuah parameter tunggal. Contohnya seperti berikut:

```
$a = 'Hello';  
$b = 'World';  
  
echo 'Say ' . $a . ' to ' . $b;  
  
// Lebih cepat  
echo 'Say ', $a, ' to ', $b;
```

## Pendekatan Berbasis Referensi

Pendekatan berbasis referensi (*passing-by-ref*) seringkali menimbulkan salah interpretasi. Pertanyaan umum yang sering dilontarkan terkait dengan topik ini adalah, apakah referensi meningkatkan performansi? Pada prinsipnya, referensi tidak menyediakan keuntungan performansi apapun untuk tipe data string, integer, dan tipe-tipe skalar lainnya. Akan tetapi, lain lagi ceritanya jika referensi diimplementasikan pada tipe data gabungan (array dan objek).

Untuk lebih jelasnya, perhatikan dua contoh fungsi sederhana berikut:

```
// Menggunakan referensi  
function UseReferensi(&$a) {  
    $b = $a;  
    $c = $a;  
}  
  
// Tanpa referensi  
function NoReferensi($a) {  
    $b = $a;  
    $c = $a;  
}
```

Pada kenyataannya, engine PHP (Zend Engine) tidak menciptakan duplikat variabel ketika melakukan “*pass by value*”, tetapi mengimplementasikan *reference-counted* dan *copy-on-write* secara internal. Dengan demikian, di fungsi pertama, `$b` dan `$c` memerlukan waktu lebih lama untuk di-set, karena referensi harus di-track. Sementara itu, `$b` dan `$c` di fungsi kedua hanya menunjuk ke nilai asli `$a`, dan *counter* referensi di-*increment*. Hasilnya, fungsi kedua akan dieksekusi lebih cepat dibanding fungsi pertama.

Di sisi lain, saat menggunakan referensi di fungsi yang menerima parameter berupa array atau objek, akan meningkatkan performansi. Ini disebabkan, tipe data array dan objek tidak menggunakan *counting* referensi. Dengan kata lain, jika menggunakan “*pass by value*” pada array atau objek, maka akan menciptakan *multiple copy*.

Sebagai tambahan, di PHP 5, semua objek secara otomatis akan di-*pass by reference*. Jadi, Anda tidak perlu lagi menambahkan operator `&` secara eksplisit.

## Flushing Output ke File

Dalam upaya mencapai efisiensi, umumnya fungsi-fungsi input dan otuput (I/O) tidak langsung

menulis data ke file, begitu kita menginstruksikannya. Ada pun yang dilakukan adalah, menumpuk ke *buffer* dan baru menyimpan (menulis) ke disk dalam satu waktu.

Agar operasi penulisan langsung dilaksanakan tanpa di-pending terlebih dahulu, gunakan fungsi `fflush()`. Contohnya seperti berikut:

```
$fp = fopen('c:/tmp/test.txt', 'w');  
fwrite($fp, 'Test flushing output');  
  
// Flushing output ke file  
fflush($fp);  
fclose($fp);
```

Dalam beberapa kasus, ada kemungkinan penulisan file yang memanfaatkan `fflush()` mengakibatkan file gagal dibaca. Apabila kasus seperti ini terjadi pada Anda, gunakan fungsi `clearstatcache()` sebelum Anda melakukan pembacaan isi file.

## Flushing Output ke Browser

Seperti halnya ketika melakukan *flushing* output ke file, Anda bisa melakukan *flushing* output ke browser. Teknik ini bisa lebih meningkatkan pengiriman output ke browser. Implementasinya, sebaiknya Anda memisahkan proses yang cepat dan proses yang memerlukan waktu lebih.

Sebagai ilustrasi, sebelum operasi query data selesai, Anda bisa mengirim informasi status terlebih dahulu. Dengan demikian, client tidak mendapati halaman blank ketika query sedang dilakukan.

Contoh sederhananya seperti berikut:

```
// Simulasi request user  
$keyword = 'Test Flushing';  
$jml = 1000;  
echo 'Searching ', $keyword;  
// Flushing output ke browser  
flush();  
  
echo '<p>Result : <br>';  
// Simulasi proses data besar  
for ($i=1; $i<$jml; $i++) {  
    echo $i, ' ', $keyword, '<br>';  
}
```

Fungsi `flush()` akan mengirimkan semua output yang secara internal di-buffer PHP ke web server. Dalam beberapa kasus, browser client mungkin tidak langsung menampilkan data begitu sudah didapatkannya. Selain itu, beberapa versi Internet Explorer tidak akan menampilkan data sampai ia menerima sedikitnya 256 byte data. Untuk mengatasi masalah di Internet Explorer ini, Anda bisa mengirim karakter kosong sebelum melakukan *flushing* output.

## Buffering Output Aplikasi

Buffering output merupakan salah satu teknik yang digunakan untuk mengurangi waktu eksekusi output kode. Ide dasar dari teknik ini adalah, menyimpan semua isi halaman web ke dalam buffer memori sebelum kemudian dikeluarkan secara bersamaan. Jadi, output aplikasi tidak langsung dikirimkan ke browser client.

Keuntungan yang bisa Anda peroleh dengan melakukan buffering output antara lain:

- Mengurangi operasi-operasi I/O guna meningkatkan performansi. Di mana operasi-operasi I/O bisa dilakukan secara sekuensial dan dengan cepat.
- Mengizinkan Anda untuk memodifikasi isi halaman web sebelum kemudian dikeluarkan.

Untuk mengimplementasikan buffering output, Anda cukup memanggil fungsi `ob_start()` di bagian atas kode program dan `ob_end_flush()` di bagian akhir program. Contoh implementasinya seperti berikut:

```
<?php  
// Me-replace string  
function callback($buff) {  
    return (str_replace('unjuk kerja',  
        'performansi', $buff));  
}
```

```
}  
// Aktifkan buffering output  
ob_start('callback');  
?>  
  
<html>  
<body>  
<p>  
Buffering output, untuk meningkatkan  
unjuk kerja  
</p>  
</body>  
</html>  
  
<?php  
echo 'ob size: ', ob_get_length(), ' byte';  
  
// Mengirim buffer output dan disable buffering  
while (ob_get_level() > 0) {  
    ob_end_flush();  
}  
?>
```

Sebenarnya, tanpa memanggil `ob_end_flush()` pun, data akan tetap dikeluarkan begitu kode selesai dieksekusi. Meskipun demikian, untuk menyeimbangkan kode, sebaiknya Anda memanggil `ob_end_flush()`.

Dalam melakukan buffering output, Anda juga bisa menentukan sendiri ukuran buffer. Ukuran ini akan dijadikan sebagai batas jumlah data yang akan di-buffer. Selanjutnya, setiap buffer penuh, data akan dikeluarkan ke browser client. Langkah ini bisa mencegah penggunaan memori yang berlebihan, ketika data cukup besar dan memori terbatas.

## Kompresi Output

Kompresi output adalah salah satu teknik yang efektif dan efisien untuk meningkatkan performansi aplikasi. Apabila kompresi diaktifkan, ia akan mendeteksi entry khusus di request browser dan mengompres setiap output yang dikeluarkan oleh aplikasi. Selanjutnya, browser akan men-dekompres data dari server sebelum ditampilkan ke user.

Bagaimanapun juga, teknik kompresi memerlukan dukungan browser. Apabila browser tidak mendukung dekompresi, proses kompresi tidak akan berpengaruh. Umumnya, kebanyakan browser saat ini memiliki kemampuan untuk menerima data terkompres.

Ada dua pendekatan yang bisa Anda lakukan untuk mengompres output aplikasi. Pertama, dengan melakukan konfigurasi langsung pada directive `zlib` di `php.ini`, dan kedua menggunakan fungsi-fungsi kontrol output. Contoh pendekatan pertama seperti berikut:

```
zlib.output_compression= On
```

Untuk mendukung kompresi, Anda juga bisa menetapkan level kompresi.

```
; minimal compression  
zlib.output_compression_level= 1  
; maximal compression  
zlib.output_compression_level= 9
```

Semakin tinggi level kompresi, akan semakin kecil ukuran data yang dihasilkan. Akan tetapi, sebagai konsekuensinya, ini memerlukan waktu lebih bagi CPU server untuk mengompres data.

Untuk pendekatan kedua, gunakan fungsi `ob_start()` dengan parameter `ob_gzhandler()`.

```
// Untuk menghindari konflik kompresi  
if (!ini_get('zlib.output_compression')) {  
    // Buffering output dengan kompresi  
    ob_start('ob_gzhandler');  
    echo 'kompresi dengan ob_gzhandler <p>';  
} else {  
    echo 'kompresi dengan zlib.output_compression <p>';  
}  
for ($i=0; $i<10; $i++) {  
    echo $i, ' Test kompresi <br>';  
}
```

```
}  
echo 'size: ', ob_get_length();
```

Pada prinsipnya, kedua pendekatan di atas memerlukan ekstensi zlib. Artinya, kedua pendekatan tersebut tidak dapat bekerja seperti yang diharapkan apabila ekstensi zlib tidak aktif. Sebagai tambahan, Anda tidak dapat menggunakan kedua pendekatan tersebut secara bersamaan. Disarankan, Anda memilih pendekatan yang kedua, dibanding menggunakan `ob_gzhandler()`.

## Caching Content Aplikasi

Caching content adalah teknik penyimpanan isi halaman (content) web yang di-generate secara dinamis dan selanjutnya bisa di-retrieve untuk ditampilkan ke browser client. Pendekatan ini sangat efisien sekali, karena memungkinkan Anda “mengubah” halaman dinamis PHP menjadi halaman statis, melakukan buffering output, dan menyimpan ke suatu file.

Untuk mengimplementasikan caching content, Anda bisa membuat kode program sendiri atau memanfaatkan paket PEAR yang sudah tersedia. Mengingat caching ini sangat kompleks, akan lebih praktis jika memanfaatkan paket Cache dari PEAR. Berikut ini contoh kode programnya:

```
// Include class PEAR::Cache_Lite Output  
require_once 'Cache/Lite/Output.php';  
  
// Konfigurasi option Cache_Lite  
$opts = array(  
    'cacheDir'      => './cache/',  
    'writeControl'  => 'true',  
    'readControl'   => 'true',  
    'readControlType' => 'md5'  
);  
  
// Membuat objek Cache_Lite_Output  
$cache = new Cache_Lite_Output($opts);  
  
// Set lifetime caching (1 minggu)  
$cache->setLifeTime(302400);  
// Start caching dengan id header  
if (!$cache->start('header', 'Static')) {  
    ?>  
    <html>  
    <head>  
    <title>Caching with PEAR::Cache_Lite</title>  
    <body> <center>  
    <p> Simulasi Header Page <br>  
    Last modified @ <?=date('H:i:s')?> <hr><p>  
  
    <?php  
    // Output di-buffer sampai pemanggilan  
    // method end(), dan simpan ke cache  
    $cache->end();  
}  
  
// Simulasi content body dinamis  
// Lifetime = 5 detik  
$cache->setLifeTime(5);  
if (!$cache->start('body', 'Dynamic')) {  
    echo 'Body already modified... <br>';  
    echo 'Last modified @ ', date('H:i:s');  
    $cache->end();  
}  
  
// Content footer, statis  
$cache->setLifeTime(302400);  
if (!$cache->start('footer', 'Static')) {  
    ?>  
  
    <p><hr> Simulasi Footer Page <br>  
    Last modified @ <?=date('H:i:s')?>  
    </body>  
    </html>
```

```
<?php  
$cache->end();  
}
```



Gambar 1 Caching content aplikasi web

## Profiling Kode

Untuk mengukur dan menyimpulkan bahwa suatu fungsi atau konstruksi bahasa dieksekusi cepat atau lambat, tentu tidak bisa hanya berdasar analisis kode. Tool pengukuran yang dapat merepresentasikan waktu eksekusi kode secara nyata adalah kode profiler.

Pada prinsipnya, semua kode profiler PHP memiliki acuan dasar untuk melakukan pengukuran, misalnya dengan fungsi `microtime()` atau `getrusage()`. Secara garis besar, profiler mendapatkan nilai eksekusi dengan rumus waktu eksekusi akhir (selesai) dikurangi waktu awal eksekusi.

Berdasarkan rumus global yang digunakan oleh kode profiler, Anda juga bisa membuat sendiri kode profiler yang sederhana. Kode programnya seperti berikut:

```
<?php  
  
// Eksekusi fungsi ketika pemrosesan kode selesai  
register_shutdown_function('testSummary');  
$timing = array();  
  
function startTest($label, $fungsi, $iter=1) {  
    global $timing;  
    echo 'Testing ', $label, '<br>';  
    // Flushing output buffer  
    ob_flush();  
    $start = currTime();  
    // Memanggil fungsi yang dispesifikasikan  
    call_user_func($fungsi, $iter);  
    $timing[$label] = currTime() - $start;  
    return $timing[$label];  
}  
  
function testSummary() {  
    global $timing;  
    if (empty($timing)) {  
        return;  
    }  
    // sort reverse elemen array  
    arsort($timing);  
    // Set pointer ke elemen pertama  
    reset($timing);  
    $slowest = current($timing);  
    // Set pointer ke elemen terakhir  
    end($timing);  
    echo '<h4>The Winner is: ', key($timing), '</h4>';  
    echo '<table border=1> <tr> <td> Contestants </td>';  
  
    foreach ($timing as $label => $time1) {  
        echo '<th>', $label, '</th>';  
    }  
    echo '</tr>';  
}
```

```
$copy = $timing;
foreach ($copy as $label => $time1) {
    echo '<tr><td><b>', $label, '</b><br>';
    printf("%.3f seconds</td>\n", $time1);
    foreach ($timing as $label2 => $time2) {
        $percent = (($time2 / $time1) - 1) * 100;
        if ($percent > 0) {
            printf("<td>%.3f seconds <br> %.1f%% slower", $time2, $percent);
        } elseif ($percent < 0) {
            printf("<td>%.3f seconds <br> %.1f%% faster", $time2, -$percent);
        } else { // sama dengan 0
            echo '<td align=center> -';
        }
        echo '</td>';
    }
    echo '</tr>';
}
echo '</table>';
}

// Get current time (format Unix)
function currTime() {
    list($usec, $sec) = explode(' ', microtime());
    return ((float)$usec + (float)$sec);
}

?>
```

Cara menggunakan kode profiler di atas seperti berikut:

```
<?php
require_once './myprofiler.php';
$arr = array('a'=> 'satu', 'b'=> 'dua',
            'c'=> 'tiga', 'd'=> 'empat', 'e'=> 'lima');

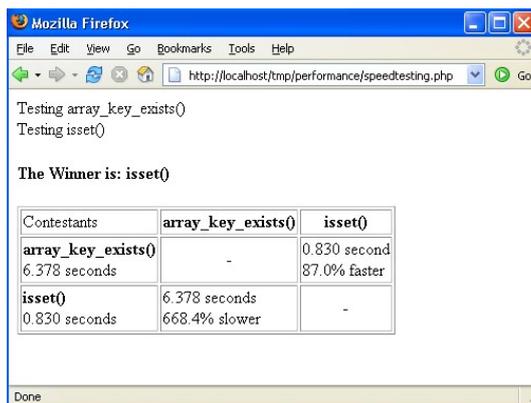
$loops = 1000000;
startTest('array_key_exists()', 'use_key', $loops);
startTest('isset()', 'use_isset', $loops);

function use_key($loops) {
    global $arr;
    for ($i=0; $i<$loops; $i++) {
        if (array_key_exists('c', $arr)) {
            // elemen ditemukan
        } else { exit('err1...'); }
    }
}

function use_isset($loops) {
    global $arr;
    for ($i=0; $i<$loops; $i++) {
        if (isset($arr['c'])) {
            // elemen ditemukan
        } else { exit('err2...'); }
    }
}

?>
```

Hasil dari pengujian kurang lebih akan terlihat seperti Gambar 2.



Testing array\_key\_exists()  
Testing isset()  
**The Winner is: isset()**

Contestants	array_key_exists()	isset()
array_key_exists() 6.378 seconds	-	0.830 second 87.0% faster
isset() 0.830 seconds	6.378 seconds 668.4% slower	-

Gambar 2 Pengujian dengan kode profiler

Apabila Anda merasa kurang puas dengan kinerja kode profiler di atas, sebagai tambahan, gunakan paket Benchmark yang tersedia di PEAR. Implementasinya seperti berikut:

```
// Include class PEAR::Benchmark Iterate
require 'Benchmark/Iterate.php';

$arr = array('a'=> 'satu', 'b'=> 'dua',
            'c'=> 'tiga', 'd'=> 'empat', 'e'=> 'lima');

$loops = 1000; // loop 1000 kali
function use_key($arr) {
    global $loops;
    for ($i=0; $i<$loops; $i++) {
        if (array_key_exists('c', $arr)) {
            // elemen ditemukan
        } else { exit('err1...'); }
    }
}

function use_isset($arr) {
    global $loops;
    for ($i=0; $i<$loops; $i++) {
        if (isset($arr['c'])) {
            // elemen ditemukan
        } else { exit('err2...'); }
    }
}

$timer =& new Benchmark_Iterate;

$timer->run($loops,'use_key', $arr);
$res1 = $timer->get();

$timer->run($loops,'use_isset', $arr);
$res2 = $timer->get();

echo '<h4>Summary</h4>';
echo "<p>Mean execution time of key: $res1[mean]</p>";
echo "<p>Mean execution time of isset: $res2[mean]</p>";
```

Akhirnya, sampai di sini pembahasan mengenai sedikit tip dan trik untuk mengoptimasi dan meningkatkan aplikasi web PHP. Bagaimanapun juga, pembahasan mengenai performansi sangatlah kompleks, dan dipengaruhi faktor-faktor terkait, seperti web server dan database server. Artikel ini merupakan salah satu pembahasan dari buku penulis yang berjudul “101 Tip dan Trik PHP“, dan mudah-mudahan bisa menjadi referensi yang bermanfaat.

## Referensi

PEAR Manual, PEAR Documentation Group, 2005.

PHP Manual, PHP Documentation Group, 2005.  
<http://www.zend.com/>

## **BIOGRAFI PENULIS**

**Didik Dwi Prasetyo.** Lahir di Bojonegoro, 30 September 1979. Menyelesaikan program S1 jurusan Teknik Informatika di Universitas Ahmad Dahlan, Jogjakarta, pada tahun 2004.

Kompetensi inti adalah pada bidang Software Engineering dan Database Management System. Kegiatan yang ditekuni sampai sekarang adalah pemrogram freelance dan sebagai penulis buku-buku komputer di PT. Elexmedia Komputindo (Gramedia Group).

Informasi lebih lanjut tentang penulis ini bisa didapat melalui:

Email: [didik\\_rpl@yahoo.com](mailto:didik_rpl@yahoo.com)

[http://groups.yahoo.com/group/didik\\_directory](http://groups.yahoo.com/group/didik_directory)