Konsep dan Perancangan *Code-Completion* untuk PHP

Tjatur Kandaga, Rinardi Budiadi Sarean
Jurusan S1 Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri no. 65 Bandung 40164
email: tjatur.kandaga@itmaranatha.org, rinardi_1518_sarean@hotmail.com

Abstract

In this information age, the development of web application is very important. Almost all corporations have one or more websites to support their business. Even now, individuals have a personal webpages. PHP is one of programming language able to create web application, PHP stands for Hypertext Preprocessor. This language capable of building web application from a simple one to a relatively complex one. To be able to create a web application rapidly, programmer need good tools. Code completion is one of tools/features that programmers find very useful. A code completion tool can complete the code according to the keywords, and standard library of a particular programming language. This paper provide an example of building code completion tool in PHP. Besides keywords and standard libraries, the tool built can read another source code in the same project as it's source of words list as well.This code completion feature is part of a project that has been successfully built an integrated development environment for PHP language.

Keyword : Code Completion, PHP, Web

1. Pendahuluan

Bahasa pemrograman *PHP* adalah salah satu bahasa pemrograman untuk aplikasi web yang terkenal. Bahasa ini dieksekusi oleh server web setiap kali ada permintaan. Banyak sekali keuntungan menggunakan *PHP*, antara lain, penggunaan sintak pemrograman bahasa *C* yang sudah terkenal, tidak adanya tipe tipe data untuk variabel yang digunakan, banyaknya fungsi yang disediakan untuk ekspansi dengan menggunakan sistem lain. Selain itu juga *PHP* merupakan bahasa pemrograman yang sifatnya *open source*. Artinya bahasa ini boleh dipakai oleh siapapun juga, dan untuk apapun tanpa harus mengeluarkan biaya untuk lisensinya. Sifat *open source* juga, menjadikan bahasa ini boleh dimodifikasi untuk memenuhi kebutuhan pengguna.

Karena sifatnya yang dieksekusi oleh web server, maka tidak dibutuhkan *compiler* khusus untuk editor *PHP*. Banyak sekali *editor* php yang tersebar di dunia ini, mulai dari yang *open source* sampai yang profesional. Akan tetapi hampir semua editor yang ada, kurang akan fitur yang sangat penting untuk sebuah editor yang bagus yaitu *code completion* (melengkapi kode secara otomatis).

Tulisan ini akan membahas mengenai salah satu konsep untuk merancang sebuah *code completion* untuk *PHP* dan sekaligus membahas perancangan-nya yang direprensentasikan melalui diagram agar dapat dikembangkan menggunakan bahasa pemrograman yang ada.

2. PHP (Hypertext Preprocessor)

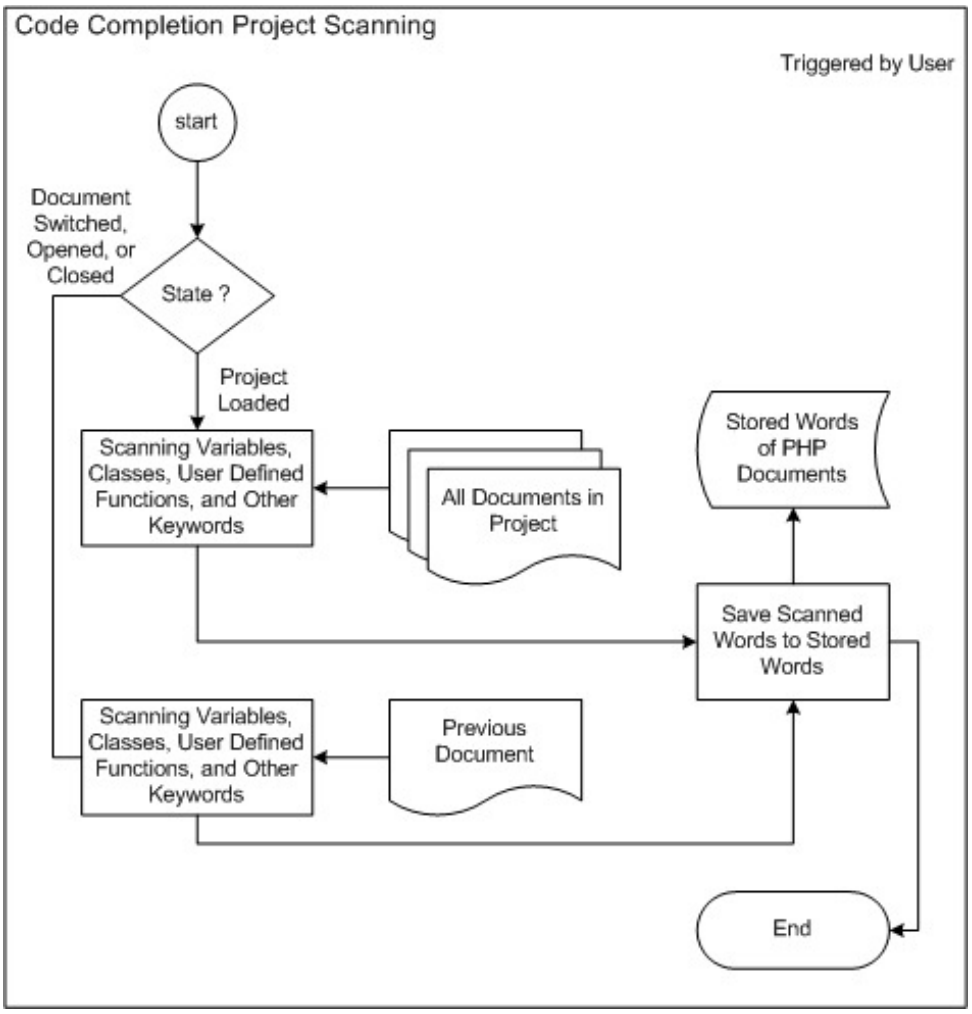
PHP, singkatan dari *Hypertext Preprosesor*, merupakan sebuah *scripting language* yang digunakan untuk menghasilkan halaman *web*. *PHP* bermula dari sebuah *scripting* yang dikembangkan oleh *Rasmus Lerdorf* pada tahun 1995[1]. Dari sumber tersebut, dijelaskan lebih lanjut, bahwa kemudian dari bahasa tersebut dikembangkan menjadi *PHP* (*PHP 3.0*) oleh *Andi Gutmans* dan *Zeev Suraski* pada tahun 1997. Bentuk ini merupakan bentuk yang mirip seperti digunakan sekarang ini. Kemudian pada tahun 1998, Mereka berdua mulai mengembang *core* dari mesin *PHP* yang dikenal dengan nama *Zend Engine*. Setahun kemudian versi *4.0* dari *PHP* menggunakan mesin ini sebagai intinya. Kemudian setelah pengembangan yang cukup lama, pada akhirnya diperkenalkanlah *PHP 5.0* dengan *Zend Engine II.* Mesin ini mempunyai fitur tambahan seperti pemrograman berorientasi objek dan banyak fungsi serta fitur lainnya.

3. Konsep Code Completion yang Dibahas

Code completion merupakan salah satu kemampuan wajib yang harus dimiliki oleh *editor* bahasa pemrograman apapun. Bila diterjemahkan, *code completion* berarti melengkapi kode. Kode disini adalah kode kode bahasa pemrograman. Bila didefinisikan, *code completion* berarti sebuah fitur yang mampu memprediksi kata-kata (kode) dengan masukan satu atau beberapa karakter saja dari pengguna, dan memunculkan hasilnya sehingga pengguna dapat memilih mana yang akan dipakai, lalu memasukannya dalam dokumen. Fitur ini akan berjalan dengan otomatis menunggu masukan dari pengguna.

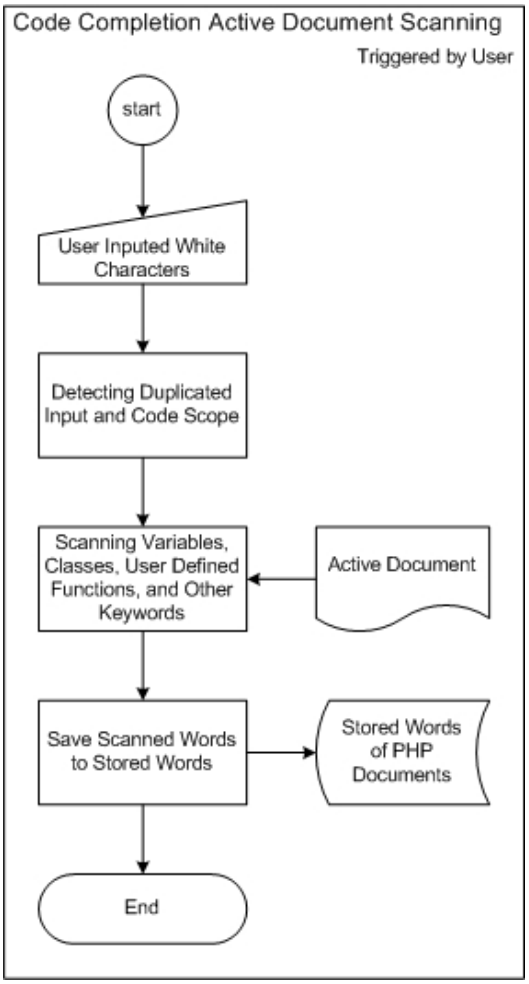
Aplikasi ini menerapkan *code completion* ini sebagai fitur yang pasti akan mempermudah pengguna dalam mengetikan kode PHP. Dibawah ini beberapa diagram alir dan penjelasannya dari cara kerja *code completion* yang akan diterapkan dalam aplikasi.

[1] http://id2.php.net/manual/en/history.php.php



Gambar *1 Code Completion Project Scanning*

Diagram diatas merupakan diagram yang menunjukan kerja *code completion* dalam men-*scan* kata kata kunci dalam level proyek. Bisa dilihat, rutinitas ini akan berjalan bila ada proyek yang dibuka, atau pengguna pindah kerja dari satu dokumen ke dokumen lainnya. Berlaku pula bila pengguna menutup atau membuka dokumen lain yang berada dalam satu proyek. Setelah keadaan tersebut, rutin ini akan memulai memeriksa dokumen – dokumen dan memasukan semua variabel, kelas, fungsi, maupun kata-kata kunci lainnya. Setelah itu memasukan semuanya kedalam penyimpanan kata sementara.



Gambar 2 Code Completion Active Document Scanning

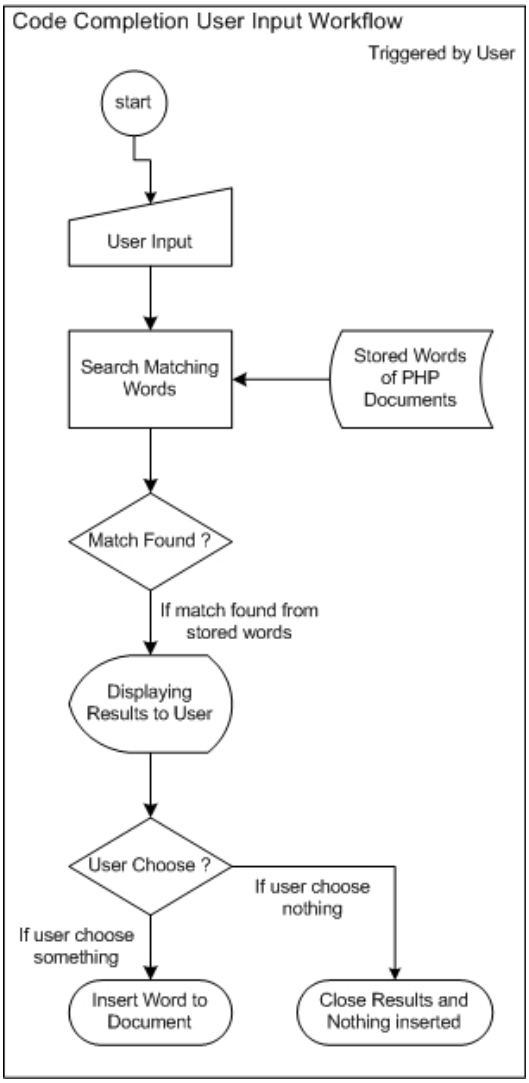
Diagram diatas merupakan diagram yang menunjukan cara kerja *code completion* dalam men-*scan* dokumen yang sedang aktif. Bila dilihat, rutin ini sebagian memiliki kesamaan fungsi dengan diagram yang sebelumnya. Hanya saja, rutinitas dimulai ketika pengguna menekan salah satu *white characters*. Kemudian rutin akan mengecek apakah penekanan tombol tombol tersebut merupakan duplikat dari yang sebelumnya atau bukan. Dan masih dalam satu fungsi yang sama, rutin juga akan mengecek keberadaan kursor, apakah dalam lingkup suatu kelas, fungsi atau global. Setelah itu baru fungsi-fungsi setelahnya sama dengan fungsi pada diagram sebelumnya.

Untuk men-*scan* dokumen, digunakan teknik *parsing* sederhana. *Parsing* merupakan teknik untuk menganalisa sekumpulan karakter atau kalimat dengan tujuan untuk mengasosiasikannya dengan suatu grup berdasarkan sintaks tertentu[2]. *Parsing* untuk *code completion* ini menggunakan *regular expression* untuk

[2] http://dictionary.reference.com/search?q=parse&x=0&y=0

membantu mencari *token* atau kata kunci yang diinginkan. *Regular Expression* adalah sekumpulan karakter dengan aturan tertentu untuk mendeskripsikan pola pencarian[3]. Untuk aplikasi yang akan dikembangkan, hanya akan dicari kata-kata kunci yang mewakili kelas *PHP*, fungsi *PHP*, variabel *PHP*, dan konstanta *PHP*. *Parser* ini tidak mempedulikan urutan dalam eksekusi kode *PHP*, karena *parser* ini hanya akan mengambil informasi dari kode *PHP* tersebut. Oleh karena itu yang terpenting adalah, informasi kepemilikan (*ownership*) bukan urutan atau *valid*-nya sebuah kata-kunci.

Parser akan berjalan untuk men-*scan* isi dokumen pada saat pengguna mengetikan kode *PHP*. Kemudian hasilnya akan disimpan.



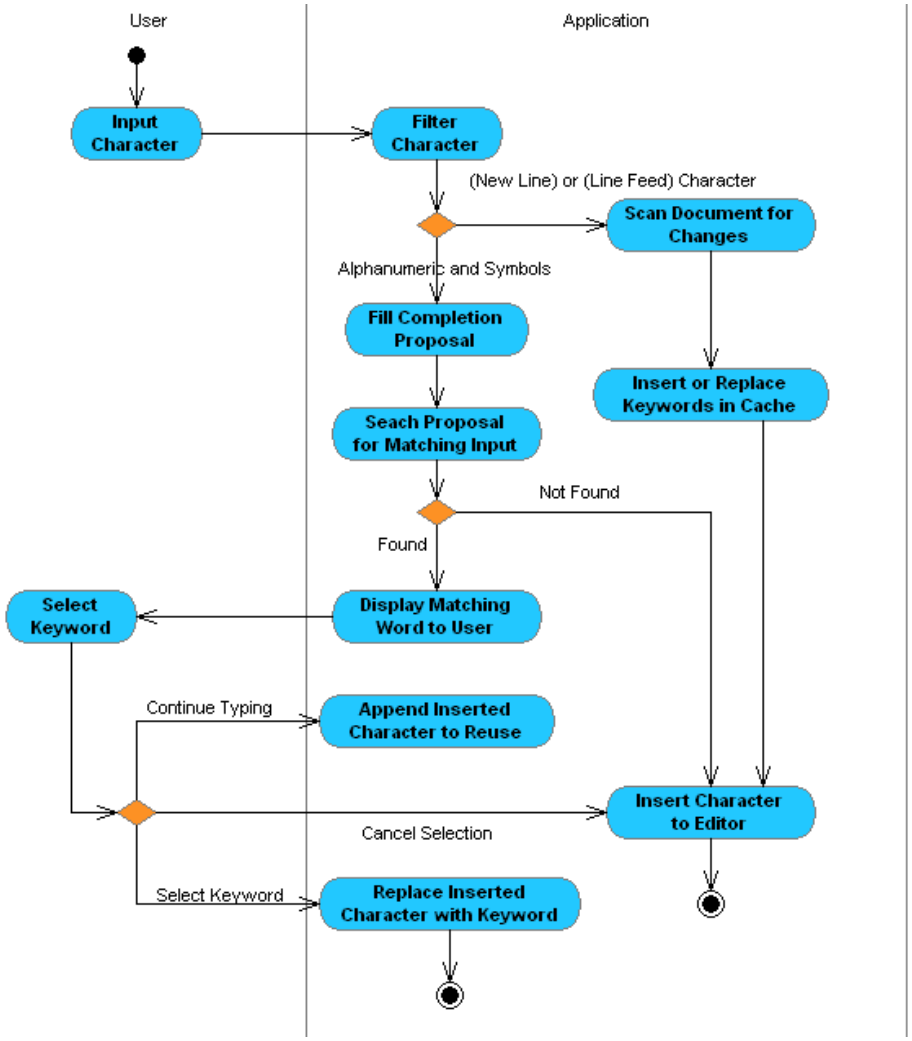
Gambar 3 Code Completion User Input Workflow

[3] http://regular-expression.info/

Diagram ini menunjukan cara kerja *code completion* bila pengguna memasukan karakter-karakter kedalam dokumen. Minimal karakter yang dibutuhkan adalah satu dan bukan *white characters*. Setelah itu, rutin akan mengambil karakter-karakter yang diketikan dan mencocokannya dengan data penyimpan kata sementara lalu mengambil semua kata yang mirip dengan yang diketikan oleh pengguna. Setelah itu, rutin ini akan menyuruh aplikasi untuk mengeluarkan kata-kata tadi untuk kemudian dipilih oleh pengguna. Bila pengguna memilih kata yang terdapat dalam hasil pencarian tadi, maka rutin akan langsung mengganti kata yang diketikan tadi dengan yang dipilih oleh pengguna. Bila pengguna tidak memilih apa-apa (dengan menekan tombol keluar atau *white characters*) maka rutin ini tidak akan melakukan perubahan pada dokumen.

4. Diagram Aktivitas

Dibawah ini merupakan diagram aktivitas untuk penggunaan konsep *code completion* ini kedalam aplikasi.



Gambar 4 Diagram Aktivitas

Aktivitas ini merupakan contoh penerapan konsep *code completion* diatas untuk aplikasi *editor PHP*. Kegiatan dimulai dari pengguna mengetikan kode-kode program. Setiap pengguna memasukan satu karakter, aplikasi akan melakukan pengecekan terhadap karakter yang dimasukan tersebut. Bila karakter merupakan *white-space* (*tab, return, space*), maka aplikasi akan melakukan *scanning* pada dokumen yang aktif. Kemudian memasukan hasilnya kedalam penyimpanan sementara. Bila karakter yang dimasukan selain *white-space*, maka aplikasi langsung mencari kata kunci yang sesuai dengan karakter tersebut. Bila ditemukan, aplikasi akan memunculkan jendela kecil untuk memberikan pilihan kepada pengguna, apa yang akan diketik sebenarnya. Bila pengguna memilih, maka karakter yang dimasukan, akan diganti dengan pilihan pengguna. Bila pengguna membatalkan pilihan, maka jendela tersebut akan menutup. Dan yang terakhir, bila pengguna melanjutkan pengetikan, maka karakter-karakter tersebut akan dicari lagi oleh aplikasi untuk ditampilkan kembali yang cocok-nya.

5. Contoh Implementasi

Sesuai dengan perancangan yang dijelaskan didalam bab III pada landasan teori, untuk menopang fitur code-completion, dibutuhkan sebuah scanner/parser untuk mencari kata-kata kunci yang dibutuhkan oleh completion proposal. Parser yang dikembangkan menggunakan teknik parsing sederhana, yaitu hanya mencari dan menyimpan. Parser untuk aplikasi ini dikembangkan dengan bentuk sebuah thread. Sehingga kerja parser tidak mengganggu kerja aplikasi. Parser ini juga menggunakan regular expression untuk mencari kata-kata kunci yang diinginkan.

Berikut dibawah ini merupakan pola regular expression yang digunakan dalam parser aplikasi ini.

Tabel 1 Pola Regular Expression

Kata Kunci	Pola Regular Expression
Require	(?<=require)([_once\(\s]+")([\.a-zA-Z0-9/_]+)
Include	(?<=include)([_once\(\s]+")([\.a-zA-Z0-9/_]+)
Konstanta	(?<=define\()([\S]+)(?=\))
Variabel	(?<=\$)([_A-Za-z0-9]+)
Fungsi	(?<=function)([_A-Za-z0-9]+)(?=\()
Kelas	(?<=class)([_A-Za-z0-9]+)

Pola-Pola tersebut dipakai untuk metoda-metoda yang dipakai untuk mencari dan menyimpan kata-kata kunci sesuai dengan pola masing-masing. Berikut merupakan contoh kode program untuk fungsi yang merekap semua fungsi untuk mencari kata-kata kunci tersebut. Kode program dibuat menggunakan *Codegear Delphi 2007* untuk aplikasi *desktop* dan menggunakan *multithread*.

```
1    function TDocumentParseParserThread.ParsePHPDocument: TPHPDocument;
```

```
 2  var I,C,SO,EO : Integer;
 3      S : String;
 4      L : TScopeStorageItems;
 5  begin
 6   Result := TPHPDocument.Create;
 7   TraceDirectives(Result.Directives);
 8   {Parse PHP Section from Document.}
 9   TracePHPSection(Result.Sections);
10   if (Result.Sections.Count > 0) then
11    for I := 0 to Result.Sections.Count - 1 do
12     begin
13      {First, scan for classes within every section.}
14      S := Result.Sections.Items[I].Info.Content;
15      SO := Result.Sections.Items[I].Info.StartOffset;
16      TraceClass(S,SO,Result.Classes);
17      {Second, scan for functions within every section, outside the class(es) scope.}
18      if (Result.Classes.Count > 0) then
19       begin{if there're function(s) within this class...}
20        for C := 0 to Result.Classes.Count do
21         begin
22          {Define offset based on function offset and class offset.}
23          if (C = 0) then SO := Result.Sections.Items[I].Info.StartOffset
24           else SO := Result.Classes.Items[C-1].Info.EndPosition;
25          if (C = Result.Classes.Count) then EO :=
    Result.Sections.Items[I].Info.EndOffset
26           else EO := Result.Classes.Items[C].Info.NameOffset;
27          {Copy string from start offset to end offset as scope.}
28          S := Copy(FLines.Text,SO + 1,EO - SO);
29          {Trace Function within the scope.}
30          TraceFunction(S,SO,'',Result.Functions);
31         end;{end for C := 0...}
32       end else begin
33        {if there's no classes within this section, use previous scope, then scan for
    functions.}
34        TraceFunction(S,SO,'',Result.Functions);
35       end; { end if (Result.Classes... else... }
36      {Third, scan for variables within every section, outside the scope of class(es)
    and function(s).}
37      S := Result.Sections.Items[I].Info.Content;
38      SO := Result.Sections.Items[I].Info.StartOffset;
39      L := Scoper(Result);
40      if (L.Count > 0) then
41       begin
42        for C := 0 to L.Count do
43         begin
44         {Define offset based on function offset and class offset.}
45          if (C = 0) then SO := Result.Sections.Items[I].Info.StartOffset
46           else SO := L.Items[C-1].EndOffset;
47          if (C = L.Count) then EO := Result.Sections.Items[I].Info.EndOffset
48           else EO := L.Items[C].StartOffset;
49          {Copy string from start offset to end offset as scope.}
50          S := Copy(FLines.Text,SO + 1,EO - SO);
51          {Trace Variables within the scope.}
52          TraceVariable(S,SO,otNoOwner,'',Result.Variables);
53         end;{end for C := 0...}
54       end else begin
55        {if there's no classes or function within this section, use previous scope, then
    scan for variables.}
56        TraceVariable(S,SO,otNoOwner,'',Result.Variables);
```

```
57        end;
58        L.Free;
59        {Fourth, scan for any inclusions within this section.Both include and require.}
60        S := Result.Sections.Items[I].Info.Content;
61        SO := Result.Sections.Items[I].Info.StartOffset;
62        EnumeratePHPInclusion(S,SO,Result.Inclusions);
63        {Fifth, scan for any defined constants within this section.}
64        EnumeratePHPConstant(S,SO,Result.Constants);
65      end; {end for I := 0...}
66    end;
```

Penjelasan dari cuplikan kode diatas adalah sebagai berikut.

Baris 6	:	Membuat objek untuk hasil *parse*.
Baris 7	:	Mengambil kata penunjuk didalam kode.
Baris 9	:	Mengambil segmen *PHP* dalam dokumen.
Baris 11	:	Memproses segmen yang terambil satu per satu.
Baris 16	:	Ambil kelas *PHP*.
Baris 18-35	:	Proses mengambil fungsi *PHP* diluar kelas.
Baris 37-57	:	Proses mengambil variabel *PHP* diluar kelas.
Baris 62	:	Mengambil *include* dan *require* dari segmen.
Baris 64	:	Mengambil konstanta *PHP* dari segmen.

6. Simpulan

Berdasarkan project yang sudah dilakukan dapat diambil beberapa simpulan, diantaranya:

1. Fitur *code completion* dapat membantu *programmer* dalam meningkatkan kenyamanan dan produktivitas ketika membuat program.
2. Dapat dibuat *code completion tool* yang mengikutsertakan kode program yang sebelumnya telah dibuat oleh pemrogram sebagai sumber kata yang dikenali, selain *keywords* dan pustaka standar dari bahasa program tujuan.
3. *Parsing* dan *regular expression* merupakan elemen yang sangat penting dalam sebuah *code completion tool*.

7. Daftar Pustaka

1. PHP: History of PHP – Manual. Retrieved July 6th 2009 from http://id2.php.net/manual/en/history.php.php

2. Parse Definition | Definition of Parse at Dictionary.com. Retrieved July 6th 2009 from http://dictionary.reference.com/search? q=parse&x=0&y=0

3. Regular-Expressions.info - Regex Tutorial, Examples and Reference - Regexp Patterns. Retrieved July 6th 2009 from http://regular-expression.info/