



**DEV2QA**

**PDT  
1.1**

**Feature Name : Code Completion  
Module name : PHPCompletionEngine**

Writer	Date	Comment	Approved
Michael Spector	6/15/2008	1. Dev2QA	
		2. Target Version - PDT 1.1	



## 1. Introduction

### 1.1 Requirement Rationale

Code Completion predicts what user wanted to type by observing word prefix and source code context, and provides a list of possible completions.

### 1.2 New Terminology

#### 1.2.1 Camel-case match

User can request code completion by providing only medial capital letters in the original PHP element, or even a string constructed from substrings that start from those capital letters.

For example:

ZF matches ZendForm  
ZeFo also matches ZendForm

#### 1.2.2 Workspace scope

This is a search scope, which is used for building PHP element proposals. This scope contains all open PHP projects including their build path (include path).

#### 1.2.3 PHP Built-in Variables

<http://php.net/reserved.variables>

#### 1.2.4 PHP magic methods

<http://php.net/oop5.magic>

### 1.2.5 Type Inference Engine

This is a mechanism that allows determine type of PHP expression (either a simple type or a complex type consisting of one or more simple types) by investigating source code as well as PHP-doc sections.

Example #1:

```
<?php
function foo() {
    if (something()) {
        return new A();
    }
    return new B();
}
?>
```

foo() has complex return type: {A, B}

Example #2:

```
<?php
/**
 * @return C
 */
function foo() {
}
?>
```

foo() has return type C

Example #3:

```
<?php
/**
 * @return C|B
 */
function foo() {
}
?>
```

foo() has complex return type: {C, B}

**!!!** More on Type Inference engine can be found by exploring Unit Tests (package *org.eclipse.php.test.headless.core.typeinference*)



---

## **1.3 Detailed Description**

### **1.3.1 Explicit Code Completion**

Code completion is shown when user presses CTRL + whitespace. The proposals shown in explicit completion contain the same list as in automatic completion.

### **1.3.2 Automatic Code Completion**

Code completion is shown automatically while user types code. The proposals shown in automatic completion contain the same list as in explicit completion.

### **1.3.3 Completion Match**

Code completion matches the user prefix either using an exact case-insensitive prefix comparison or camel-case match.

### **1.3.4 Code Completion Base**

Code completion is based on type inference engine that makes use of PHP-doc description as well as of the code.

---

## Completion Proposals

The following table defines what completion proposals will be shown depending on the current code context (where the cursor is now) and on the prefix that user has already typed.

Reminder: PHP elements defined inside of function are belonging to the global scope.

<b>Code Context</b>	<b>Typed Prefix</b>	<b>Expected Proposals</b>
Script		All keywords
Script	new Abc	All classes from workspace scope that start match 'Abc'
Script	Abc	All keywords, classes, interfaces, methods and non-class constants from workspace scope that start match 'Abc'
Script	Abc::	All static fields declared in class Abc, or in its super-class hierarchy, e.g. constants, variables, methods.
Script	Abc::XYZ	All static constants and methods declared in class Abc, or in its super-class hierarchy that match 'XYZ'.
Script	Abc::\$	All static variables declared in class Abc, or in its super-class hierarchy.
Script	Abc::\$xyz	All static variables declared in class Abc, or in its super-class hierarchy that match '\$xyz'.
Script	<i>exp</i> ->	All members declared in type(s) represented by the object referenced by <i>exp</i> expression, or in its super-class hierarchy, e.g. constants, variables, methods.

Script	<code>exp-&gt;xyz</code>	All members declared in type(s) represented by the object referenced by <code>exp</code> expression, or in its super-class hierarchy that match 'xyz', e.g. constants, variables and methods.
Script	<code>\$xyz</code>	Variables visible in current scope, e.g. function arguments, local variables, global variables (even without global declaration), <code>\$this</code> variable in case we are inside of class method and PHP built-in variables that match 'xyz'.
Class Declaration	<code>class A</code>	Two keywords: 'extends' or 'implements'. If there where a prefix, it should complete to the keyword that starts from this prefix.
Class Declaration	<code>class A implements Xyz</code>	All interfaces that match 'Xyz'
Class Declaration	<code>class A extends Xyz</code>	All classes that match 'Xyz'
Interface Declaration	<code>interface A extends Xyz</code>	All interfaces that match 'Xyz'
In class body		Nothing
In class body	<code>abc</code>	All keywords used for declaring class members, e.g. accessibility (private, public, etc...), member type (var, function, etc...), etc... that match 'abc'.

In class body	function xyz	Class magic methods, class constructor, all derived from super classes' non-private methods that match 'xyz'. Prefix 'xyz' is not mandatory – in this case all results will be available.
In method parameters	function foo(ABC	All classes or interfaces that match 'ABC'
In PHP-doc	* @	All PHP-doc tags
In PHP-doc	* @param \$	All method parameters
In PHP-doc	* @return	All method return types determined from source code.

## 2. Testing Highlights

You have to understand what the abilities of Type Inference engine are before testing Code Completion feature.

## 3. Unit Testing

Please observe existing Unit Tests from the package *org.eclipse.php.test.headless.core.codeassist*, and add more.

## 4. Future Development

- Improve completion placeholder support: when completing function call show arguments in a placeholder that can be easily edited, as in example:

```
stop(context);
```

- Add PHP 5.3 support.