



The PHP Company

Namespaces in the Wild with PHP 5.3

By Stas Malyshev and Michael Spector

Why have namespaces?

- Organize OO code into self-contained units
- Reduce conflicts
- Shorten names
 - ▶ class
Zend_Search_Lucene_Analysis_Analyzer_Common_TextNum_CaseInsensitive
extends Zend_Search_Lucene_Analysis_Analyzer_Common_TextNum
- Isolate and abstract functions, classes, etc.

What are Namespaces?

- “In the broadest definition namespaces are a way of encapsulating items. “
- designed to solve two problems when creating re-usable code:
 - ▶ Name collisions between code you create, and internal PHP classes/functions/constants or third-party classes/functions/constants.
 - ▶ Ability to alias (or shorten) Extra_Long_Names designed to alleviate the first problem, improving readability of source code.

Java

```
package illustration;

import java.awt.*;

public class Drawing {
    ...
}
```

C++

```
using namespace std;

namespace first
{
    int var = 5;
}
```

PHP

```
namespace my\name;

use your\name;

class MyClass {}
function myfunct () {}
const MYCONST = 1;
```

Namespaces Properties

- Compile-time definition
- Mostly compile-time resolution
- Many NS per file, many files per NS

First look at Namespaces

```
<?php
namespace my\name; // see "Defining Namespaces" section
use My\Full\Classname as Another;

class MyClass {}
function myfunction() {}
const MYCONST = 1;

$a = new MyClass;
$obj = new Another;
$c = new \my\name\MyClass; // see "Global Space" section

$a = strlen('hi'); // see "Using namespaces: fallback to global
                  // function/constant" section

$d = namespace\MYCONST; // see "namespace operator and __NAMESPACE__
                        // constant" section

$d = __NAMESPACE__ . '\MYCONST';
echo constant($d); // see "Namespaces and dynamic language features" section
?>
```

Introducing namespaces

```
define("MY_HTTP_GET", 1);
define("MY_HTTP_POST", 2);

class My_Http_Request {
    function __construct(
        $method =
        ZEND_HTTP_GET)
    {
    }
}

function my_http_send_request(
    My_Http_Request $request) {
}
}
```

```
namespace My\Http;

const GET = 1;
const PUT = 2;

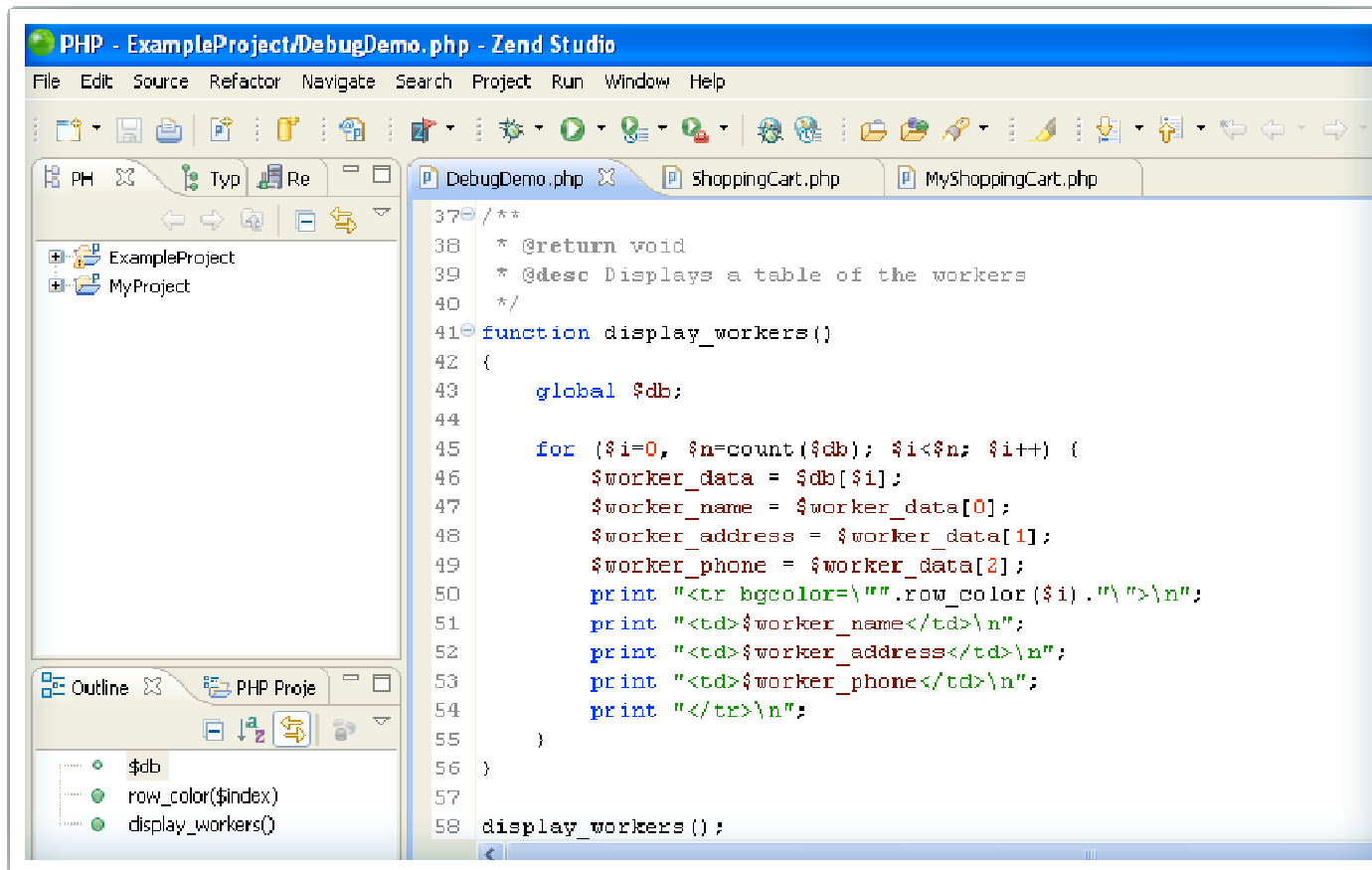
class Request {
    function __construct(
        $method = GET)
    {
    }
}

function send_request(
    Request $request) {
}
```

Should I use Namespaces?

- **Yes!** if your application is based on a framework or component that exposes namespaces
- **Probably!** if you write a PHP 5.3 framework or component that will be used by others
- **Probably not!** if you write PHP components that are unlikely to be embedded/reused by others, or if you use PHP for short scripting or for fun!
- **No!** if you use legacy code or other cases that namespaces are not required by code structure

Demo #1 – Simple usage



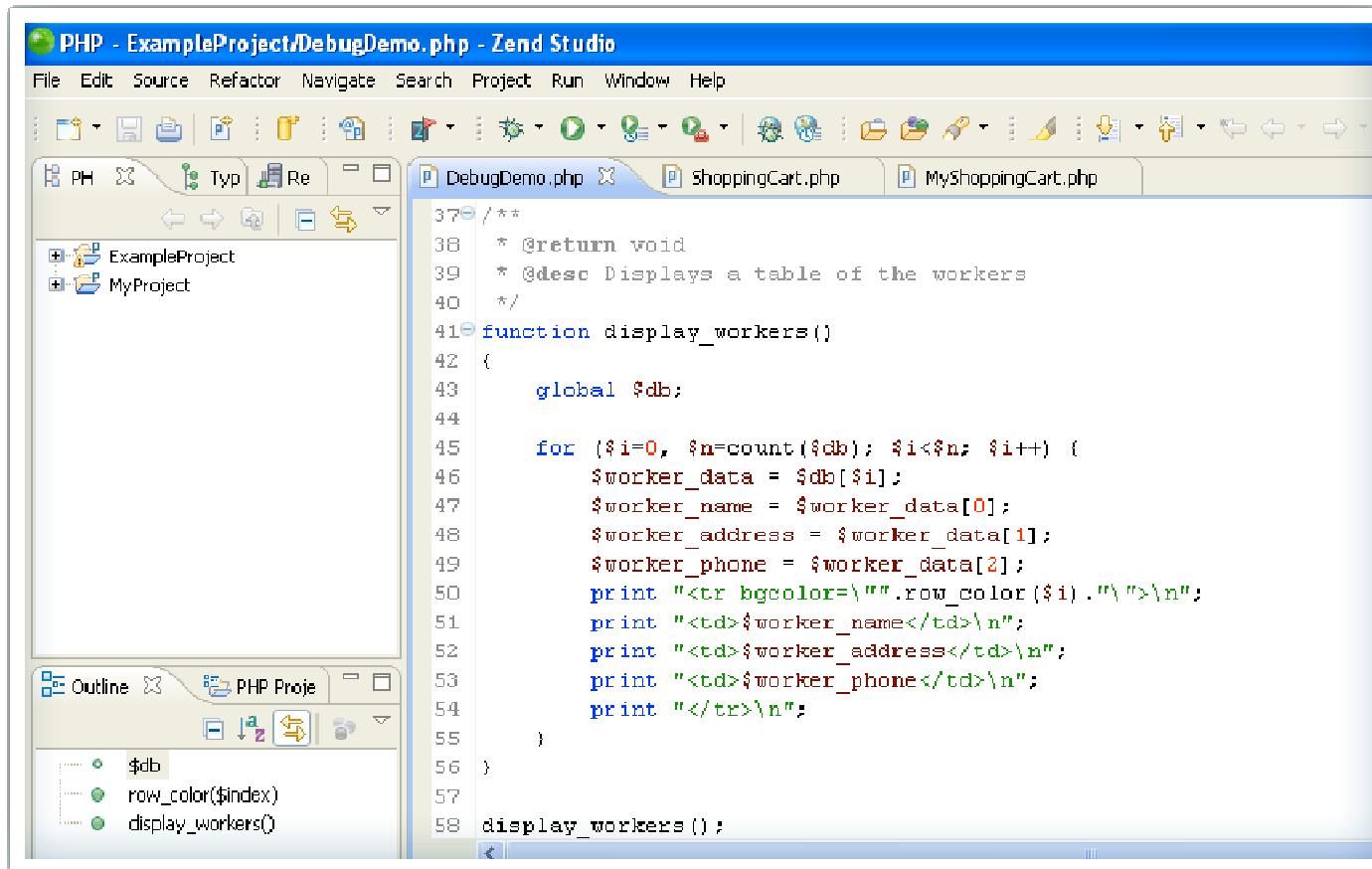
```
PHP - ExampleProject/DebugDemo.php - Zend Studio
File Edit Source Refactor Navigate Search Project Run Window Help

PH Typ Re
ExampleProject
MyProject

Outline PHP Proje
$db
row_color($index)
display_workers()

37 /**
38  * @return void
39  * @desc Displays a table of the workers
40  */
41 function display_workers()
42 {
43     global $db;
44
45     for ($i=0, $n=count($db); $i<$n; $i++) {
46         $worker_data = $db[$i];
47         $worker_name = $worker_data[0];
48         $worker_address = $worker_data[1];
49         $worker_phone = $worker_data[2];
50         print "<tr bgcolor=\"".row_color($i).\" \">\n";
51         print "<td>$worker_name</td>\n";
52         print "<td>$worker_address</td>\n";
53         print "<td>$worker_phone</td>\n";
54         print "</tr>\n";
55     }
56 }
57
58 display_workers();
```

Demo #2 – Debugging



The screenshot shows the Zend Studio IDE interface. The main editor window displays the following PHP code:

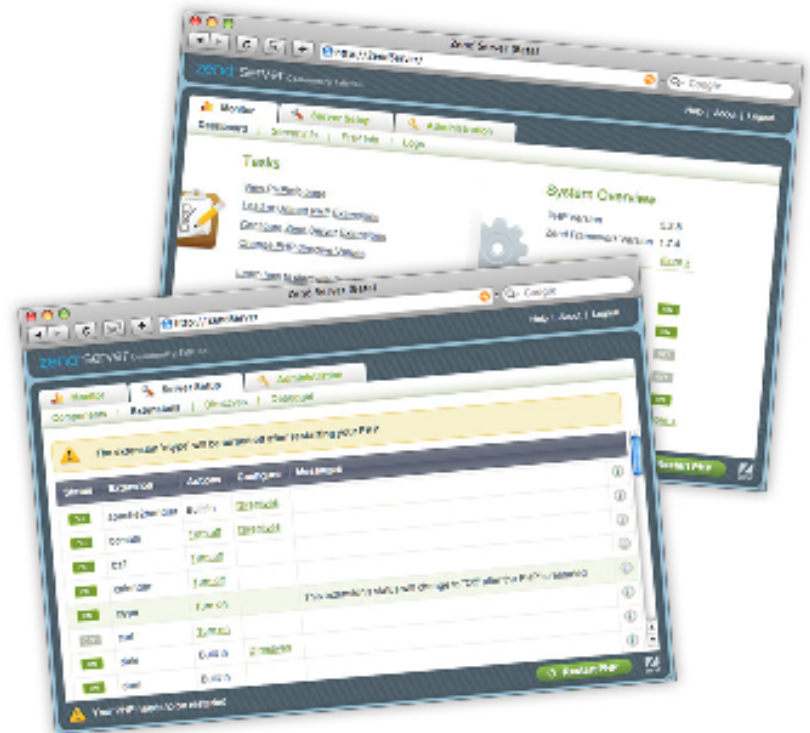
```
37 /**
38  * @return void
39  * @desc Displays a table of the workers
40  */
41 function display_workers()
42 {
43     global $db;
44
45     for ($i=0, $n=count($db); $i<$n; $i++) {
46         $worker_data = $db[$i];
47         $worker_name = $worker_data[0];
48         $worker_address = $worker_data[1];
49         $worker_phone = $worker_data[2];
50         print "<tr bgcolor=\"\".row_color($i).\" \">\n";
51         print "<td>$worker_name</td>\n";
52         print "<td>$worker_address</td>\n";
53         print "<td>$worker_phone</td>\n";
54         print "</tr>\n";
55     }
56 }
57
58 display_workers();
```

The Outline view on the left shows the following structure:

- \$db
- row_color(\$index)
- display_workers()

Supporting Products

- **Zend Server Community Edition**
 - ▶ Including PHP 5.3
- **Zend Studio and Eclipse PDT 2.1**
 - ▶ New syntax highlighter
 - ▶ Support new language features in PHP 5.3, such as namespaces and closures
 - ▶ Navigation and Exploration of namespaces
 - ▶ Code assist based on new type inference
 - ▶ Hyperlinks to namespaces or classes
 - ▶ PHP 5.3 debugging information



Recap and takeaways

- Namespaces are here to solve name collisions and to shorten elements names
- How to use (or not-use) namespaces
- There are many new things in PHP 5.3 that you should read about!

Q&A