

PHP - An Introduction

Wim Molenberghs

Agenda

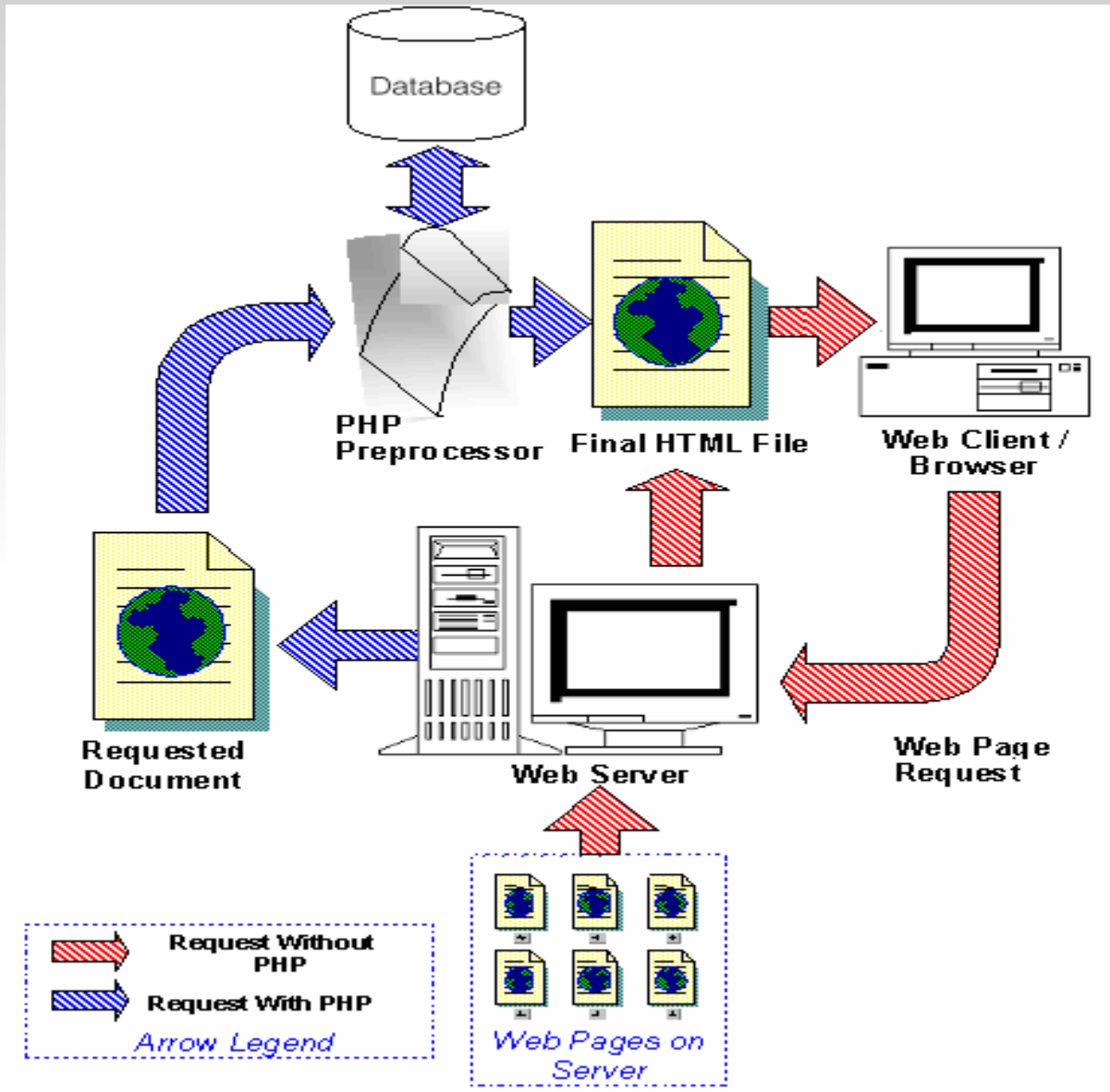
- What is PHP
- Getting Started
- OO in PHP
- Popular tools
- More info
- Q & A

History of PHP

- 1995: PHP/FI was created by Rasmus Lerdorf
- 1997: PHP/FI 2.0
- 1997: PHP 3 was introduced by Zeev Suraski and Andi Gutmans
- 1998: PHP 4 was released
- 2000: PHP 4 with Zend Engine
- 2004: PHP 5 with Zend Engine 2.0

What is PHP

- PHP (PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML
- Code is executed on the server
- Extremely simple for a newcomer, but offers many advanced features for a professional programmer



Types

- PHP supports eight primitive types
 - scalar types: boolean, integer, float (aka double), string
 - compound types: array, object
 - special types: resource, NULL
- PHP does not require or support explicit type definition in variable declaration

```
$foo = "0";           // $foo is string (ASCII 48)
$foo += 2;           // $foo is now an integer (2)
$foo = $foo + 1.3;   // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";   // $foo is integer (15)
```

```
$bar = 'car';        // $bar is string car      $bar[0] = 'b';     // $bar is string bar
```

Variables

- Are represented by a dollar sign followed by the name of the variable
- The names are case-sensitive
- Has to start with a letter or an underscore

```
$foo = "0";
```

```
$_foo = 2;
```

- By default, assigned by value
- Assign by reference, prepend an ampersand to the beginning of the variable

```
$foo = "0";
```

```
$bar = &$foo;
```

- The scope of the variable is the context within which is defined

Predefined variables

- PHP provides a large number of predefined variables to any script which it runs.
- The variables represent everything from external variables to built-in environment variables, last error messages to last retrieved headers
- Most of the variables can be viewed with the phpinfo function.

PHP Variables

Variable	Value
PHP_SELF	/phpinfo.php
_REQUEST["test"]	ctg
_GET["test"]	ctg
_SERVER["HTTP_HOST"]	zelhory.schkocr.cz
_SERVER["HTTP_USER_AGENT"]	Mozilla/5.0 (Windows; U; Windows NT 5.1; nl; rv:1.9.0.8) Gecko/2009032609 Firefox/3.0.8
_SERVER["HTTP_ACCEPT"]	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
_SERVER["HTTP_ACCEPT_LANGUAGE"]	en,en-us;q=0.7,nl;q=0.3
_SERVER["HTTP_ACCEPT_ENCODING"]	gzip,deflate
_SERVER["HTTP_ACCEPT_CHARSET"]	ISO-8859-1,utf-8;q=0.7,*;q=0.7
_SERVER["HTTP_KEEP_ALIVE"]	300
_SERVER["HTTP_CONNECTION"]	keep-alive
_SERVER["PATH"]	(\bin;\bin;\usr\bin;\usr\bin;\usr\local\bin

Built-in predefined variables

- `$GLOBALS` Variables available in global scope
- `$_SERVER` Server & execution env. information
- `$_GET` HTTP GET variables
- `$_POST` HTTP POST variables
- `$_FILES` HTTP File upload variables
- `$_REQUEST` HTTP REQUEST variables
- `$_SESSION` Session variables
- `$_ENV` Environment variables
- `$_COOKIE` HTTP Cookies

Built-in predefined variables example

index.php:

```
<?php
    session_start();
    $_SESSION['sessionvar'] = 'Session variable';
?>
<form method="get" action="test.php">
    <input type="input" name="getVar" />
    <input type="submit" />
</form>
```

test.php:

```
<?php
    session_start();
    echo $_SESSION['sessionvar'];           //Prints Session variable
    echo $_SERVER['SERVER_NAME'];          //Prints localhost
    echo $_GET['getVar'];                  //Prints getVar parameter
?>
```

Variables - scope

```
<?php
function test(){
    echo $a;                //Error Undefined variable
    echo $GLOBALS['a'];    //Prints 1
    $c = 9;
    global $d;
    $d = 8;
    $GLOBALS['e'] = 8;
}

$a =1;
test();
echo $c; //Error Undefined variable
echo $d; //Prints 8
echo $e; //Prints 8
?>
```

Variable variables

- A variable name is a name which can be set dynamically

```
$a = 'Hello';
```

```
$$a = 'world';
```

```
echo "$a $hello"; //Prints Hello World
```

Operators and Control Structures

- Comparison operators
 - Difference in strict comparison(===) and loose comparison(==)

```
echo false == 0 ? 'true' : 'false'; //Prints true
```

```
echo false === 0 ? 'true' : 'false'; //Prints false
```

- Switch structure allows usage of strings
- PHP supports the goto operator

```
goto a;  
echo 'Foo';  
a:  
echo 'Bar'; //outputs Bar
```

Functions

- Any type may be returned by a function

```
function sum($a,$b){  
    return $a + $b;  
}  
echo sum(1,2);
```

- Support for variable functions

```
function foo(){  
    echo "inside foo";  
}  
$func = 'foo';  
$func();           //Outputs inside foo
```

Classes and Objects

- Every class definition begins with the keyword `class`, followed by a name.
- The functions inside a class can be called by the `->` operator.

```
class Foo {  
    public function displayVar() {  
        echo "a default value";  
    }  
}
```

```
$foo = new Foo();  
$foo->displayVar();    //Displays ' a default value'
```

Constructors and Destructors

- Constructors and destructors can be called with the magic keywords `__construct` and `__destruct`

```
class Foo {  
    public function displayVar() {  
        echo "a default value";  
    }  
    function __construct(){  
        echo "inside constructor";  
    }  
    function __destruct(){  
        echo "inside destructor";  
    }  
}
```

```
$foo = new Foo();  
$foo->displayVar(); /* Displays inside constructor, a default value and  
    inside destructor */
```

Visibility

- The visibility of properties or methods in a class can be declared with the keywords `public`, `protected` or `private`
- Methods without any declaration are defined as `public`

Scope Resolution operator (::)

- Is a token that allows access to static, constant and overridden members or methods of a class

```
class Foo {  
    const VALUE = 'A constant';  
}  
class Bar extends Foo{  
    public static $my_static = 'static var';  
    public static function doubleColon(){  
        echo parent::VALUE;  
        echo self::$my_static;  
    }  
}  
echo Bar::doubleColon(); //Displays A constant and static var
```

Abstract classes and interfaces

- It is not allowed to create an instance of a class that has been defined as abstract.
- Any class that contains at least one abstract method must also be abstract.
- When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child.
- A class can extend only one class but implements many interfaces

Abstract classes and interfaces

```
abstract class AbstractClass{
    abstract function getValue();
    public function printOut(){
        print 'function inside abstract class';
    }
}
interface Iface{
    public function foo();
}
class Bar extends AbstractClass implements Iface{
    public function getValue(){
        echo 'inside getValue';
    }
    public function foo(){
        echo 'inside foo';
    }
}
```

Magic Methods

- PHP has some Magic Methods
 - `__construct` Constructor for classes
 - `__destruct` Destructor for classes
 - `__call` Triggered when invoking inaccessible methods
 - `__callStatic` Triggered when invoking inaccessible static methods
 - `__get` Utilized for reading data from inaccessible members
 - `__set` Utilized for writing data from inaccessible members
 - `__isset` Calling `isset()` or `empty()` on inaccessible members
 - `__unset` When `unset()` is used on inaccessible members
 - `__sleep` Executed prior to any serialization
 - `__wakeup` Executed with `unserialize`
 - `__toString` Convert a class to a string
 - `__invoke` Called when a script tries to call an object as function
 - `__set_state` Static method exported by `var_export`
 - `__clone` Executed after `clone()` method

Reflection

- PHP 5 comes with a complete reflection API that adds the ability to reverse-engineer classes, interfaces, functions and methods as well as extensions

```
class Foo {  
    function counter(){  
    }  
}
```

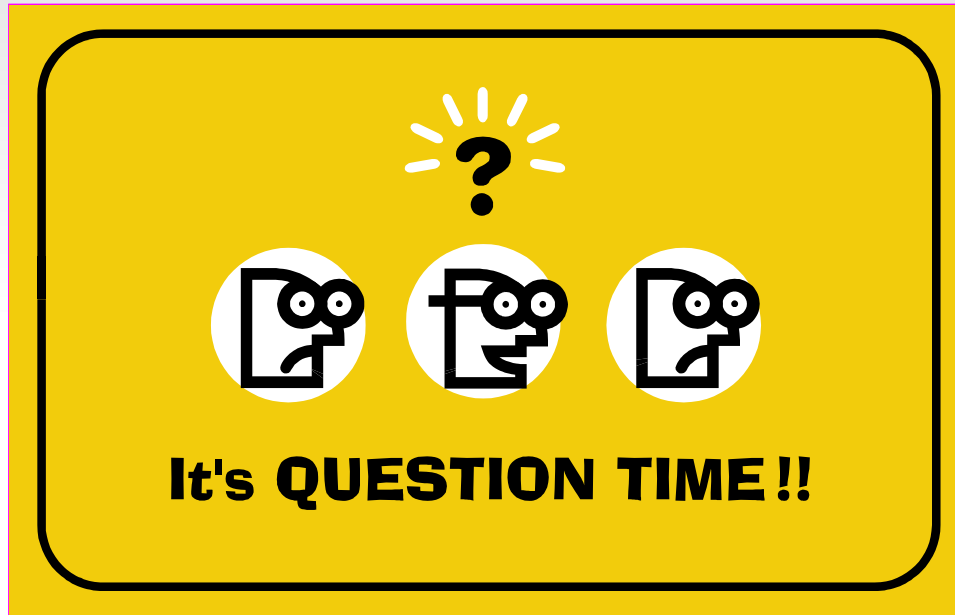
```
$bar = new ReflectionClass('Foo');  
echo $bar->getName(); //Displays Foo  
echo $bar->hasMethod('counter'); //Displays 1
```

Popular PHP tools

- Frameworks
 - Zend
 - CakePHP
 - Symfony
 - Seagull
- CMS
 - Drupal
 - Joomla
 - Typo 3
 - Nucleus
- Database
 - phpMYAdmin
- Blogs
 - Wordpress
 - Serendipity
- E-commerce
 - osCommerce
 - Magento
- Forums
 - phpBB
- Wiki
 - MediaWiki

More info

- PHP Home page
<http://www.php.net>
- Zend
<http://www.zend.com>



Thank You!

Questions/Comments

wim AT eek.be