PHP Best Practices

Conventions, Documentation, Security, Performance

Mario Peshev
Technical Trainer
http://peshev.net/
Software University
http://softuni.bg

Table of Contents

1. Writing conventions
2. Type safe code
3. Exceptions, being E_STRICT
4. Documentation
5. Security
6. Performance
7. Deployment

Writing conventions

- Write your code so you and others can read and understand it
- Don't invent your own standards and conventions
- Use established styles
- Use naming conventions
- Example – PEAR Coding Standards:

  http://pear.php.net/manual/en/standards.php

Writing conventions (2)

- Consider converting **underscores(_)** to **slashes(/)** when packaging classes:

```
Spreadsheets_Excel_Writer.php => Spreadsheets/Excel/Writer.php
```

- Use **PascalCase** for class names:

```
CatchingExceptions, ExcelWriter
```

- Name variables with **camelCase**:

```
$number, $isEqual, $currentWord
```

Writing conventions (3)

- Name constants with **ALL_CAPS**:

```
ABSOLUTE_ZERO, MIN_VALUE, MAX_VALUE
```

- Prefix private methods and properties of classes with **underscore(_)**

```
Private function _findMaxValue()
```

- Use four spaces instead of tabs to indent the code

Type-safe coding

- PHP is loosely typed language
  - May lead to unexpected results and errors
  - Be careful when using normal comparison operators
    - Replace with type-safe where needed

```
== becomes ===                    != becomes !==
```

  - Use type casting and explicit type conversions

```
$str = “2”;
$intgr = (int) $str;
```

Short open tags

- `<?`, `<?=`, `<%` are short opening tags and are being deprecated
  - `<?` is XML opening tag
  - `<?=` is complete invalid XML
  - `<%` is ASP style tag
  - If there is code in more than one language in one file, short open tags may lead to confusion of parsers
- Use `<?php` for opening tag

Exceptions

- Handling exceptions and warnings is cool but dangerous
  - Exceptions can lead to more problems than solutions
    - Use only when really needed
  - Exceptions may leak memory

```
for ($i = 10000; $i > 0; $i --) {
        throw new Exception ('I Leak Memory!');
}
```

    - The memory, allocated for the for-loop does not get freed

Being E_STRICT

- A lot of functions are being deprecated
- In PHP 5 using certain functions will raise E_STRICT error
  - In PHP 6 those will become E_FATAL
- Example:
  - Function `is_a` is deprecated

```
if (is_a($obj, 'FooClass')) $obj->foo();
```

  - Use `instanceof` instead

```
if ($obj instanceof 'FooClass') $obj->foo();
```

Source Documentation

- phpDocumentor is a tool for generating documentation
  http://www.phpdoc.org/
- phpDocumentor and its tags are similar to Javadoc
  - Standard for generating documentation
  - Describes functions and classes, parameters and return values
  - Different tools use them to generate code-completion, technical documentation, etc.

Source Documentation – Example

- phpDocumentor tags

```
/**
* MyClass description
*
* @category MyClasses
* @package MyBaseClasses
* @copyright Copyright © 2008 LockSoft
* @license GPL
**/
```

Follow to next page

Source Documentation – Example

SOFTWARE UNIVERSITY FOUNDATION

```
class MyClass extends BaseClass {
        /**
        * Easily return the value 1
        *
        * Call this function with whatever parameters
        * you want – it will always return 1
        *
        * @param string $name The name parameter
        * @return int The return value
        **/
        protected function foo ($name) {
                return 1;
        }
}
```

Security

- Never use variables that may not be initialized

```
$a = true; $b = false;
if ($a == $b)  $login = true;
echo $login
```

- Never trust the user input
  - Always be careful about the content of:

    $_POST, $_GET, $_COOKIE

  - Use white list of possible values

Security

- Always hide errors and any output that may contain system information
  - Knowledge about paths and extensions may make it easier to exploit the system
  - Never leave `phpinfo()` calls
  - Turn off `display_errors` on deployment server
  - Turn off `expose_php`

Security

- Check file access rights
  - No writeable and executable files should be kept in the web root
  - No writeable PHP files
  - Block access to files that contain configuration on a file system level
  - Never give permission to OS user accounts if they do not need access

Security

- Always check for and turn off magic quotes
  - Use **addslashes** and other escaping functions

```
$str = addslashes('We are studying in "Software University"!');
echo($str);
```

  - Pay special attention to user input that goes into SQL statements
    - Consider using prepared statements
- Always check for and turn off **register_globals**

Performance

- PHP internal function are much faster than user functions
  - They are inbuilt and coded in C
  - Read the manual, so you don't reinvent the wheel

    http://php.net/docs.php
  - If you have slow functions, consider writing them in C

    and adding them as extensions to PHP

Performance

- Simple optimizations save a lot time
  - Use `echo` with multiple parameters instead of multiple calls or concatenation

```
//slow
echo "Hello";
echo " ";
echo $world;
```

```
//fast
echo "Hello ", $world;
```

  - Optimize loops

```
//slow
for ($i = 0; $i < count($arr); $i++) echo $i;
//fast
for ($i = 0, $n = count($arr); $i<$n; ++$i) echo $i;
```

Performance

- Keep objects and classes in limit
  - PHP 5 adds cool OO features
  - Each object consumes a lot of memory
  - Method call and property access take twice more time than calling function and accessing variable
  - Do not implement classes for everything, consider using arrays
  - Don't split the methods too much

Performance

- Most content is static content
  - Always check your site with online tools
    - Yslow – https://developer.yahoo.com/yslow/
  - Apply caching for all the static content
  - Use `Last-Modified` for database content with the date of the recorded last update
- Consider using PHP accelerators
  - http://en.wikipedia.org/wiki/List_of_PHP_accelerators
  - Compiles the code and uses it instead, until source file changes

Performance

- Use `mod_gzip` when you can afford it
  - Consumes a lot CPU, because it compresses the data on the fly
  - Saves up to 80% data transfer
  - Be careful – some browsers may have issues if some file formats are delivered with `gzip` compression
    - Example: Internet Explorer 6 and PDF

Performance

- Think twice before using regular expressions
  - They take a lot of time because of the back tracking
  - Check if it can be optimized with possessive operators and non-capturing groups
  - If you want to validate data, use **filter**, instead of **preg**

```
//slow
if (preg_match("/[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}/",
        $str)) { echo "regex match"; }
//fast
if (filter_var($str,FILTER_VALIDATE_EMAIL))
        { echo "filter match"; }
```

Design Patterns

- Always check what is out there
  - PEAR, Zend Framework and others are proven
    - Issues have been cleared
    - Object Oriented, slower
- Use standard architectures like MVC
  - Strip the database abstraction layer and object from the core logic and the view (the HTML files)

Deployment

SOFTWARE UNIVERSITY FOUNDATION

- NEVER edit files on a production server, live site or system
  - Use source repositories with versions and deployment tags
  - When developing, use development server
    - Must match the production server in parameters, or
    - Run a staging server that mimics the deployment environment
      - Deploy there for testers

Deployment

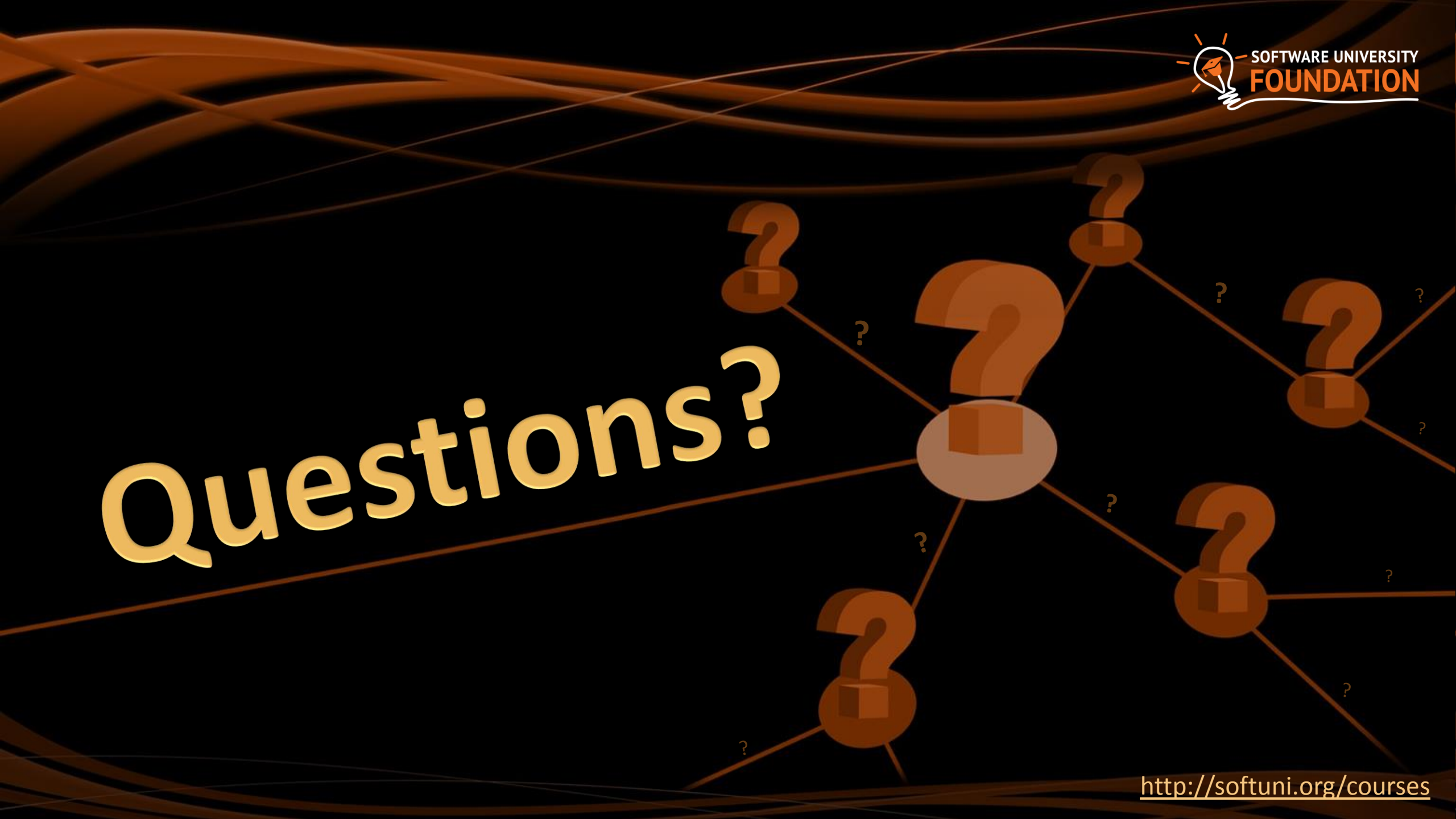
- Never override files on the server
  - Use `symlinks`
    - create a separate directory with the new files
    - link to it with `symlink()`
- Never manually interact with the server
  - Write a script that deploys the files without human interaction
- Always run a second test on the deployed project

Summary

- Writing conventions
- Type safe code
- Exceptions, being E_STRICT
- Documentation
- Security
- Performance
- Deployment

SOFTWARE UNIVERSITY FOUNDATION

Questions?



http://softuni.org/courses

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "C# Part I" course by Telerik Academy under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg