PHPotter – Doing magic with PHP

International PHP Conference Spring Edition

Ludwigsburg, 22nd of May 2007

eZ® The information sharing company

Agenda

Virtual properties

Fluent interface

If “if” wasn't “if”

Lambda style

call_user_func_array()

SPL – if time left...

About us – Kore

Kore Nordmann

- Studying computer science at the University Dortmund
- Working for eZ systems on eZ components
- Maintainer and Developer in multiple open source projects: Image_3D, KaForkl, KPortal, Busimess, PHPUnit

About us – Toby

- Tobias Schlitt
  - IT specialist
  - Studying computer science at the University Dortmund
  - Working for eZ systems on the eZ components project
  - Contributer to multiple open source projects: PHPUnit, Serendipity, Image_3D,...

Virtual properties 1/3

- The problem
  - PHP is loosely typed
    - Every variable may contain any type at any time
    - Relying on attribute values is dangerous
  - Nonexistent attributes
    - Notice on get access
    - Silently created on set access

Virtual properties 2/3

- What could be done to fix these problems?
  - Check on use
    - Need to be in place everywhere
    - Dangerous to be forgotten
  - Getter / Setter methods
    - Commonly known from Java
    - Need 2 methods for each attribute
    - Unintuitive API
    - Much more to type

Virtual properties 3/3

- The solution: Virtual properties
  - No more direct access to properties
  - Use overloading
    - __get($propertyName)
    - __set($propertyName, $propertyValue)
    - __isset($propertyName);
    - __unset($propertyName);

eZ®
The information sharing company

Code browsing 1/6

Tiny configuration class

Fluent interfaces

- PHP community hype in 2006
- Can produce nice APIs
- Often simply adds complexity
- Work with “chaining” methods
  - Key mechanism is returning $this

Code browsing 2/6

Very simple SQL query abstraction

If “if” wasn't “if”

- Stacked conditional statements are...
  - ... unreadable
  - ... not nice to type
- So, what are the alternatives?
  - Ternary operator
  - switch statement
- Use at your own risk...

Ternary operator 1/2

- $foo ? “foo” : “not foo”;
- Only operator which takes 3 arguments
- “Shortcut” language construct for if statement
  - If statements don't return anything!
- Often used as “ifsetor”
  - $foo = isset($foo) ? $foo : “default”;

Ternary operator 2/2

- PHP manual says:
  - *Is is recommended that you avoid "stacking" ternary expressions.*
- But you can...
  - Pay attention for proper bracing
  - Code indentation is important here!

Switch

- Only compares 2 expressions
  - Head
  - Case
- Makes a nice shortcut
  - switch (true) ...
- Allows to execute more statements with one condition

Code browsing 3/X

Some funny examples...

Lambda style 1/2

- Anonymous functions
- Commonly known from functional programming languages
- In PHP: create_function()
  - Returns a unique name for the function
  - Stored in a variable

Lambda style 2/2

- Pros
  - Easily create functions on the fly
  - Reduces pollution of global function name space
  - You can play funny games with it
- Cons
  - The little brother of eval()
  - No source code highlighting
  - Lots of escaping needed

Code browsing 4/6

Some small examples

call_user_func_array()

- PHP allows variable functions
  - $func($param1, $param2);
- Problem: Variable parameter count
- Solution
  - call_user_func_array()
  - Submit an array of parameters

Code browsing 5/6

A real world example

SPL

- Standard PHP Library
- Extension since PHP 5
- Default enabled
- Contains a lot of OOP stuff
  - Interfaces
  - Classes
  - ...

Iterator

- Interface Iterator in SPL
  - Can be implemented by user space classes
  - Allows you to deal with an object in foreach

Iterators

- Iterators themselves
  - DirectoryIterator
- Recursive iterators
  - RecursiveDirectoryIterator
- Outer iterators
  - FilterIterator
  - RegexIterator
  - RecursiveIteratorIterator

Code browsing 6/6

Iterators in action

The end

Thanks for listening!

We hope you heard what you expected

Any questions left?

Contact us:

- kore@php.net
- toby@php.net