

The PHP OO candy store

An introduction to PHP5s cool OO features

International PHP Conference, Frankfurt, 2006-11-07

Kore Nordmann <kn@ez.no>
Tobias Schlitt <ts@ez.no>

0.0) Speaker - Toby

Tobias Schlitt

IT specialist

Currently studying computer science

Working for eZ systems on the
eZ components project

Member of the PEAR project

Member of the Zend Certification Board

0.1) Speaker - Kore

Kore Nordmann

Studies computer science at the University
Dortmund

Working as a software developer for eZ
systems on
eZ components and eZ publish

Maintainer and Developer of Image_3D
PHPUnit developer

Agenda

Introduction

Property overloading

Object comparison

Object or Array?

Fluent Interfaces

1.0) Background

Scope model for properties since PHP 5

Private/protected properties disable
external property access

Unknown class properties

Silently created on write access

E_NOTICE on read access

Setters and getters offer

Value checks on the fly

Read-/write-only properties

1.1) Property overloading

Allows to wrap around unknown properties

`__get($name)`

Wraps property read access

`__set($name, $value)`

Wraps property write access

Methods called for not accessible properties

1.2) Code

Example code

Tiny configuration class

1.3) More magic

Create objects on write access

Example: ezcConsoleTable

Clone objects on write access

Example: ezcGraph

`__isset()` and `__unset()`

Not implemented will cause no notices or other errors

1.4) Documenting

Simulated properties were impossible to document

New doc tags for phpDocumentor

Defined on class level

`@property <type> $name <desc>`

Documents a read/write property

`@property-read ...`

Documents a read-only property

`@property-write ...`

Documents a write-property

Agenda

Introduction

Property overloading

Object comparision

Object or Array?

Fluent Interfaces

2.0) Object comparison

==, and === work on objects, too

== checks for...

- ...same class of checked instances

- ...same property values (no type check)

=== checks for

- Reference to the exactly same object

2.1) Object comparison in PHPUnit

assertEquals() accepts delta in tests

assertEquals() is used for non typesafe
checks on variable equality

DualIterator used to compare arrays

PHPUnit reimplements synchronus
iteration on two complex datatypes for
delta comparison

2.2) Code

Example code

Object comparison

2.3) Object storage

Class from ext/SPL

“Standard PHP Library”

Check for object repetition in recursive algorithms

Allows storing and processing of a set of objects

2.4) Code

Example code

Process multiple mail recipients, but do not send 2 times to the same recipient

Agenda

Introduction

Property overloading

Object comparison

Object or Array?

Fluent Interfaces

3.0) Object or Array?

Interfaces allow objects to be handled almost as arrays

Found in ext/SPL

“Standard PHP Library”

Available interfaces:

ArrayAccess

Iterator

Countable

3.1) **ArrayAccess**

Allows the usage of [] syntax on objects:

```
$object["foo"]
```

Methods defined by the interface:

```
offsetExists()
```

```
offsetGet()
```

```
offsetSet()
```

```
offsetUnset()
```

3.2) Code

Example code:

Simple table structure

3.3) Iterator

Allows to throw an object into foreach:

```
foreach ( $obj as $key => $value )
```

Methods defined by the interface:

```
current()
```

```
key()
```

```
next()
```

```
rewind()
```

```
valid()
```

3.4) Code

Example code:

Enhanced table structure

3.5) Countable

Allows to use count() on objects:

```
count( $obj )
```

Methods defined by the interface:

```
count()
```

3.6) Code

Example code:

Enhanced table structure

Agenda

Introduction

Property overloading

Object comparision

Object or Array?

Fluent Interfaces

4.0) Fluent interfaces

Buzzword in PHP scene since January

Make sense in very few situations for nice APIs

Often simply adds complexity

4.1) What are fluent interfaces?

Methods return the modified object

Works since PHP 5 due to automatic dereferencing

4.2) Code

Example code:

Building SQL queries

5.0) Conclusion

Great set of classes and interfaces to improve your APIs

Fun to create APIs using these design methods

Pitfalls in implementation:

`reset($obj)`

does not work as expected!

5.1) Open part

Questions?

Feedback?

Ideas?

Critics?

....

5.2) The bitter end...

Thanks for listening!

I hope you...

... learned what you expected.

... had an interesting time.

... possible slept well? ;)

More Information can be found here:

The PHP website

<http://php.net>

The eZ components website

http://ez.no/products/ez_components

Talk available

<http://schlitt.info>

<http://kore-nordmann.de>

Our email addresses:

Kore Nordmann <kn@ez.no>

Tobias Schlitt <ts@ez.no>