

# PHP meets UML

Marcus Börger

# Overview

- ☑ PHP 5 vs. PHP 4
- ☑ PHP 5 OO
- ☑ PHP and UML



# PHP 4 and OO ?



## Poor Object model

### ✓ Methods

- ✗ No visibility
- ✗ No abstracts, No final
- ✗ Static without declaration

### ✓ Properties

- ✗ No default values
- ✗ No static properties

### ✓ Inheritance

- ✗ No abstract, final inheritance, no interfaces

### ✓ Object handling

- ✗ Copied by value
- ✗ No destructors

# PHP 5's revamped OO Model

- ☑ PHP 5 has really good OO
  - ☑ Better code reuse
  - ☑ Better for team development
  - ☑ Easier to refactor
  - ☑ Some patterns lead to much more efficient code
  - ☑ Fits better in marketing scenarios

# Completely new XML support

- ☑ PHP 4's XML was pathetic
  - ☑ SAX was OK
  - ☑ DOM was crappy, DOM was fake
  - ☑ There was nothing else
  
- ☑ PHP 5's XML is brilliant
  - ☑ SAX is OK
  - ☑ DOM is functional
  - ☑ SimpleXML is the solution to all your problems
  - ☑ Native SOAP support



# Reflection API

- ☑ Can reflect nearly all aspects of your PHP code
  - ☑ Functions
  - ☑ Classes, Methods, Properties
  - ☑ Extensions

```
<?php
class Foo {
    public $prop;
    function Func($name) {
        echo "Hello $name";
    }
}
ReflectionClass::export('Foo');
ReflectionObject::export(new Foo);
ReflectionMethod::export('Foo', 'Func');
ReflectionProperty::export('Foo', 'prop');
ReflectionExtension::export('standard');
?>
```

# Reflection API

- ☑ Can be used to generate XMI
- ☑ Can be used to reverse engineer PHP

# Unified Database API

- ☑ PHP 5.1 will come with PDO
  - ☑ Unified object oriented API
  - ☑ Support for MSSql, MySQL, Oracle, Postgres, ...
  - ☑ Support for transactions
  - ☑ Support for LOBs
  - ☑ Support for iterators



# Why UML

- ☑ Improve Communication
  - ☑ With customer
  - ☑ With management
  - ☑ Between developers
- ☑ Visualize Concepts
- ☑ Improve documentation
- ☑ Model verification
  - ☑ Mistakes during design are cheap
  - ☑ Mistakes during implementation are expensive
- ☑ Automatic code generation

# PHP and UML

- ☑ Support for OO syntax/semantic
- ☑ Support through open source tools
  - ☑ Doxygen
  - ☑ Umbrello
  - ☑ ArgoUML
- ☑ Support through commercial/proprietary tools
  - ☑ Poseidon
  - ☑ Waterproof UML
  - ☑ BITPlan smartGENERATOR
  - ☑ Sybase PowerDesigner

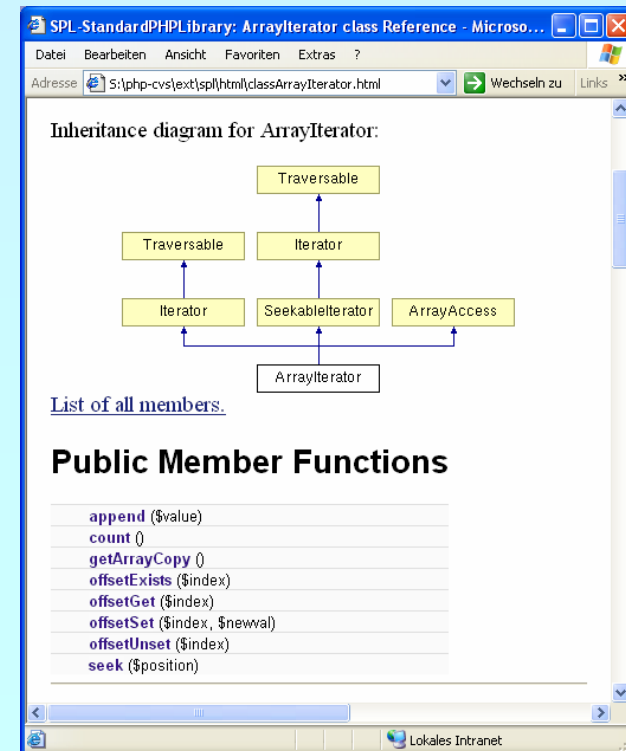
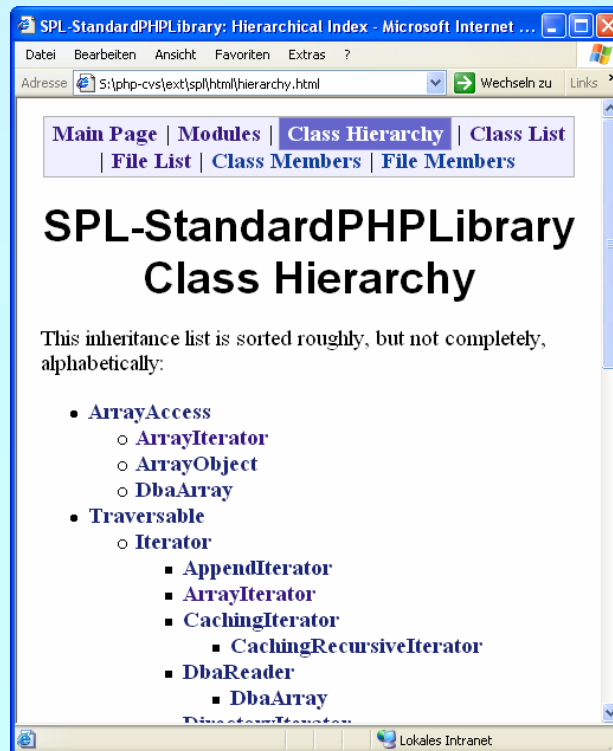


# Doxygen



Doxygen is a documentation system

- ☑ Code comments to documentation
- ☑ Creates UML like graphs automatically
- ☑ Generates html, chm, TEX, PostScript, PDF, man pages
- ☑ Runs on Linux/Windows

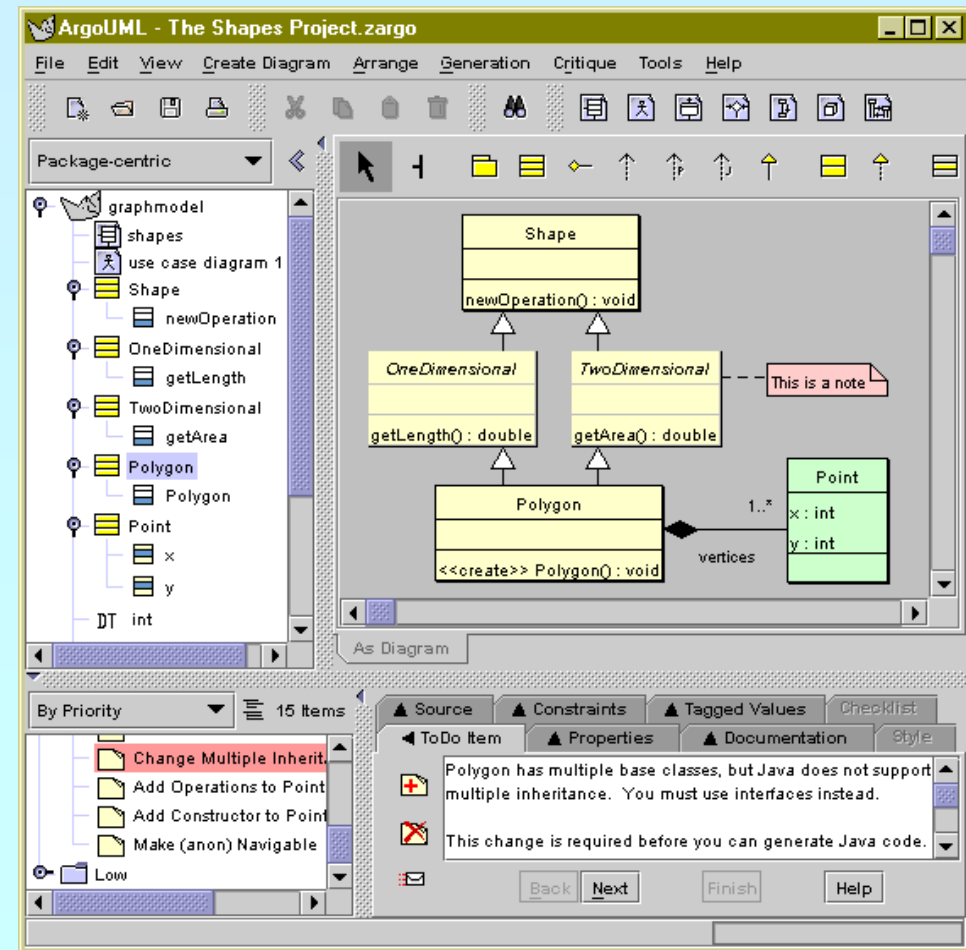


# Umbrello

- ✓ Most of UML 1 diagram types
- ✓ Code generation for
  - C++, Java, PHP, Perl, Python, SQL, ADA, AS,
  - JavaScript, IDL, XML Schema
- ✓ Export to PNG and XMI
- ✓ Runs on Linux/KDE
- ✓ Manual in English and German

# ArgoUML

- ☑ Runs on any platform with Java 1.3+
- ☑ Exports GIF, PS, EPS, PGML and SVG
- ☑ XMI Support
- ☑ OCL Support
- ☑ Code generation
  - ☑ C++
  - ☑ Java
  - ☑ PHP
- ☑ Support through commercial version Poseidon



# BITPlan smartGENERATOR

- Code generation from XMI
  - This allows to use all XMI compliant UML tools
    - Rational Rose
    - Poseidon Gentleware
    - microTOOL objectivF
    - Together Soft Together/J oder C++
    - Object Domain from Object Domain Systems
    - JDBC-Quellen and other Repositories
  - At the moment only a prototype

# Sybase PowerDesigner

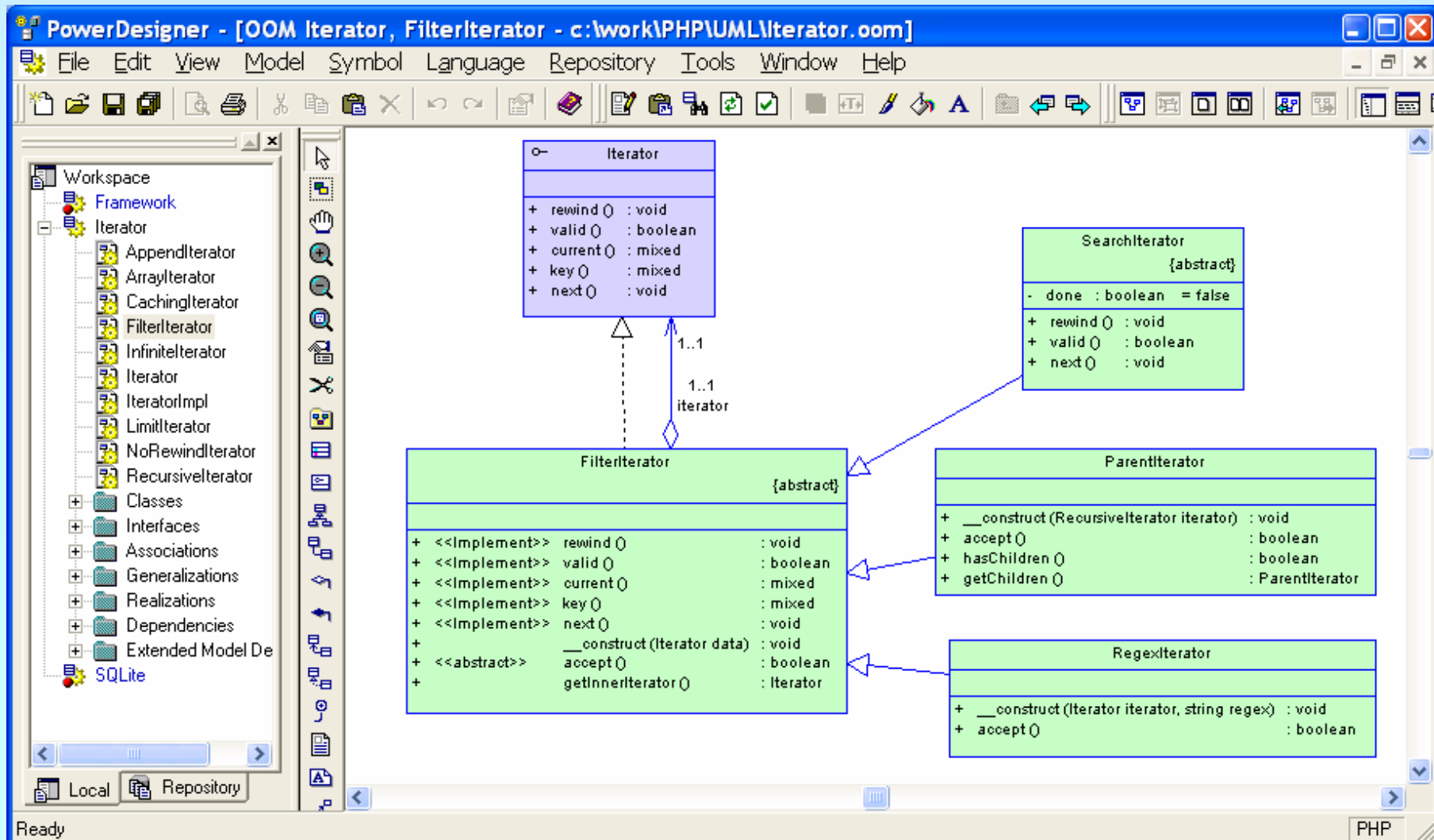
- ✓ Complete UML 1.3 support
- ✓ Process Execution
  - ✓ Service Oriented Architecture support
  - ✓ ebXML, BPEL4WS
- ✓ Data Modeling
  - ✓ Conceptual, Logical, Physical and Warehouse Data
  - ✓ Roundtrip engineering for databases
- ✓ Enterprise Modeling
  - ✓ Model Driven Architecture
  - ✓ Object/Relational, XML to Database and Warehouse Source mapping Techniques
  - ✓ Roundtrip engineering for C#, C++, Java, VBA, XML, ...
- ✓ Documentation generation
- ✓ Highly Extensible and Customizable



# Sybase PowerDesigner



## Working with class diagrams





# Sybase PowerDesigner



## Teaching PHP

The screenshot shows the 'Object Language Properties (For All Models)' dialog box in Sybase PowerDesigner. The 'General' tab is active, and the path is set to 'PHP::Profile\Class\Templates\definition'. The left-hand tree view shows the project structure, with 'definition' selected under 'Class' > 'Templates'. The right-hand pane shows the following code template:

```

[%javaDocComment%\n]\
.if (%Stereotype% == defines)
%members%
.else
[%flags% ]class %Code%[ %extends%][ %implements%]
{
[ %members%\n\n]\
}
.endif
    
```

The code is displayed in a text editor with a toolbar and status bar showing 'Ln 1, Col 1'. At the bottom of the dialog, there are buttons for 'OK', 'Abbrechen', 'Übernehmen', and 'Hilfe'.



# Sybase PowerDesigner



## Teaching PHP

The screenshot shows the 'Object Language Properties (For All Models)' dialog box in Sybase PowerDesigner. The 'General' tab is active, and the path is set to 'PHP::Profile\Attribute\Templates\declaration'. The left-hand tree view shows the project structure, with 'declaration' selected under 'Attribute\Templates\Helpers'. The right-hand pane shows the following code template:

```

[%javaDocComment%\n]\  

.if (%Parent.Stereotype% == defines)  

define(''%Code%'', %initialValue%);  

.elseif (%Parent.Stereotype% == enum)  

const %Code%[ = %initialValue%];  

.elseif (%Stereotype% == const)  

const %Code% = %initialValue%;  

.else  

[%visibility% ][%flags% ]%fieldCode%[ = %initialValue%];  

.endif
    
```

The dialog box includes standard Windows window controls, a toolbar with icons for undo, redo, and other actions, and buttons for 'OK', 'Abbrechen', 'Übernehmen', and 'Hilfe' at the bottom.



# Sybase PowerDesigner



## Teaching PHP

The screenshot shows the 'Object Language Properties (For All Models)' dialog box in Sybase PowerDesigner. The 'General' tab is active, and the path is set to 'PHP::Profile\Attribute\Stereotypes\const\Custom Checks\CheckValuePresent'. The left pane shows a tree view of the project structure, with 'CheckValueP' selected under 'Custom Checks'. The right pane displays the following PHP script:

```

Function %Check%(obj)
  If obj.DataType = "string" Then
    %Check% = True
  ElseIf IsEmpty(obj.initialValue) OR obj.initialValue = "" Then
    %Check% = False
  Else
    %Check% = True
  End If
End Function
    
```

The script is located at line 5, column 24. The dialog box has buttons for 'OK', 'Abbrechen', 'Übernehmen', and 'Hilfe' at the bottom.



# Sybase PowerDesigner



## Generating PHP scripts from UML

The screenshot displays the Sybase PowerDesigner interface. On the left, a tree view shows the project structure. The main window is titled "Class Properties - SearchIterator (SearchIterator)". It features a tabbed interface with "Script" selected, showing the generated PHP code for the SearchIterator class. The code is as follows:

```

abstract class SearchIterator extends FilterIterator
{
    private $done = false;

    public rewind()
    {
        parent::rewind();
        $this->done = false;
    }

    /** @return (boolean) */
    public valid()
    {
        return !$this->done && parent::valid();
    }

    public next()
    {
        $this->done = true;
    }
}
    
```

On the right, a UML class diagram shows the SearchIterator class as an abstract class with a private attribute `done : boolean = false` and three public methods: `rewind() : void`, `valid() : boolean`, and `next() : void`. Below it, two concrete classes are shown: `ParentIterator` and `RegexIterator`. `ParentIterator` has methods `__construct(RecursiveIterator iterator) : void`, `accept() : boolean`, `hasChildren() : boolean`, and `getChildren() : ParentIterator`. `RegexIterator` has methods `__construct(Iterator iterator, string regex) : void` and `accept() : boolean`. The PHP Source tab at the bottom of the class properties window is active, and the "OK" button is visible at the bottom of the dialog.



# Sybase PowerDesigner



## Generating PHP code for constants

The screenshot shows the Sybase PowerDesigner interface. The main window displays the 'Class Properties' dialog for the class 'CahingIteratorMode'. The 'Script' tab is active, showing the generated PHP code:

```
<?php
/*****
 * Module: CahingIteratorMode.php
 * Author: marcus
 * Purpose: Defines the Class CahingIteratorMode
 *****/

define('CIT_CALL_TOSTRING', 1);
define('CIT_CATCH_GET_CHILD', 2);

?>
```

The background shows a UML class diagram with the following details:

- Class: RecursiveIterator**
  - Operations: `hasChildren() : boolean`, `getChildren() : RecursiveIterator`
- Class: CahingIteratorMode** (stereotype: `<<defines>>`)
  - Attributes: `+ CIT_CALL_TOSTRING : int = 1`, `+ CIT_CATCH_GET_CHILD : int = 2`
- Relationships:**
  - A dashed arrow points from `RecursiveIterator` to `CahingIteratorMode`, indicating a generalization or realization relationship.
  - A solid arrow points from `RecursiveIterator` to `CahingIteratorMode`, indicating an association.

# What to do next

- ☑ Automatically generate data query classes
- ☑ Extending PHP's reflection API to write XMI
- ☑ Typehints for properties and return values
- ☑ Implement roundtrip engineering
- ☑ Implement a script packaging extension
- ☑ Extend PHP with the keyword 'package'



# What else?

- ☑ Using PHP to script UML tools
- ☑ Generate PHP code from Activity Diagrams
- ☑ Validating generated PHP code

# Links

- ☑ This slides  
<http://somabo.de/talks>
- ☑ PHP  
<http://php.net>
- ☑ Doxygen  
<http://doxygen.org>
- ☑ ArgoUML  
<http://argouml.tigris.org>
- ☑ Poseidon  
<http://www.gentleware.com>
- ☑ Umbrello  
<http://uml.sourceforge.net>
- ☑ Waterproof  
<http://www.waterproof-software.com/uml/>
- ☑ PowerDesigner  
<http://www.sybase.com/products/enterprisemodeling>

