PHP 5

The Year After

Sebastian Bergmann

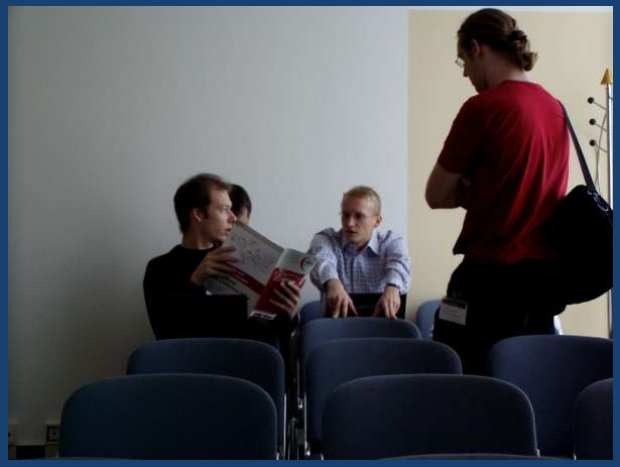
May, 4th 2005

Who I Am

- Sebastian Bergmann.
- Born 1978.
- CS Student in Bonn, Germany.
- Committer: PHP, PEAR, Gentoo Linux, ...
- Author of Open Source PHP projects like PHPUnit or phpOpenTracker.
- Author for online and print media.

Who Are You?

- Who of you uses PHP 5?
- Who of you have heard of
  - PHPUnit2?
  - Phing?
  - Creole?
  - Propel?



A Look Back

- PHP 3
  - Released on June 6, 1998.
  - Very rudimentary support for object-oriented programming.
  - No session management.
- PHPLIB
  - Maybe the most important library (written in PHP) for PHP 3.
  - Developed by Kristian Köhntopp, Ulf Wendel et al. at NetUSE AG in Kiel, Germany.
  - Sascha Schumann's code for serialization of PHP variables and session management, which was part of PHPLIB, was the de-facto standard for PHP 3.

A Look Back

- PHP 4
  - Released on May 22, 2000.
  - Rudimentary support for object-oriented programming.
  - Sascha Schumann rewrote his serialization and session management code (which was part of PHPLIB) in C and it became (as ext/session) part of PHP.
- PEAR
  - Repository of reusable classes and components written in PHP.
  - Infrastructure (PEAR Installer) to install and maintain packages.

Today

- PHP 5
  - Released on July 13, 2004.
  - Good support for object-oriented programming.
    - Language Features.
    - Standard PHP Library (SPL).
    - Extensions like DOM and MySQLi offer OOP APIs.
  - Good support for working with XML technologies and Web Services.
- PECL
  - PHP Extension Community Library.
  - Extensions to PHP written in C.
  - Not commonly used extensions (for example dio, fam, yp, ...) are unbundled from the standard PHP distribution and moved to PECL.
  - Commonly used PECL extensions (for example SQLite) are bundled with the PHP standard distribution.

A Look Ahead

- PHP 5.1
  - To be released later this year.
  - Optimized Virtual Machine (VM).
    - Up to 40% better performance compared to PHP 5.0.
  - PHP Data Objects (PDO)
    - Built-In Database Abstraction Layer on the API level.
      - Unified `new PDO(...)` instead of `mysql_connect(...)`, `pg_connect(....)`, …
    - Makes use of Zend Engine 2 features like the `Iterator` interface.
  - Improved Standard PHP Library (SPL)
    - Standard hierarchy of `Exception` classes.
    - `Countable` interface.
    - `Subject` and `Observer` interfaces.
  - New XMLReader API for XML processing.

A Look Ahead

- PEAR 1.4
  - To be released later this year.
  - Most important update to the PEAR/PECL infrastructure to date.
    - Channels.
    - Mirroring.
    - Binary PECL packages.
    - New package.xml 2.0 format.
    - Pre-Download dependency validation and full dependency validation on uninstall.
    - Self-Installing PEAR from a single file (thanks to PHP_Archive).
    - Representational State Transfer (ReST) for client/server communication (in addition to or as a replacement for XML-RPC).

Projects that use PHP 5

- PHPUnit2
  - Framework for Unit Tests based on JUnit.
- Phing
  - Project build system based on Apache Ant.
- Creole
  - Database abstraction layer loosely based on JDBC.
- Propel
  - Full-Service object persistence and query toolkit based on Apache Torque.
- Reflection_Annotation
  - Extension to the Reflection API that supports annotation-based programming.

PHPUnit2

- Framework for Unit Testing based on JUnit.
- Features include
  - Full port of JUnit 3.8.1.
  - Support for Agile Documentation (TestDox).
  - Support for Code Coverage analysis with Xdebug.
  - Support for Incomplete Tests.
  - Integration with Phing.
  - Skeleton Generator for test classes.
  - Logging to PEAR::Log sinks and in XML format.
- Installation
  - `pear install PHPUnit2`

PHPUnit2

```
<?php
require_once "PHPUnit2/Framework/TestCase.php";

class ArrayTest extends PHPUnit2_Framework_TestCase {
    public function testNewArrayIsEmpty() {
        $fixture = array();
        $this->assertEquals(0, sizeof($fixture));
    }

    public function testArrayContainsElement() {
        $fixture = array('element');
        $this->assertEquals(1, sizeof($fixture));
    }
}
?>
```

- Conventions
  - Tests for class `Class` in class `ClassTest`.
  - Tests written as methods with prefix `test`.

PHPUnit2

Terminal

File Edit View Terminal Tabs Help

```
sb@wopr-mobile ~ % phpunit ArrayTest
PHPUnit 2.1.3 by Sebastian Bergmann.

..

Time: 0.007176

OK (2 tests)
sb@wopr-mobile ~ %
```

PHPUnit2

```
<?php
require_once "PHPUnit2/Framework/TestCase.php";

class ArrayTest extends PHPUnit2_Framework_TestCase {
    public function testNewArrayIsEmpty() {
        $fixture = array();
        $this->assertEquals(0, sizeof($fixture));
    }

    public function testArrayContainsElement() {
        $fixture = array('element');
        $this->assertEquals(2, sizeof($fixture));
    }
}
?>
```

PHPUnit2

```
sb@wopr-mobile ~ % phpunit ArrayTest
PHPUnit 2.1.3 by Sebastian Bergmann.

.F

Time: 0.007051
There was 1 failure:
1) testArrayContainsElement
expected same: <2> was not: <1>

FAILURES!!!
Tests run: 2, Failures: 1, Errors: 0, Incomplete Tests: 0.
sb@wopr-mobile ~ %
```

Phing

- Project build system based on Apache Ant.
- Features include
  - file transformations,
    - e.g. token replacement, XSLT transformation, Smarty template transformations
  - file system operations,
  - SQL execution,
  - CVS and Subversion operations,
  - tools for creating PEAR packages,
  - running PHPUnit2 tests and generating test reports.
- Installation
  - `pear install http://phing.info/pear/phing-current.tgz`

Phing

```
<?xml version="1.0"?>

<project name="FooBar" default="dist" basedir=".">

  <target name="prepare">
    <echo msg="Preparing build..." />
    <mkdir dir="./build" />
  </target>

  <target name="build" depends="prepare">
    <echo>Building...</echo>
    <copy file="./src/File.php" to="./build/File.php"/>
    <copy file="./src/File2.php" to="./build/File2.php"/>
  </target>

  <target name="dist" depends="build">
    <echo message="Creating archive..." />
    <tar outfile="furbee.tar.gz" basedir="./build"/>
  </target>

  <target name="clean">
    <echo msg="Cleaning up..."/>
    <delete file="./build"/>
  </target>
</project>
```

Database Abstraction

- If a PHP application is to be used with different database systems a *database abstraction layer* is needed.
- What needs to be abstracted:
  - PHP's API for the different database systems.
    - A non-problem with PHP 5.1 and PDO.
  - Differences in regard to the SQL standard(s) between the different database systems.

Creole

- Database Abstraction Layer for PHP 5.
  - Object-Oriented, makes use of PHP 5 features like `Iterator`.
  - Designed after the Java Database Connectivity (JDBC) API.
- Installation
  - `pear install http://creole.phpdb.org/pear/creole-current.tgz`
  - `pear install http://creole.phpdb.org/pear/jargon-current.tgz`

Creole

```
<?php
require_once 'creole/Creole.php';

try {
  $connection = Creole::getConnection(
    array(
      'phptype'  => 'mysql',
      'hostspec' => 'localhost',
      'username' => 'root',
      'password' => '',
      'database' => 'test'
    )
  );
}

catch (SQLException $e) {
  // Ausnahme $e behandeln.
}
?>
```

Creole

```
<?php
try {
  $resultSet = $connection->executeQuery(
    'SELECT foo FROM bar;'
  );

  foreach ($resultSet as $rowNumber => $row) {
    printf(
      "%d: %s\n",

      $rowNumber,
      $row['foo']
    );
  }
}

catch (SQLException $e) {
  // Ausnahme $e behandeln.
}
?>
```

Creole

```
<?php
try {
  $numAffectedRows = $connection->executeUpdate(
    'DELETE FROM bar;'
  );
}

catch (SQLException $e) {
  // Ausnahme $e behandeln.
}
?>
```

Creole

```
<?php
try {
  $statement = $connection->createStatement();
  $statement->setOffset(5);
  $statement->setLimit(10);

  $resultSet = $statement->executeQuery(
    'SELECT foo FROM bar;'
  );
}

catch (SQLException $e) {
  // Ausnahme $e behandeln.
}
?>
```

Creole

```
<?php
$foo = 1978;

try {
  $statement = $connection->prepareStatement(
    'DELETE FROM bar WHERE foo = ?;'
  );

  $statement->setInt(1, $foo);

  $numAffectedRows = $statement->executeUpdate();
}

catch (SQLException $e) {
  // Ausnahme $e behandeln.
}
?>
```

Object-Relational Mapping

- Mapping from classes to tables in an RDBMS.
  - Storing of objects in a database.
  - Encapsulating database entities in objects.
- An object-relational mapper is a bridge between the database and the application.
  - The programmer does not need to formulate SQL queries.
  - When the data model gets changed the corresponding classes are automatically updated.
  - When the application gets migrated to another RDBMS only the configuration needs to be changed.

Propel

- Propel is such a bridge for PHP 5.
  - Designed after Apache Torque.
  - Based upon Creole and Phing.
- Two components:
  - Propel Generator
    - Input: XML specification for the data model.
    - Output: Database Schema and PHP classes.
  - Propel Runtime
    - Runtime Environment.
    - Framework for the use of the classes generated by the Propel Generator.
- Installation
  - `pear install http://propel.phpdb.org/pear/propel_runtime-current.tgz`
  - `pear install http://propel.phpdb.org/pear/propel_generator-current.tgz`

Data Model

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>

<database name="books" defaultIdMethod="native">
  <table name="author">
    <column name="author_id" required="true" primaryKey="true"
            type="INTEGER"/>
    <column name="first_name" required="true"
            type="VARCHAR" size="128"/>
    <column name="last_name" required="true"
            type="VARCHAR" size="128"/>
  </table>

  <table name="publisher">
    <column name="publisher_id" required="true" primaryKey="true"
            type="INTEGER"/>
    <column name="name" required="true"
            type="VARCHAR" size="128"/>
  </table>

  <table name="book">
    <column name="book_id" required="true" primaryKey="true"
            type="INTEGER"/>
    <column name="title" required="true"
            type="VARCHAR" size="255"/>
    <column name="isbn" required="true"
            type="VARCHAR" size="24"/>
    <column name="author_id" required="true"
            type="INTEGER"/>
    <column name="publisher_id" required="true"
            type="INTEGER"/>

    <foreign-key foreignTable="author">
      <reference local="author_id" foreign="author_id"/>
    </foreign-key>

    <foreign-key foreignTable="publisher">
      <reference local="publisher_id" foreign="publisher_id"/>
    </foreign-key>
  </table>
</database>
```

Object Store

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<config>
  <log>
    <ident>propel-books</ident>
    <level>7</level>
  </log>
  <propel>
    <datasources default="books">
      <datasource id="books">
      <adapter>mysql</adapter>
      <connection>
        <phptype>mysql</phptype>
        <hostspec>localhost</hostspec>
        <database>books</database>
        <username>root</username>
        <password></password>
      </connection>
      </datasource>
    </datasources>
  </propel>
</config>
```

Propel Generator

```
propel-gen /home/sb/books
```

Database Schema

```
DROP TABLE IF EXISTS author;
CREATE TABLE author(
  author_id     INTEGER      NOT NULL,
  first_name    VARCHAR(128) NOT NULL,
  last_name     VARCHAR(128) NOT NULL,

  PRIMARY KEY (author_id)
) TYPE=InnoDB;

DROP TABLE IF EXISTS publisher;
CREATE TABLE publisher(
  publisher_id  INTEGER      NOT NULL,
  name          VARCHAR(128) NOT NULL,

  PRIMARY KEY (publisher_id)
) TYPE=InnoDB;

DROP TABLE IF EXISTS book;
CREATE TABLE book(
  book_id       INTEGER      NOT NULL,
  title         VARCHAR(255) NOT NULL,
  isbn          VARCHAR(24)  NOT NULL,
  author_id     INTEGER      NOT NULL,
  publisher_id  INTEGER      NOT NULL,

  PRIMARY KEY (book_id),
  INDEX (author_id),
  FOREIGN KEY (author_id)    REFERENCES author    (author_id),
  INDEX (publisher_id),
  FOREIGN KEY (publisher_id) REFERENCES publisher (publisher_id)
) TYPE=InnoDB;
```

Object Model

- Table `author`
  - Abstract base class `BaseAuthor`.
    - Methods like
      - `getFirstName()` or
      - `setFirstName($v)`.
  - Concrete class `Author extends BaseAuthor`.
    - Empty.
    - Will not be overwritten if the Propel Generator is run again.

Creating a Book

```
<?php
require_once 'propel/Propel.php';
Propel::init('books/conf/runtime-conf.php');

require_once 'books/Author.php';
require_once 'books/Book.php';
require_once 'books/Publisher.php';

$sebastian = new Author;
$sebastian->setFirstName('Sebastian');
$sebastian->setLastName('Bergmann');

$dpunkt = new Publisher;
$dpunkt->setName('dpunkt.verlag');

$psmp5 = new Book;
$psmp5->setIsbn('3898642291');
$psmp5->setTitle(
  'Professionelle Softwareentwicklung mit PHP 5'
);
$psmp5->setAuthor($sebastian);
$psmp5->setPublisher($dpunkt);
$psmp5->save();

$hakan = new Author;
$hakan->setFirstName('Hakan');
$hakan->setLastName('Kücükyilmaz');

$php5 = new Book;
$php5->setIsbn('3898642364');
$php5->setTitle('PHP 5');
$php5->setAuthor($hakan);
$php5->setPublisher($dpunkt);
$php5->save();
?>
```

Updating an Author

```
<?php
require_once 'propel/Propel.php';
Propel::init('books/conf/runtime-conf.php');

require_once 'books/AuthorPeer.php';

// Autor mit Primärschlüssel 1 holen.
$author = AuthorPeer::retrieveByPK(1);

// Den Vornamen des Autors ändern.
$author->setFirstName('Johannes Sebastian');

// Die geänderten Daten speichern.
$author->save();
?>
```

Deleting a Book

```
<?php
require_once 'propel/Propel.php';
Propel::init('books/conf/runtime-conf.php');

require_once 'books/BookPeer.php';

// Buch mit Primärschlüssel 1 holen.
$book = BookPeer::retrieveByPK(1);

// Das Buch löschen.
$book->delete();
?>
```

Lookup by Criteria

```
<?php
require_once 'propel/Propel.php';
Propel::init('books/conf/runtime-conf.php');

require_once 'books/AuthorPeer.php';

$criteria = new Criteria;

$criteria->add(
  AuthorPeer::FIRST_NAME,
  'Sebastian'
);

$criteria->add(
  AuthorPeer::LAST_NAME,
  'Nohn',
  Criteria::NOT_EQUAL
);

$authors = AuthorPeer::doSelect($criteria);

foreach ($authors as $author) {
    print 'Sebastian ' . $author->getLastName();
}
?>
```

```
Sebastian Bergmann
```

Lookup by Criteria

```
<?php
require_once 'propel/Propel.php';
Propel::init('books/conf/runtime-conf.php');

require_once 'books/AuthorPeer.php';

$criteria  = new Criteria;

$sebastian = $criteria->getNewCriterion(
  AuthorPeer::FIRST_NAME,
  'Sebastian'
);

$hakan = $criteria->getNewCriterion(
  AuthorPeer::FIRST_NAME,
  'Hakan'
);

$sebastian->addOr($hakan);
$criteria->add($sebastian);

$authors = AuthorPeer::doSelect($criteria);

foreach ($authors as $author) {
    print $author->getFirstName() . $author->getLastName();
}
?>
```

```
Sebastian Bergmann
Hakan Kücükyilmaz
```

Reflection_Annotation

- Extension to PHP 5's Reflection API that supports annotation-based programming.
- Annotation `Test` declared in class `TestAnnotation` that extends `Reflection_Annotation`.
- Classes, methods, and properties can be annotated using `@Test(foo="bar")` inside the annotated element's Doc Comment.
  - Parameter foo is set to "bar" by calling `setFoo($bar)` on the Annotation object.

Reflection_Annotation

```
<?php
require_once 'Reflection/Annotation.php';

class TestAnnotation extends Reflection_Annotation {
    private $foo;

    public function getFoo() {
        return $this->foo;
    }

    public function setFoo($foo) {
        $this->foo = $foo;
    }
}
?>
```

Reflection_Annotation

```
<?php
/**
 * A class.
 *
 * @Test(foo="bar")
 */
class TestClass {
}

require_once 'Reflection/Annotation/Class.php';

$testClass = new Reflection_Annotation_Class('TestClass');

if ($testClass->hasAnnotation('Test')) {
    print $testClass->getAnnotation('Test')->getFoo() . "\n";
}
?>
```

Reflection_Annotation

- Development on the Reflection_Annotation package has only recently started.
- Eventually it will be committed to PEAR.
- A future version of PHPUnit2 might support Annotations like `@UnitTest` to denote test methods instead of using the method's name (`test*`) for this.
- Annotations could also ease the use of Web Services with PHP 5's SOAP extension.

Conclusion

- With PHP 5 it is possible to take ideas from the Java world and implement them in PHP.
- After the release of PHP 5 a couple of well-known Java solutions have already been ported to PHP.
  - JUnit -> PHPUnit2
  - Apache Ant -> Phing
  - Java JDBC -> Creole
  - Apache Torque -> Propel
- Solutions like PHPLIB's serializer and concepts like database API abstraction tend to be integrated into the PHP Interpreter itself.

License