

AzPHP

An Introduction to PHPUnit

Chris Daley

Lodestone Systems

www.lodesys.com

Sept. 28, 2004

Overview

- What's OOP?
- How can we test objects?
- PHPUnit Nuts & Bolts
- A Coding Example
- PHPUnit Tips & Tricks
- General Q & A

What's Object Oriented Programming?

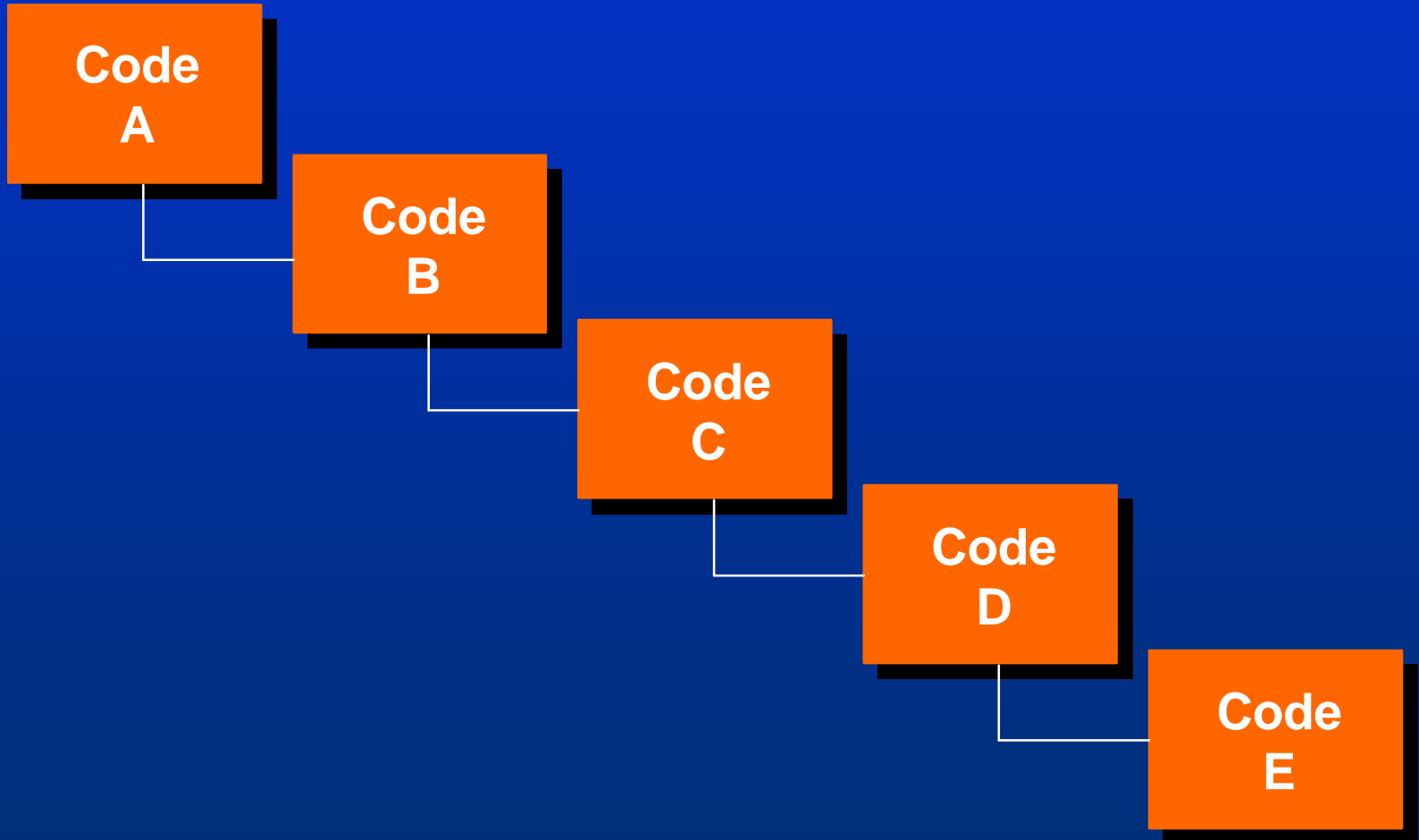
And why should I care?

Object Oriented Programming Concepts

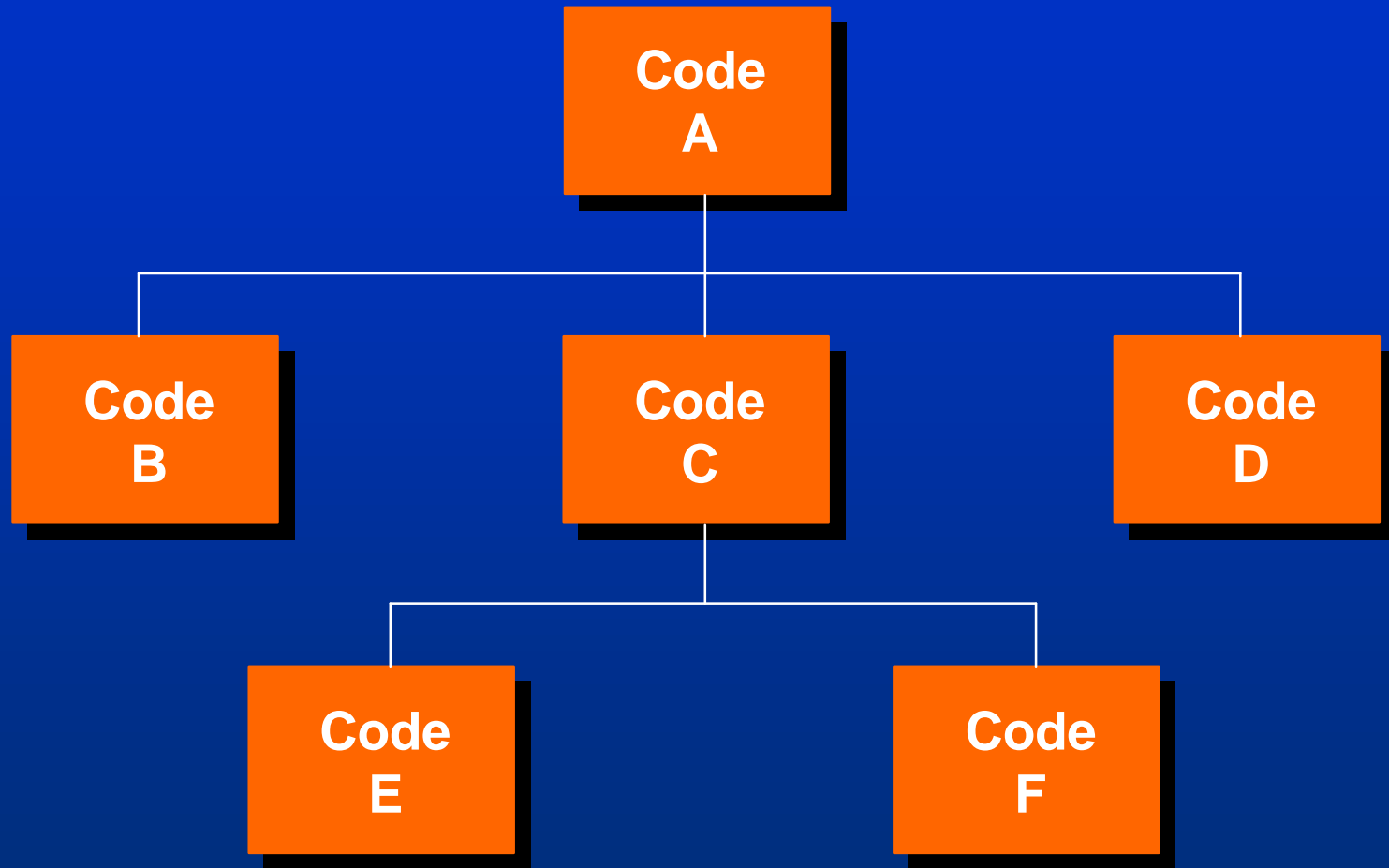
- OOP uses objects to represent the real-world.
- OOP is a tool. It is not a solution.
- **IT IS NOT** a totally new way of programming!
- **IT IS** a new way of thinking about and organizing your code and data.
- It models the way the world really works.

In the Beginning...

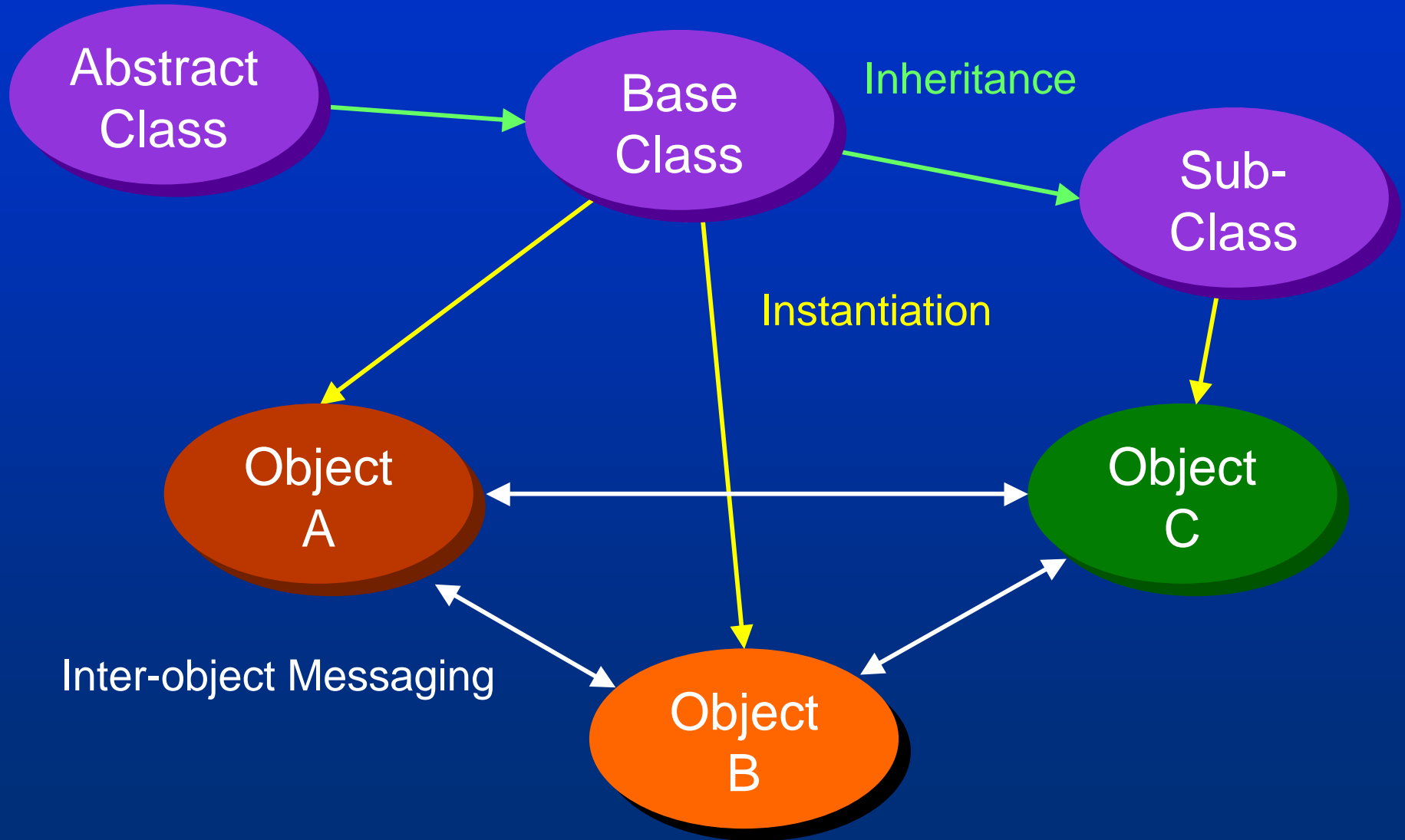
Straight Line Programming



Then... Structured Design



Now... Object Oriented Design



Objects/Classes...

are modules that contain

PHP Object

Behaviors:	Methods	Code	Function
Attributes:	Properties	Values	Value

Example Objects...

- A Bank Account
- An Employee
- A Data Table
- The Output Page (HTML, XML, PDF, ???)
- An Inventory Item
- The Shipping Clerk
- A Starship
- The Data Base

Benefits of OOP

- Quicker design - use of building blocks
- Real-world modeling
- Less complex code
- Reusability and modularity
- Easier to support
- The ability to modify existing code without modifying it

How Can We Test Objects?

I'd like to thank...

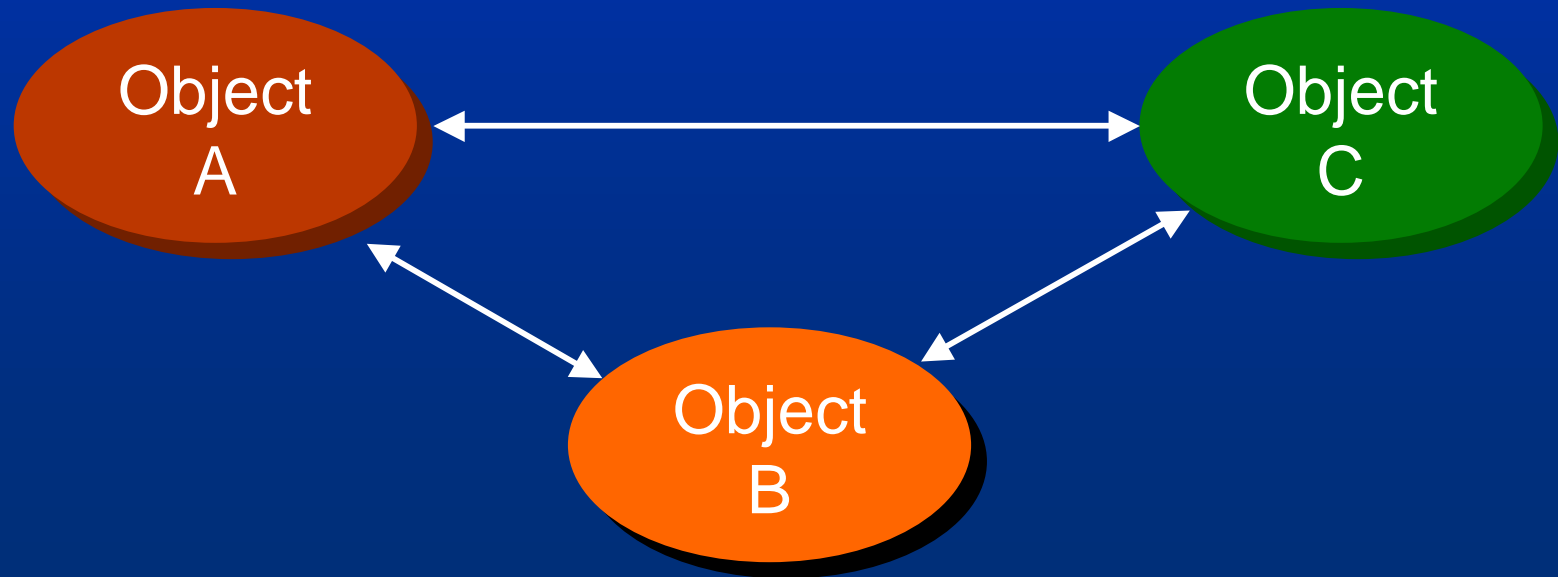
Sun Microsystems

JAVA

JUnit

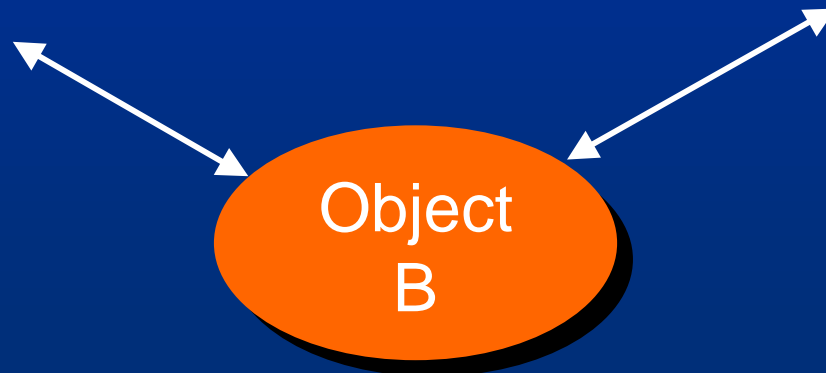
Sebastian Bergmann

In Object Oriented Programming...



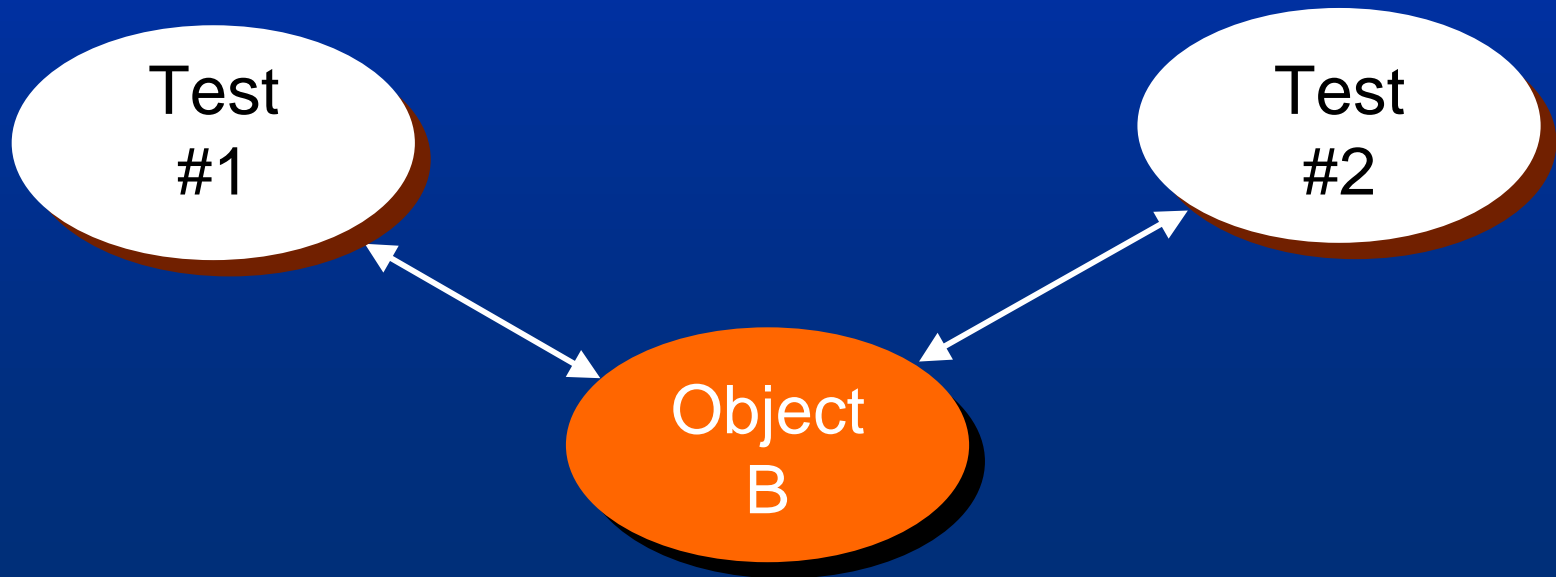
Each Object Is...

“Independently Owned and Operated”



So, we can...

Replace the rest of the program
with test modules.



The Evolution of Unit Testing

- Testing code, especially objects, is a pain.
- Erich Gamma and Kent Beck did something about it, writing the original version of JUnit for JAVA on their way to OOPSLA97.
- JUnit was originally designed to support regression testing.
- Sebastian Bergmann took the concepts of JUnit and ported it to PHP, creating PHPUnit.

PHPUnit Nuts & Bolts

Getting from here to code...

PHPUnit vs. PHPUnit2

There are two versions of PHPUnit.

- PHPUnit (version 1)
 - Loosely based on JUnit.
 - Will run on all current versions of PHP.
- PHPUnit2
 - Implements the functionality of JUnit 3.8.1
 - Requires PHP 5.
 - Quality level: Release Candidate

Installing PHPUnit

- Both available via PEAR
 - PHPUnit & PHPUnit2
- PHPUnit is easier to understand
- PHPUnit2 is the future, but requires PHP 5
- Documentation is minimal

Creating a PHPUnit Test Framework

- Subclass `PHPUnit2_Framework_TestCase`
- Code the `setUp()` and `tearDown()` functions as needed
 - `setUp()` used to initialize and organize the object's environment.
 - `tearDown()` used to clean up after the test.
- And add the tests
 - Each test is independent, using a fresh copy of the Object.
 - Each test should test just one aspect or function of the Object.

Test Example

```
<?php
require_once "BankAccount.php";
require_once "PHPUnit2/Framework/Testcase.php";

class BankAccountTest extends PHPUnit2_Framework_TestCase {

    public function testBalanceIsInitiallyZero() {
        $ba = new BankAccount;
        $this->assertEquals(0, $ba->getBalance());
    }

    public function testBalanceCannotBecomeNegative() {
        $ba = new BankAccount;
        $ba->withdrawMoney(1);
        $this->assertEquals(0, $ba->getBalance());

        $ba = new BankAccount;
        $ba->setBalance(-1);
        $this->assertEquals(0, $ba->getBalance());
    }
}
?>
```

Test Types Include...

- assertEquals()
- assertFalse()
- assertNotNull()
- assertNotSame()
- assertNull()
- assertRegExp()
- assertSame()
- assertTrue()
- assertType()
- convertToString()
- fail()
- failNotEquals()
- failNotSame()
- failSame()

PHPUnit can support...

- **Test-Driven Development:** write the test, then write the code.
- **Design-by-Contract Development:** Define the object requirements via tests.
- **Agile Programming:** design as you go -- the “Junkyard Wars” approach to programming.
- **Extreme Programming:** combines all of the above ... and more.

Wrapping PHPUnit2

- The output from PHPUnit2 tests can be as text, HTML, XML, E-Mail(s) – just about anything.
- PHPUnit2 can be launched...
 - From the command line
 - As a web page
- This functionality was not very robust.
 - What I get for using the “1.0” version.
- I use a wrapper object to allow me to run PHPUnit2 tests the way I want using my web browser.

A Coding Example

Let The Games Begin!

The Requirements

- Create an Object called “TextDecode” with a method called “decodeAmount”.
 - More decoding methods will be added later.
- Returns FALSE if an invalid text value.
- Returns a number if valid.
- Must support both U.S. and European formatting systems.
 - Can enter 12,345.6 or 12 345,6
- Punctuation is optional and whole numbers are allowed.
 - 12345 & 12345.6 & 12345,6 are all valid



Let's Go Do It!

PHPUnit Tips & Tricks

OOP too!

Test Philosophy

- Use Test-Driven Development.
 - Write the test case,
 - Then write just the code needed to pass the test.
- “More and Simple” is better.
 - 100’s of test cases are normal.
- If you find a bug, create a test that fails because of it first, then fix it.
 - If it ever reappears, you’ll quickly catch it!

When Getting Started...

- Learn Object Oriented Programming first!
 - PHPUnit won't help you learn OOP.
- Start with a simple test project.
- PHPUnit will be of limited value for an existing project.
 - Unless you were planning on rewriting (“refactoring”) it anyway.
- Use the web resources.

PHP's PROTECTED vs. PRIVATE

- Use PROTECTED, not PRIVATE in Objects.
- PRIVATE means no one outside the current class can access the function or value.
 - Useful for passwords, etc.
 - Bad if you ever want to subclass the object.
- PROTECTED means only “outside” objects cannot access the function or method.
 - Subclassed objects function correctly.
- Using PRIVATE will not cause PHP errors, but makes for some strange behaviors.

Testing HTML Output

You can use PHPUnit to test HTML output...

- Use `ob_start()` to capture the output of `Print`, `Echo`, etc.
- Use `ob_get_contents()` to save the output to a string variable, then call `ob_end_clean()` to throw away the output.
- Look for the required text in the string variable.

External Package Trick

- Creating a wrapper object to implement changes is best.
- If you have to modify the package code...
 - Create PHPUnit tests to identify what needs to be changed.
 - » i.e., don't create the default DB object in ezSQL.
 - When you download a new version, you can quickly identify everything that needs to be changed, and confirm that the changes were made.

Resources

- sebastian-bergmann.de/en/phpunit.php
 - Main page for PHPUnit
- pear.php.net/package/PHPUnit
- pear.php.net/package/PHPUnit2
 - Code and some additional information here
- c2.com/cgi/wiki?JUnit
 - Great wiki describing JUnit, testing frameworks, etc., etc.
- en.wikipedia.org/wiki/Design_by_contract
 - More information on Design By Contract

Q & A

Any comments?