Zend PHP 5.0 Certification Course

by Paul Reinheimer

Contents

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

PHP Basics

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - **PHP Basics**
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

PHP Basics

- PHP Basics refers to a number of things that are likely already second nature to you:
  - Syntax
  - Operators
  - Variables
  - Constants
  - Control Structures
  - Language Constructs and Functions

Notes

PHP Tags

- Since PHP was designed to be incorporated with HTML, the interpreter needs to distinguish between PHP code and the HTML markup.
- The most common tags are these:
  - <?php
  - //Long Tags
  - ?>

Notes

PHP Tags Continued

- While the regular style of tags is preferred, and can not be disabled, you are also guaranteed access to the script style tags

```
<script language="php">
  //Long or Script style tags
</script>
```


Notes

PHP Tags Continued

- Short tags have also been quite popular, however they can be disabled (often for XML compliance) and are falling from favour
  <? echo 'Short PHP tags'; ?>
  <?= $expression ?>
- The latter set of tags are a shortcut for
  <? echo $expression ?>
- Both sets of tags can be enabled/disabled via use of the short_open_tag directive in php.ini

Notes

PHP Tags Continued

- ASP style tags are also available, but must be enabled specifically in the php.ini file via the asp_tags directive
  <% echo 'ASP-style tags'; %>
  <%= $variable; %>

Notes

PHP Tags Conclusion

- Use the long or regular PHP tags to ensure your application will be portable across multiple servers regardless of configuration options.

Notes

Comments

- Good programmers comment their code, PHP supports multiple syntaxes for comments:

```
<?php
//This is a single line comment

# This is also a single line comment

/*
   This is a multi-line
   comment
*/
```


Notes

A few comments about comments

- Single line comments are ended by either the newline character or a closing PHP tag, so a single like comment like
  // This tag ?> can end PHP code
- Can have unintended output

Notes

Variables

- Variables provide containers for data, with PHP there are two main naming rules for variables and other identifiers:
  - Variables names can only contain letters, numbers or underscores
  - Variables names must start with a letter or an underscore

Notes

Variables

- Even though PHP is loosely typed, PHP variables are still stored and managed in one of PHP's core types:
  - Scalar types: boolean, string, integer, float
  - Compound types: array, object
  - Special Types: resource, null

Notes

Scalar Types - Strings

- Strings: A string is a series of characters, natively within PHP each character is represented by a single byte, as such PHP has no native support for multi-byte character sets (like Unicode)
  - This changes with PHP6
- There is no PHP imposed limit on the length of the strings you can use

Notes

Scalar Types - Strings

- Strings can be quoted in one of three ways
  - ‘ ’ – The single quote or string literal, characters within single quotes will be recorded as is, without variable or escape sequences interpreted and replaced
  - “ “ – The double quotes, variables and escape sequnces will be interpreted and replaced
  - <<< – Heredoc (next slide)

Notes

Scalar Types - Strings

- The heredoc syntax functions in a similar manner to double quotes, but is designed to span multiple lines, and allow for the use of quotes without escaping

```
$greeting = <<<GREETING
She said "That is $name's" dog!
While running towards the thief
GREETING;
```

- The closing heredoc tag must be the first thing on the line, the only other permissible character on the line is the semi-colon.

Notes

Scalar Types - Integer

- Integers (whole numbers) can be specified in decimal (base 10), hexadecimal (base 16), or octal (base 8) notation and optionally preceded by a sign (+, -)
  - To use octal precede the number with a 0
  - To use hex precede the number with a 0x
- The maximum size of an integer is platform dependent, a maximum of ~2Billion is common

Notes

Scalar Types – Float (double)

- Floats, or doubles, can be used to represent really large or really small values. Floats can be entered via several syntaxes

```
$a = 1.234;       //Standard decimal notation
$b = 1.2e3;       //Scientific notation (1200)
$c = 7E-10;
        //Scientific notation (0.0000000007)
```

- The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value

Notes

Scalar Types - Boolean

- Boolean values hold either True (1) or False (0)
  - Any integer other than 0 is cast to TRUE
- TRUE & FALSE are case-insensitive, though the all caps representation is common

Notes

Quick Quiz

- Things to ponder:
  - What is the output of:

```
echo 09;
```

  - What is the output of:

```
$a = 3e4;
$b = <<<EOD
   This is a variable: $a
   Wasn't that nice?
   EOD;
echo $b;
```


Notes

Compound Types - Arrays

- Arrays can contain any combination of other variable types, even arrays or objects
- Arrays will be the subject of further discussion further on

Notes

Compound Types - Objects

- Objects allow data and methods to be combined into one cohesive structure
- Objects will be covered in much greater details in a future section

Notes

Special Types - Resource

- A resource is a special variable that represents some sort of operating system resource, generally an open file or database connection
- While variables of the type resource can be printed out, their only sensible use is with the functions designed to work with them

Notes

Special Types - null

- A null variable is special in that it has no value and no type
- null is not the same as the integer zero or an zero length string (which would have types)

Notes

Constants

- Constants allow you to create a special identifier that can not be changed once created, this is done using the define() function. Constants can not be changed once set.

```
define('username', 'bob');
```


Notes

Variable Variables

- Variable Variables are both useful and confusing:

```
$a = 'name';
$$a = "Paul";
echo $name; //Paul
```

- There are several situations in which Variable Variables can be useful, but use them with care as they can easily confuse other programmers.

Notes

Operators

- PHP supports a multitude of Operators:
  - Assignment
  - Logical
  - Arithmetic
  - String
  - Comparison
  - Bitwise
  - Other

Notes

Assignment Operators

- Variables are useless unless you can store something in them:

```
$name = "Paul";
$greeting = "Hello $name";
echo $greeting;
```


Notes

Assignment Operators

- By Value vs By Reference:

```
$a = 5;
$b = $a;
$b = 7;

//What is a, b?

$d = &$a;
$d = 9;
//What is a, d?
```


Notes

Arithmetic Operators

- Arithmetic operators just like we remember from school:

```
$a = 5;
$a = $a + 4;
$a = $a - 3;
$a = $a * 6;
$a = $a / 6;
$a = $a % 5;
```


Notes

Pre and Post Increment

- Very useful for loops

```
$a = 5;
echo $a++;     //5
echo $a;       //6

$b = 5;
echo ++$b;     //6
echo $b;       //6

$c = 5;
echo $c--;     //5
echo $c;       //4
$d = 5;
echo --$d;     //4
echo $d--;     //4
```


Notes

Operator plus Assignment

- Frequently you run into situations where you need to modify a variable in place:
  $a = 5;
  $a = $a + 5;
- Alternatively you can simply combine the desired operator with the assignment operator
  $a += 5;

Notes

String Operators

- The . or concatenation operator is used to “glue” two strings together:
- echo "His Name is: " . $name;
- This can also be combined with the assignment operator
- $greeting .= $name;

Notes

Crash Course in Binary

- We use a base 10 numbering system (0-9), computers use base 2 (0-1).

Binary									Base 10
128	64	32	16		8	4	2	1	
0	0	1	0		1	0	1	0	42
0	0	1	0		0	1	1	1	39
0	1	1	0		0	1	0	1	101
1	0	0	0		0	0	0	0	128
1	0	1	0		0	1	1	0	166
1	1	1	1		1	1	1	1	255

Notes

Bitwise Operators

- & (and)
  - If the bit in the same location for both the left and right operand is 1, the output bit will be one
- | (or)
  - If the bit in either the left or right operand is one, the output bit will be one
- ^ (XOR)
  - If the bit in either the left or right operand is one, but not both, the output will be one

Notes

Using Bitwise Operations

- What is the output of:
  - 1 & 2
  - 2 & 7
  - 15 & 24
  - 1 | 2
  - 16 | 15
  - 8 ^ 6
  - 15 ^ 7

Notes

Bitwise Operators, Cont'd

- ~ (NOT)
  - Inverts every bit, 0s become 1s, 1s become 0s
- >> (Shift Right)
  - All of the bits in the binary number shift N places to the right in the number, the right most digit(s) falls out, and the resulting number is padded on the left with 0s. A right shift of one is equivalent to dividing a number by two, and tossing the remainder
- << (Shift Left)
  - All of the digits shift N places to the left, padding the right with 0s. A left shift of one is equivalent to multiplying a number by two

Notes

Using Bitwise Operators

- What is the output of (within the context of our 8 bit binary world):
  - ~254
  - 4 >> 2
  - 1 << 2
  - 5 >> 2
  - 7 << 3

Notes

Other Operators

- @
  - Suppresses errors from the following expression
- `` (backticks)
  - Execute the contents as a shell command (shortcut for shell_exec())
- Instanceof
  - Returns true if the indicated variable is an instance of the designated class, one of it's subclasses or a designated interface

Notes

Using Other Operators - @

- @ - Error Suppression

```
ini_set("ERROR_LEVEL", E_STRICT);
$a = @myFunction();
function myFunction($a)
{
    array_merge(array(), "");
}
```


Notes

Using Other Operators - ``

- The backtick is dangerous, it executes with the same permissions as the webserver, and is easy to miss when skimming code, use shell_exec() instead, if you must.

```
$ls = `ls`;
```


Notes

Using Other Operators – instanceof

- The instanceof operator is useful (often combined at the function level with typehinting) to ensure you are given what you expect.

```
$a = new stdClass;
if ($a instanceof stdClass) { ... }
if ($a !instanceof stdClass) { ... }
     //Parse Error
```


Notes

Control Structures

- If
  - Alternate
  - Short
- Switch
- While
- do
- For
  - Continue
  - break
- Foreach
- Functions
  - Paramaters
    - Optional
    - Byref
    - Byval
    - Objects
    - Variable functions
- Objects
  - Later!

Notes

if

- if – The most basic, and frequently used conditional

```
if ($a === 5)
{
  echo "a is identical to 5";
}elseif ($a == 5)
{
  echo "a is equal to 5";
}else
{
    echo "a is neither equal to or identi
cal to 5";
}
```


Notes

if – Alternate Syntax

- Rarely used, but it works:

```
if ($a == 5):
    echo"hi";
    echo "they are equal";
elseif ($a === 5):
    echo "they are also identical";
endif;
```


Notes

if – Short Syntax

- This one's actually used

```
$action = (empty($_POST['action']))
          ? 'default' : $_POST['action'];
```

- it a lot in places where they want to keep these sorts of things short

Notes

Switch

- Switch statements operate on a single item, but are faster than if statements:

```
switch($a)
{
    case 1:
        echo "1";
    case 2:
        echo "2";
        break;
    default:
        echo "3";
}
```


Notes

do

- With the do loop the condition is executed at the termination of the condition, rather than the beginning

```
$a = 12;
do
{
    echo $a;
}while($a++ < 10);
echo $a;
```

- Output?

Notes

while

- With a while loop the condition is executed at the head of the loop

```
$a = 0;
while ($a++ < 10)
{
    echo $a;
}
echo $a;
```

- Output?

Notes

for

- Easy way to combine all the regular parts of an iteration loop:

```
for ($a = 0; $a < 10; $a++)
{
    echo $a;
}
for ($a = 0, $b = 10; $a * $b < 100;
     $a++, $b++)
{
    echo $a;
}
```


Notes

foreach

- Fast quick way to iterate over an array

```
foreach ($students as $name => $grade)
{
    echo "$name got a $grade";
}
```

- A note about objects in foreach loops

Notes

continue

- Jump out of current iteration of the loop, and continue with the next one

```
for($a = 0; $a < 100; $a++)
{
    if (!($a % 7))
    {
        continue;
    }
    echo $a . "\n";
}
```


Notes

break

- Exit loop unconditionally and continue with code execution

```
$a = 0;
$b = 7;
while ($a++ < 100)
{
    while ($b++ < 40)
    {
        if (($a * b) > 80)
        {
            break 2;
        }
    }
}
```


Notes

return

- Return ends execution and returns control to the calling scope, functions both inside an included file (returning control to the calling file) and from inside a function/method (returning control to the calling code)

Notes

exit;

- Halt execution of the script, can optionally return a message, or an exit status (0-255)
  - exit();
  - die();
  - exit('couldn't load file');
  - exit(1);
  - exit(0);

Notes

A Basic Conclusion

- PHP Tags, multiple types with different use rules (which ones are on/off by default, which ones a guaranteed?)
- PHP is loosely typed, but there is typing going on underneath the hood
- Some assumptions are made (octal format)
- Binary is a pain in the neck, but it has its uses
- Control structures have formats other than the ones commonly uses

Notes

Questions for further study

- Practice casting back and forth between variables of different types (int, string, double, boolean, array, object, null, resource)
- What's the output? Trace **carefully** then test:

```
for($a = 013; $a < 27; $a++)
{
    $b = $a;
    $a += $b % 2;
    $b = $b >> 2;
    echo $b . " ";
}
echo $b * $a;
```


Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

PHP 4 / 5 Differences

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - **PHP 4 vs PHP 5**
  - Web Features
  - XML

Notes

Shortest Section

- The majority of differences between PHP 4 and PHP 5 have been covered in their relevant subsections
- If you handed the transition yourself, you're likely already familiar with most of them
- This is just a re-cap

Notes

Object Orientation

- ByRef rather than copying
- $myClone = clone $original;
- __clone() function called on clone if it exists
- Constructor and Destructor
- Magic Methods (in OO Section)
- Class constants
- SimpleXML
- PDO
- SPL
- instanceOf
- Type hinting

Notes

E_STRICT

- Raises notices when using old or outdated coding methods
- Not included in E_ALL
  - Non static method calls
  - By reference issues
  - var key word

Notes

Summary & Homework

- Go read
  - http://ca3.php.net/manual/en/faq.migration5.php
- Objects got better, faster
- The way we handled objects got better
- E_STRICT is available, it's not included in E_ALL

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

XML and Web Services

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - **XML and Web Services**

2

Notes

XML and Web Services

- XML Basics
- SimpleXML
- XML Extensions
- Xpath
- Webservices Basics
- SOAP
- REST

3

Notes

XML

- Well Formed
  - Single root level tag
  - Tags must be opened and closed properly
  - Entities must be well formed
- Valid
  - Well formed
  - Contain a DTD
  - Follow that DTD

4

Notes

XML Sample

```
<!DOCTYPE books SYSTEM "http://
  www.example.com/formats/books.dtd">
<bookshelf>
  <book>Tom &amp; Jerry&apos;s, a history
  of</book>
  <book>2 + 2 &lt 5, Math for beginners</
  book>
</bookshelf>
```

5

Notes

Parsing XML

- SAX
  - Simple API for XML
- Event based approach where every 'event' encountered must be coded for and handled
- Events include open tag, close tag, tag data
- DOM
  - Document Object Model
- Loads entire document into ram, then creates an internal tree representation

6

Notes

SimpleXML

- Uses DOM method
- Very easy to use
- Memory Intensive
- http://ca.php.net/simplexml

7

Notes

SimpleXML

```
$library = <<<XML
<library>
    <metaInfo>
        <owner>Paul Reinheimer</owner>
        <location>Montreal</location>
        <tag>My Books</tag>
    </metaInfo>
    <books>
        <book>
            <author>Paul Reinheimer</author>
            <title>Professional Web APIs with PHP</title>
        </book>
        <book>
            <author>Orson Scott Card</author>
            <title>Ender's Game</title>
        </book>
        <book>
            <author>Tom Hold</author>
            <title>In Your Dreams</title>
        </book>
    </books>
</library>
XML;
```

8

Notes

SimpleXML

```
$xml = simplexml_load_string($library);

echo $xml->metaInfo->owner . "'s Library \n";
foreach($xml->books->book AS $book)
{
    echo $book->author . " wrote " . $book->title . "\n";
}
```

```
Output:
Paul Reinheimer's Library
Paul Reinheimer wrote Professional Web APIs with PHP
Orson Scott Card wrote Ender's Game
Tom Hold wrote In Your Dreams
```

9

Notes

XPath

- W3C Standard supported by many languages
- Combines the mojo of Regex with some SQL like results

10

Notes

XPath Searches

- xpath(“item”) - will return an array of item objects
- xpath(“/bookshelf”) - will return all children of the forum node
- xpath(“//book”) - will return an array of title values
- xpath(“.”) - will return the current node, <bookshelf>
- xpath(“..”) - will return an empty array, root node

<bookshelf> does not have a parent element.

11

Notes

Web Services

- Web Services is a generic umbrella that describes how disparate systems can integrate with each other over the web
- Most major sites offer some form of web services:
  - Amazon
  - FedEx
  - eBay
  - PayPl
  - del.icio.us

12

Notes

Why use Web Services?

- Someone else has data you need
- Easier than scraping pages
  - also more reliable
- Automate processes
- Provide additional information to your clients

13

Notes

REST

- Representational State Transfer
- Requests look like filled out forms, can use either GET or POST
- Response is an XML document
- Request and Response is defined by the service provider
- Very simple and popular

14

Notes

REST Request

- `http://library.example.com/api.php?devkey=123&action=search&type=book&keyword=style`

15

Notes

REST Response

```
<?xml version="1.0" encoding="UTF-8"?>
<LibraryAPI
    xmlns="http://library.example.com/api/spec">
<Request>
 <RequestId>123a456</RequestId>
 <Paramaters>
  <Argument Name="devkey" Value="123" />
  <Argument Name="action" Value="search" />
  <Argument Name="type" Value="book" />
  <Argument Name="keyword" Value="style" />
 </Paramaters>
</Request>
```

16

Notes

REST Response Continued

```
<Response>
 <ResultCount>2</ResultCount>
 <Item>
  <Title>Style Book Vol 1</Title>
  <Status>Out</Status>
  <Holds>3</Holds>
  <CopiesOnHand>2</CopiesOnHand>
  <Author>Jon Doe</Author>
 </Item>
 <Item>
  <Title>Style Book Vol 2</Title>
  <Status>In</Status>
  <Holds>0</Holds>
  <CopiesOnHand>1</CopiesOnHand>
  <Author>Jon Doe</Author>
 </Item>
</Response>
```

17

Notes

SOAP

- Simple Object Access Protocol
  - Doesn't actually stand for anything anymore
- Requests are sent using POST
- Requests are encapsulated within a SOAP Envelope
- Responses are XML documents with similar Envelope & Body sections

18

Notes

WSDL Documents

- SOAP web services are described by WSDL files
- They are designed for automated (not human) consumption
- Describes the methods offered by the web service, the parameters required, and the return type, as well as all complex types required by the service

19

Notes

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <SOAP-ENV:Body>
    <devkey xsi:type="xsd:int">123</devkey>
    <action xsi:type="xsd:string">search</action>
    <type xsi:type="xsd:string">book</type>
    <keyword xsi:type="xsd:string">style</keyword>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

20

Notes

SOAP Response

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <SOAP-ENV:Body>
 <LibrarySearchResponse xmlns="http://
library.example.com/api/ns">
  <RequestInfo>
    <devkey xsi:type="xsd:string">123</devkey>
    <action xsi:type="xsd:string">search</action>
    <type xsi:type="xsd:string">book</type>
    <keyword xsi:type="xsd:string">style</keyword>
  </RequestInfo>
```

21

Notes

SOAP Response Cont'd

```
<ResponseInfo>
   <ResultCount>2</ResultCount>
   <Item>
    <Title xsi:type="xsd:string">Style Book Vol 1</Title>
    <Status xsi:type="xsd:string">Out</Status>
    <Holds xsi:type="xsd:int">3</Holds>
    <CopiesOnHand xsi:type="xsd:int">2</CopiesOnHand>
    <Author xsi:type="xsd:string">Jon Doe</Author>
   </Item>
   <Item>
    <Title xsi:type="xsd:string">Syle Book Vol 2</Title>
    <Status xsi:type="xsd:string">In</Status>
    <Holds xsi:type="xsd:int">0</Holds>
    <CopiesOnHand xsi:type="xsd:int">1</CopiesOnHand>
    <Author xsi:type="xsd:string">Jon Doe</Author>
   </Item>
  </ResponseInfo>
 </LibrarySearchResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

22

Notes

PHP5's SOAP Support

- PHP5 has built in SOAP support
- The functions (generally) take a WSDL file as input, and create an object that mimics the services of the webservice:

  $client = new SoapClient("http://soap.amazon.com/schemas2/AmazonWebServices.wsdl");

23

Notes

PHP5's Built in SOAP Support

- Making API calls is now trivial:

```
$result = $client->KeywordSearchRequest
($params);
```

24

Notes

PHP5 SOAP Server

- The internal SOAP support doesn't include creation of WSDL files
  - NuSOAP does
  - Zend Studio does
- You can still provide a service either by using an external WSDL file, or merely defining the access methods

25

Notes

XML & Web Services Conclusion

- XML is a great storage device, SimpleXML makes it easy to use
- XPath is powerfull, and scary
- REST is very easy to use, requests are trivial, responses are just XML
- SOAP is a bit tougher if you try to do it from scratch, built in support makes it equally trivial

26

Notes

Homework

- Find some XML files (from this class, RSS feeds, etc)
  - Open them up in SimpleXML
  - Read various elements directly
  - Deal with namespaces
- Find a simple REST web service, make a few calls
- Find a simple SOAP web service, make a few calls

27

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Web Features

Course Author: **Paul Reinheimer**

Forms

- GET vs POST
- Form Tokens
- Default Values
- Re-populating data

2

Notes

Cookies

- Client side data store
  - Can be deleted, manipulated, copied
  - Behavior is inconsistent
- Header();
  - Set cookies manually, using the RFC for the appropriate headers
- Set_cookie();
  - Wraps the Header function, sets default values when nothing is passed.

3

Notes

Sessions

- Sessions, The safer way to state
- Use a cookie that contains a Session ID
- That Session ID corresponds with a local(ish) data store that contains the user's information
- The Session ID is the only data that is transmitted to a client, and is also the only thing that identifies them
- Session Hijacking and Session Fixation

4

Notes

HTTP Headers

- Header(); redirects
  - Do they work?
- Other arbitrary headers
  - Header injection attacks
  - Caching
  - Content-Type
  - Meta Information

5

Notes

Web Features Conclusion

- Data comes in from GET or POST, frequently from forms, don't trust it
- Cookies are a client side data store
- Sessions use cookies to offer a localish data-store
- Header allows us to send meta information to the client
-

6

Notes

Homework

- Create a cookie acceptance checking script
  - Set a cookie manually
  - Use a redirect
  - see if it's there
  - Do it with 1 script
- Play with sessions
  - What can you store in them
  - Can you write your own session handler?

7

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Supplementary Material + Review

Course Author: **Paul Reinheimer**

Supplementary Material

- If you're reading this, we've got some extra time left over, so let's look at some material in greater detail, or cover some things just assumed previously

Notes

Strings - String Functions

- PHP has a *large* collection of built in functions to handle strings, we've looked at strings already, let's look at different ways to interact with them in greater detail

Notes

strlen()

- Returns the length of the string passed to it:

```
echo strlen('Jello'); //5
$a = array("Hello World", "a");
echo strlen($a); //?
```


Notes

str_replace() & strtr()

- Searches the given string for the needle, replacing it with the presented string:

```
echo str_replace("a", "b", "abc"); //bbc
echo str_replace(array('a','b','c'), array('b','c','a'), 'abc');
   //?

echo strtr('abc','ab','bc'); //bcc
```


Notes

strcmp() & strcasecmp()

- These functions compare two strings, returning zero when they match, greater than one when string 1 > string 2, and less than one when string 1 < string 2

```
echo strcmp('hello', 'Hello'); // > 0
echo strcasecmp('hello', 'Hello'); // ?
```


Notes

strpos() & strstr()

- strpos will return the position of the search string within the target. strstr will return the portion of target from the beginning of the first match until the end. Both function have case insensitive counterparts

```
echo strpos('hello new world', 'new');    //6
echo strstr('hello new world', 'new');  //new world
```


Notes

strspn() & strcspn()

- Apply a mask to a given string, strspn() returns the number of characters matching the mask, strcspn() uses a blacklist and returns the number of non-matching characters:

```
$mask = "abcdefg ";
$string = "beg bad fed billy"; //strlen($string) = 17
echo strspn($string, $mask); //13

$mask = "abcdefg";
$string = "joy to the world"; //strlen($string) = 16
echo strcspn($string, $mask); //9
```


Notes

substr()

- substr() – Used to retrieve a portion of a string

```
$string = "I am the very model of the modern major general";
echo substr($string, 2);
  //am the very model of the modern major general
echo substr($string, 9, 10);    //very model
echo substr($string, -7);       //general
echo substr($string, 5, -8);
  //the very model of the modern major
```


Notes

Printf formatting specifiers:

- % - a literal percent character. No argument is required.
- b - the argument is treated as an integer, and presented as a binary number.
- c - the argument is treated as an integer, and presented as the character with that ASCII value.
- d - the argument is treated as an integer, and presented as a (signed) decimal number.
- e - the argument is treated as scientific notation (e.g. 1.2e+2).
- u - the argument is treated as an integer, and presented as an unsigned decimal number.
- f - the argument is treated as a float, and presented as a floating-point number (locale aware).
- F - the argument is treated as a float, and presented as a floating-point number (non-locale aware). Available since PHP 4.3.10 and PHP 5.0.3.
- o - the argument is treated as an integer, and presented as an octal number.
- s - the argument is treated as and presented as a string.
- x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
- X - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

Notes

String Questions:

- strlen(null);
- Use printf() to format $a = 123.4567 as “$123.45”
- Using a combination of string functions write one code block that will extract “hello world” from “he said hello world” and “hello molly” from “hello molly he said”
- What does a string cast to an array give you?
- Which syntax is preferred in PHP5? $a[0] or $a{0}

Notes

Arrays

- Arrays are a way of storing multiple peices of related information, you can store data in either an associative or an enumerated array
- In enumerated array values are keyed numerically, when you allow PHP to assign the key it takes the highest numeric key used thus far, and adds one
- In associative arrays strings are used to key the data, keys containing only digits are cast to integers

Notes

array_keys() & array_values()

- These two functions will split an array up, giving you either the keys or values from an array respectively:

```
print_r(array_keys(array('a' => 'b', 'c'=>'d')));
Array
(
    [0] => a
    [1] => c
)
print_r(array_values(array('a' => 'b', 'c'=>'d')));
Array
(
    [0] => b
    [1] => d
)
```


Notes

Merging Arrays

- You can merge arrays using array_merge(), when different associative arrays have duplicate keys, the values in the rightmost array take precedence

```
$a = array('a' => 'b', 'c'=>'d');
$b = array('a' => 'e', 'b'=>'d');
print_r(array_merge($a, $b));
/*
Array
(
    [a] => e
    [c] => d
    [b] => d
)
*/
```


Notes

array_splice()

Cut out a chunk of an array:

```
$a = range(0, 20, 2);
$a = array_slice($a, 4, 3);
print_r($a);
/*
Array
(
    [0] => 8
    [1] => 10
    [2] => 12
)
*/
```


Notes

Array Questions

- $a = array(1,2,3,4);
  Remove 2
- Describe how you would hold information on a series of students, their name, email and grade
- What does array_merge_recursive() do?
- $b = array('23' => 'cat', 'dog', 'beaver', 7);
  $b[] = $b;
  Give me both ways to get 'dog'
- Give me the first letter of dog

Notes

SQLite

- SQLite is a database, without the database
- Basically, rather than using a separate program to persistently maintain the database, SQLite on the other hand requires the C libraries that comprise the DB to be built into whatever program would like to use them
- SQLite was built into PHP by default as of PHP5, which is cool, some twits turn it off, which isn't
- It's fast, free, and has nice licensing terms

Notes

SQLite Continued

- Apart from not needing to connect to a remote server or process, SQLite is no different from other database systems:

```
$db = sqlite_open('mysqlitedb', 0666, $sqliteerror)
sqlite_query($db, 'CREATE TABLE foo (bar varchar(10))');
sqlite_query($db, "INSERT INTO foo VALUES ('fnord')");
$result = sqlite_query($db, 'select bar from foo');
```

- Take a look at the manual pages for more information on the function library

Notes

MySQL vs MySQLi

- Recent versions of PHP no longer ship with the MySQL extension built in, you must enable it at configure time.
- We new have two versions of the MySQL extension available, MySQL and MySQLi
- The MySQLi extension makes some of MySQL's more recent functionality available, things like prepared statements

Notes

Objects

- Objects can be used to merge togethor data and the methods that manipulate it
- You can define the interface of an object using either abstract classes or interfaces
- Abstract classes can define the methods that extending classes must offer by declaring them with the abstract keyword. They may also create their own methods
- Interfaces simply define the methods that an implementing object must include

Notes

Abstraction

```
abstract class example
{
    public $name;
    abstract public function displayExample();
    abstract protected function prepExample();
    public function __construct($name)
    {
        $this->name = $name;
    }
}

class myExample extends example
{
    public function displayExample() { echo $this->name; }
    public function prepExample() {
                        $this->name = md5($this->name);;}
}
```


Notes

Interfaces

```
interface inputData
{
    public function alpha ($data);
    public function alnum ($data);
}
interface outputData
{
    public function web ($data);
    public function sql ($data);
}
class templateEngine implements inputData, outputData
{
    public function alpha($a) {}
    public function alnum($b) {}
    public function web($c, $d = "") {}
    public function sql($e) {}
    public function iStillWork() {}
}
```


Notes

Built in Interfaces

```
print_r(get_declared_interfaces());
/*
Array
(
    [0] => Traversable
    [1] => IteratorAggregate
    [2] => Iterator
    [3] => ArrayAccess
    [4] => Serializable
    [5] => RecursiveIterator
    [6] => OuterIterator
    [7] => SeekableIterator
    [8] => Countable
    [9] => SplObserver
    [10] => SplSubject
    [11] => Reflector
)
*/
```


Notes

Exercises

- Create an abstract class, extend it
  - Can you define an abstract method as private?
- Create an interface, implement it
  - Can you define an interface as having private methods?
- Implement the Iterator interface
  - foreach() over your new object :)
- Play with some of those string and array function

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Exam

Zend Certification Course

Functions

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - **Functions**
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

Functions

- If you don't know what functions are…
- We use functions for several major reasons
  - Readability
  - Maintainability
  - Code separation
  - Modularity
  - Our Comp-Sci teacher told us to
  - To do otherwise would subject us to mockery

Notes

Functions.. Examples?

```
function myName()
{
  echo "Paul";
}

function yourName($name)
{
  echo $name;
}
```


Notes

Functions - Scope

```
function scope($count)
{
  $count = 100;
  echo $count;
}

$count = 3;
scope($count);
echo $count;
```


Notes

Functions - Examples

```
function incrementID()
{
  global $id;  //GLOBALS['id'];
  $id++;
}

function x10($number)
{
  return $number * 10;
}
```


Notes

Functions - Examples

```
function printList($string, $count = 5)
{
  for ($i = 0; $i < $count; $i++)
  {
    echo $string;
  }
}
```


Notes

Functions - Examples

- function newTo5(&$number = 2)
{
 echo ++$number;
}

Notes

Functions - Examples

- function takeWhatYouGet()
{
 $a = func_get_args();
 foreach ($a as $key => $value)
 {
  echo "In slot $key I found $value how excit
ing! \n";
 }
}
//takeWhatYouGet(3, 2, "paul", "bob", 28);

Notes

Functions - Examples

- function andLikeIt($shouldILook = TRUE)
  {
  if ($shouldILook)
  {
  $argCount = func_num_args();
  for ($i = 0; $i < $argCount; $i++)
  {
  echo "In slot $i I found " . func_get_arg
  ($i) . " how very exciting! \n";
  }

Notes

Functions & Objects, Friends at last

- In PHP4 Objects were thrown around ByVal, this meant that you made copies without even thinking about it.
- In PHP5 Objects are always passed ByRef unless you explicitly clone it, keep that in mind

Notes

Functions - Conclusion

- They're terribly useful
- You have to remember how scope works
  - Practice tracing
- Objects fly around by reference now
- Return statements can exist anywhere within the function, you can even have several
- Avoid situations where the same function may return nothing, or something (Zend Studio's Code Analyzer tests for this)

Notes

Questions for further study

- Develop a function that accepts a variable number of parameters
- Experiment with variable scope, ensure you are comfortable with how local and global variables interact
- What happens if a function is defined then the PHP scope ends, then resumes again later with the remainder of the function
- Is "_" a valid function name?

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Strings

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - **Strings**
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

Little bits of string

- Strings are the most commonly used variable type in PHP, because they are both central to web development, and the common method of data transmission from the user
- Within strings many characters take a special meaning, matching quotes, back slashes, octal numbers, variables, if you wish to accept them as they are you must escape them with the \ character.

Notes

Escape Sequences

- PHP Accepts several escape sequences
  - \n linefeed (LF or 0x0A (10) in ASCII)
  - \r carriage return (CR or 0x0D (13) in ASCII)
  - \t horizontal tab (HT or 0x09 (9) in ASCII)
  - \\ backslash
  - \$ dollar sign
  - \" double-quote
  - \[0-7]{1,3} the sequence of characters matching the regular expression is a character in octal notation

Notes

Matching Strings

- strcmp() – Compare two strings. Returns < 0 if str1 is less than str2 ; > 0 if str1 is greater than str2 , and 0 if they are equal.
- strcasecmp() – Case insensitive version of strcmp()
  - Remember 0 == FALSE and 0 !== FALSE

Notes

Extracting

- substr() – Used to retrieve a portion of a string
-

```
$string = "I am the very model of the modern major general";
echo substr($string, 2);
  //am the very model of the modern major general
echo substr($string, 9, 10);    //very model
echo substr($string, -7);       //general
echo substr($string, 5, -8);
  //the very model of the modern major
```


Notes

Extracting – Array Syntax

-
```
$string = "I am the very model of the modern major general";
echo $string[5];    //t, the t from the
echo $string[0];    //I
echo $string{2};    //a, the a from am
exit;
```
- The {} syntax is depreciated in PHP 5.1 and will produce a warning under E_STRICT

Notes

Formatting

- number_format() – By default formats a number with the comma as the thousands separator, and no decimal

```
echo number_format("1234567.89");
    //  1,234,568
echo number_format("9876543.698", 3, ",", " ");
    // Shows 9 876 543,698
```


Notes

money_format()

- Only available when the underlying c library strfmon() is available (not windows)

```
$number = 1234.56;
setlocale(LC_MONETARY, 'en_US');
echo money_format('%i', $number) . "\n";
// USD 1,234.56

setlocale(LC_MONETARY, 'it_IT');
echo money_format('%.2n', $number) . "\n";
// L. 1.234,56
```


Notes

Printf family

- Part of a family: print(), sprintf(), vprintf(), sscanf(), fscanf(). All of which accept the same formatting options

Notes

Printf formatting specifiers:

- % - a literal percent character. No argument is required.
- b - the argument is treated as an integer, and presented as a binary number.
- c - the argument is treated as an integer, and presented as the character with that ASCII value.
- d - the argument is treated as an integer, and presented as a (signed) decimal number.
- e - the argument is treated as scientific notation (e.g. 1.2e+2).
- u - the argument is treated as an integer, and presented as an unsigned decimal number.
- f - the argument is treated as a float, and presented as a floating-point number (locale aware).
- F - the argument is treated as a float, and presented as a floating-point number (non-locale aware). Available since PHP 4.3.10 and PHP 5.0.3.
- o - the argument is treated as an integer, and presented as an octal number.
- s - the argument is treated as and presented as a string.
- x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
- X - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

Notes

Printf examples

- ```
  $number = 1234.5678;
  printf("%d", $number);      //1234
  printf("%f", $number);
          //1234.567800 (6 Decimal Places)
  printf("%8d", $number);     //    1234
  printf("%'x8d", $number);   //xxxx1234
  printf("%8.2f", $number);   //1234.57
  printf("%8.2f", $number);   //1234.57
  ```


Notes

Replacement

- ```
  $string = "I want steak with a side of salad";
  echo str_replace("salad", "fries", $string);
          //I want steak with a side of fries
  echo str_replace(array("steak", "salad"), array
  ("burger", "taco"), $string);
          //I want burger with a side of taco
  echo str_replace(array("steak", "burger"), array
  ("burger", "taco"), $string);
          //I want taco with a side of salad
  ```


Notes

PCRE

- PCRE – Perl Compatible Regular Expressions, PHP's most popular RegEx engine
- Technically you can turn it off, no one does
- Exceedingly powerful, not always the fastest choice, if you can use a built in function do so.

Notes

PCRE – Meta Characters

- \d Digits 0-9
- \D Anything not a digit
- \w Any alphanumeric character or an underscore (_)
- \W Anything not an alphanumeric character or an underscore
- \s Any whitespace (spaces, tabs, newlines)
- \S Any non-whitespace character
- . Any character except for a newline

Notes

PCRE – Meta Characters

- ? Occurs 0 or 1 time
- * Occurs 0 or more times
- + Occurs 1 or more times
- {n} Occurs exactly n times
- {,n} Occurs at most n times
- {m,} Occurs m or more times
- {m,n} Occurs between m and n times

Notes

PCRE – Pattern Modifiers

- The way the PCRE engine works can be modified by modifiers placed after the terminating character
  - i – Case insensitive search
  - m – Multiline, $ and ^ will match at newlines
  - s – Makes the dot metacharacter match newlines
  - x – Allows for commenting
  - U – Makes the engine un-greedy
  - u – Turns on UTF8 support
  - e – Matched with preg_replace() allows you to call

Notes

preg_match()

- preg_match() – Returns the number of matches found by a given search string under this format, also capable of returning the match:

```
$string = "124 abc";
var_dump(preg_match("/\w\s
\w/", $string)); // 1 (matches);
var_dump(preg_match("/\d{3}\s[a-z]
{3}/", $string)) // 1 (matches)
```


Notes

preg_replace()

- Searches the subject for matches to a given pattern replaced with the given replacement text
-
```
$pattern = "!(java|ruby)!i";
$string = "I love programming in Java, I als
o love ruby!";
echo preg_replace
($pattern, "php", $string);
```
- //I love programming in php, I also love php!

Notes

preg_replace()

```
$bbcode = "Hi !!Paul!!, how are !!you!! today\n";
echo preg_replace("/!!(.+)!!/e", "strtoupper('\
     \1')", $bbcode);
//Hi PAUL!!, HOW ARE !!YOU today

echo preg_replace("/!!(.+)!!/eU", "strtoupper('
    \1')", $bbcode);
//Hi PAUL, how are YOU today
```

- Security concerns, escape input

Notes

preg_split()

- preg_split() allows you to split strings based on more complex rules:

```
$ws = "Too      much     white
      space";
print_r(preg_split("!\s+!", $ws));
Array(
 [0] => Too
 [1] => much
 [2] => white
 [3] => space
)
```


Notes

Strings & Patterns Conclusion

- Escaping, worth thinking about
- Heredoc vs “ vs ‘
- printf() and its family
- PCRE
  - meta characters
  - pattern modifiers
  - functions

Notes

Questions for further study

- Develop a regular expression to pull email addresses from a web page using the standard paul at preinheimer dot com format.
- Experiment with the printf() modifiers and research the related functions
- There are **many** string functions in PHP experiment and research a few
- Try using the various escape sequences with all three quote types (single, double, heredoc)

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Exam

Zend Certification Course

Arrays

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - **Arrays**
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

Enumerated Arrays

- Numeric keys, either auto assigned or assigned by you, great when you don't care how the data is indexed

```
$students = array();
$students[] = "Paul";
$students[] = "Fred";
$students[] = "Bob";
print_r($students);
```


Notes

Enumerated Arrays

```
Array
(
    [0] => Paul
    [1] => Fred
    [2] => Bob
)
```


Notes

Associative Arrays

- Frequently used with databases, or when there should be some key=>value association

```
$students = array();
$students['Paul'] = 'Canada';
$students['Bob'] = 'USA';
$students['Winston'] = 'United Kingdom';
$students['David'] = 'South Africa';
print_r($students);
```


Notes

Associative Arrays

```
Array
(
    [Paul] => Canada
    [Bob] => USA
    [Winston] => United Kingdom
    [David] => South Africa
)
```


Notes

Multi Dimensional Arrays

- Multi-Dimensional arrays, or arrays of arrays can contain a lot of data, but can also be complicated to access

```
$students = array();
$students[] = array
('name' => 'paul', 'location' => 'canada');
$students[] = array
('name' => 'bob', 'location' => 'USA');
print_r($students);
```


Notes

Multi-Dimensional Arrays

```
Array
(
    [0] => Array
        (
            [name] => paul
            [location] => canada
        )
    [1] => Array
        (
            [name] => bob
            [location] => USA
        )
)
```


Notes

Arrays - Questions

```
$a = array(5=>"b", 7, 9, "23" => 'c', 'j');
$b = array(1, 2, 3, 'f', 'fred' => $a);
```

- How do I access the value:
  - 7
  - ‘c’
  - ‘b’ through the array $b
  - ‘f’

Notes

Array Iteration

- When iterating through an array via foreach() keep in mind it operates on a copy of the array
- Using the internal array pointer doesn't have this problem

Notes

Array Iteration

- **end()** – Move pointer to last element
- **key()** - Retreives key from current position
- **next()** – Advances array pointer one spot then returns current value
- **prev()** – Rewinds the array pointer one spot, then returns the current value
- **reset()** – Resets the array pointer to the beginning of the array
- **each()** – Returns the current key and value from the array, then iterates

Notes

Array Iteration - Example

```
$fruit = array('a' => 'apple', 'b' => 'banana', 'c'
    => 'cranberry');
reset($fruit);
while (list($key, $val) = each($fruit)) {
  echo "$key => $val\n";
}
//a => apple b => banana c => cranberry
```


Notes

Foreach()

- In PHP 5 items can be iterated over by reference rather than simply by value
- foreach($array AS &$value) { … }

Notes

Sorting Arrays

- Sorting arrays comes up frequently, but can be frequently be offloaded to an external system (databases are a common system) which may be able to do it faster
- The type of sorting that must be done depends entirely on the data contained within the array

Notes

Array Sort

- Original Array:

```
Array
(
    [Paul] => Canada
    [Bob] => USA
    [Winston] => United Kingdom
    [David] => South Africa
)
```


Notes

Array Sort - sort()

- sort()

```
Array
(
    [0] => Canada
    [1] => South Africa
    [2] => USA
    [3] => United Kingdom
)
```


Notes

Array Sort - ksort()

- Ksort()

```
Array
(
    [Bob] => USA
    [David] => South Africa
    [Paul] => Canada
    [Winston] => United Kingdom
)
```


Notes

Array Sort - asort()

- asort()

```
Array
(
   [Paul] => Canada
   [David] => South Africa
   [Bob] => USA
   [Winston] => United Kingdom
)
```


Notes

Stacks & Queues

- Arrays can also be treated like a stack or a queue:

```
$fruit = array
("apple", "banana", "cantaloupe");
array_push($fruit, "dragonfruit");
//apple,banana, cantaloupe, dragonfruit

echo array_pop($fruit); //dragonfruit
//apple, banana, canteloupe
```

- Stack, implementing LIFO

Notes

Stacks and Queues

- Generally array_shift() will be combined with array_push() for a Queue implementation

```
$fruit = array("banana", "cantaloupe");
array_unshift($fruit, "apple");
//apple, banana, cantaloupe

echo array_shift($fruit); //apple
//banana, cantaloupe
```


Notes

Array Functions

- There's a lot: http://ca.php.net/array/
- A few of note:
  - array_walk()
  - array_keys()
  - array_values()
  - array_change_key_case()

Notes

An array of conclusions

- Arrays are perfect for storing a set of related data
- Remember precisely how to access various elements, the syntax can be tricky
- There's lots of functions, play around a bit
- Internal array pointers can be reset, but don't reset on their own

Notes

Questions for further study

- Develop implementations of FIFO and LIFO using built in array functions
- Experiment with various array sorting methods, try mixing variable types (string, integer, float)
- Implement a sorting mechanism using the usort() function
- Research other array functions (array_keys, array_values, _splice, _walk, etc)

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Files and Stream

Course Author: **Paul Reinheimer**

Files

- Files provide a simple temporary to permanent data store
- PHP5 presents two functions which make handling files trivial, namely file_get_contents () and file_put_contents()
  - file_get_contents() was backported to PHP4 because it was so useful
- For large amounts of data, or in issues where a race condition is possible a database would be a better data store

Notes

Files

- Many file functions still use the file pointer provided by fopen()

```
$fp = fopen(__FILE__, 'r');
if ($fp === FALSE)
{
    echo "Failed to open file";
    exit;
}
//fread(), fgets(), fwrite(), fstat()
fclose($fp);
```


Notes

File Modes

- ***mode*** **Description**
- *'r'* Open for reading only; place the file pointer at the beginning of the file.
- *'r+'* Open for reading and writing; place the file pointer at the beginning of the file.
- *'w'* Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- *'w+'* Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- *'a'* Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- *'a+'* Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- *'x'* Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the **fopen()** call will fail by returning **`FALSE`** and generating an error of level **`E_WARNING`**. If the file does not exist, attempt to create it.
- *'x+'* Create and open for reading and writing; place the file pointer at the beginning of the file. If the file already exists, the **fopen()** call will fail by returning **`FALSE`** and generating an error of level **`E_WARNING`**.

Notes

File System Functions

- Copy
- Move/rename
- Unlink
- Stat
- Read
- Write
- Append

Notes

fstat

```
<?php
        $fp = fopen(__FILE__, "r"); // open file
        print_r(fstat($fp)); // get array of info about the file
        /*
        [dev] => 5633 // device
        [ino] => 1059816 // inode
        [mode] => 33188 // permissions
        [nlink] => 1 // number of hard links
        [uid] => 1000 // user id of owner
        [gid] => 102 // group id of owner
        [rdev] => -1 // device type
        [size] => 106 // size of file
        [atime] => 1092665414 // time of last access
        [mtime] => 1092665412 // time of last modification
        [ctime] => 1092665412 // time of last change
        [blksize] => -1 // blocksize for filesystem I/O
        [blocks] => -1 // number of blocks allocated
        */
?>
```

- Stat does it without the file being open

Notes

Streams

- Steams are the way that PHP handles working with network resources
- Whenever you open up a file PHP creates a stream in the background
- Each stream consists of several components:
  - file wrapper
  - one or two pipe-lines
  - an optional context
  - metadata about the stream

Notes

Stream Meta Data

- ```
  $fp = fopen("http://www.php.net/", "r");
  print_r(stream_get_meta_data($fp));
  Array(
      [wrapper_data] => Array(
              [0] => HTTP/1.1 200 OK
              [1] => Date: Thu, 12 Oct 2006 15:34:57 GMT
              [2] => Server: Apache/1.3.37 (Unix) PHP/5.2.0-dev
              [3] => X-Powered-By: PHP/5.2.0-dev
              [4] => Last-Modified: Thu, 12 Oct 2006 15:21:50 GMT
              [5] => Content-language: en
              [6] => Set-Cookie: COUNTRY=CAN%2C24.201.61.253; expires=Thu, 19-
  Oct-2006 15:34:57 GMT; path=/; domain=.php.net
              [7] => Connection: close
              [8] => Content-Type: text/html;charset=utf-8)
      [wrapper_type] => http
      [stream_type] => tcp_socket
      [mode] => r+
      [unread_bytes] => 1088
      [seekable] =>
      [uri] => http://www.php.net/
      [timed_out] =>
      [blocked] => 1
      [eof] => )
  ```


Notes

Stream Context

```
<?php
$ctx = array('http' => array(
 'method' => 'HEAD', // make a HEAD request
 'header' => "Accept-language: de\r
             \n") // HTTP headers
);
$c = stream_context_create($ctx);
$fp = fopen('http://
www.php.net' , 'r', false, $c);?
?>
```


Notes
```

More about files

- Files can be locked to prevent race conditions

```
flock($fp, LOCK_SH);  //Place Shared lock
flock($fp, LOCK_EX);  //
Place Exclusive (write) Lock
flock($fp, LOCK_UN);  //Release lock
```

- Placing and removing locks can inhibit performance as traffic increases, consider using a different data store

Notes

Convenience Functions

- The information obtained via these functions is available in other ways (like using stat) but these wrappers make it easy

```
• is_dir($path);            //Returns if path is a directory
• is_executable($path);    //Returns if path is an executable
• is_file($path);
  - //Returns if path exists and is a regular file
• is_link($path);
  - //Returns if path exists and is a symlink
• is_readable($path);
  - //Returns if path exists and is readable
• is_writable($path);
  - //Returns if path exists and is writable
```


Notes

Sockets

- Using stream context and a few other techniques a lot can be done without resorting to sockets, however for more detailed control sockets can be used.
- Sockets allow read/write access to various socket connections, while things like HTTP may seem read only, at the protocol level information is being sent to identify the desired resource.

Notes

Sockets

```
$fp = fsockopen
("example.preinheimer.com", 80, $errno, $errstr, 30)
;
if (!$fp) {
   echo "$errstr ($errno)<br />\n";
} else {
   $out = "GET / HTTP/1.1\r\n";
   $out .= "Host: example.preinheimer.com\r\n";
   $out .= "Connection: Close\r\n\r\n";
    fwrite($fp, $out);
    while (!feof($fp)) {
       echo fgets($fp, 128);
   }
   fclose($fp);
}
```


Notes

Sockets

```
HTTP/1.1 200 OK
Date: Wed, 14 Mar 2007 13:39:55 GMT
Server: Apache/2.2.3 (Debian) mod_ssl/2.2.3 OpenSSL/0.9.8c
Last-Modified: Mon, 28 Aug 2006 01:56:37 GMT
ETag: "5f0348-59-3c7e3f40"
Accept-Ranges: bytes
Content-Length: 89
Connection: close
Content-Type: text/html; charset=UTF-8

This domain exists for demonstrations, You should be
looking at one of
those, not this.
```


Notes

Files of Conclusions

- Files are a great, easy to use datastore
- Locking is a pain in the neck, and may not handle load well
- fopen() returns a resource, this resource is used by many other file functions
- Modern applications should make use of the file_(get|put)_contents() functions

Notes

Questions for further study

- Using streams, submit a search request to google, print those results to the screen
- Retrieve the HTTP response headers from that request
- Take a look at file locking
- Try doing a submission using POST rather than get
- Store the results locally using file_put_contents()

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Security

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - **Security**
  - Databases
  - Design and Theory
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

Security Concepts

- Defense in Depth
- Principle of Least Privilege
- Terms: Validation vs Escaping

Notes

Defense in Depth

- When you plan with defense in depth, you plan for failure. Rather than internal functions assuming the data they receive is already validated or escaped, they will check and confirm.
- Applications that demonstrate defense in depth are not only more resistant to attack when they are developed, they also remain more resistant over time as new attacks are developed, and as more code is added to them

Notes

Principle of Least Privilege

- Your PHP applications by default has a lot of power, they execute as the web user (often apache or www-data) with all the rights and privileges thereof
- If an attacker gains control of PHP through an application vulnerability this can be used and abused

Notes

Validation vs Escaping

- These terms are often confused, or used interchangeably
- Validation or Filtering is the process by which you subject data to a series of rules, either it passes (validates) or does not
- Escaping is the process by which you prepare data for a specific resource by "escaping" certain portions of the data to avoid confusion of instruction and data

Notes

Validate Input

- Whenever you receive data one of three things can be said about it:
  - It is valid, the user entered the data you want, in the format you desire
  - It is invalid because the user either did not comply or did not understand the rules on the data you requested (eg. Poorly formatted phone number)
  - It is invalid because the user is attempting to compromise your system

Notes

Validate Input

- Data from the end user can not be trusted, it must be validated before it can be used
  - How does user data get into your PHP application?
- Validate data **first** don't save it for last
- Fail early, tell the user what went wrong

Notes

Whitelist vs Blacklist

- There are two major approaches to data validation, the whitelist and blacklist approach
- Under the whitelist approach you select a series of valid characteristics (frequently characters) only data that follows these rules is accepted as valid
- Under the blacklist approach you select a series of invalid characteristics, any data that contains these characteristics is considered invalid
- Which do you prefer?

Notes

Functions to examine

- ctype_alnum -- Check for alphanumeric character(s)
- ctype_alpha -- Check for alphabetic character(s)
- ctype_cntrl -- Check for control character(s)
- ctype_digit -- Check for numeric character(s)
- ctype_graph -- Check for any printable character(s) except space
- ctype_lower -- Check for lowercase character(s)
- ctype_print -- Check for printable character(s)
- ctype_punct -- Check for any printable character which is not whitespace or an alphanumeric character
- ctype_space -- Check for whitespace character(s)
- ctype_upper -- Check for uppercase character(s)
- ctype_xdigit -- Check for character(s) representing a hexadecimal digit
- preg_match – Apply regular expression rules to data
- is_array – Check to see if variable is an array

Notes

Register Globals

- With register globals enabled PHP automatically creates global variables for all variables received
- Register globals was great, when you were learning, and no one else saw your applications
- Register globals on it's own is **not** a security risk, register globals combined with *sloppy* coding is the problem

Notes

Register Globals

- To write secure code that remains secure in an environment with register globals enabled you must pre-initilize all your variables
- Code with E_STRICT enabled, it will warn you of variables that are used before initialization

Notes

Magic Quotes

- Magic Quotes automatically escapes quote characters, backslash and null to make it safer to send data to databases
- This isn't really a good thing, data goes places other than databases, database accept meta characters other than the ones magic quotes deals with

Notes

Magic Quotes

- Gone completely in PHP6
- Disabled by default in PHP5
- A pain in the neck, you end up with code like this:

```
if (get_magic_quotes_gpc())
{
  $string = stripslashes($string);
}
```


Notes

Escaping Output

- When you send data to a resource (database, web browser, etc.) that resource likely interprets certain characters as having a specific meaning, by escaping the output you tell the resource to ignore the meaning for that character

Notes

Escaping Output – Web Page

- strip_tags() – Remove anything that looks like an HTML tag from a string
- htmlentities() – Convert any character that has a specific HTML entity into it
- Htmlspecialchars() – Convert &, “, ‘, <, > into their entities

Notes

Escaping Output - SQL

- You have two choices, escape your data or use prepared statements
  - To escape your data use a database specific function, the newest one available
  - mysql_real_escape_string()
- Prepared statements are preferred as they offer perfect separation
  - http://usphp.com/manual/en/function.mysqli-prepare.php

Notes

XSS

- Under a cross site scripting attack an attacker injects code into your page (forum post, shout box, etc) that contains code that re-writes the page to do something nefarious
- This can be prevented through proper escaping and data validation techniques

Notes

CSRF

- Under a cross site request forgery attack a site exploits another sites persistent user trust relationship to make something happen
- `<img src=http://www.amazon.com/buy/my/book height="1" width="1">`
- iFrames are another common tool leveraged in this technique

Notes

Sessions

- Sessions provide safer state, it may not nesesarily be safe
- Basically, sessions combine cookies containing a session ID with a local(ish) data store corresponding to that session id
- If the session id is compromised, or the data store is not secure (/tmp on a shared machine) sessions are still vulnerable to attack

Notes

Session Fixation

- This attack has two sides, either an attacker tricks another user into clicking on a link providing a session id, or an innocent user pastes a link containing their session id to someone they shouldn't have
- To defend against this type of attack don't allow session IDs to come in over GET, and regenerate session ids when a user authenticates themself

Notes

Session Hijacking

- Session IDs are **very** random
- Predicting them is hard, it's much easier to:
  - Check out /tmp on a shared server, see what people have
  - Intercept communications
  - Implement session fixation
- To defend, implement some browser fingerprinting

Notes

Additional Topics

- Filesystem Security
- Dynamic Includes
- Code Injections

Notes

Other Security Settings

- open_basedir – Restricts PHP's file acces to one or more specified directores
  - Relatively quick and easy
- safe_mode – limits file access based on uid/gid of running script and file to be accessed
  - Slower, doesn't work well with uploaded files

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Course

Databases

Course Author: **Paul Reinheimer**

SQL

- You will not be tested on database specific code
  - *In Theory* In reality many cross database experts have stated a belief that the exam has a MySQL base
- A basic understanding of SQL and it's concepts is required

Notes

Some Key Terminology

- DataBase Management System (DBMS)
- Database
- Table
- Column or Field
- Record or Column
- Query
- Primary Key

Notes

Data Types

- While PHP is loosely types, databases are not. Detailed information about the data being stored is required for efficient storage and retrieval, common data-types include:
  - int - Signed integer number, 32bits
  - char - Fixed length character string
  - varchar - Variable length character string
  - float - Signed floating point number, 64bits

Notes

Relationships

- Most modern database systems fall into the relational category, the basis of which is the relationship between various tables
- Relationships link records based on some common data, relationships can be one to one, one to many, many to many
- Database systems can be configured to enforce those relationships to maintain *referential integrity*

Notes

Indices

- Databases are smart and fast, but the database designer has foreknowledge of how the database will be used
- With this knowledge the designer can create an “index” on appropriate columns
- This index instructs the DBMS to store additional information about the data in that column, to make locating data within it as fast as possible

Notes

SQL CREATE

- CREATE TABLE users(
  username varchar(32),
  password varchar(32),
  first_name varchar(50),
  last_name varchar(50),
  PRIMARY KEY(username));

Notes

INSERT

- INSERT
  INTO users
  VALUES (‘paul', md5(‘horriblePassword'),
  ‘Paul', ‘Reinheimer');

Notes

SELECT

- SELECT * FROM users;
- SELECT first_name, last_name FROM users;
- SELECT * FROM users WHERE first_name = ‘paul’

Notes

JOINS

- SELECT u.username, p.title, p.message FROM users AS u JOIN posts AS p ON u.username = p.username
- Outer Joins
  - Right
  - Left

Notes

Analyzing Queries

- EXPLAIN EVERYTHING!
- EXPLAIN SELECT * FROM 07_amazon_monitor_rank_results WHERE asin = '0672324547';

Notes

Explain Example

```
mysql> EXPLAIN SELECT * FROM 07 WHERE asin = '0672324547';
+----+-------------+-----------------------------------+------+---------------+------+---------+------+------+-------------+
|id| select_type|table| type | possible_keys | key  | key_len | ref  | rows | Extra
+----+-------------+-----------------------------------+------+---------------+------+---------+------+------+-------------+
| 1| SIMPLE      | 07 | ALL  | NULL     | NULL | NULL    | NULL |  552 | Using where
+----+-------------+-----------------------------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.02 sec)


mysql> EXPLAIN SELECT * FROM 07 WHERE asin = '0672324547';
+----+-------------+-----------------------------------+------+---------------+------+---------+------+------+-------------+
|id| select_type | table | type | possible_keys | key  | key_len | ref    rows| Extra
|1| SIMPLE      | 07 | ref  | asin      | asin | 12      | const |   69 | Using where
+----+-------------+-----------------------------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.00 sec)
```


Notes

Prepared Statements

- Prepared statements (They're in MySQLi I promise) allow you to increase speed of repeated queries, and isolate data from command
- First you Prepare the statement, then you bind parameters to it, then you execute it
- http://ca.php.net/manual/en/function.mysqli-prepare.php

Notes

Prepared Statement Example

```
$link = mysqli_connect("localhost", "u", "p", "ex");
$city = "Montreal";
$stmt = mysqli_stmt_init($link);
if ($stmt = mysqli_stmt_prepare
($stmt, "SELECT Province FROM City WHERE Name=?"))
{
   mysqli_stmt_bind_param($stmt, "s", $city);
   mysqli_stmt_execute($stmt);
   mysqli_stmt_bind_result($stmt, $province);
   mysqli_stmt_fetch($stmt);
   printf("%s is in district %s\n", $city, $province);
   mysqli_stmt_close($stmt);
}

mysqli_close($link);
```


Notes

Transactions

- Transactions allow you to merge multiple queries into one atomic operation, either they ALL execute successfully, or none do
- BEGIN TRANSACTION #name;
- … queries here
- COMMIT;

Notes

PDO

- PDO stands for PHP Data Objects, and it presents a consistent object oriented method to interact with various databases
- PDO on its own is not able to access any databases, a database specific PDO driver must also be installed
- In some situations PDO actually allows for greater performance than the native database driver (eg. MySql with prepared statements)

Notes

Connecting with PDO

- Regardless of which database you are connecting to, you create a new instance of the same object. The connection type is defined in the first parameter

```
define(USERNAME, "preinheimer");
define(PASSWORD, "sillyPassword");
$pdoConnection = new PDO
    ('mysql:host=localhost;dbname=example',
    USERNAME, PASSWORD);
```


Notes

Executing a Query

- To execute a query, access the query method of the created PDO object. It can be iterated over to access the various rows in the result.

```
foreach ($pdoConnection->
     query("SELECT * FROM users") AS $user)
{
    echo "User number {$user['id']}
       has a username of {$user['userName']}\n";
}
```


Notes

Prepared Statements

```
$query = "SELECT * FROM posts WHERE topicID =
     :tid AND poster = :userid";
$statement = $pdoConnection->prepare($query,
     array(PDO::ATTR_CURSOR,
           PDO::CURSOR_FWDONLY));

$statement->execute(array(':tid' => 100,
      ':userid' => 12));
$userAPosts = $statement->fetchAll();

$statement->execute(array(':tid' => 100,
      ':userid' => 13));
$userBPosts = $statement->fetchAll();
```


Notes

Closing the connection

- To close a PDO connection, simply set the variable containing the PDO object to null

```
$pdoConnection = null;
```


Notes

Database Conclusions

- Databases are likely the fastest datastore you can access
- SQL is a standardized language, most DBMS providers tweak or extend it in some way
- The power lies in the SQL syntax which can be used to locate, update or remove data with an almost terrifying level of complexity
- Indices are a great idea, use EXPLAIN or equivalent syntax to confirm their use

Notes

Database Homework

- Using your favourite DBMS:
  - Create a new table to store the following information: first & last name, phone number, username, password, email address
  - Populate the table manually with at least four records
  - Add an index
  - Select data from the table based on various selection criteria
  - Update the table based on the same

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Exam

Zend Certification Course

Object Oriented Programming

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - Design and Theory
  - **Object Oriented Programming**
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

Object Oriented Programming

- Prior to PHP5 OOP was a hack on top of the array implementation
- PHP5 Changed everything, in a great way

Notes

Simple OO Example

```
class person
{
  public $age;
  public $name;
  function __construct($name, $age)
  {
    $this->name = $name;
    $this->age = $age;
  }

  function birthday()
  {
    echo "Happy Birthday " . $this->name . "!<br>";
    $this->age++;
  }

}
```


Notes

Simple OO Example

```
$myChild = new person("Cayce", 1);
$myFather = new person("Mike", 50);
echo $myChild->name . " is " . $myChild->age . " years old <br>";
echo $myFather->name . " is " . $myFather->age . " years old <br>";
$myChild->birthday();
$myFather->birthday();

echo $myChild->name . " is " . $myChild->age . " years old <br>";

//

Cayce is 1 years old
Mike is 50 years old
Happy Birthday Cayce!
Happy Birthday Mike!
Cayce is 2 years old
```


Notes

PPP & F?

- Public – Method or property can be accessed externally
- Private – Method or property is private to that class and can not be accessed externally
- Protected – Method or property is private and can also be accessed by extending classes
- Final – Method can not be overridden by extending classes

Notes

Extending

```
class friend extends person
{
 private $relationship;
 const MYNAME = "paul";

 function __construct($name, $age, $relationship)
 {
   parent::__construct($name, $age);
   $this->relationship = $relationship;
 }

 function showStatus()
 {
  echo "{$this->name} is my {$this->relationship}";
 }

 function fight()
 {
   $this->relationship = "enemy";
 }

 static function phoneCall()
 {
  echo friend::MYNAME . " the phone is for you";
 }
}
```


Notes

Extending

```
$laura = new friend("Laura", "20", “friend");
$laura->birthday();
$laura->showStatus();
$laura->fight();
$laura->showStatus();
echo friend::MYNAME;
friend::phoneCall();
```


Notes

Autoloading

```
function __autoload($class_name)
{
   require_once "/www/phpClasses/
                        {$class_name}.inc.php";
}
$a = new friend;
```


Notes

Typehinting

```
ageFast($laura);

function ageFast(person $aboutToGetOld)
{
  for ($i = 0; $i< 10; $i++)
  {
    $aboutToGetOld->birthday();
  }
}
```


Notes

Special Functions

- __construct()
- __destruct()
- __toString()
- __sleep()
- __wakeup()
- __call()
- __get()
- __set()

Notes

Exceptions

- Unified method of error handling
- Makes proper error handling possible
- Can be accomplished with try catch blocks or by setting a unified error handler once
- Try catch blocks take precedence if the error raising code is within them

Notes

Exceptions

```
try {
  $fp = fopen("c:/blah.txt", "w");
  if (!$fp)
    throw new Exception("Unable to open file.");
  if (fwrite($fp, "test") != 3)
    throw new Exception("Cannot write data.");
  if (!fclose($fp))
    throw new Exception("Cannot close file.");
} catch (Exception $e)
{
    printf("Error on %s:%d %s\n",
    $e->getFile(), $e->getLine(), $e->getMessage());
    exit;
}
```


Notes

Exceptions

```
<?php
function exception_handler($exception) {
  echo "Uncaught exception: " ,
     $exception->getMessage(), "\n";
}

set_exception_handler('exception_handler');

throw new Exception('Uncaught Exception');
echo "Not Executed\n";
?>
```


Notes

Reflection API

- The reflection API provides a manner to obtain detailed information about code
- Documentation: http://ca3.php.net/reflection

Notes

OOP Conclusions

- Slower than procedural code, but allows complex tasks to be understood more readily
- Data and related methods are grouped into one cohesive structure
- Multiple instances of the same object can be created
- Objects are now dealt with by reference rather than by value, objects must be explicitly cloned to be copied

Notes

OOP Homework

- Develop a small OOP class, including a constructor, destructor, sleep, and wakeup
- Extend it
- Instantiate at least two copies of the class, use them independently, compare them to each other
- Pass an object to another function, use type hinting
- Try the instanceOf operator out

Notes

php | architect
The Magazine For PHP Professionals

Live Interactive
PHP TRAINING

Zend Certification Exam

Zend Certification Course

Design and Theory

Course Author: **Paul Reinheimer**

Course Progress

- The twelve major topics of the exam:
  - PHP Basics
  - Functions
  - Strings
  - Arrays
  - Files and Streams
  - Security
  - Databases
  - **Design and Theory**
  - Object Oriented Programming
  - PHP 4 vs PHP 5
  - Web Features
  - XML

Notes

SPL

- Standard PHP Library
- Extension available and compiled by default in PHP 5
- Improved in PHP 5.1
- http://www.php.net/~helly/php/ext/spl/

Notes

Code Reuse

- We solve the same problems every day
- We solve the same problems as our neighbor every day
- We solve the same problems on every project
- Code Reuse allows us to cut development time while increasing code quality

Notes

OOP Theory

- Design Patterns - Some time ago people realized that we were solving the same problems again and again, they then decided to sit down and come up with really good solutions to those problems, they're called design patterns
- The advantages of design patterns are two fold: first they present a well defined solution to common problems, second they provide a common language for developers

Notes

Factory Pattern

- The factory pattern is used to provide a common interface to a series of classes with identical functionality but different internals (think data storage, varying shipping systems, payment processing)
- The “factory” provides an instance of the appropriate class

Notes

Factory Pattern

- Code Example:

```
class shippment
{ public static function getShippmentClass($method) {
    switch ($method) {
       case "Fedex":
         return new shipFedEx();
         break;
       case "USPS":
         return new shipUSPS();
         break;
       case "UPS":
         return new shipUPS();
         break;
       default:
         throw new Exception("Unknown shipment method requested");
    }
  }
}
$shippingInterface = shippment::getShippmentClass("Fedex");
```

Notes

Singleton Pattern

- The singleton pattern is used to ensure that you only have one instance of a given class at a time
- This is useful in a myriad of circumstances, resource connections (database, file, external) most notably

Notes

Singleton Pattern

- Code Example:

```
class singleton
{
    private static $instance = null;
    protected function __construct()
    { /* ... */  }

    public function getInstance()
    {
        if (self::$instance === null)
        {
            $class = __CLASS__;
            self::$instance = new $class;
        }
        return self::$instance;
    }
}
$connection = new singleton(); //Error
$connection = singleton::getInstance();
```


Notes

Registry Pattern

- An extension of the Singleton Pattern, that allows for differing functionality based on some input data
- Imagine using the Singleton pattern on a database, but occasionally requiring a connection to an alternate database, Registry provides this

Notes

ActiveRecord Pattern

- Encapsulates a data source, allowing external code to concentrate on using the data while the active record pattern provides a consistent interface, hiding the work that goes into iterating over records, making changes etc.
- Most commonly implemented with databases
  - One of the things people love about Ruby

Notes

MVC Pattern

- Model-View-Controller
- Complex pattern, the user initiates an action via the controller, which interfaces with the model, finally the view is called which takes care of dealing with the user interface
- This clear distinction between various layers allows for modularity in code, entire levels of code can be swapped in and out

Notes

Design and Theory Conclusion

- SPL contains a number of very useful code sets, take a look, they're really useful
- Design patterns present common solutions to common problems, they also provide programmers with a common language when describing solutions

Notes

Design and Theory Homework

- Develop applications using at least two of the SPL libraries
- Implement the Singleton pattern on some database code you're already using
- Implement the factory pattern using some basic example code
- Read up on other design patterns

Notes