PHP 5

Robert Richards

http://xri.net/=rob.richards

www.cdatazone.org

- Material is based on PHP 5.2
  - Coded against PHP 5.2.4
- Material from PHP 4 is based on PHP 4.4
  - Coded against PHP 4.4.7

Object Oriented Programming (OOP)

OOP Goals

- Allow compartmentalized refactoring of code.
- Promote code re-use.
- Promote extensibility, flexibility and adaptability.
- Better for team development.
- Many patterns are designed for OOP.
- Some patterns lead to much more efficient code.
- Do you need to use OOP to achieve these goals?
  - ✧ Of course not.
  - ✧ It’s designed to make those things easier though.

OOP Features

- Encapsulation
- Inheritance
- Polymorphism

OOP in PHP 4

- Methods
  - No visibility
  - No abstracts, no final
  - Static without declaration
- Properties
  - No static properties
  - No constants
- Inheritance
  - No abstract, final inheritance, no interfaces
  - No prototype checking, no types
- Object handling
  - Copied by value
  - No destructors

Cannot Re-Assign $this

```
<?php
class MyError {}

class MyClass {
    function MyClass() {
        $this = new MyError;
    }
}

$b = new MyClass;
var_dump($b);
?>
```

PHP 4 Results:

```
object(myerror)(0) {
}
```

PHP 5 Results:

```
Fatal error: Cannot re-assign
    $this in foo.php on line 7
```

Object References

```
<?php
 class OS {
     var $name;
     function OS($name) {
         $this->name = $name;
     }
 }

 function changeName(&$obj,  $name) {
     $obj->name = $name;
 }

 $linux = new OS('linux');
 $win = $linux;
 changeName($win, 'windows');
 echo $linux->name. "\n";
 echo $win->name;
?>
```

Output in PHP4:

linux

windows

Output in PHP 5:

windows

windows

LOCAL THUNDER

Object References (PHP 5)

```
<?php
 class OS {
     var $name;
     function OS($name) {
         $this->name = $name;
     }
 }

 function changeName($obj,  $name) {
     $obj->name = $name;
 }

 $linux = new OS('linux');
 $win = clone $linux;
 changeName($win, 'windows');
 echo $linux->name. "\n";
 echo $win->name;
?>
```

Output:

linux

windows

Indirect Referencing

```
<?php
class foo {
    var $bar = 3;
}

class bar {
    var $o;
    function bar($foo) {
        $this->o = $foo;
    }
}

$foo = new foo();
$a = new bar($foo);

$foo->bar = 5;
echo $a->o->bar;
?>
```

Output in PHP 4.4:

3

Output in PHP 5:

5

Object Cloning

```
<?php
class MyClass {
    var $name;
    function MyClass($name) {
        $this->name = $name;
    }

    function __clone() {
        $this->name = 'Cloned '.$this->name;
    }
}

$objClass = new MyClass("My Class");
$objClone = clone $objClass;

var_dump($objClone);
?>
```

Object Cloning Results

Output:

```
object(MyClass)#2 (1) {
    ["name"]=>
    string(15) "Cloned My Class"
}
```

Constructors/Destructors

```php
<?php
class MyClass {
    function __construct($name) {
        $this->name = $name;
        echo "Constructor called\n";
    }

    function __destruct() {
        echo "Destructor called\n";
    }

    function __clone() {
        echo "Cloning Object\n";
    }
}

echo "Instantiating Class\n";
$objClass = new MyClass("My Class");

echo "Setting \$refClass = \$objClass\n";
$refClass = $objClass;

echo "Setting \$objClass = NULL\n";
$objClass = NULL;

echo "Setting \$refClass = NULL\n";
$refClass = NULL;
?>
```

Constructors/Destructors Contd

Output:

Instantiating Class

Constructor called

Setting $refClass = $objClass

Setting $objClass = NULL

Setting $refClass = NULL

Destructor called

Constructors/Destructors Contd

```
class old {
    function old() { $this->val = 123; }

    function __destruct() { echo "Destructor called\n"; }
}

class newer extends old {
    function __construct() {
        parent::__construct();
    }

    function __destruct() {
        parent::__destruct();
    }
}

$a = new newer();
echo $a->val."\n";
```

Output:

123

Destructor called

Visibility

- Public
  - Everyone has access
  - Equivalent to *var* keyword in PHP 4
- Protected
  - Internally accessible by class
  - Internally accessible by inherited class
- Private
  - Internally accessible by class

Private Visibility

```php
<?php
class Bedroom {
    private $action;

    function __construct() {
        $this->action = 'fun';
        echo "Action set to: ".$this->action."\n\n";
    }
}

$br = new Bedroom();

echo $br->action. "\n";
?>
```

Private Visibility Results

Output:

Action set to: fun

Fatal error: Cannot access private property Bedroom::$action in foo.php on line 12

Protected Visibility

```
<?php
class Bedroom {
    protected $action;

    function __construct() {
        $this->action = 'fun';
    }

    protected function peek() {
        echo $this->action. "\n";
    }
}

class Keyhole extends Bedroom {
    function peek() {
        echo $this->action. "\n";
    }
}
```

```
$kh = new Keyhole();
$kh->peek();

$br = new Bedroom();
$br->peek();

?>
```

Protected Visibility Results

Output:

fun

Fatal error: Call to protected method Bedroom::peek() from context '' in foo.php on line 25

Public Visibility

```php
<?php
class Bedroom {
    protected $action;

    function __construct() {
        $this->action = 'fun';
    }

    public function peek() {
        echo $this->action. "\n";
    }
}

class Keyhole extends Bedroom {
    function peek() {
        echo $this->action. "\n";
    }
}
```

```php
$kh = new Keyhole();
$kh->peek();

$br = new Bedroom();
$br->peek();

?>
```

Public Visibility Results

Output:

fun
fun

__get()/__set()

```
class MyClass {
    private $names = array('a'=>1);

    public function __set($name, $value) {
        if (array_key_exists($name, $this->names)) {
            $this->names[$name] = $value;
        }
    }

    public function __get($name) {
        return isset($this->names[$name])?$this->names[$name]:NULL;
    }
}

$myObj = new MyClass();
echo $myObj->a."\n";
$myObj->b = 2;
echo $myObj->b;
```

__isset()/__unset()

```
class MyClass {
    private $names = array('a'=>1);

    public function __isset($name) {
        return isset($this->names[$name]);
    }

    public function __unset($name) {
        unset($this->names[$name]);
    }
}

$myObj = new MyClass();
var_dump(isset($myObj->a));

unset($myObj->a);
var_dump(isset($myObj->a));
var_dump(empty($myObj->a));
```

Output:

bool(true)
bool(false)
bool(true)

Dynamic Methods

```
class math {
    function __call($name, $args) {
        switch ($name) {
            case 'add':
                return array_sum($args);
            case 'divide':
                $val = array_shift($args);
                foreach ($args as $v) $val /= $v;
                return $val;
        }
    }
}

$m = new math();
echo $m->add(1,2)."\n";      /* will output 3 */
echo $m->divide(8,2);        /* will output 4 */
```

Class Constants

```
<?php
class Base {
    const greeting ="Hello\n";
}

class Derived extends Base {
    const greeting ="Hello World\n";

    static function func() {
        echo parent::greeting;
    }
}

echo Base::greeting;
echo Derived::greeting;
Derived::func();
?>
```

Final Members

```
class Base {
    final function invariant() { echo "Hello\n"; }
}

class Derived extends Base { }

final class Leaf extends Derived { }

class InvalidClass extends Derived {
    function invariant() { echo "Hello\n"; }
}

class BadClass extends Leaf {}
```

Final Members Result

Output with InvalidClass declaration

Fatal error: Cannot override final method Base::invariant() in foo.php on line 12

Output with BadClass declaration

Fatal error: Class BadClass may not inherit from final class (Leaf) in foo.php on line 15

Static Members

Bound to class not object

Can be initialized

Static Members Example

```
<?php
class Object {
    static $refCount = 0;

    public function __construct() {    self::$refCount++;    }

    static function test() {
        echo "Constructor called ".self::$refCount." times\n";
    }
}

Object::test();
$obj1 = new Object;

Object::test();
$obj2 = new Object;

Object::test();
echo $obj1->refCount;   /* Invalid – refCount is not a property */
```

Static Members Results

Constructor called 0 times
Constructor called 1 times
Constructor called 2 times

Notice: Undefined property: Object::$refCount in foo.php on line 19

Abstract Members

- Abstract methods declare signature
- Abstract methods do not provide an implementation
- A Class with an abstract method must be declared abstract
- Abstract classes cannot be instantiated
- Classes can inherit only a single abstract class
- Abstract classes define how behavior is received

Abstract Class

```
<?php
abstract class Shape {
    protected  $base;
    protected $height;

    public function __construct($base, $height) {
        $this->base = $base;
        $this->height = $height;
    }

    abstract public function surface();
}
```

Abstract Class

```
class Triangle extends Shape {
    public function surface(){
        return round(((($this->base)*($this->height)/2),2);
    }
}

class Rectangle extends Shape {
    public function surface(){
        return round(((($this->base)*($this->height)),2);
    }
}

$r = new Rectangle(15,3);
echo $r->surface() ."\n";                    // output 45

$t = new Triangle(15,3);
echo $t->surface() ."\n";                    // output 22.5
```

Interfaces

- Interfaces indicate support for certain behavior
- Classes may inherit multiple Interfaces
- Ensure that a given class provides a certain set of functions

Abstract Classes and Interfaces

```
abstract class Canine {
    abstract function howl();
}

interface Speech {
    function speak();
}

class Dog extends Canine implements Speech {
    function howl() {    return "Bark! Bark!";    }
    function speak() {    return $this->howl();    }
}
class Wolf extends Canine implements Speech {
    function howl() {    return "Hoooowwwlll!!";    }
    function speak() {    return $this->howl();    }
}
```

Abstract Classes and Interfaces

```
class Human implements Speech {
    function speak() {
        return "Hello World!";
    }
}

$dog = new Dog();
echo $dog->howl() . "\n";
echo $dog->speak() . "\n\n";

$human = new Human();
echo $human->speak();
```

Output:

Bark! Bark!

Bark! Bark!

Hello World!

Abstract Classes and Interfaces

Simplified Example

```
interface Speech {
    function speak();
}

abstract class Canine implements Speech {
   abstract function howl();

   function speak() {  return $this->howl();  }
}

class Dog extends Canine {
    function howl() {  return "Bark! Bark!";  }
}

$dog = new Dog();
echo $dog->howl()."\n";
echo $dog->speak()."\n";
```

Output:
Bark! Bark!
Bark! Bark!

Objects As Strings

__toString() method implements object conversion to string

```
class foo {
    public $vals = array('foo', 'bar', 'baz');

    function __toString() {
        return implode(' ', $this->vals);
    }
}

$a = new foo();
echo $a . "\n"; // will print "foo bar baz"
$str = (string) $a;
echo $str;  // will print "foo bar baz"
```

Exceptions

- Basic Rules
  - ✧ Exceptions are exceptions
  - ✧ Never use exceptions for control flow
  - ✧ Never ever use exceptions for parameter passing
- Exceptions should be specialized
- Exceptions should inherit from exception class
- Exception blocks can be nested
- Exceptions can be re thrown

Exceptions

```
<?php
try {                                   /* begin exception block */
    $fp = fopen(__FILE__, "r");
    if (!$fp) {
        /* throw exception, error occurred */
        throw new Exception('no such file', 9);
    }
} catch (Exception $e) {          /* handle error (thrown exception)*/
    echo $e->getCode() . "\n";              // error code
    echo $e->getFile() . "\n";              // file exception was thrown
    echo $e->getLine() . "\n";              // line exception was thrown
    echo $e->getMessage() . "\n";       // error message
    print_r($e->getTrace());                // print backtrace
}                                       /* end exception block */
?>
```

Uncaught Exceptions

```
<?php
function inverse($x) {
    if (!$x)
        throw new Exception('Division by zero.');
    return 1/$x;
}

echo inverse(5) . "\n";
echo inverse(0) . "\n";

echo 'Hello World';
?>
```

Uncaught Exception Result

0.2

Fatal error: Uncaught exception 'Exception' with message 'Division by zero.' in foo.php:4

Stack trace:

#0 foo.php(9): inverse(0)

#1 {main}

thrown in foo.php on line 4

Constructor Failure (PHP 4)

```
<?php
class MyClass {
    function MyClass() {
        /* Constructor error */
        $this = NULL;
    }
}

$b = new MyClass;
if (! $b) {
    echo "Error constructing MyClass";
}
?>
```

Constructor Failure (PHP 5)

```
<?php
class MyClass {
    function __construct() {
        /* Constructor error */
        throw new Exception;
    }
}

try {
    $b = new MyClass;
} catch (Exception $e) {
    echo "Error constructing MyClass";
}
?>
```

Simplify Error Handling

```
<?php
try {
    $db = new PDO('CONNECTION');
    $db->setAttribute(PDO::ATTR_ERRMODE,
                      PDO::ERRMODE_EXCEPTION);
    $res= $db->query('SELECTdata');
    $res2= $db->query('SELECTother');
    // handle data

} catch (Exception $e) {
    echo 'Service currently unavailable';
    error_log($e->getMessage());
}
?>
```

Specializing Exceptions

```php
<?php
class MyMathException extends Exception { }

function inverse($x) {
    if (!$x)
        throw new MyMathException('Division by zero.');
    return 1/$x;
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

echo 'Hello World';
```

Specializing Exceptions Contd

0.2

Caught exception: Division by zero.

Hello World

Specializing Exceptions Contd

```
class MyMathException extends Exception { }

function inverse($x) {
    if (!$x)
        throw new MyMathException('Division by zero.');
    return 1/$x;
}

try {
    echo inverse(5) . "\n";
    echo inverse(0) . "\n";
} catch (MyMathException $e) {
    echo 'Caught MyMathException: ',  $e->getMessage(), "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

echo 'Hello World';
```

Specializing Exceptions Contd

0.2

Caught MyMathException: Division by zero.

Hello World

Nested and Re-Thrown Exceptions

```
class MyMathException extends Exception { }

function myfunction($x) {
   throw new MyMathException('This function errors.');
}

try {
     try {
          myfunction(5);
     } catch (MyMathException $e) {
          throw $e;                          /* Handle and re-throw exception */
     } catch (Exception $e) {
         echo 'Caught exception: ',  $e->getMessage(), "\n";
    }
} catch (MyMathException $e) {
     echo 'Handle exception: ',  $e->getMessage(), "\n";
}
```

Nested and Re-Thrown Exceptions

Handle exception: This function errors.

Type Hinting

```
class YourClass {
     public $var = 1;
}

class MyClass {
    public function test(YourClass $otherclass) {
        echo $otherclass->var."\n";
    }
}

$objYourClass = new YourClass();
$objMyClass = new MyClass();

$objMyClass->test($objYourClass);


$objMyClass->test("12345");
```

Type Hinting

1

Catchable fatal error: Argument 1 passed to MyClass::test() must be an instance of YourClass, string given, called in foo.php on line 16 and defined in foo.php on line 7

Dynamic Class Loading

- Requires a single class per file
- Loads classes only when necessary
- Requires additional user space code
- Only a single loader model is possible

```
<?php
function __autoload($class_name) {
    require_once $class_name . '.php';
}

$obj  = new MyClass1();
$obj2 = new MyClass2();
?>
```

The Standard PHP Library (SPL)

- Captures some common patterns
- Advanced Iterators
- Advanced Array access
- File and directory handling
- Makes __autoload() useable
- Exception hierarchy with documented semantics
- http://www.php.net/manual/en/ref.spl.php

Iterators

```
class Alphabet implements Iterator {
    private $cur = 65;

    public function current() {
        return new Letter($this->cur);
    }

    public function key() {     return $this->cur;     }

    public function next() {    $this->cur++;   }

    public function rewind() {   $this->cur = 65;   }

    public function valid() {
        return ($this->cur <= 90);
    }
}
```

Iterators

```
class Letter {
   private $char;

   public function __construct($char) {    $this->char = $char;   }

   public function __toString() {   return chr($this->char);   }
}

$alphabet = new Alphabet();

foreach ($alphabet AS $ndx=>$letter) {
   print $letter;
}
```

SPL: IteratorAggregate

```
class AlphabetIterator implements Iterator {
    private $alphabet;

    public function __construct(Alphabet $alphabet) {
        $this->alphabet = $alphabet;
    }
    public function current() {
        return new Letter($this->alphabet->cur);
    }
    public function key() {     return $this->alphabet->cur;     }
    public function next() {    $this->alphabet->cur++;   }
    public function rewind() {   $this->alphabet->cur = 65;   }
    public function valid() {   return ($this->alphabet->cur <= 90);    }
}
```

SPL: IteratorAggregate

```
class Alphabet implements IteratorAggregate {
    public $cur = 65;
    public function getIterator() {
        return new AlphabetIterator($this);
    }
}

class Letter {
   private $char;
   public function __construct($char) {    $this->char = $char;   }
   public function __toString() {   return chr($this->char);   }
}

$alphabet = new Alphabet();

foreach ($alphabet AS $ndx=>$letter) {
   print $letter;
}
```

SPL: RecursiveIterator

- Interface for recursion
- Allows for multi-dimensional iteration
- Extends Iterator interface
- Adds hasChildren() method
- Adds getChildren() method

SPL: SeekableIterator

```
class Alphabet implements SeekableIterator {
    public $cur = 65;

    // Implement existing functionality

    public function seek($index) {
        if ($index < 65 || $index > 90)
            throw new OutOfBoundsException('Invalid index');
        $this->cur = $index;
    }
}

$alphabet = new Alphabet();

$alphabet->seek(ord("Q"));
echo $alphabet->current();
$alphabet->seek(100);
```

SPL: ArrayIterator

```
class Alphabet implements SeekableIterator {
    public $cur = 65;

    // Implement existing functionality

    public function seek($index) {
        if ($index < 65 || $index > 90)
            throw new OutOfBoundsException('Invalid index');
        $this->cur = $index;
    }
}

$alphabet = new Alphabet();

$alphabet->seek(ord("Q"));
echo $alphabet->current();
$alphabet->seek(100);
```

SPL: Directory Iterator

```
$path = new DirectoryIterator(<path to directory>);

foreach ($path as $file) {
    echo $file->getFilename() . "\t";
    echo $file->getSize() . "\t";
    echo $file->getOwner() . "\t";
    echo $file->getMTime() . "\n";
}
```

SPL: Exceptions

- BadMethodCallException
  - ✧ The method call was illegal.
- InvalidArgumentException
  - ✧ The arguments passed were invalid.
- LengthException
  - ✧ The parameter exceeds the allowed length.
- LogicException
  - ✧ Generic error occured in program logic.
- OutOfBoundsException
  - ✧ An illegal index was requested.
- UnexpectedValueException
  - ✧ An unexpected value was received.

New Database Support

- SQLite
  - New database extension
  - Self Contained
- MySQLi
- PHP Data Objects (PDO)

SQLite

- Bundled with PHP 5 (including library)
- Fast and easy interface to 'flatfiles'
- Implements most of SQL92
- Supports both OO and procedural API
- No daemon required
- Enabled by default
- Can use PHP to extend SQL language

SQLite

```
/* open connection to memory database */
$db = new SQLiteDatabase(":memory:");

/* execute a regular query */
$db->query("CREATE TABLE test(a,b)");
$db->query("INSERT INTO test VALUES('1','2')");

/* retrieve data using an unbuffered query */
$r = $db->unbufferedQuery("SELECT * FROM test",
                          SQLITE_ASSOC);

/* use object iterators to retrieve the data */
foreach ($r as $row) {
    print_r($row);
}
```

Output

```
Array
(
    [a] => 1
    [b] => 2
)
```

SQLite: Navigating Buffered Results

```
$db = new SQLiteDatabase("test.db");
$db->query("CREATE TABLE test(first,last)");
$db->query("INSERT INTO test VALUES('Rob','Richards')");
$db->query("INSERT INTO test VALUES('Joe','Smith')");
$db->query("INSERT INTO test VALUES('Jane','Doe')");
$db->query("INSERT INTO test VALUES('Mike','Jones')");

/* retrieve data using a buffered query */
$r = $db->query("SELECT * FROM test", SQLITE_ASSOC);

if ($r->numRows() > 3) {
    if ($r->seek(2) && $r->prev()) {
        $row = $r->fetch();
        print "Row: ".$r->key()." First:".$row['first'];
    }
}
```

SQLite: fetchObject

```
class MyPerson {
   private $curdate;
   function __construct($curdate) {
       $this->curdate = $curdate;
   }
   function whoami() {
       echo "I am ".$this->first." ".$this->last." created ".$this->curdate;
   }
}
$db = new SQLiteDatabase("test.db");
$r = $db->query("SELECT * FROM test", SQLITE_ASSOC);

$ctor_params =  array(date("m/d/Y"));
while ($obj = $r->fetchObject('MyPerson',$ctor_params)) {
    print $obj->whoami()."\n";
}
```

SQLite: Additional Info

Supports Triggers

User-Defined Functions (UDFs)

Aggregate User Defined Functions

ISO-8859-1 and UTF-8 character set support

Transaction Support

MySQLi

- Supports MySQL 4.1+ Client API
- Prepared statement support
- Support for tracing, debugging, load balancing and replication functionality
- Multi-Statement queries
- SQLSTATE error codes

MySQL Syntax

```
/* Connect to a MySQL server */
$link = mysql_connect('localhost', <username>, <password>);

if (!$link)
   die("Can't connect . Errorcode: " . mysql_error());
mysql_select_db('mydb', $link);

/* Send a query to the server */
if ($result = mysql_query('SELECT state, name FROM lkp_state')) {
    while( $row = mysql_fetch_assoc($result) ){
        printf("%s (%s)\n", $row['state'], $row['name']);
    }

    mysql_free_result($result); // Destroy result set and free memory
}

mysql_close(); // Close the connection
```

MySQLi Syntax

```
/* Connect to a MySQL server */
$link = mysql_connect('localhost', <username>, <password>, 'mydb');

if (!$link)
   die("Can't connect . Errorcode: " . mysqli_connect_error());

/* Send a query to the server */
if ($result = mysqli_query($link, 'SELECT state, name FROM lkp_state'))
{
    while( $row = mysqli_fetch_assoc($result) ){
        printf("%s (%s)\n", $row['state'], $row['name']);
    }

    mysqli_free_result($result); // Destroy result set and free memory
}

mysqli_close($link); // Close the connection
```

MySQLi: OOP

```
/* Connect to a MySQL server */
$objLink = new mysqli('localhost', <username>, <password>, 'mydb');

if (!$objLink)
   die("Can't connect . Errorcode: " . mysqli_connect_error());

/* Send a query to the server */
if ($objResult = $objLink->query('SELECT state, name FROM
lkp_state')) {
    while( $row = $objResult->fetch_assoc() ){
        printf("%s (%s)\n", $row['state'], $row['name']);
    }

    $objResult->free(); // Destroy result set and free memory
}

$objLink->close(); // Close the connection
```

MySQLi: Multi-Resultset

```
$objLink = new mysqli('localhost', <username>, <password>, 'mydb');

$sql1 = 'SELECT name from lkp_state ORDER BY name ASC LIMIT 2';
$sql2 = 'SELECT name from lkp_state ORDER BY name DESC LIMIT 2';

if ($objLink->multi_query($sql1 . ';'. $sql2)) {

    do {
        if ($result = $objLink->store_result()) {  // store first result set
            while ($row = $result->fetch_row()) {
                printf("%s\n", $row[0]);
            }
            $result->close();
        }
        if ($objLink->more_results())
            printf("-----------------\n");
    } while ($objLink->next_result());
}
```

MySQLi: Prepared Statements

```
$objLink = new mysqli('localhost', <username>, <password>, 'mydb');

$state = "ME";
$sql = 'SELECT name FROM lkp_state WHERE state=?';

if ($objStmt = $objLink->prepare($sql)) {

    $objStmt->bind_param("s", $state);
    $objStmt->execute();

    $objStmt->bind_result($name);

    $objStmt->fetch();

    print "State: $name\n";
}
```

Output:

State: Maine

PHP Data Objects (PDO)

- Lightweight, consistent interface for accessing databases
- Provides a data-access abstraction layer
- It doesn't rewrite SQL or emulate missing features

PDO: Supported Databases

Driver Name	Supported Database
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and Win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2

PDO: Connection Management

```
try {
    $dbh = new PDO($dsn, $user, $pw);
    // use the database here
    // ...
    // done; release the connection
    $dbh = null;
} catch (PDOException $e) {
    echo "connect failed";
    $e->getMessage();
}
```

PDO: DSNs

- Database Source Name
- data structures used to describe a connection to a database
- In general
  - ✧drivername:<driver-specific-stuff>

PDO: DSNs

- mysql:host=name;dbname=dbname
- pgsql:native_pgsql_connection_string
- odbc:odbc_dsn
- oci:dbname=dbname;charset=charset
- sqlite:/path/to/db/file
- sqlite::memory:
- sqlite2:/path/to/sqlite2/file
- sqlite2::memory:

PDO: DSN example

```
try{
   // use SQLite inmemory database
   $dbh = new PDO('sqlite::memory:');

   // use the database then release
   connection
   $dbh=null;
}catch(PDOException $e){
   echo $e->getMessage();
}
```

PDO: DSN Aliasing

- uri:uri
  - ✧Specify location of a file containing actual DSN on the first line
  - ✧Works with streams interface, so remote URLs can work too
- name (with no colon)
  - ✧Maps to pdo.dsn.name in your php.ini
  - ✧pdo.dsn.name=sqlite:/path/to/name.db

PDO: DSN Aliasing

```
pdo.dsn.name=sqlite:/path/to/name.db

$dbh = new PDO("name");

Same as:
$dbh = new PDO("sqlite:/path/to/name.db");
```

PDO: Executing SQL

```
$createSQL = 'CREATE TABLE staff (
id INT NOT NULL ,
first VARCHAR( 50 ) NOT NULL ,
last VARCHAR( 50 ) NOT NULL
)';

try {
   $dbh = new PDO('sqlite::memory:');
   $dbh->exec($createSQL);
   $dbh->exec("insert into staff (id, first, last) values " .
                                                "(1, 'Rob', 'Richards')");
   $dbh = null;
} catch (PDOException $e) {
   print "Error!: " . $e->getMessage() . "\n";
}
```

PDO: Retrieve Data

```
try {
   $dbh = new PDO('sqlite::memory:');
   $dbh->exec($createSQL);    /* Previous SQL Statement */
   $dbh->exec("insert into staff (id, first, last) values (1, 'Rob',
    'Richards')");

   foreach ($dbh->query('SELECT * from staff') as $row) {
      print_r($row);
   }

   $dbh = null;
} catch (PDOException $e) {
   print "Error!: " . $e->getMessage() . "\n";
}
```

PDO: Autonumber/Sequences

```
$dbh->exec(
    "insert into foo values (...)");
echo $dbh->lastInsertId();

$dbh->exec(
    "insert into foo values (...)");

echo $dbh->lastInsertId("seqname");
```

PDO: Prepared Statements

- The query only needs to be parsed (or prepared) once
- The parameters to prepared statements don't need to be quoted
- Prepared statements are so useful that they are the only feature that PDO will emulate for drivers that don't support them

PDO: Binding columns for output

```
$stmt = $dbh->prepare(
     "SELECT extension, name from CREDITS");

if ($stmt->execute()) {
   $stmt->bindColumn('extension', $extension);
   $stmt->bindColumn('name',      $name);

   while ($stmt->fetch(PDO::FETCH_BOUND)) {
      echo "Extension: $extension\n";
      echo "Author: $name\n";
   }
}
```

PDO: Transactions

```
$dbh->beginTransaction();
try {
   $dbh->query("UPDATE ...");
   $dbh->query("UPDATE ...");
   $dbh->commit();
} catch (Exception $e) {
   $dbh->rollBack();
}
```

XML and Web Services

- PHP 5 introduced numerous interfaces for working with XML
- The libxml2 library (http://www.xmlsoft.org/) was chosen to provide XML support
- The sister library libxslt provides XSLT support
- I/O is handled via PHP streams
- ext/domxml was removed in favor of ext/dom
- Choice between working with XML and JSON
- Built-in SOAP support

SimpleXML

- Provides simple access to XML documents
- Operates only on elements and attributes
- Contains XPath support
- Allows for modifications to the XML
- Zero copy interoperability with DOM
- Objects implement __toString()
- New in PHP 5.1.3
  - ✧ Elements and attributes can be added using addChild() and addAttribute() methods.
  - ✧ Node names can be retrieved by calling getName().

SimpleXML: Yahoo WebSearch

```
<ResultSet xmlns="urn:yahoo:srch" totalResultsAvailable="34800000"
    totalResultsReturned="3" . . .>

  <Result>
    <Title>XML and PHP 5</Title>
    <Summary>The goal for XML support in PHP 5 was not ...</Summary>
    <Url>http://devzone.zend.com/article/2387-XML-and-PHP-5</Url>
    <ClickUrl>http://uk.wrs.yahoo.com/_ylt=A9iby4x...</ClickUrl>
    <DisplayUrl>devzone.zend.com/article/2387-XML-and...</DisplayUrl>
    <ModificationDate>1193036400</ModificationDate>
    <!-- Abbreviated content -->
  </Result>

  <!-- Abbreviated Results -->

</ResultSet>
```

SimpleXML: Yahoo WebSearch

```
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';
$url .= '?query='.rawurlencode('php 5 xml');          // encode arguments
$url .= "&appid=zzz&results=3&language=en";     // limit 3 English results

$sxe = simplexml_load_file($url);

/* Check for number of results returned */
if ($sxe['totalResultsReturned'] > 0) {

   /* Loop through results and print title, url and modification date */
   foreach ($sxe->Result AS $result) {
      print 'Title: '.$result->Title."\n";
      print 'Url: '.$result->Url."\n";
      print 'Mod: '.date ('M d Y', (int)$result->ModificationDate)."\n\n";
   }
}
```

SimpleXML: Yahoo WebSearch

Title: XML and PHP 5
Url: http://devzone.zend.com/article/2387-XML-and-PHP-5
Mod: Oct 22 2007

Title: XML in PHP 5 - What's New?
Url: http://devzone.zend.com/node/view/id/1713
Mod: Oct 25 2007

Title: ONLamp.com -- Using PHP 5's SimpleXML
Url: http://www.onlamp.com/pub/a/php/2004/01/15/simplexml.html
Mod: Oct 23 2007

SimpleXML: Namespaces

```
$xml = <<< EOXML
<books xmlns="urn::books" xmlns:bk="urn::book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
  </bk:book>
  <book qty="33" xmlns="urn::classics">
    <name>To Kill a Mockingbird</name>
  </book>
</books>
EOXML;

$books = simplexml_load_string($xml);

foreach ($books->book AS $book) {
    print $book->name;
}
```

Output:

To Kill a Mockingbird

SimpleXML: Namespaces

```
$xml = <<< EOXML
<books xmlns="urn::books" xmlns:bk="urn::book">
  <bk:book qty="25">
    <bk:name>Grapes of Wrath</bk:name>
  </bk:book>
  <book qty="33" xmlns="urn::classics">
    <name>To Kill a Mockingbird</name>
  </book>
</books>
EOXML;

$books = simplexml_load_string($xml);

$ns_book = $books->children("urn::book");
foreach ($ns_book->book AS $book) {
    print $book->name;
}
```

Output:

Grapes of Wrath

LOCAL THUNDER

SimpleXML: XPath

```
$xml = <<< EOXML
<books xmlns="urn::books" xmlns:bk="urn::book">
    <bk:book>Grapes of Wrath</bk:book>
    <book xmlns="urn::classics">To Kill a Mockingbird</book>
</books>
EOXML;

$books = simplexml_load_string($xml);

$nodelist = $books->xpath("//book");
print "Title: " . $nodelist[0] . "\n";

$nodelist = $books->xpath("//bk:book");
print "Title: " . $nodelist[0] . "\n";

$books->registerXPathNamespace("ec", "urn::classics");
$nodelist = $books->xpath("//ec:book");
print "Title: " . $nodelist[0] . "\n";
```

Output:

Title:
Title: Grapes of Wrath
Title: To Kill a Mockingbird

SimpleXML: Editing Documents

```
$xml = <<< EOXML
<books>
  <book>Grapes of Wrath</book>
  <book>To Kill a Mockingbird</book>
</books>
EOXML;
$books = simplexml_load_string($xml);

$books->book[2] = "Pro PHP XML and Web Services";

unset($books->book[0]);

$books->book[0] .= " II";

print $books->asXML();
```

Output:

```
<?xml version="1.0"?>
<books>
  <book>To Kill a Mockingbird II</book>
  <book>Pro PHP XML and Web . . .</book>
</books>
```

DOM

- Tree based parser
- Allows for creation and editing of XML documents
- W3C Specification : DOM Level 2/3 compliant
- Provides XPath support
- Provides XInclude Support
- Ability to work with HTML documents
- Zero copy interoperability with SimpleXML
- Replacement for ext/domxml from PHP 4

DOM: Sample Document

```
<courses>
    <course>
        <title>Basic Languages</title>
        <description>Introduction to Languages</description>
    </course>
    <course>
        <title>French I</title>
        <description>Introduction to French</description>
    </course>
    <course>
        <title>French II</title>
        <description>Intermediate French</description>
    </course>
</courses>
```

DOM: Navigation

```
$dom = new DOMDocument();
$dom->loadXML($xml);
$root = $dom->documentElement;

foreach ($root->childNodes AS $child) {
   if ($child->nodeType == XML_ELEMENT_NODE &&
       $child->nodeName == "course") {
         $node = $child->firstChild;

         while($node) {
              if ($node->nodeName == "title") {
                 echo "Title: ".$node->textContent."\n";
                 break;
              }
         $node = $node->nextSibling;
      }
   }
}
```

Output:

```
Title: Basic Languages
Title: French I
Title: French II
```

DOM: Creating Simple Tree

```
$doc = new DOMDocument();

$root = $doc->createElement("tree");
$doc->appendChild($root);

$root->setAttribute("att1", "att1 value");

$child = $root->appendChild(new DOMElement("child"));

$comment = $doc->createComment("My first Document");
$doc->insertBefore($comment, $root);

$text = $doc->createTextNode("some content");
$child->appendChild($text);

$doc->formatOutput = TRUE;
print $doc->saveXML();
```

DOM: Creating Simple Tree

```
<?xml version="1.0"?>
<!--My first Document-->
<tree att1="att1 value">
    <child>some content</child>
</tree>
```

DOM: Yahoo WebSearch

```
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';
$url .= '?query='.rawurlencode('php 5 xml');          // encode arguments
$url .= "&appid=zzz&results=3&language=en";      // limit 3 English results
$outputElements = array("Title", "Url");

$doc = new DOMDocument();
$doc->load($url);

$resultNodes = $doc->getElementsByTagName("Result");

foreach ($resultNodes AS $result) {
   foreach ($result->childNodes AS $child) {
      if (in_array($child->nodeName, $outputElements))
          print $child->nodeName . ": " . $child->textContent . "\n";
   }
   print "\n";
}
```

XMLReader

- Forward moving stream based parser
- It is a Pull parser
- Based on the C# XmlTextReader API
- Advantages:
  - ✧ Low memory footprint
  - ✧ Namespace support
  - ✧ Simple API
  - ✧ Validation support
  - ✧ Faster Processing

XMLReader: Reading

```
$xml = <<< EOXML
<courses>
    <course>
        <title>Basic Languages</title>
    </course>
</courses>
EOXML;

$xmlreader = new XMLReader();

$xmlreader->XML($xml);

while ($xmlreader->read()) {
   print str_repeat(" ", $xmlreader->depth * 3).
         $xmlreader->name."\n";
}
```

Output:

```
courses
   #text
   course
      #text
      title
         #text
      title
      #text
   course
   #text
courses
```

XMLReader: Yahoo WebSearch

```
$url = 'http://api.search.yahoo.com/WebSearchService/V1/webSearch';
$url .= '?query='.rawurlencode('php 5 xml');          // encode arguments
$url .= "&appid=zzz&results=3&language=en";     // limit 3 English results

$reader = new XMLReader();
if (! $reader->open($url)) die("Cannot access Webservice");

while($reader->name != "Result") { $reader->read(); }

do {
    processResult($reader);
    print "\n";
} while($reader->next('Result'));
```

XMLReader: Yahoo WebSearch

```
function getTextValue($reader) {
    if ($reader->nodeType != XMLReader::ELEMENT
        || $reader->isEmptyElement
        || ($reader->read() &&
            $reader->nodeType == XMLReader::END_ELEMENT))
                return;

    $retVal = $reader->value;
    $reader->read();
    return $retVal;
}
```

XMLReader: Yahoo WebSearch

```
function processResult($reader) {
    $depth = $reader->depth;
    if ($reader->isEmptyElement || ($reader->read() &&
        $reader->nodeType == XMLReader::END_ELEMENT))
            return;
    $nodes = array("Title", "Url", "ModificationDate");
    while($depth < $reader->depth) {
      if ( in_array ($reader->name, $nodes)) {
          print $reader->name.": ".getTextValue($reader)."\n";
      }
      $reader->next();
    }

   /* Read until </Result> is encountered */
   while($depth < $reader->depth) { $reader->read(); }
}
```

XMLReader: Yahoo WebSearch

Title: XML and PHP 5
Url: http://devzone.zend.com/article/2387-XML-and-PHP-5
ModificationDate: 1193036400

Title: XML in PHP 5 - What's New?
Url: http://devzone.zend.com/node/view/id/1713
ModificationDate: 1193295600

Title: ONLamp.com -- Using PHP 5's SimpleXML
Url: http://www.onlamp.com/pub/a/php/2004/01/15/simplexml.html
ModificationDate: 1193122800

XMLWriter

- Lightweight and forward-only API for generating well formed XML
- Automatically escapes data
- Works with PHP 4.3+ available at http://pecl.php.net/package/xmlwriter
- Object Oriented API available for PHP 5+
- Part of core PHP distribution since PHP 5.1.2

XMLWriter: Simple Example

```
$xw = new XMLWriter();
$xw->openMemory();

// Turn on indenting to make output look pretty and set indent string
$xw->setIndent(TRUE);
$xw->setIndentString('    ');

// Write out the optional XML declaration only specifying version
$xw->startDocument('1.0');

// Create the opening document element, which is namespaced
$xw->startElementNs(NULL, "chapter", "urn::default");

// Write namespace declaration that is used later in the document
$res = $xw->writeAttribute('xmlns:a', 'urn::namespace-a');

// Write complete elements with text content
$xw->writeElement('a:title', 'XMLReader');
$xw->writeElement('para', 'spec chars < > & " inside para element');
```

XMLWriter: Simple Example

```
// start an element and add an attribute to it
$xw->startElement('a:section');
$xw->writeAttribute('a:id', 'about');

// Write out an element with special characters
$xw->writeElement('title', 'Pro PHP XML & Webservices');

$xw->startElement('para'); // This opens the para element

$xw->writeComment("this is a comment");
$xw->text("        ");
$xw->writePi("php", "echo 'Hi!  This is PHP version ' . phpversion(); ");
$xw->text("\n      ");

$xw->endElement();              // This will close the open para element

$xw->endDocument();             // Closes all open tags

echo $xw->flush(true);          // Flush and clear the buffer
```

XMLWriter: Simple Example

```
<?xml version="1.0"?>
<chapter xmlns:a="urn::namespace-a" xmlns="urn::default">
  <a:title>XMLReader</a:title>
  <para>spec chars &lt; &gt; &amp; &quot; inside para
element</para>
  <a:section a:id="about">
    <title>Pro PHP XML &amp; Webservices</title>
    <para>
      <!--this is a comment-->
      <?php echo 'Hi!  This is PHP version ' . phpversion(); ?>

    </para>
  </a:section>
</chapter>
```

SOAP

- An XML XML-based protocol for exchanging information between applications
- It allows for remote invocation of methods in a distributed environment
- Uses existing transport protocols such as HTTP
- Can operate with or without a Web Service Definition Language (WSDL) document
- A W3C standard and the core component to the Web Services Interoperability Organization (WS WS-I) Basic Profile

SOAP: Weather Forecast

```
$wsdl = 'http://www.webservicex.net/WeatherForecast.asmx?WSDL';

$client = new SOAPClient($wsdl);

$response = $client->GetWeatherByZipCode(array("ZipCode"=>"04101"));

$obj = $response->GetWeatherByZipCodeResult;

print "Weather for: ".$obj->PlaceName." ".$obj->stateCode."\n";

foreach ($obj->Details->WeatherData AS $weather) {
    if (empty($weather->Day))
        continue;

    print "  ".$weather->Day."\n";
    print "     Max: " . $weather->MaxTemperatureF .
            " Min: ".$weather->MinTemperatureF."\n";
}
```

SOAP: Weather Forecast

```
$wsdl = 'http://www.webservicex.net/WeatherForecast.asmx?WSDL';

$client = new SOAPClient($wsdl);

$functions = $client->__getFunctions();
foreach ($functions AS $function) {
    print $function."\n";
}

$types = $client->__getTypes();
foreach ($types AS $type) {
    print $type."\n";
}
```

SOAP: Weather Forecast

Methods

GetWeatherByZipCodeResponse GetWeatherByZipCode(GetWeatherByZipCode $parameters)

GetWeatherByPlaceNameResponse GetWeatherByPlaceName(GetWeatherByPlaceName $parameters)

SOAP: Weather Forecast

```
struct GetWeatherByZipCode {
 string ZipCode;
}

struct WeatherForecasts {
 float Latitude;
 float Longitude;
 float AllocationFactor;
 string FipsCode;
 string PlaceName;
 string StateCode;
 string Status;
 ArrayOfWeatherData Details;
}

struct ArrayOfWeatherData {
 WeatherData WeatherData;
}

struct WeatherData {
 string Day;
 string WeatherImage;
 string MaxTemperatureF;
 string MinTemperatureF;
 string MaxTemperatureC;
 string MinTemperatureC;
}

struct GetWeatherByZipCodeResponse {
 WeatherForecasts GetWeatherByZipCodeResult;
}
```

LOCAL THUNDER

Filter Extension

- Included starting with PHP 5.2
- Enabled by default
- Provides a default filter
- Provides two groups of accessing filters: sanitizing and logical
- Filters can have options to configure their behavior

magic_quotes_gpc

“The magic quotes option was introduced to help protect developers from SQL injection attacks. It effectively executes addslashes() on all information received over GET, POST or COOKIE. Unfortunately this protection isn't perfect: there are a series of other characters that databases interpret as special not covered by this function. In addition, data not sent direct to databases must un-escaped before it can be used.

Because it's inconsistent and ineffective, it's not recommended that magic_quotes_gpc be enabled. Rely on input filtering done by your scripts.”

PHP Security Consortium

http://phpsec.org/projects/phpsecinfo/tests/magic_quotes_gpc.html

Default Filters

```
<form action="" method="get">
<input type="text" name="data" maxlength="64" size="64"/>
<input type="submit"/>
</form>
<?php
if ( isset( $_GET['data'] ) ) {
    $filter = ini_get('filter.default');
    echo "The data filterered through '$filter' is:";
    var_dump( $_GET['data'] );
}
?>
```

Sanitize Data

```
<form action="" method="get">
Data: <input type="text" name="data" maxlength="64"/><br/>
<input type="submit"/>
</form>
<?php
$options = FILTER_FLAG_STRIP_HIGH;

if (isset($_GET['data'])) {
    $data = filter_input(
        INPUT_GET,              // source
        'data',                 // parameter name
        FILTER_SANITIZE_STRING,  // filter
        $options                // options);
    var_dump( $data );
}
?>
```

Sanitize Arrays

```
<form action="" method="get">
Data 1: <input type="text" name="data[]" size="64"/><br/>
Data 2: <input type="text" name="data[]" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$options = FILTER_FLAG_STRIP_HIGH | FILTER_REQUIRE_ARRAY;

if (isset($_GET['data'])) {
    $data = filter_input(
        INPUT_GET,              // source
        'data',                 // parameter name
        FILTER_SANITIZE_STRING,  // filter
        $options                // options);
    var_dump( $data );
}
?>
```

Dealing with Arrays

- FILTER_REQUIRE_SCALAR (default): requires the input variable to be not an array
- FILTER_REQUIRE_ARRAY: requires the input variable to be an array
- FILTER_FORCE_ARRAY: converts the input variable to an array, even if a scalar was submitted

Getting Input Arrays

```
<form action="" method="get">
Number: <input type="text" name="int" maxlength="64" size="64"/><br/>
String: <input type="text" name="string" maxlength="64" size="64"/><br/>
<input type="submit"/></form><?php

$definition = array(
    'int' => array(
        'filter' =>  FILTER_VALIDATE_INT,
        'options' => array( "min_range" => 1, "max_range" => 10 )
    ),
    'string' => FILTER_SANITIZE_SPECIAL_CHARS
);

if (isset($_GET['int'])) {
    $data = filter_input_array( INPUT_GET, $definition );
    var_dump( $data );
}
?>
```

Filtering Variables

```
<?php
$text = "This tekßt is göing\tto be changed\n";
$data = filter_var(
    $text,
    filter_id( 'special_chars' ), //Sanitize special chars
    FILTER_FLAG_STRIP_HIGH
);
var_dump( $data );
?>
```

Output

```
string(39) "This tekt is ging to be changed "
```

Sanitizing Filters

- Allows or disallows characters in a string
- Does not take care about data formats
- Does not transform types, but keeps strings

Validating Filters

- Analyses the data in a logical way
- Understands data formats
- Can transform type

Sanitizing Filters

- FILTER_SANITIZE_EMAIL
  - Remove all characters except letters, digits and !#$%&'*+-/=?^_`{|}~@.[].
- FILTER_SANITIZE_URL
  - Remove all characters except letters, digits and $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=.
- FILTER_SANITIZE_NUMBER_INT
  - Remove all characters except digits and +-.
- FILTER_SANITIZE_NUMBER_FLOAT
  - Remove all characters except digits, +- and optionally .,eE.

Sanitizing Filters

- FILTER_SANITIZE_MAGIC_QUOTES
  - Apply addslashes().
- FILTER_SANITIZE_STRING
  - Strip tags, optionally strip or encode special characters.
- FILTER_SANITIZE_STRIPPED
  - Alias of "string" filter.
- FILTER_SANITIZE_ENCODED
  - URL-encode string, optionally strip or encode special characters.
- FILTER_SANITIZE_SPECIAL_CHARS
  - HTML-escape '"<>& and characters with ASCII value less than 32, optionally strip or encode other special characters.

Sanitize Email

```
<form action="" method="post">
Data: <input type="text" name="data"
size="64"/><br/>
<input type="submit"/>
</form>
<?php
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                                FILTER_SANITIZE_EMAIL);
    var_dump( $data );
}
?>
```

Validate Email

```
<form action="" method="post">
Data: <input type="text" name="data"
size="64"/><br/>
<input type="submit"/>
</form>
<?php
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                                FILTER_VALIDATE_EMAIL);
    var_dump( $data );
}
?>
```

Sanitize Float

```
<form action="" method="post">
Data: <input type="text" name="data" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$flags = FILTER_FLAG_ALLOW_FRACTION |
         FILTER_FLAG_ALLOW_THOUSAND |
         FILTER_FLAG_ALLOW_SCIENTIFIC;
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                          FILTER_SANITIZE_NUMBER_FLOAT,
                          $flags);
    var_dump( $data );
}
?>
```

Validate Float

```
<form action="" method="post">
Data: <input type="text" name="data" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$flags = FILTER_FLAG_ALLOW_FRACTION |
         FILTER_FLAG_ALLOW_THOUSAND |
         FILTER_FLAG_ALLOW_SCIENTIFIC;
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                          FILTER_VALIDATE_FLOAT,
                          $flags);
    var_dump( $data );
}
?>
```

Validate URL

```
<form action="" method="post">
URL: <input type="text" name="data" size="64"/><br/>
<input type="submit"/>
</form>
<?php
$flags = FILTER_FLAG_SCHEME_REQUIRED |
         FILTER_FLAG_HOST_REQUIRED |
         FILTER_FLAG_PATH_REQUIRED |
         FILTER_FLAG_QUERY_REQUIRED;
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                          FILTER_VALIDATE_URL, $flags);
    var_dump( $data );
}
?>
```

Validate RegExp

```
<form action="" method="post">
Data: <input type="text" name="data" size="64"/><br/>
Regex: <input type="text" name="regexp"
            value="/^[2-9][0-9]{2}[-][2-9][0-9]{2}[-][0-9]{4}$/" size="64"/><br/>
<input type="submit"/></form>
<?php

if (!empty($_POST['regexp']))
$options = array('options' =>
                                array( 'regexp' => $_POST['regexp']));
if (isset($_POST['data'])) {
    $data = filter_input( INPUT_POST, "data",
                            FILTER_VALIDATE_REGEXP, $options);
    var_dump( $data );
}
?>
```

Callback Filter

- Allows you to write your own callback filter
- Can not be used as default filter

Filter Callback

```
<form action="" method="get">
Data: <input type="text" name="data" size="64"/> (PHP)<br/>
<input type="submit"/></form>
<?php
$callback = array( 'options' => array( 'Validate', 'My' ) );

if (isset($_GET['data'])) {
    $data = filter_input(INPUT_GET, 'data', FILTER_CALLBACK, $callback);
    var_dump( $data );
}

class Validate {
    function My( $text ) {
        if ( $text == 'PHP' )
            return 'PHP Rocks!';
        return false;
    }
}
?>
```

Questions?

PHP 5

Robert Richards

http://xri.net/=rob.richards

www.cdatazone.org