Curso de Programación en PHP

Nivel I

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología - Oro Verde - v2.1

Capítulo 5: POO y patrones en PHP

Introducción a patrones de diseño

Tipos de patrones de diseño

Patrones más utilizados en el diseño

Singleton en PHP

Active Record en PHP

Factory en PHP

Registry en PHP

Value Object en PHP

Clase 10: Introducción a patrones de diseño

¿Qué es un patrón de diseño?

Es una solución a problemas comunes en el desarrollo de Sistemas, teniendo como principales características la **eficiencia** y la **reusabilidad**.

Nacimiento de los patrones de diseño

La definición de patrón de diseño aparece en el año 1979 por el arquitecto Christopher Alexander.

Éxito de los patrones de diseño

En 1990 toman carrera los patrones de diseño en el mundo de la informática con el libro *Design Patterns* escrito por el grupo *Gang of Four (GoF)*.

Eric Gamma
Richard Helm
Ralph Johnson
John Vlisides

Algunos Objetivos de los patrones de diseño

Proporcionar elementos reusables.

Disminuir la búsquedas de soluciones a problemas comunes ya resueltos.

Otorgar un vocabulario común entre diseñadores de Software.

Facilitar el aprendizaje para las nuevas generaciones de desarrolladores.

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

Tipos de patrones de diseño

Patrones más utilizados en el diseño

Singleton en PHP

Active Record en PHP

Factory en PHP

Registry en PHP

Value Object en PHP

Del libro Design Patterns del grupo GoF

Se proponen 23 patrones.

Los cuales varían según su granularidad y nivel de abstracción.

Teniendo en cuenta lo anterior se propone clasificar los mismos por tipos o familias.

Clase 10: Introducción a patrones de diseño

Tipos o familias de patrones

De **creación**
- Abstract Facotry
- Factory
- Builder
- Prototype
- Singleton

Estructurales
- Adapter
- Bridge
- Composite
- Decorator
- Facade

De **comportamiento**
- Command
- Strategy
- Observer
- Chain of responsibility

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

Patrones más utilizados en el diseño

Singleton en PHP

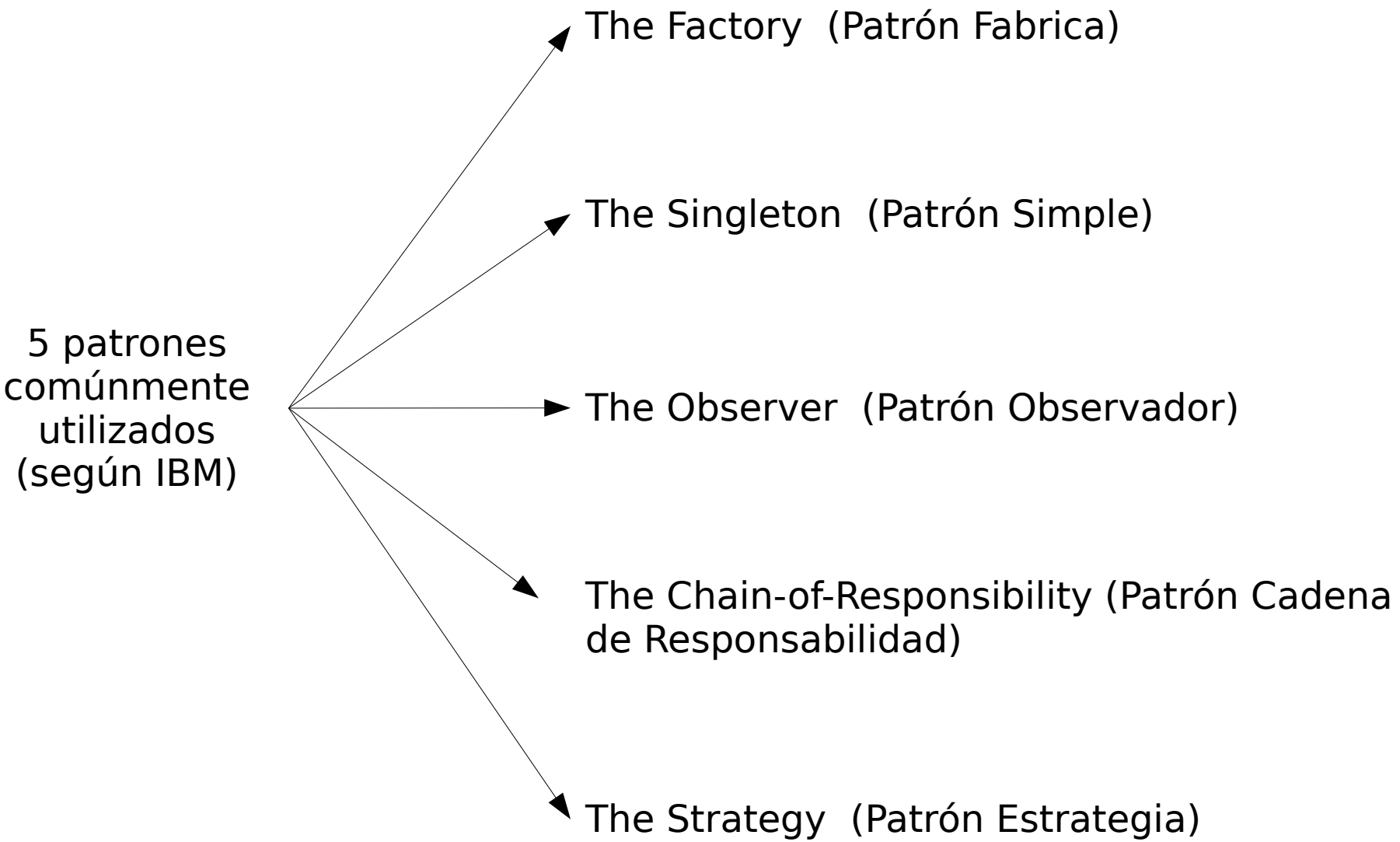
Active Record en PHP

Factory en PHP

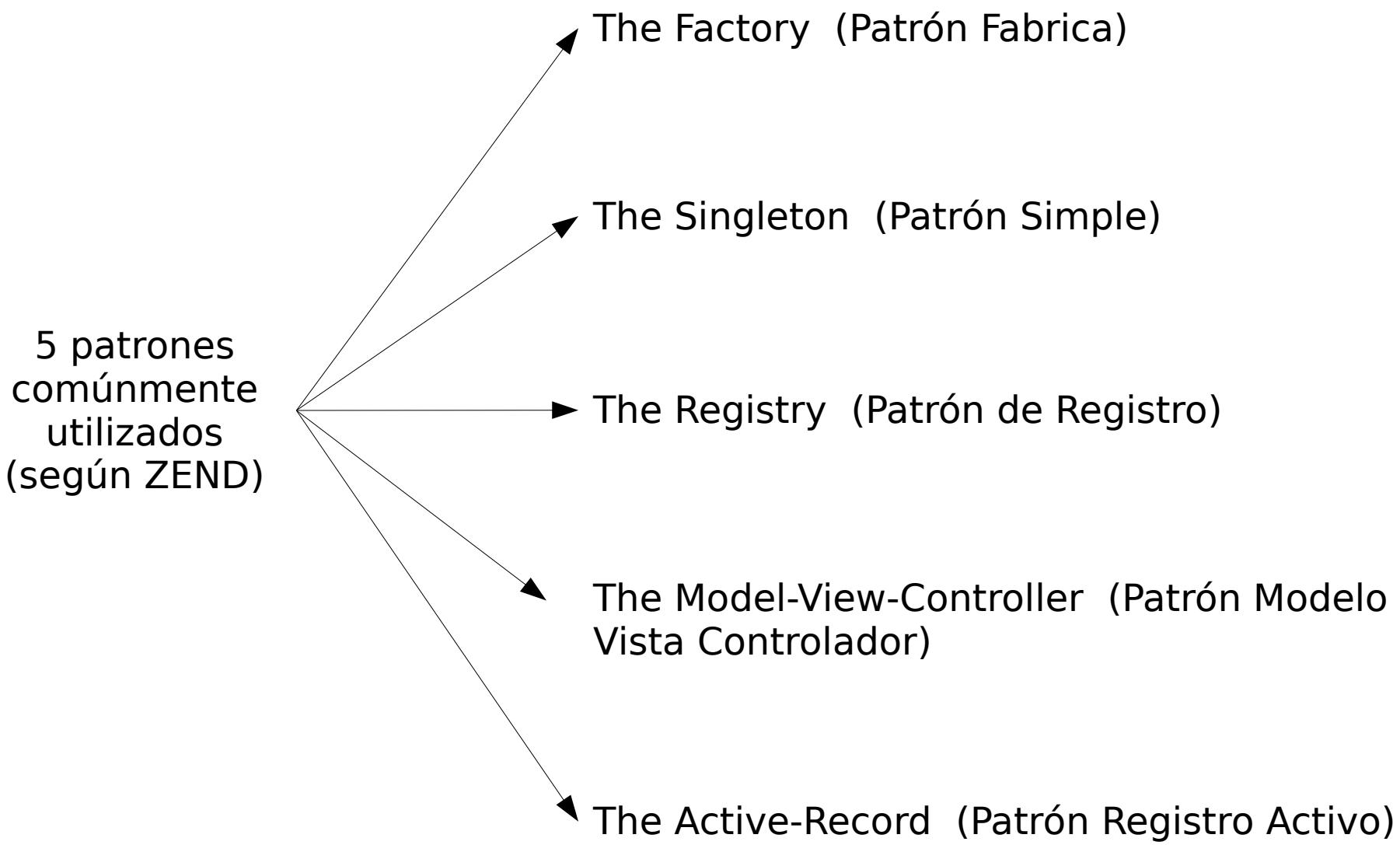
Registry en PHP

Value Object en PHP

Clase 10: Introducción a patrones de diseño



Clase 10: Introducción a patrones de diseño



Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

~~Patrones más utilizados en el diseño~~

Singleton en PHP

Active Record en PHP

Factory en PHP

Registry en PHP

Value Object en PHP

Clase 10: Introducción a patrones de diseño

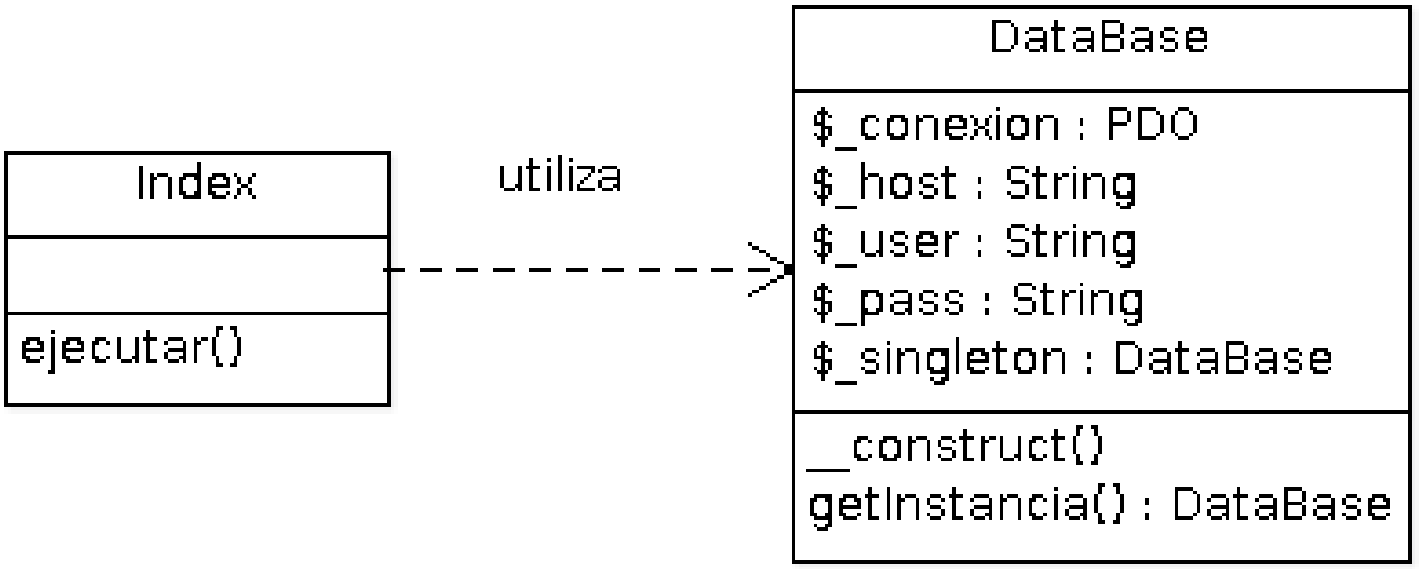
Patrón Singleton

Nos permite garantizar la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Es uno de los patrones más simple de utilizar.

Clase 10: Introducción a patrones de diseño

UML (diagrama de clases) para Singleton



Clase 10: Introducción a patrones de diseño

DataBase.php

```
<?php
class DataBase
{
	private static $_singleton;
	private static $_host = '127.0.0.1';
	private static $_user = 'root';
	private static $_pass = '';
	private $_conexion;
	private function __construct()
	{
		$this->_conexion = new PDO( 'mysql:host='.self::$_host, self::user,
self::pass);
	}
	public static function getInstancia()
	{
		if (is_null(self::$_singleton)) {
			self::$_singleton = new DataBase();
		}
		return self::$_singleton;
	}
}
```

Clase 10: Introducción a patrones de diseño

index.php

```
<?php

require_once 'DataBase.php';

class Index
{
    public function ejecutar()
    {
        $oDataBase = DataBase::getInstancia();
        var_dump($oDataBase);

        $oDataBase1 = DataBase::getInstancia();
        var_dump($oDataBase1);
    }
}

$oIndex = new Index();
$oIndex->ejecutar();
```

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

~~Patrones más utilizados en el diseño~~

~~Singleton en PHP~~

Active Record en PHP

Factory en PHP

Registry en PHP

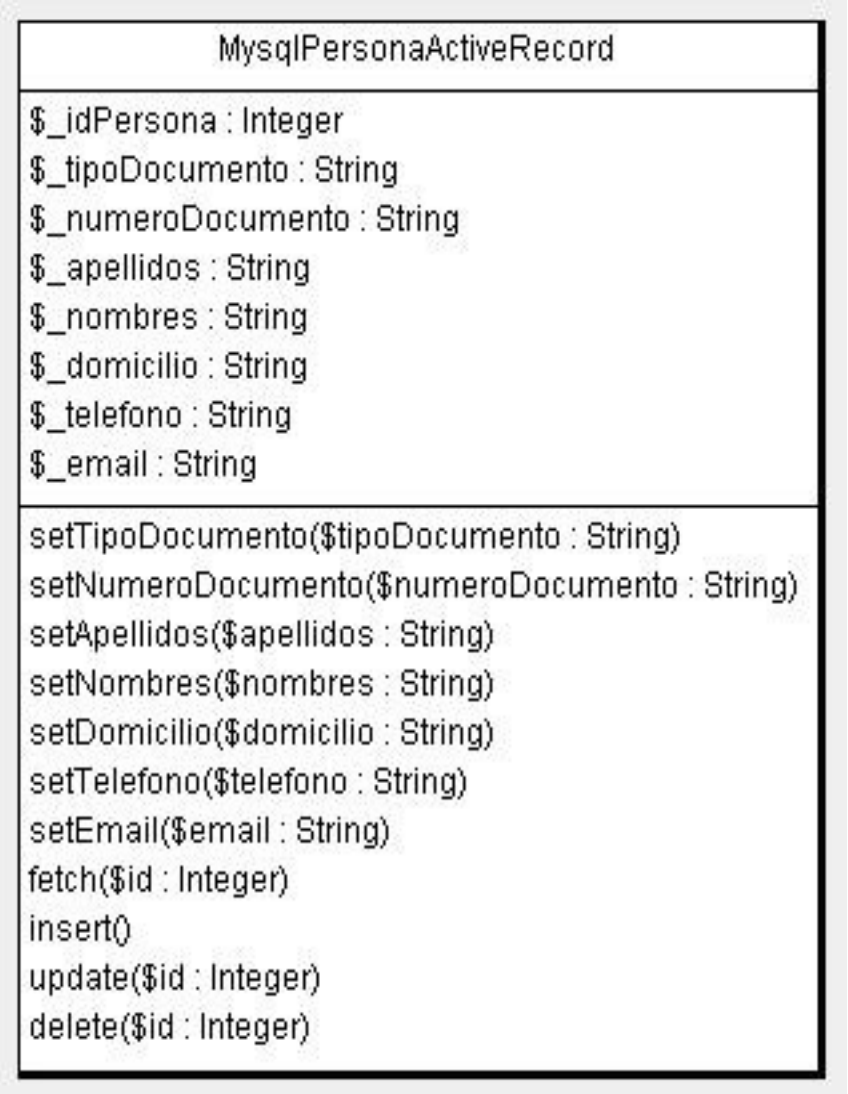
Value Object en PHP

Patrón Active Record

Es un objeto que representa una fila en una tabla de una base de datos, encapsula su acceso a la base de datos y añade lógica de negocio.

Es la aproximación más obvia, poniendo el acceso a la base datos en el propio objeto de negocio. De este modo es evidente como manipular la persistencia a través de el mismo.

UML (diagrama de clases) para Active Record



Clase 10: Introducción a patrones de diseño

MysqlPersonaActiveRecord.php

```
<?php
class MysqlPersonaActiveRecord
{
    private $_idPersona;
    private $_tipoDocumento;
    private $_numeroDocumento;
    private $_apellidos;
    private $_nombres;
    private $_domicilio;
    private $_telefono;
    private $_email;
    public function setTipoDocumento($tipoDocumento)
    {
        $this->_tipoDocumento = $tipoDocumento;
    }
    public function setNumeroDocumento($numeroDocumento)
    {
        $this->_numeroDocumento = $numeroDocumento;
    }
    // Se definen los métodos setter que faltan para cada propiedad
    ...
```

Clase 10: Introducción a patrones de diseño

MysqlPersonaActiveRecord.php

```
// Obtener una fila
    public function fetch($id)
    {
        $sql = “SELECT * FROM persona WHERE id_persona=$id”;
        $stmt = $dbh->query($sql);
        return $stmt->fetch();
    }

// Insertar una fila
    public function insert()
    {
        $sql = “INSERT INTO persona VALUES “;
        $sql .= ('', '$this->_tipoDocumento', '$this->_numeroDocumento',
              '$this->_apellidos', '$this->_nombres', '$this->_domicilio',
              '$this->_telefono', '$this->_email')”;

             if ($dbh->exec($sql) >= 1) {
                  return true;
             } else { return false }
    }
```

Clase 10: Introducción a patrones de diseño

MysqlPersonaActiveRecord.php

```
// Actualizar una fila
    public function update($id)
    {
        $sql = “UPDATE persona SET
            numero_documento=$this->_numeroDocumento,
            apellidos = $this->_apellidos, nombres = $this->_nombres,
            $this->_email WHERE id_persona=$id”;
            if ($dbh->exec($sql) >= 1) {
                return true;
            } else { return false }
    }
// Borrar una fila
    public function delete($id)
    {
            $sql = “DELETE FROM persona WHERE id_persona = $id”;
            if ($dbh->exec($sql) >= 1) {
                return true;
            } else { return false }
    }
} // Fin de la definición de la clase
```

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

~~Patrones más utilizados en el diseño~~

~~Singleton en PHP~~

~~Active Record en PHP~~

Factory en PHP

Registry en PHP

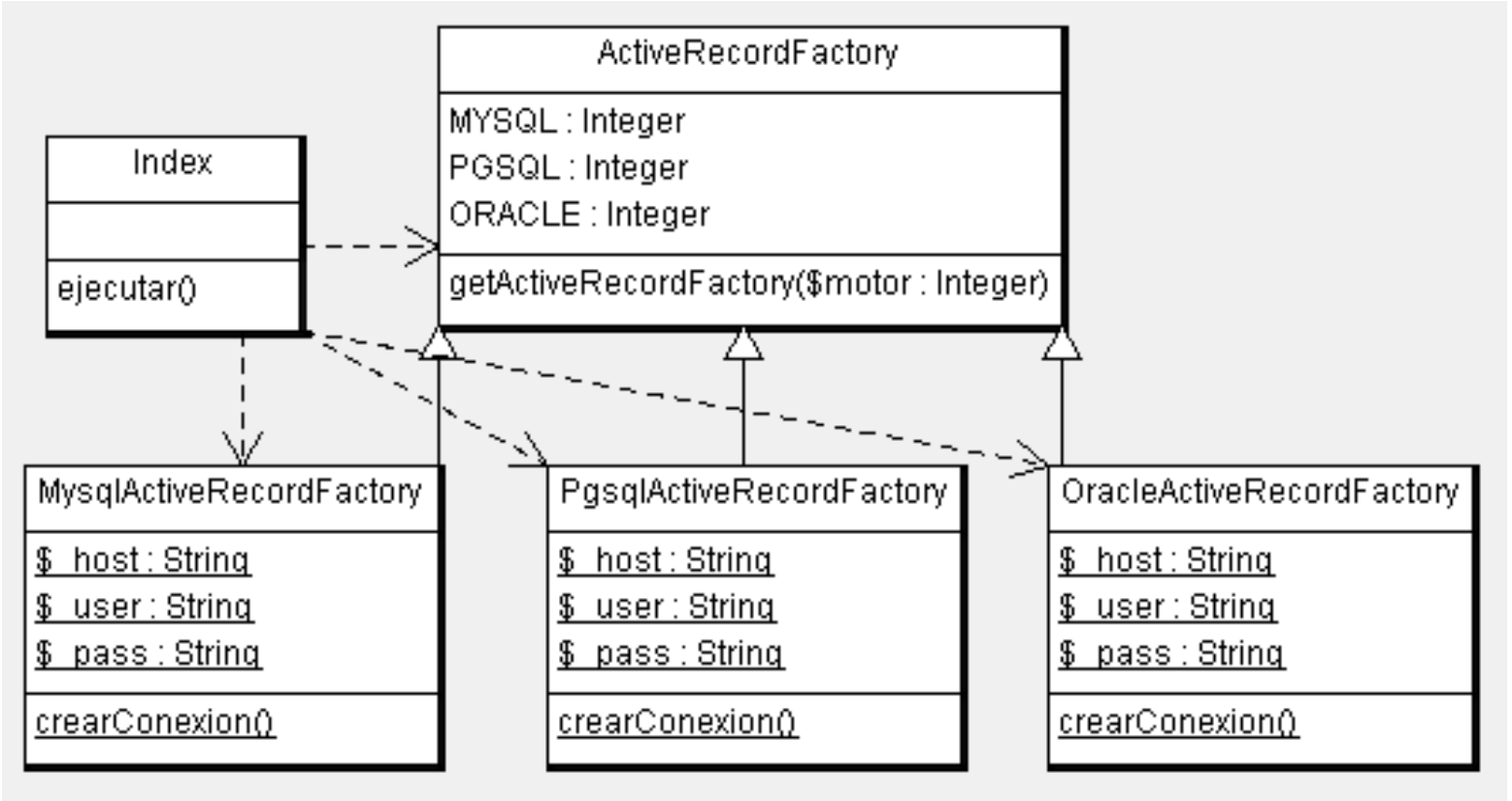
Value Object en PHP

Patrón Factory

Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario los detalles particulares para elegir el subtipo que crear.

Clase 10: Introducción a patrones de diseño

UML (diagrama de clases) para Factory



Clase 10: Introducción a patrones de diseño

ActiveRecordFactory.php

```
<?php
require_once 'MysqlActiveRecordFactory.php';
require_once 'PgsqlActiveRecordFactory.php';
require_once 'OracleActiveRecordFactory.php';
class ActiveRecordFactory
{
    const MYSQL = 1;
    const PGSQL = 2;
    const ORACLE = 3;
    public static function getActiveRecordFactory($motor = selft::MYSQL)
    {
        switch ($motor) {
            case selft::MYSQL:
                return new MysqlActiveRecordFactory();
            case selft::PGSQL:
                return new PgsqlActiveRecordFactory();
            case self:ORACLE:
                return new OracleActiveRecordFactory();
        }
    }
}
```

Clase 10: Introducción a patrones de diseño

MysqlActiveRecordFactory.php

```
<?php
class MysqlActiveRecordFactory
{
    // ...
}
```

PgsqlActiveRecordFactory.php

```
<?php
class PgsqlActiveRecordFactory
{
    // ...
}
```

OracleActiveRecordFactory.php

```
<?php
class OracleActiveRecordFactory
{
    // ...
}
```

Clase 10: Introducción a patrones de diseño

index.php

```
<?php

require_once 'ActiveRecordFactory.php';
require_once 'MysqlActiveRecordFactory.php';
require_once 'PgsqlActiveRecordFactory.php';
require_once 'OracleActiveRecordFactory.php';

class Index
{
    public function ejecutar()
    {

        $oMysqlActiveRecord =
ActiveRecordFactory::getActiveRecordFactory(ActiveRecordFactory::MYSQL);

    }
}

$oIndex = new Index();
$oIndex->ejecutar();
```

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

~~Patrones más utilizados en el diseño~~

~~Singleton en PHP~~

~~Active Record en PHP~~

~~Factory en PHP~~

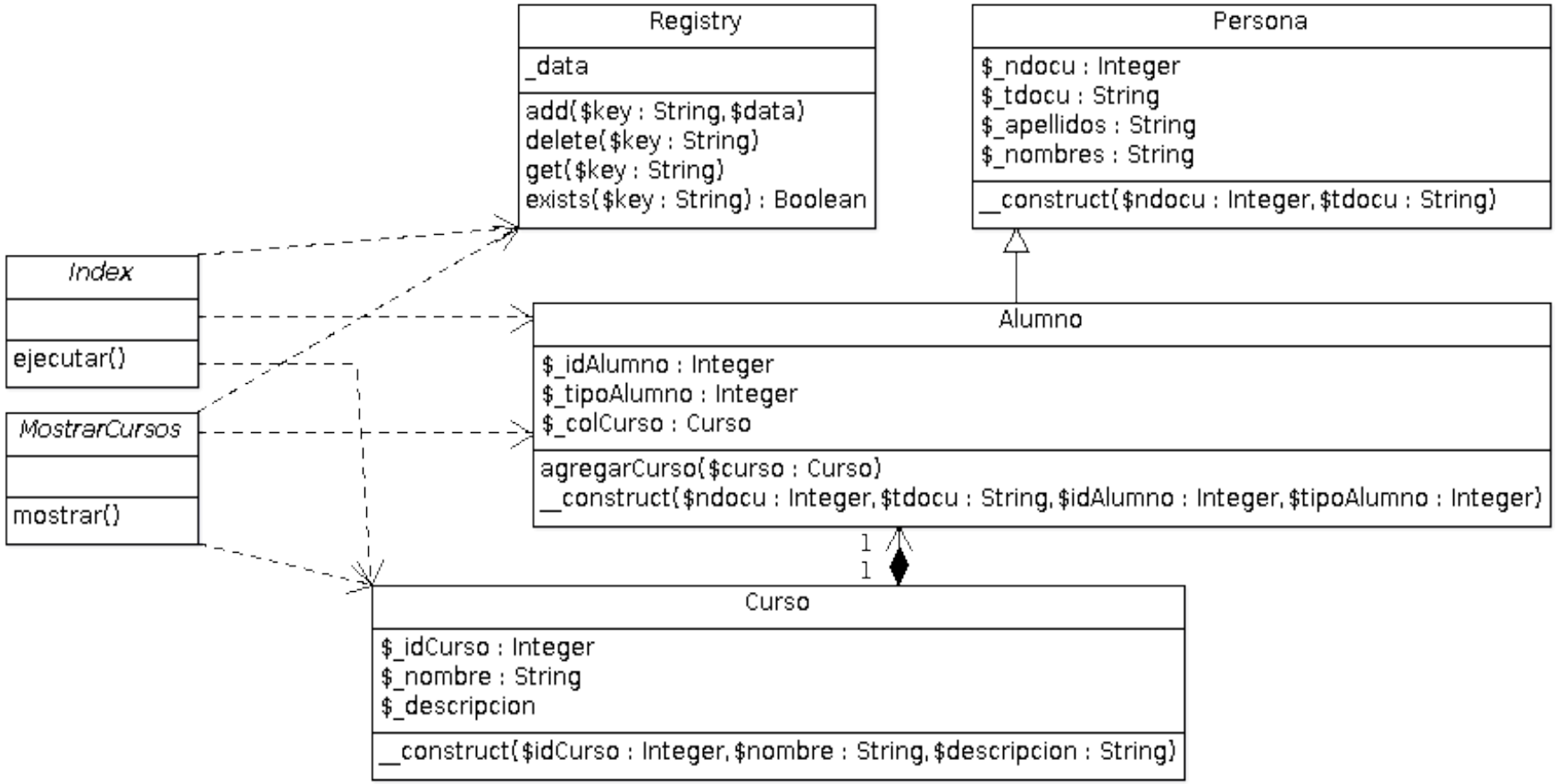
Registry en PHP

Value Object en PHP

Patrón Registry

Es un medio simple y eficiente de compartir datos y objetos en nuestra aplicación sin tener que preocuparse de mantener numerosos parámetros o hacer uso de variables globales.

UML (diagrama de clases) para Registry



Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

Registy.php

```
class Registry
{
	private $_datos = array();

	public function add($key, $data)
	{
		$this->_datos[$key] =  $data;
	}

	public function delete($key)
	{
		unset($this->_datos[$key]);
	}

	public function get($key)
	{
		return $this->_datos[$key];
	}

	public function exists($key)
	{
		if (isset($this->_datos[$key])) {
			return true;
		} else {
			return false;
		}
	}
}
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

Index.php

```
<?php
session_start();

require_once 'Alumno.php';
require_once 'Curso.php';
require_once 'Registry.php';

abstract class Index
{
	public function ejecutar()
	{
		$_SESSION['oRegistry'] = new Registry();
		$oAlumno = new Alumno(25546113, 'DNI', 1, 1);
		$oCurso = new Curso(1, 'Programación en PHP', 'Curso de Desarrollo WEB');
		$oAlumno->agregarCurso($oCurso);
		$_SESSION['oRegistry']->add('Alumno', $oAlumno);
		header("Location: MostrarCursos.php");
	}
}
Index::ejecutar();
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

Persona.php

```
<?php

class Persona
{
	private $_ndocu;
	private $_tdocu;
	private $_apellidos;
	private $_nombres;

	public function __contruct($ndocu, $tdocu)
	{
		$this->_ndocu = $ndocu;
		$this->_tdocu = $ndocu;
	}
}
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

Alumno.php

```
<?php
require_once 'Persona.php';
require_once 'Curso.php';

class Alumno extends Persona
{
	private $_idAlumno;
	private $_tipoAlumno;
	private $_colCurso = array();
	public function __construct($ndocu, $tdocu, $idAlumno, $tipoAlumno)
	{
		parent::__contruct($ndocu, $tdocu);
		$this->_idAlumno = $idAlumno;
		$this->_tipoAlumno = $tipoAlumno;
	}
	public function agregarCurso(Curso $curso)
	{
		$this->_colCurso[] = $curso;
	}
}
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

Curso.php

```
<?php

class Curso
{
	private $_idCurso;
	private $_nombre;
	private $_descripcion;

	public function __construct($idCurso, $nombre, $descripcion)
	{
		$this->_idCurso = $idCurso;
		$this->_nombre = $nombre;
		$this->_descripcion = $descripcion;
	}
}
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

MostrarCursos.php

```
<?php

require_once 'Alumno.php';
require_once 'Curso.php';
require_once 'Registry.php';

session_start();

abstract class MostrarCursos
{
	public function mostrar()
	{
		echo "<h2>Obtengo y muestro el objeto Alumno: </h2><br />";

		$oAlumno = $_SESSION['oRegistry']->get('Alumno');

		var_dump($oAlumno);
		echo "<br />";
```

Clase 10: Introducción a patrones de diseño

Patrón de Diseño Registry
Implementación del patrón en PHP

MostrarCursos.php (continuación...)

```
        echo "<h2>Borro y verifico la existencia del objeto Alumno: </h2><br />";

        $_SESSION['oRegistry']->delete('Alumno');
        echo "<br />";

        echo "Objeto Alumno borrado!!!<br />";
        echo "<br />";

        if ($_SESSION['oRegistry']->exists('Alumno')) {
            echo "El objeto Alumno existe como elemento registrado!";
        } else {
            echo "El objeto Alumno NO existe como elemento registrado!";
        }
    }
}

MostrarCursos::mostrar();
```

Capítulo 5: POO y patrones en PHP

~~Introducción a patrones de diseño~~

~~Tipos de patrones de diseño~~

~~Patrones más utilizados en el diseño~~

~~Singleton en PHP~~

~~Active Record en PHP~~

~~Factory en PHP~~

~~Registry en PHP~~

Value Object en PHP

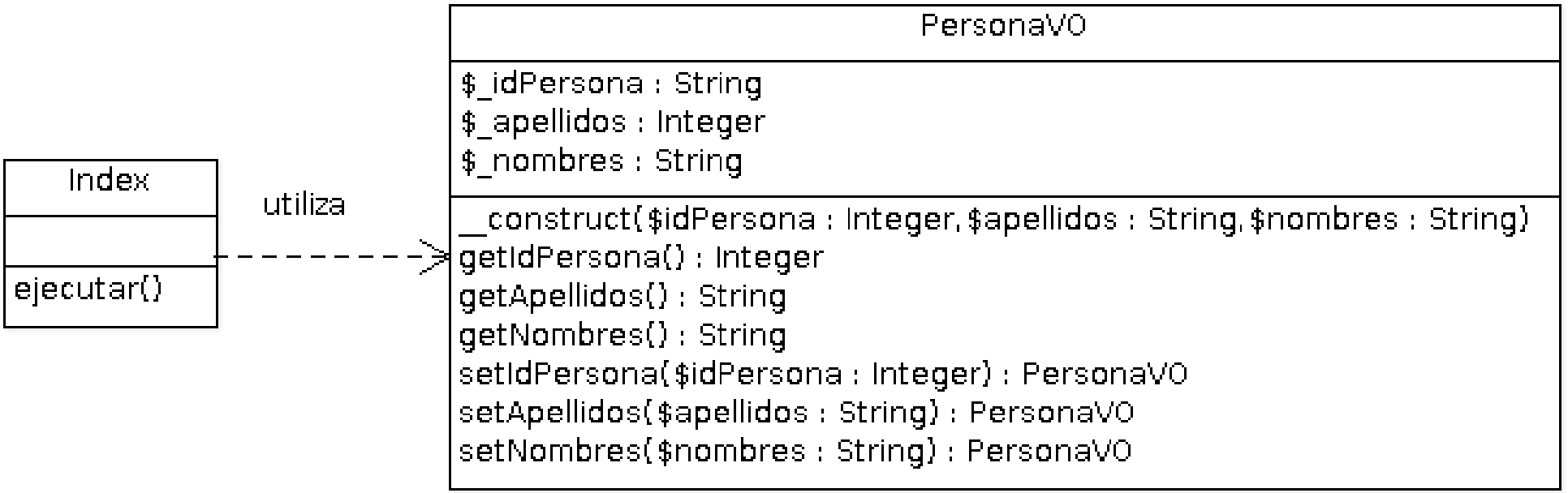
Patrón Value Object

Es un objeto simple que se utiliza para intercambiar información. En su estructura se definen métodos getter y algunas implementaciones métodos setter.

Respetando rigurosamente el patrón no debemos implementar métodos setter en la clase y dejar al constructor la responsabiliadad de establecer valores.

De ahí en más los valores de ese objeto se mantienen inmutables.

UML (diagrama de clases) para Value Object



Clase 10: Introducción a patrones de diseño

Patrón de Diseño Value Object
Implementación del patrón en PHP

PersonaVO.php

```
<?php

class PersonaVO
{
	private $_idPersona;
	private $_apellidos;
	private $_nombres;

	public function __contruct($idPersona=0, $apellidos='', $nombres='')
	{
		$this->_idPersona = $idPersona;
		$this->_apellidos = $apellidos;
		$this->_nombres = $nombres;
	}

	public function getIdPersona()
	{
		return $this->_idPersona;
	}
```

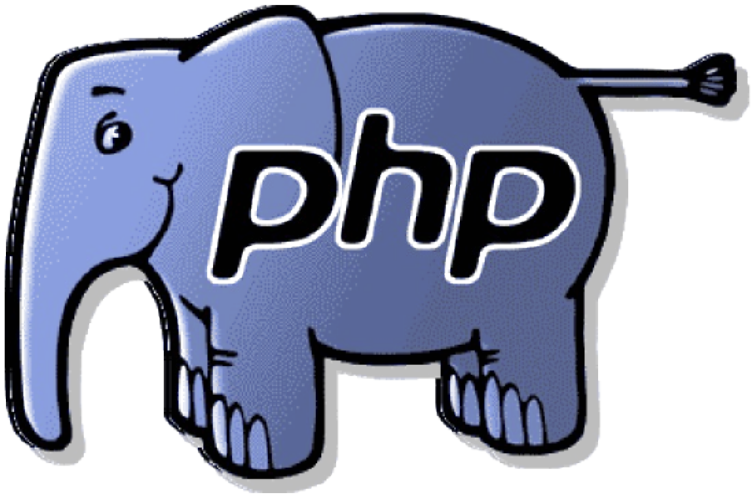
Clase 10: Introducción a patrones de diseño

Patrón de Diseño Value Object
Implementación del patrón en PHP

PersonaVO.php

```
        Public function getApellidos()
        {
            return $this->_apellidos();
        }
        public function getNombres()
        {
            return $this->_nombres();
        }
        public function setIdPersona($idPersona)
        {
            return new Persona($idPersona,$this->_apellidos,$this->_nombres);
        }
        public function setApellidos($apellidos)
        {
            return new Persona($this->_idPersona,$apellidos,$this->_nombres);
        }
        // Setter de los atributos restantes
        ...
}
```

¿Dudas?



¿Consultas?

Información de contacto

Web:

http://www.gugler.com.ar

http://campusvirtual.gugler.com.ar

http://www.facebook.com/gugler.com.ar

http://www.twitter.com/cgugler

Mail:

contacto@gugler.com.ar

academica@gugler.com.ar

administracion@gugler.com.ar

Versión 2.1