Kumbia ActiveRecord Documentation

Release 2.0.0

Kumbia Team

Aug 28, 2017

Contents

Active Record es un enfoque al problema de acceder a los datos de una base de datos relacionales en forma orientada a objetos. Una fila en la tabla de la base de datos (o vista) se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado.

Cuando se crea uno de estos objetos, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos.

CHAPTER 1

Características Principales

- Facilidad de uso e implementación
- Functiona y aprovecha las ventajas de PDO
- Puede ser usado en cualquier proyecto de manera independiente
- Aprovecha las nuevas funcionalidades de PHP 5.3

CHAPTER 2

Indice de Contenidos

Introducción

En esta sección a nivel de introducción explicaremos porqué debería usarse un ORM como ActiveRecord y las ventajas que representa este para una aplicación profesional en PHP.

¿Qué es?

ActiveRecord es un componente que implementa el patrón ORM (Object Relational-Mapping), gracias a este, podemos trabajar con bases de datos de manera orientada a objetos, encapsulando muchos detalles de bajo nivel, facilitando la manutención del código y haciendo las aplicaciones menos propensas a errores.

¿Por qué usar ActiveRecord?

En el pasado era normal usar código como el siguiente para acceder a bases de datos:

```
<?php

mysql_query("SELECT * FROM usuarios LIMIT 10");
while ($usuario = mysql_fetch_array()) {
        echo $usuario['nombre'];
}
```

Y si queriamos actualizar los registros consultados haciamos algo como esto:

```
<?php

mysql_query("SELECT * FROM usuarios LIMIT 10");
while ($usuario = mysql_fetch_array()) {
        mysql_query("UPDATE usuarios SET fecha = '".date('Y-m-d')."' WHERE id = '.
↪$usuario['id']);
}
```

Esta forma de trabajar representa muchas desventajas para nuestra aplicación:

- Usamos funciones que están atadas a un motor especifico (MySQL en este caso) haciendo dificil usar la aplicación en otros motores
- Escribir SQL nos lleva a usar extensiones que puede que no sean compatibles en otros motores (por ejemplo LIMIT)
- Escribir SQL requiere de cuidado pues al olvidar una simple comilla podriamos hacer fallar la aplicación
- Si nuestra aplicación es orientada a objetos estaríamos mezclandola con programación estructurada lo cual es una mala práctica
- El código podría ser potencialmente suceptible a ataques de inyección de SQL si no se toman las debidas precauciones

¿Qué es un modelo?

Las bases de datos relacionales almacenan los datos en tablas de esta manera se organiza la naturaleza de la información. Las tablas tienen columnas, cada columna representa un atributo de una entidad. La siguiente tabla "personas" nos muestra esto:

```
mysql> desc personas;
+------------------+------------------+------+-----+---------+----------------+
| Field            | Type             | Null | Key | Default | Extra          |
+------------------+------------------+------+-----+---------+----------------+
| id               | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| apellidos        | varchar(120)     | NO   |     | NULL    |                |
| nombres          | varchar(120)     | NO   |     | NULL    |                |
| fecha_nacimiento | date             | YES  |     | NULL    |                |
| estado_civil     | varchar(24)      | YES  |     | NULL    |                |
| nacionalidad     | varchar(50)      | YES  |     | NULL    |                |
+------------------+------------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Esta tabla tiene algunos atributos para representar a una persona, algunos datos en esta tabla podrían ser los siguientes:

```
mysql> select * from personas;
+----+-----------+---------+------------------+--------------+--------------+
| id | apellidos | nombres | fecha_nacimiento | estado_civil | nacionalidad |
+----+-----------+---------+------------------+--------------+--------------+
|  1 | Perez     | Juan    | 1979-05-17       | Casado       | Mexicano     |
|  2 | Martinez  | Rosario | 1985-11-10       | Soltero      | Peruano      |
|  3 | Arenas    | Manuel  | 1963-08-24       | Viudo        | Boliviano    |
+----+-----------+---------+------------------+--------------+--------------+
3 rows in set (0.00 sec)
```

En ActiveRecord para representar una tabla creamos una clase, una clase asociada a una tabla es un modelo. La siguiente clase "Personas" es una clase que "mapea" la tabla "personas":

```
<?php

class Personas extends \ActiveRecord\Model
{

}
```

Un modelo no requiere una implementación muy sofisticada inicialemente, solamente extender la clase "\ActiveRecord\Model". Las tablas son clases y las filas son objectos. Las columnas son atributos de las clases. De la siguiente manera entendemos que:

- Crear una instancia de una clase es crear una fila
- Actualizar los atributos de un objeto es actualizar los valores de sus columnas
- Eliminar un objeto es eliminar la fila

Veámolo en código:

```
<?php

//Creamos una nueva persona
$persona = new Personas();

//Asignamos valores a sus atributos
$persona->apellidos = 'Alvarez';
$persona->nombre = 'Carolina';
$persona->fecha_nacimiento = '1991-10-14';
$persona->estado_civil = 'Soltera';
$persona->nacionalidad = 'Uruguaya';

//Guardamos el nuevo registro
$persona->save();
```

El anterior código agregaría un nuevo registro a la tabla "personas":

```
mysql> select * from personas;
+----+-----------+----------+------------------+--------------+--------------+
| id | apellidos | nombres  | fecha_nacimiento | estado_civil | nacionalidad |
+----+-----------+----------+------------------+--------------+--------------+
|  1 | Perez     | Juan     | 1979-05-17       | Casado       | Mexicano     |
|  2 | Martinez  | Rosario  | 1985-11-10       | Soltero      | Peruano      |
|  3 | Arenas    | Manuel   | 1963-11-10       | Viudo        | Boliviano    |
|  4 | Alvarez   | Carolina | 1991-10-14       | Soltera      | Uruguaya     |
+----+-----------+----------+------------------+--------------+--------------+
4 rows in set (0.00 sec)
```

Como pudimos ver el registro es insertado sin requerir escribir SQL tan solo usando programación orientada a objetos. ActiveRecord también nos permite consutlar registros existentes, actualizarlos y eliminarlos si es necesario:

```
<?php

//Consultar la persona con id=1
$persona = Personas::first(1);

//Cambiar su estado civil
$persona->estado_civil = 'Soltero';

//Guardar los cambios
$persona->save();

//Consultar todas las personas Bolivianas
foreach (Personas::find("nacionalidad='Boliviano'") as $persona) {
        echo $persona->nombres;
}

//Eliminar todas las personas Casadas
```

```
foreach (Personas::find("estado_civil='Casado'") as $persona) {
        $persona->delete();
}
```

Configuración Inicial

ActiveRecord permite administrar varias configuraciones de bases de datos que podrán ser usadas por nuestros modelos.

Estableciendo parámetros de conexión

La clase Parameters nos permitirá describir conexiones, esta clase nos da la libertad de definir los parámetros de varias formas:

```
<?php

use ActiveRecord\Config\Parameters;

//en el siguiente ejemplo crearemos la configuración en el constructor de la clase:

$mysql = new Parameters("configMysql", array(
        "type"     => "mysql"
        'username' => "root"
        "password" => "123456"
        "host"     => "localhost"
        "name"     => "mi_base_de_datos"
));

//Ahora configuraremos otra conexión, pero a traves de los métodos setters de la␣
↪clase:

$postgres = new Parameters("configPostgres"); //no pasamos el arreglo con la␣
↪configuración.

$postgres->setUsername("administrador")
        ->setPassword("admin123Admin")
        ->setHost("192.168.1.3")
        ->setPort("2020")
        ->setDbName("clinicas")
        ->setType("pgsql");
```

Como se puede ver, esas son las dos formas de crear la configuración de acceso a una base de datos.

> NOTA: en el constructor el indice para el nombre de la base de datos debe ser “name” aunque realmente dicho valor se almacena en la propiedad $dbName de la clase Parameters, por eso el método para establecer el nombre de la base de datos es “setDbName()”.

Anteriormente hemos usado los nombres “configMysql” y “configPostgres”, con estos nombres podremos identificar la conexión y asignarla al modelo:

```
<?php

use ActiveRecord\Model;
```

```
class Clientes extends Model
{
        protected $connection = "config_postgres"
}

class Clientes extends Model
{
        protected $connection = "config_mysql"
}

class Clientes extends Model
{
        //si no establecemos la conexión a usar, la libreria
        //tomará la primera configuración establecida en la clase Config
}
```

Administrador de Conexiones

De acuerdo al nombre de conexión que asignemos al modelo, ActiveRecord solicitará a la clase Config estos parámetros cuando los requiera. Config actúa como un administrador de conexiones, por esto debemos agregar los parametros a esta para que estén disponibles para nuestros modelos:

```
<?php

use ActiveRecord\Config\Parameters;
use ActiveRecord\Config\Config;

//creamos los objetos Parameters necesarios, como en el ejemplo anterior.
$mysql = new Parameters("configMysql", array(
        "type"     => "mysql"
        'username' => "root"
        "password" => "123456"
        "host"     => "localhost"
        "name"     => "mi_base_de_datos"
));

//luego agregamos los parametros a la configuración
Config::add($mysql);

//multiples conexiones pueden ser agregadas a Config
Config::add($postgres);
```