Librería Idiorm

Que es Idiorm

IdiOrm es una librería fácil de usar, su énfasis es la comodidad de otorgar simplicidad y desarrollo rápido para proyectos pequeños/medianos. Librería para construir abstracciones de bases de datos de alto nivel y ahorrar códigos engorrosos de SQL.

GitHub: j4mie/idiorm
Documentación: Docs Idiorm

Instalación

Composer

Descargue y ejecute Composer-Setup.exe Instalará la última versión de Composer y configurará su PATH para que pueda llamar desde cualquier directorio en su línea de comandos `.composer`

Descargar

Puede clonar el repositorio git, descargar idiorm.php o una etiqueta de lanzamiento y luego soltar el archivo idiorm.php en el directorio vendors/3rd party/libs o libreria de su proyecto.

Configuración

Como punto principal de idiOrm es que no es necesario una clase de modelo para usarlo, con idiOrm puede usar el ORM de inmediato.

Primero, el archivo fuente de Idiorm:`require`

```
<?php
```

```
require_once 'idiorm.php';
```

A continuación, pase una cadena de conexión *Nombre de origen de datos* al método de la clase ORM. Esto es utilizado por PDO para conectarse a su base de datos
Es posible que también deba pasar un nombre de usuario y una contraseña al controlador de base de datos, mediante las opciones de configuración.Por ejemplo, si está utilizando MySQL:`username password`

```
<?php

ORM::configure('mysql:host=localhost;dbname=my_database');
ORM::configure('username', 'database_user');
ORM::configure('password', 'top_secret');
```

Consultas (Querys)

Como ya sabemos las cuatro consultas que utilizamos para manipular una base de datos son “INSERT”, “SELECT”,”UPDATE” y “DELETE”, con idiorm podemos simplificar estas consultas, en esta documentacion vamos a mostrar las mas basicas, como por ejemplo:

INSERT: añade filas en una tabla.

Para generar un insert solo debemos inicializar una variable llamando a la clase ORM, le indicamos a que tabla queremos insertar e iniciamos la funcion create(), una vez inicializado llamamos a los campos que queremos completar con datos, para finalizar y guardar la consulta finalizamos la variable con save().

```
<?php
//Inicializamos la consulta
$person = ORM::for_table('person')->create();

//Cargamos los campos con sus respectivos datos
$person->name = 'Joe Bloggs';
$person->age = 40;

//Finaliza la query y devuelve true si se guardo, caso contrario devuelve false
$person->save();

//Una vez que se utiliza save podemos recuperar la id autogenerada
$autoID = $person->id();
```

SELECT: permite consultar registros de la base de datos.

Tenemos varias formas de generar un select, que van desde recuperar un dato de una tabla o recuperar todos los datos.

Por ejemplo, si queremos buscar un dato especifico que este dentro de una tabla podemos hacer uso de where, inicializamos una variable con la clase ORM, le indicamos la tabla, y luego en where le indicamos el campo de la tabla y el dato que estamos buscando, de todo esto finalizamos con find_one(el cual nos devuelve el objeto person).

```php
<?php
//inicializamos la consulta para buscar una coincidencia
$person = ORM::for_table('person')->where('name', 'Fred Bloggs')->find_one();

//Podemos obtener los campos como si fuesen getters de la tabla consultada
$name = $person->get('name');
$age = $person->get('age');

// o llamarlos de la siguiente forma
$name = $person->name;
$age = $person->age;

//Otra consulta basica que podemos hacer es buscar por id
$person = ORM::for_table('person')->find_one($id);

//Para consultar todos los datos de la tabla se utiliza
$person = ORM::for_table('person')->find_many();

//Para que nos devuelva en formato de array usamos
$person = ORM::for_table('person')->find_array();

//Para consultar todos los datos que contengan un valor, por ej, la edad usamos
$person = ORM::for_table('person')->where('age', '25')->find_many();
```

UPDATE: actualiza los valores de campos y registros de una tabla.

Para actualizar datos, podemos hacerlo de dos formas, una es que si necesitamos cambiar campos del dato debemos buscarlo de forma individual, otra es buscar un conjunto de datos y cambiar uno de sus campos, algo similar a “update on cascade”

```php
<?php

// Buscamos el id de valor 5 de la tabla 'person'
$person = ORM::for_table('person')->find_one(5);

// Podemos editar los campos de las siguientes formas
$person->set('name', 'Bob Smith');
$person->age = 20;
```

```php
// Tenemos la posibilidad de setear con un array de datos
$person->set(array(
    'name' => 'Bob Smith',
    'age'  => 20
));

// Una vez que terminamos de setear guardamos con save, este devuelve un booleano
$person->save();
```

DELETE: borra filas de una tabla

```php
<?php
// Buscamos por id un dato de la tabla 'person'
$person = ORM::for_table('person')->find_one(5);
// Una vez que lo tenemos lo borramos con delete(), este devuelve un booleano
$person->delete();

// Podemos borrar tambien varios datos de una tabla, en este caso los que
// contengan en su campo 'zipcode' el dato "55555".
$person = ORM::for_table('person')
    ->where_equal('zipcode', 55555)
    ->delete_many();
```

Esta libreria cuenta con varias formas de generar consultas, Tiene opciones como:

- Utilizar JOIN(join), LIKE(where_like), ORDER BY (order_by_desc, order_by_asc), GROUP_BY (group_by), entre otros similares.
- Hacer Count´s de los datos consultados.
- Configurar id's compuestas de tablas.
- Renombrar campos al estilo AS (...->select(campo, nuevo nombre)->...).
- Deshabilitar los CamelCase "_" para pasar de "for_table()" a "forTable()"