Active Record

Грушевский Ю.В.

Active Record

Active Record обеспечивает объектно-ориентированный интерфейс для доступа и манипулирования данными, хранящимися в базах данных. Класс

Active Record соответствует таблице в базе данных, объект Active Record соответствует строке этой таблицы, а атрибут объекта Active Record представляет собой значение отдельного столбца строки. Вместо непосредственного написания SQL-выражений вы сможете получать доступ к атрибутам Active Record и вызывать методы Active Record для доступа и манипулирования данными, хранящимися в таблицах базы данных.

Получение данных

```php
// возвращает покупателя с идентификатором 123
// SELECT * FROM `customer` WHERE `id` = 123
$customer = Customer::find()
    ->where(['id' => 123])
    ->one();

// возвращает всех активных покупателей, сортируя их по идентификаторам
// SELECT * FROM `customer` WHERE `status` = 1 ORDER BY `id`
$customers = Customer::find()
    ->where(['status' => Customer::STATUS_ACTIVE])
    ->orderBy('id')
    ->all();

// возвращает количество активных покупателей
// SELECT COUNT(*) FROM `customer` WHERE `status` = 1
$count = Customer::find()
    ->where(['status' => Customer::STATUS_ACTIVE])
    ->count();

// возвращает всех покупателей массивом, индексированным их идентификаторами
// SELECT * FROM `customer`
$customers = Customer::find()
    ->indexBy('id')
    ->all();
```

Доступ к данным

```
// "id" и "email" - названия столбцов в таблице "customer"
$customer = Customer::findOne(123);
$id = $customer->id;
$email = $customer->email;
```

Сохранение данных

```php
// вставить новую строку данных
$customer = new Customer();
$customer->name = 'James';
$customer->email = 'james@example.com';
$customer->save();

// обновить имеющуюся строку данных
$customer = Customer::findOne(123);
$customer->email = 'james@newexample.com';
$customer->save();
```

```php
public function save($runValidation = true, $attributeNames = null)
{
    if ($this->getIsNewRecord()) {
        return $this->insert($runValidation, $attributeNames);
    } else {
        return $this->update($runValidation, $attributeNames) !== false;
    }
}
```

Массовое присваивание

```
$values = [
    'name' => 'James',
    'email' => 'james@example.com',
];

$customer = new Customer();

$customer->attributes = $values;
$customer->save();
```

Удаление данных

```
$customer = Customer::findOne(123);
$customer->delete();
```

```
Customer::deleteAll(['status' => Customer::STATUS_INACTIVE]);
```

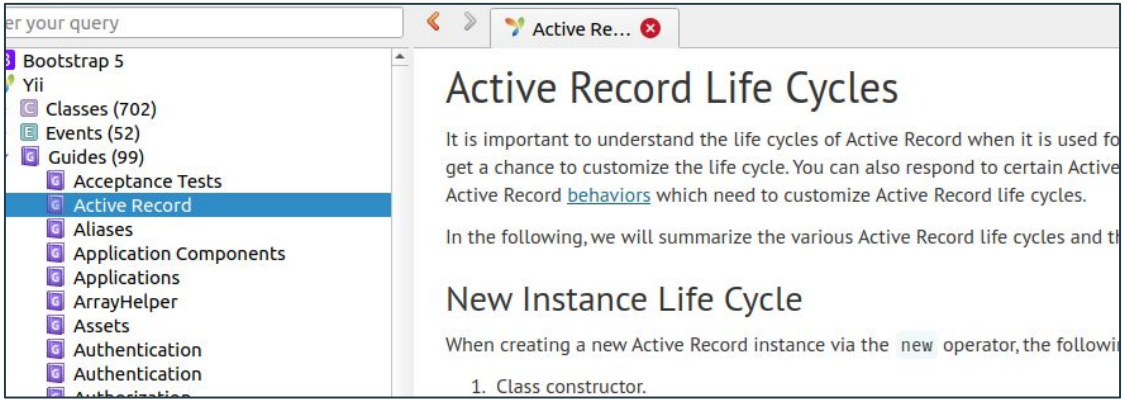
Жизненные циклы Active Record

создания нового объекта

получения данных

сохранения данных

удаления данных



Работа со связанными данными

Помимо работы с отдельными таблицами баз данных, Active Record также имеет возможность объединять связные данные, что делает их легкодоступными для получения через основные объекты данных.

Например, данные покупателя связаны с данными заказов, потому что один покупатель может осуществить один или несколько заказов. С помощью объявления этой связи вы можете получить возможность доступа к информации о заказе покупателя с помощью выражения $customer->orders, которое возвращает информацию о заказе покупателя в виде массива объектов класса Order, которые являются Active Record объектами.

Объявление связей

Для работы со связными данными посредством Active Record вы прежде всего должны объявить связи в классе Active Record.

Эта задача решается простым объявлением методов получения связанных данных для каждой интересующей вас связи

```php
class Customer extends ActiveRecord
{
    public function getOrders()
    {
        return $this->hasMany(Order::class, ['customer_id' => 'id']);
    }
}

class Order extends ActiveRecord
{
    public function getCustomer()
    {
        return $this->hasOne(Customer::class, ['id' => 'customer_id']);
    }
}
```

Доступ к связанным данным

После объявления связей вы можете получать доступ к связанным данным с помощью имён связей. Это происходит таким же образом, каким осуществляется доступ к свойству объекта объявленному с помощью метода получения связанных данных.

```
// SELECT * FROM `customer` WHERE `id` = 123
$customer = Customer::findOne(123);

// SELECT * FROM `order` WHERE `customer_id` = 123
// $orders - это массив объектов Order
$orders = $customer->orders;
```

Связывание посредством промежуточной таблицы

Один заказ будет соотноситься с несколькими товарами, в то время как один товар будет также соотноситься с несколькими заказами.

```php
// SELECT * FROM `order` WHERE `id` = 100
$order = Order::findOne(100);

// SELECT * FROM `order_item` WHERE `order_id` = 100
// SELECT * FROM `item` WHERE `item_id` IN (...)
// возвращает массив объектов Item
$items = $order->items;
```

```php
class Order extends ActiveRecord
{
    public function getItems()
    {
        return $this->hasMany(Item::class, ['id' => 'item_id'])
            ->viaTable('order_item', ['order_id' => 'id']);
    }
}
```

```php
class Order extends ActiveRecord
{
    public function getOrderItems()
    {
        return $this->hasMany(OrderItem::class, ['order_id' => 'id']);
    }

    public function getItems()
    {
        return $this->hasMany(Item::class, ['id' => 'item_id'])
            ->via('orderItems');
    }
}
```