













Phalcon Model (ORM, Raw Query, PHQL, Transaction)

Pemrograman Web F

Sebelum dapat melakukan transaksi data, kita harus terlebih dahulu memastikan bahwa aplikasi kita sudah dapat berkomunikasi dengan database. Contoh table database yang digunakan pada tutorial ini bernama **todos** yang memiliki 2 kolom, yaitu : **id** (int), **content** (varchar(255))

						id	content	
<input type="checkbox"/>		Edit		Copy		Delete	1	makan
<input type="checkbox"/>		Edit		Copy		Delete	4	minum
<input type="checkbox"/>		Edit		Copy		Delete	5	tidur
<input type="checkbox"/>		Edit		Copy		Delete	6	coding

1. Object Relational Mapping (ORM)

ORM adalah suatu metode untuk mengkonversi data dari lingkungan Pemrograman Berorientasi Objek dari / ke lingkungan database. Sehingga, kita bisa melakukan operasi query ke database menggunakan sintaks OOP. Dalam *framework* Phalcon, ORM ini biasa disebut **Model**.

Untuk dapat melakukan operasi database menggunakan model, terlebih dahulu kita harus membuat class **Todos** yang adalah turunan dari class **Phalcon\Mvc\Model**. (Nama class **harus** dalam format **Camel Case**).

```
<?php
use Phalcon\Mvc\Model;

class Todos extends Model
{
    public $id;
    public $content;
}
```

Class **Todos** secara default akan merujuk ke table **todos** di database. Apabila terdapat class **TodoCategories** secara default akan merujuk ke table **todo_categories** di database. Apabila anda ingin mengatur table yang dirujuk secara manual, anda dapat melakukannya dengan cara memanggil fungsi **setSource()** dengan parameter nama table yang hendak dirujuk.

```
<?php

use Phalcon\Mvc\Model;

class Todos extends Model
{
    public $id;
    public $content;

    public function initialize()
    {
        $this->setSource('my_todos');
    }
}
```

Setiap objek dari model merepresentasikan satu baris data pada table database. Anda dapat membaca data *record* dengan cara membaca attribute dari objek tersebut.

```
<?php

// Mencari record dengan id 4
$todo = Todos::findFirst(4);

// Akan mencetak "Minum"
$todo->content;
```

Ketika anda sudah mendapatkan record yang diinginkan dalam bentuk objek, anda dapat merubah kemudian menyimpan datanya.

```
<?php

// Mencari record dengan id 4
$todo = Todos::findFirst(4);

$todo->content = "Mengerjakan proyek web 3";

$todo->save();
```

Untuk lebih lengkapnya mengenai penggunaan class Model untuk transaksi ke database, silahkan dibaca di link berikut : <https://docs.phalconphp.com/en/3.2/db-models>

2. Raw Query

Selain menggunakan ORM untuk melakukan transaksi ke database, kita juga dapat melakukan transaksi ke database secara langsung dengan menggunakan sintaks SQL.

Tambahkan parameter **options** seperti contoh berikut pada bagian database di **services.php**

```
$di->set(
    'db',
    function () use ($config) {
        $dbAdapter = $config->database->adapter;

        return new $dbAdapter([
            "host" => $config->database->host,
            "username" => $config->database->username,
            "password" => $config->database->password,
            "dbname" => $config->database->dbname,
            "options" => [PDO::ATTR_DEFAULT_FETCH_MODE=>PDO::FETCH_ASSOC],
        ]);
    }
);
```

Tujuannya agar setiap data yang didapatkan sebagai hasil query dikembalikan dalam bentuk *associative_array*.

Pada controller anda bisa melakukan *query* ke database dengan cara seperti berikut :

```
// Mengambil data dari database
// Dikembalikan dalam bentuk associative_array
$todos = $this->db->query("SELECT content FROM todos")->fetchAll();

// $count akan berisi 4
$count = count($todos);

// Akan mencetak :
// makan
// minum
// tidur
// coding

foreach ($todos as $todo) {
    echo $todo['content']."\n";
}
```

3. PHQL

PHQL (Phalcon Query Language) adalah sebuah SQL parser yang bersifat *object-oriented* yang memungkinkan kita untuk membuat sebuah sintaks query seperti pada umumnya.

Pada controller atau view, anda dapat menggunakan PHQL ini menggunakan properti *models manager* yang telah diinjeksi kan oleh Phalcon.

```
$todos = $this->modelsManager->executeQuery(
    'SELECT * FROM Todos'
);

// Atau dengan parameter
$todos = $this->modelsManager->executeQuery(
    'SELECT * FROM Todos WHERE content = :content_body:',
    [
        'content_body' => "coding"
    ]
);
```

Hasil dari query tersebut disimpan dalam bentuk objek dari class **Phalcon\Mvc\Model\ResultSetSimple** . Objek ini menampung semua object model dari hasil query.

```
// Akan mencetak
// makan
// minum
// tidur
// coding
foreach ($todos as $todo) {
    echo $todo->content."\n";
}
```

Untuk lebih detilnya tentang penggunaan PHQL, bisa dibaca di link berikut :

<https://docs.phalconphp.com/uk/3.2/db-phql>

4. Transaction

Transaction pada Phalcon adalah sebuah cara untuk memastikan bahwa semua transaksi pada database berhasil dilakukan. Ketika semua transaksi berhasil, Transaction akan melakukan **commit** agar semua data disimpan ke dalam database. Jika terjadi kesalahan, Transaction akan melakukan **rollback** (membatalkan) semua transaksi yang dilakukan.

Untuk menggunakan transaction, kita memerlukan sebuah Transaction Manager yang mengatur setiap transaksi yang dibuat secara global, memastikan bahwa mereka (setiap transaksi) berhasil di commit / di roll back sebelum mengakhiri *request*.

Pertama, kita daftarkan dulu Transaction Manager ke Service Container agar setiap kali kita akan menggunakannya, kita tidak perlu melakukan instansiasi terlebih dahulu.

```
$di->set(
    'transactions',
    function () {
        return new \Phalcon\Mvc\Model\Transaction\Manager();
    }
);
```

```

// Memanggil object Transaction yang telah disimpan di Service Container
$transaction = $this->transactions->get();

$todos = new Todos();

// Menggunakan transaction manager pada Model
$todos->setTransaction($transaction);

$todos->content = $content;

// Jika terjadi kesalahan saat menyimpan data ke dalam database
if ($todos->save() === false)
{
    // Batalkan transaksi database tersebut
    $transaction->rollback(
        "Cannot save new todo item"
    );
}

// Jika tidak terjadi kesalahan, maka commit semua transaksi
$transaction->commit();

```

Transaction pada Phalcon berguna untuk menjaga integritas data selama melakukan operasi database. Misalkan, ketika melakukan delete, update pada record yang saling terhubung melalui Foreign Key.

Untuk lebih jelasnya mengenai penggunaan Transaction, bisa dibaca di link berikut :
<https://docs.phalconphp.com/uk/3.3/db-models-transactions#isolated>