

í»j

Autor: Felipe Bastos

Website: <http://www.felipebastosweb.com.br>

GeraÃ§Ã£o: 22-02-2012 20:43:00

[Kohana Buscando dados do banco](#)

Categoria: [Kohana](#) Publicado dia: 21/03/2011 11:07:23 ComentÃ¡rios:

O Kohana fornece ao desenvolvedor duas formas de acesso ao Banco de Dados. A mais bÃ¡sica Ã© o MÃ³dulo Database, que faz a interface com o banco de dados que deseja manipular atravÃ©s de classes bÃ¡sicas como o MySQL ou o PDO, ainda o mantendo dependente de consultas SQL. Outra Ã© o MÃ³dulo ORM, que mapeia o banco de dados na forma de objetos.

Uma coisa que vocÃª deve pensar ao escolher uma das metodologias Ã© o quÃ£o dependente sua aplicaÃ§Ã£o ficarÃ¡ do Banco de Dados. Como o SQL nÃ£o Ã© implementado da mesma forma entre os diferentes fabricantes de bancos de dados, de um fabricante para outro ou de uma versÃ£o para outra, existem diferenÃ§as sutis entre os bancos. E com a escrita de SQL vocÃª torna aquele trecho do sistema dependente do banco. Um exemplo hipotÃ©tico:

```
<?php

//Oracle
$sql = "SELECT TO_DATE(data, 'YYYY-MM-DD') from tabela;";

$query = $this->db->query($sql);

//MySQL
$sql = "SELECT DATE_FORMAT(data, '%Y-%m-%d') from tabela;";

$query = $this->db->query($sql);

?>
```

Observe que o nome das funÃ§Ãµes podem mudar de um banco para outro, bem como a quantidade e formato dos parÃ¢metros. EntÃ£o, ou vocÃª trabalha com apenas um banco e escreve consultas SQL voltadas para este, ou usa o ORM e trabalha de forma independente do banco de dados. E assim, poderÃ¡ migrar sua aplicaÃ§Ã£o

para qualquer banco de dados que deseje.

Por tais motivos, falarei apenas do ORM. Caso deseje saber mais sobre o Database, por favor, consulte a página: [Kohana Module Database](#)

Primeiramente, vamos criar um Model chamado Contato na pasta application/classes/model.

```
<?php

class Model_Contato extends ORM {

    protected $_belongs_to = array(
        'area' => array(
            'model' => 'area',
            'foreign_key' => 'area_id'
        ),
        'endereco' => array(
            'model' => 'endereco',
            'foreign_key' => 'endereco_id'
        )
    );

    protected $_has_one = array();

    protected $_has_many = array();

    protected $_table_name = 'contatos';
    protected $_primary_key = 'id';

}

?>
```

Este objeto será responsável por mapear a tabela Contatos existente no banco de dados e os relacionamentos que esta tabela tem com as tabelas de Área (aplicado como categorias) e Endereços.

Vamos fazer o Controller contato em application/classes/controller para gerenciar os contatos e efetuar consultas nele :


```

<?php

class Controller_Contato extends Controller_Template {

    public $template = 'index';

    public function action_index()
    {
        //Localiza todos os contatos em ordem decrescente de data de
        //criação
        $contatos = ORM::factory('contato')->order_by('criacao', 'DESC')->
        find_all();

        $area = ORM::factory('area')->order_by('nome')->find_all();

        //Instancia a view
        $view = View::factory('contato_index');

        //Passa os contatos para a view
        $view->contatos = $contatos;

        //Passa as áreas para a view
        $view->areas = $areas;

        //Passa a view para o template
        $this->template->content = $view;
    }

    public function action_salvar()
    {
        //Gera instancia de Model contato
        $contato = ORM::factory('contato');
        $post = Validation::factory($_POST);

        //Seta dados do contato
        $contato->nome = $post->nome;
        $contato->telefone = $post->telefone;
        $contato->celular = $post->celular;
        $contato->email = $post->email;
        $contato->site = $post->site;

        //Busca area pelo id
        $area = ORM::factory('area', $post->area->id);

        //Associa area ao novo contato
        $contato->area = $area;

        //Salva o contato no banco de dados
        if($contato->save())
        {
            $this->request->redirect('contato/index');
        }
    }
}

```

gravados e passa para a view contato_index que será responsável por exibir estes contatos encontrados. Durante a busca, aproveitamos para ordenar todos os contatos de acordo com a data em que foram registrados no sistema. Já as áreas foram ordenadas de acordo com o nome delas, para ficar em ordem alfabética.

Neste mesmo action buscamos também todas as áreas [categorias de contato] para exibir como menu.

Vejamos como ficou a view contato_index que demonstrar os resultados obtidos.

Sendo editado.