Migrations

Quem não evita as pequenas faltas, pouco a pouco cai nas grandes. (T. Kempis)

Migrations é um plugin suportado pela equipe oficial do CakePHP que ajuda você a fazer mudanças no schema do banco de dados utilizando arquivos PHP, que podem ser versionados utilizando um sistema de controle de versão.

Ele permite que você atualize suas tabelas ao longo do tempo. Ao invés de escrever modificações de schema via SQL, este plugin permite que você utilize um conjunto intuitivo de métodos para fazer mudanças no seu banco de dados.

Esse plugin é um wrapper para a biblioteca Phinx(https://phinx.org/).

Uma migração é basicamente um arquivo PHP que descreve as mudanças a serem feitas no banco de dados. Um arquivo de migração pode criar ou excluir tabelas, adicionar ou remover colunas, criar índices e até mesmo inserir dados em seu banco de dados.

Uma migração é basicamente um arquivo PHP que descreve as mudanças a serem feitas no banco de dados. Um arquivo de migração pode criar ou excluir tabelas, adicionar ou remover colunas, criar índices e até mesmo inserir dados em seu banco de dados.

Muito útil para quando desejamos distribuir um plugin ou aplicativo do CakePHP, já incluindo os arquivos de migration e/ou seed quando o usuário apenas executa o comando para criar as tabelas e adicionar os registros.

Migrate

Não se preocupe com falhas, preocupe-se com as chances que você perde quando não tenta.
(Jack Canfield)

Aqui segue um exemplo de migração:

```
<?php
use Migrations\AbstractMigration;

class CreateProducts extends AbstractMigration
{
    /**
     * Change Method.
     *
     * More information on this method is available here:
     * http://docs.phinx.org/en/latest/migrations.html#the-change-method
     * @return void
     */
    public function change()
```

```
    {
        $table = $this->table('products');
        $table->addColumn('name', 'string', [
            'default' => null,
            'limit' => 255,
            'null' => false,
        ]);
        $table->addColumn('description', 'text', [
            'default' => null,
            'null' => false,
        ]);
        $table->addColumn('created', 'datetime', [
            'default' => null,
            'null' => false,
        ]);
        $table->addColumn('modified', 'datetime', [
            'default' => null,
            'null' => false,
        ]);
        $table->create();
    }
}
```

Salvar em
config/Migrations/20190519085224_Products.php

Veja o formato: YYYYMMDDHHMMSS_Products.php

Também podemos criar a migration usando o bake:

bin/cake bake migration CreateProducts name:string description:text created modified

Você pode criar um arquivo de migração vazio caso deseje ter um controle total do que precisa ser executado. Para isto, apenas omita a definição das colunas:

bin/cake migrations create MyCustomMigration

Criar a tabela ou as tabelas do(s) Migration(s)

bin/cake migrations migrate

Limpar todo o banco, excluindo todas as tabelas

bin/cake migrations rollback

Nomes de campos e tipos

fieldName:fieldType[length]:indexType:indexName

Por exemplo, veja formas válidas de especificar um campo de e-mail:

- email:string:unique
- email:string:unique:EMAIL_INDEX
- email:string[120]:unique:EMAIL_INDEX

O parâmetro length para o fieldType é opcional e deve sempre ser escrito entre colchetes
Os campos created e modified serão automaticamente definidos como datetime.
Os tipos de campos são genericamente disponibilizados pela biblioteca Phinx. Eles podem ser:

- string
- text
- integer
- biginteger
- float
- decimal
- datetime
- timestamp
- time
- date
- binary
- boolean
- uuid

Há algumas heurísticas para a escolha de tipos de campos que não são especificados ou são definidos com valor inválido. O tipo de campo padrão é string;

- id: integer
- created, modified, updated: datetime

Adicionando colunas a uma tabela existente

Se o nome da migração na linha de comando estiver na forma "AddXXXToYYY" e for seguido por uma lista de nomes de colunas e tipos, então o arquivo de migração com o código para criar as colunas será gerado:
bin/cake bake migration AddPriceToProducts price:decimal

A linha de comando acima irá gerar um arquivo com o seguinte conteúdo:

```
<?php
use Migrations\AbstractMigration;

class AddPriceToProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->addColumn('price', 'decimal')
              ->update();
    }
}
```

Especificando o tamanho do campo

Se você precisar especificar o tamanho do campo, você pode fazer isto entre colchetes logo após o tipo do campo, ex.:

bin/cake bake migration AddFullDescriptionToProducts full_description:string[60]

Executar o comando acima irá gerar:

```
<?php
use Migrations\AbstractMigration;

class AddFullDescriptionToProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->addColumn('full_description', 'string', [
            'default' => null,
            'limit' => 60,
            'null' => false,
        ])
        ->update();
    }
}
```

Se o tamanho não for especificado, os seguintes padrões serão utilizados:

- string: 255
- integer: 11
- biginteger: 20

Removendo uma coluna de uma tabela

Da mesma forma, você pode gerar uma migração para remover uma coluna utilizando a linha de comando, se o nome da migração estiver na forma "RemoveXXXFromYYY":

bin/cake bake migration RemovePriceFromProducts price

Cria o arquivo:

```
<?php
use Migrations\AbstractMigration;

class RemovePriceFromProducts extends AbstractMigration
{
    public function change()
    {
        $table = $this->table('products');
        $table->removeColumn('price');
    }
}
```

Gerando migrações a partir de uma base de dados existente

Se você está trabalhando com um banco de dados pré-existente e quer começar a usar migrações, ou para versionar o schema inicial da base de dados da sua aplicação, você pode executar o comando migration_snapshot:

bin/cake bake migration_snapshot Financas

Ele gerará um arquivo de migrations com todas as tabelas do banco, mas somente as estruturas, sem os registros:

YYYYMMDDHHMMSS_Financas.php

Migrações também podem ser executadas para plugins. Simplesmente utilize a opção ``--plugin`` ou ``-p``

bin/cake migrations migrate -p MyAwesomePlugin

Lista das migrações

bin/cake migrations status

Seed - inserindo registros nas tabelas

Diante de uma dificuldade substitua o eu não consigo pelo vou tentar outra vez.

bin/cake bake seed Posts

Ele gera um esqueleto de classe.

Editar config/Seeds/Posts e adicionar os registros diretamente como abaixo ou usando o Faker

```
<?php
use Phinx\Seed\AbstractSeed;

class PostsSeed extends AbstractSeed
{
    public function run()
    {
        $data = [
            [
                'body'    => 'foo',
                'created' => date('Y-m-d H:i:s'),
            ],
            [
                'body'    => 'bar',
                'created' => date('Y-m-d H:i:s'),
            ]
        ];
```

```
        $posts = $this->table('posts');
        $posts->insert($data)
            ->save();
    }
}
```

Integrando seed com a biblioeca faker

Instalar a Faker

composer require fzaninotto/faker

Detalhes de uso:
https://github.com/fzaninotto/Faker

Criar um novo seed

bin/cake bake seed Despesas

Então usar na classe gerada

```
<?php

use Phinx\Seed\AbstractSeed;

class DespesasSeed extends AbstractSeed
{
    public function run()
    {
        $faker = Faker\Factory::create('pt_BR');
        $data = [];
        for ($i = 0; $i < 20; $i++) {
            $data[] = [
                'descricao'     => $faker->userName,
                'valor'         => $faker->numberBetween($min = 0, $max = 9000),
                'mes'           => $faker->regexify('0[1-9]\/2019|1[1-2]\/2019'),
                'receita_id'    => $faker->numberBetween($min = 1, $max = 2),
                'created'       => date('Y-m-d H:i:s'),
            ];
        }

        $table = $this->table('despesas');
        $table->insert($data)->save();
    }
}
```

Adicionar os registros do Seed no banco

bin/cake migrations seed

Métodos da Faker

Como tive dificuldade de encontrar alguns exemplos de uso da Faker, então seguem alguns exenplos que colecionei.

```
$faker = Faker\Factory::create('pt_BR');

$cpf = $faker->numberBetween($min = 10000000000, $max = 99999999999);
$nome = addslashes($faker->name);
$credito_liberado = $faker->regexify('[sn]');
$nascimento = $faker->date;
$email = $faker->email;
$user_id = $faker->numberBetween($min = 1, $max = 4);
$quantidade = $faker->randomNumber($nbDigits = NULL, $strict = false);
$preco_venda = $faker->numberBetween($min = 20, $max = 1200);

randomNumber($nbDigits = NULL, $strict = false) // 79907610
randomFloat($nbMaxDecimals = NULL, $min = 0, $max = NULL) // 48.8932
numberBetween($min = 1000, $max = 9000) // 8567

$faker->regexify('[sn]'); // s ou n
$faker->randomElement($array = array ('s','n'));
$faker->randomLetter;
$faker->regexify('[A-Z]+[a-z]{2,5}'); // 2 a 5 letras
$faker->regexify('[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}'); // sm0@y8k96a.ej
$faker->randomElement($array = array ('a','b','c')); // 'b'
print $faker->sentence($nbWords = 3, $variableNbWords = true);
$faker->sentence($nbWords = 6, $variableNbWords = true);
$faker->address; // rua, número e cep
$faker->text; // Para grandes quantidades de texto
$faker->sentence($nbWords = 6, $variableNbWords = true);
$faker->text($maxNbChars = 200);
$faker->title($gender = null|'male'|'female');     // 'Ms.'
$faker->name($gender = null|'male'|'female');      // 'Dr. Zane Stroman'
$faker->cityPrefix;
$faker->state;
$faker->stateAbbr;
$faker->buildingNumber;
$faker->city;
$faker->streetName;
$faker->streetAddress;
$faker->postcode;
$faker->country;
$faker->PhoneNumber;
$faker->company;
```

$faker->date($format = 'Y-m-d', $max = 'now');
$faker->time($format = 'H:i:s', $max = 'now');
$faker->freeEmail;
$faker->password;
$faker->domainName;
$faker->url;
$faker->ipv4;
$faker->macAddress;
$faker->creditCardType;
$faker->creditCardNumber;
$faker->creditCardExpirationDateString;
$faker->hexcolor;
$faker->colorName;
$faker->fileExtension;
$faker->mimeType;
$faker->locale;
$faker->countryCode;
$faker->randomHtml(2,3);

A partir da versão 1.5.5 do plugin, você pode usar a shell de migrations para popular seu banco de dados. Essa função é oferecida graças ao recurso de seed da biblioteca Phinx. Por padrão, arquivos seed ficarão no diretório config/Seeds de sua aplicação. Por favor, tenha certeza de seguir as instruções do Phinx para construir seus arquivos de seed.

Podemos especificar um plugin

bin/cake bake seed Articles --plugin PluginName

As opções --data, --limit e --fields foram adicionadas para exportar dados da sua base de dados.
A partir da versão 16.4, o comando bake seed permite que você crie um arquivo de seed com dados exportados da sua base de dados com o uso da flag --data:

Exportando a tabela Articles juntamente com seus dados:

bin/cake bake seed --data Articles

Por padrão, esse comando exportará todas as linhas encontradas na sua tabela. Você pode limitar o número de linhas a exportar usando a opção --limit:

Exportar apenas os 10 primeiros registros encontradas

bin/cake bake seed --data --limit 10 Articles

Para popular seu banco de dados, você pode usar o subcomando seed:

bin/cake migrations seed

Para plugins

bin/cake migrations seed -p MeuPlugin

Você pode especificar apenas um seeder para rodar usando a opção --seed:

bin/cake migrations seed --seed ArticlesSeed

Limpar o cache

Se você usa o plugin ao fazer o deploy de sua aplicação, garanta que o cache ORM seja limpo para renovar os metadados das colunas de suas tabelas.

bin/cake orm_cache clear

Migrando somente uma migration

bin/cake migrations migrate -p CakeDC/Users
bin/cake migrations migrate -p Acl

Resumindo

Você nunca sabe que resultados virão da sua ação. Mas se você não fizer nada, não existirão resultados. (Mahatma Gandhi)

Gerando um backup do banco juntamente com os dados em Migrations:

Exportar banco de dados existente para uma Migration (apenas a estrutura das tabelas)

bin/cake bake migration_snapshot NomeBanco

Gera a migration para todo o banco em config/Migrations/20190520132718_NomeBanco.php

Exportar uma tabela juntamente com seus registros

bin/cake bake seed --data users

Gera o arquivo config/Seeds/UsersSeed.php

Limpar todo o banco com rollback

Para remover todas as tabelas do banco execute:

bin/cake migrations rollback

Restaurar todas as tabelas com seus dados

Trazer de volta a estrutura das tabelas e os registros, após ter feito a migration e seeds.

bin/cake migrations migrate
bin/cake migrations seed

Vantagem Extra

Obs.: os arquivos de migrations tem mais uma vantagem. São independentes de SGBD. Funcionam em todos os SGBD suportados pelo CakePHP.

Atualizar aplicativo para a versão atual (3.7.7)

cd aplicativo
composer require --update-with-dependencies "cakephp/cakephp:3.7.*"

Mudando de SGBD

Imagina só, você criou seu aplicativo com o MySQL então precisa mudar para o PostgreSQL. Basta exportar as migrations e as seeds no MySQL, depois criar o banco no PostgreSQL, ajustar a configuração e importar as migrations e seeds. Realmente muito prático!

Referências

https://book.cakephp.org/3.0/pt/appendices/3-0-migration-guide.html
https://book.cakephp.org/3.0/en/upgrade-tool.html