# Tinker

tinker是一個REPL (read-eval-print-loop)，REPL是指交互式命令行界面，它可以讓你輸入一段程式碼去執行，並把執行結果直接打印到終端機上面去。

這樣就不用每次都使用route才能修改資料，透過tinker可以很快速的修改資料庫的資料。

新增一個今天要用的專案

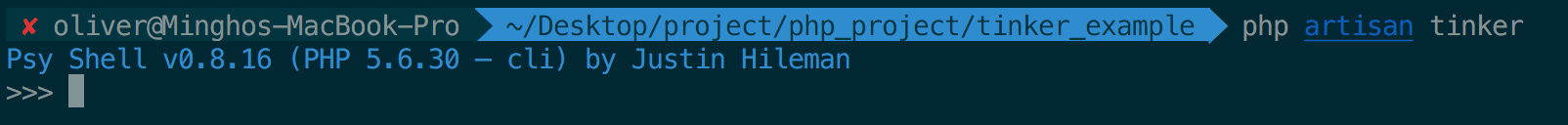
composer create-project --prefer-dist laravel/laravel tinker\_example

新增migrate

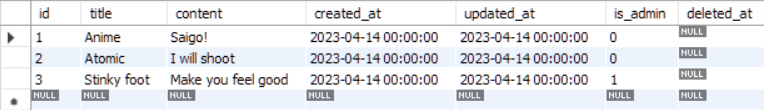
php artisan migrate

在終端機中輸入下面的指令

php artisan tinker

開啟成功後就可以在這邊下指令了  


# Insert Data



posts中有三筆資料，現在使用tinker直接新增一筆資料



> $post = App\Models\Post::create(['title'=>'PHP post from tinker', 'content'=>'PHP content from tinker']);

透過Post模型建立一筆新的資料

= App\Models\Post {#4444

title: "PHP post from tinker",

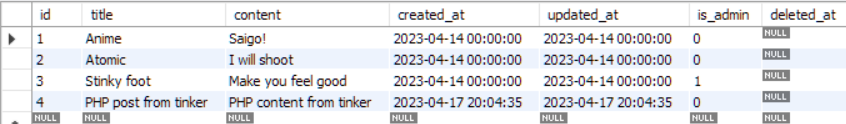
content: "PHP content from tinker",

updated\_at: "2023-04-17 20:04:35",

created\_at: "2023-04-17 20:04:35",

id: 4,

}



# New Object

> $post = new App\Models\Post;

= App\Models\Post {#4484}

新建立一個Post模型的物件

> $post;

= App\Models\Post {#4484}

呼叫$post會回傳他是屬於Post的物件

* 建立物件的內容

> $post->title = "The title of the Tinker";

= "The title of the Tinker"

建立title

> $post->content = "This is the content from the Tinker";

= "This is the content from the Tinker"

建立content

> $post;

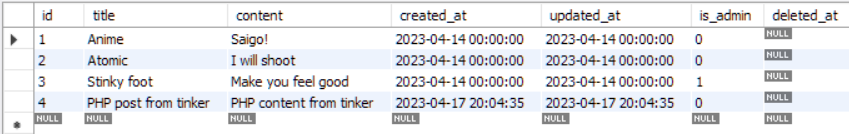
= App\Models\Post {#4484

title: "The title of the Tinker",

content: "This is the content from the Tinker",

}

透過呼叫$post可以看到目前物件儲存title和content的內容



但是資料庫重整後，沒有看到posts表格中有新增上面的title和content的資料

> $post->save();

= true

將目前的post資料儲存到資料庫中

> $post;

= App\Models\Post {#4484

title: "The title of the Tinker",

content: "This is the content from the Tinker",

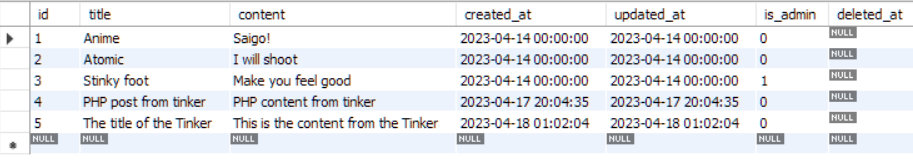
updated\_at: "2023-04-18 01:02:04",

created\_at: "2023-04-18 01:02:04",

id: 5,

}

此時的post會成功儲存到資料庫中



# 查找資料

> $post = App\Models\Post::find(5);

= App\Models\Post {#4714

id: 5,

title: "The title of the Tinker",

content: "This is the content from the Tinker",

created\_at: "2023-04-18 01:02:04",

updated\_at: "2023-04-18 01:02:04",

is\_admin: 0,

deleted\_at: null,

}

查找posts中的第五筆資料

> $post = App\Models\Post::where('id', 5)->first();

= App\Models\Post {#4716

id: 5,

title: "The title of the Tinker",

content: "This is the content from the Tinker",

created\_at: "2023-04-18 01:02:04",

updated\_at: "2023-04-18 01:02:04",

is\_admin: 0,

deleted\_at: null,

}

使用條件式查找posts中的第五筆資料

> $post = App\Models\Post::whereId(5)->first();

= App\Models\Post {#4718

id: 5,

title: "The title of the Tinker",

content: "This is the content from the Tinker",

created\_at: "2023-04-18 01:02:04",

updated\_at: "2023-04-18 01:02:04",

is\_admin: 0,

deleted\_at: null,

}

一樣用條件式查找id=5的post資料

# 更新資料

> $post->title = "Updated record with id 5";

= "Updated record with id 5"

更新title資料

> $post->content = "Updated record content where id = 5";

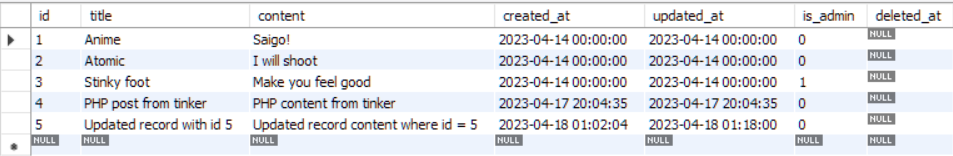
= "Updated record content where id = 5"

更新content資料

> $post->save();

= true

將資料送進資料庫中

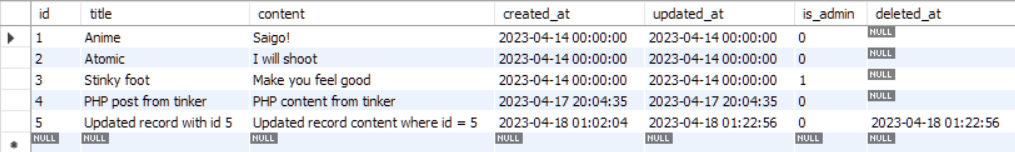


可以看到id=5的title和content已經被更新

# 刪除資料

> $post->delete();

= true



可以看到id=5的資料並沒有被刪除，因為先前有在post表格中增加soft\_deletes的功能，因此該筆資料只是丟進垃圾桶中，並沒有完全刪除。

> $post = App\Models\Post::onlyTrashed();

= Illuminate\Database\Eloquent\Builder {#4086}

查找垃圾桶中有哪幾筆資料 => deleted\_at中有資料的欄位為在垃圾筒的資料

> $post;

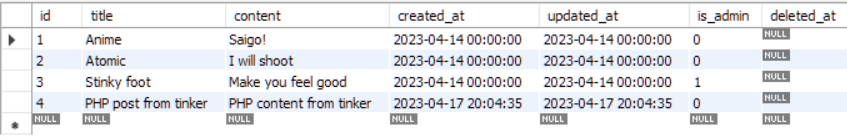
= Illuminate\Database\Eloquent\Builder {#4086}

$post儲存了垃圾筒中的資料

> $post->forceDelete();

= 1

強制刪除垃圾桶中的資料



可以看到id=5的資料已經被刪除了