




## **LARAVEL FRAMEWORK**

### **BÀI 6: DATABASE: ELOQUENT**

- ⊙ Tổng quan Eloquent
- ⊙ Sử dụng Eloquent thao tác với database
- ⊙ Hoàn thành CRUD



## Phần I: Tổng quan Eloquent

 Các thao tác giữa ứng dụng và database

 Giới thiệu Eloquent

 Tạo model với Eloquent

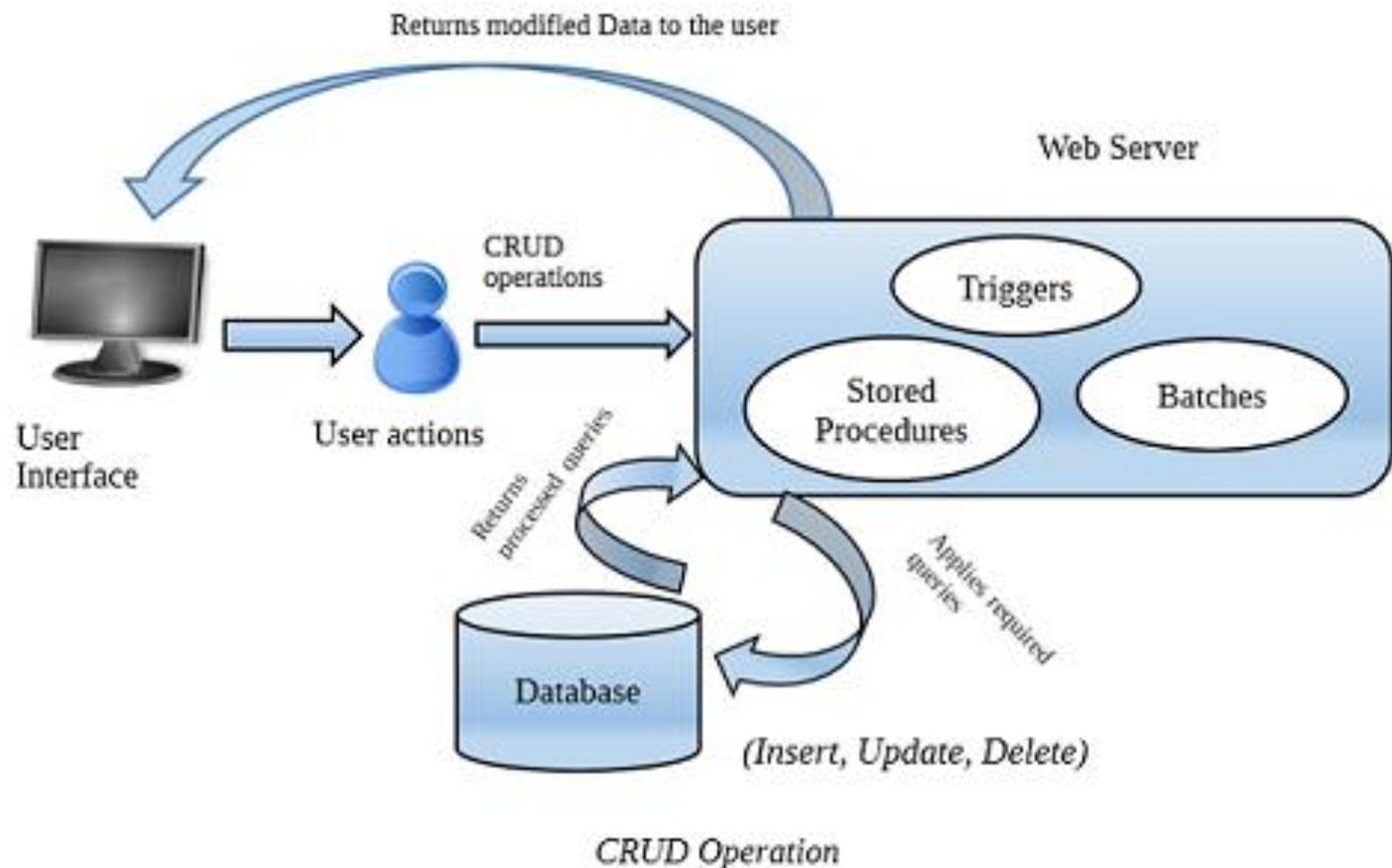
## Phần II: Các thao tác CRUD

 Inserting & Updating Models

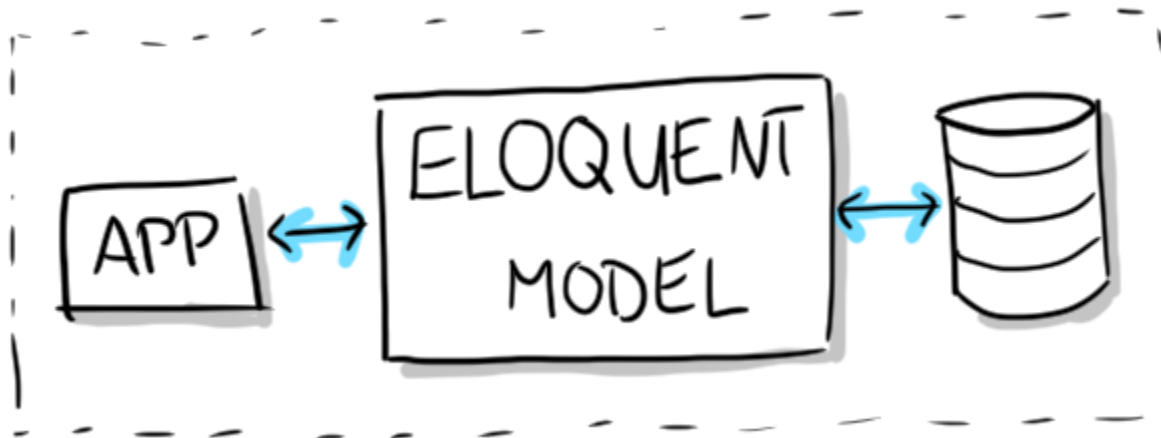
 Deleting Models



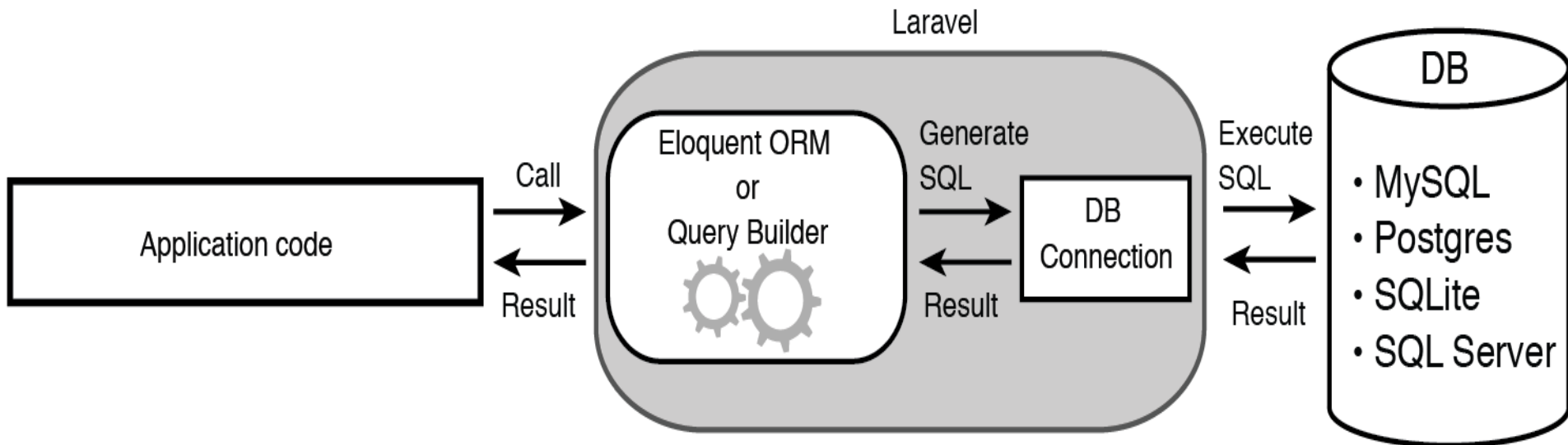
- ❑ Database và ứng dụng luôn có mối quan hệ gắn bó



- ❑ Eloquent ORM đi kèm với Laravel cung cấp các API ActiveRecord đơn giản và tiện lợi trong giao tiếp với database. Mỗi table sẽ có một "Model" tương ứng để tương tác với table đó. Model cho phép bạn query dữ liệu trong table, cũng như chèn thêm các dữ liệu mới thông qua các phương thức trong Eloquent



- ❑ Eloquent models kế thừa từ Illuminate\Database\Eloquent\Model trong mô hình kiến trúc MCV
- ❑ Eloquent ORM là một Object Relational Mapper do Laravel phát triển



## ❑ Tạo model

- ❖ Sử dụng Cmd và đến thư mục gốc của project
- ❖ php artisan make:model Flight : tạo model tên Flight

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    //
}
```

## ❑ Tạo model

- ❖ Table Name: Khi tạo modle nếu không chỉ định table nào thì Laravel lấy tên modle thêm 's' vào cuối để làm tên table tương ứng
- ❖ Tạo modle tên Flight và không chỉ định table thì mặc định model sẽ liên kết với table Flight**s**
- ❖ Có thể chỉ định table và PrimaryKey khi tạo model

```
protected $primaryKey = 'id';  
protected $table = 'categories';
```



## ❑ Tạo model

- ❖ Record Timestamps: Mặc định thêm 2 field vào table là: create\_at và update\_at.
- ❖ Nếu không muốn những field này tự động được quản lý bởi Eloquent, thiết lập thuộc tính \$timestamps thành false

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * Indicates if the model should be timestamped.
     *
     * @var bool
     */
    public $timestamps = false;
}
```

## ❑ Tạo model

- ❖ Database Connection: Eloquent model sẽ sử dụng kết nối database mặc định được cấu hình khi tạo Migration. Muốn sử dụng một kết nối khác cho model, sử dụng thuộc tính \$connection

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The connection name for the model.
     *
     * @var string
     */
    protected $connection = 'connection-name';
}
```

## ❑ Tạo model

- ❖ Retrieving Models (truy vấn dữ liệu): Sau khi tạo model và chỉ định table tương ứng thì việc truy vấn dữ liệu từ database sẵn sàng

```
<?php

use App\Flight;

$flights = App\Flight::all();

foreach ($flights as $flight) {
    echo $flight->name;
}
```

## ❑ Tạo model

- ❖ Các ràng buộc: Hàm all sẽ trả về tất cả các kết quả trong table của model. Vì mỗi Eloquent model phục vụ như một query builder, nên có thể tạo ràng buộc cho các query, và cuối cùng sử dụng hàm get để lấy kết quả:

```
$flights = App\Flight::where('active', 1)  
    ->orderBy('name', 'desc')  
    ->take(10)  
    ->get();
```

## ❑ Tạo model

- ❖ Collections: Eloquent có các phương thức như all, get trả về tập nhiều kết quả thì cũng cung cấp nhiều method để tương tác với các kết quả đó

```
$flights = $flights->reject(function ($flight) {  
    return $flight->cancelled;  
});
```

- ❖ Thực hiện lặp collection này như một array

```
foreach ($flights as $flight) {  
    echo $flight->name;  
}
```

## ❑ Tạo model

- ❖ Chunking Results: xử lý khối lượng dữ liệu lớn bằng "chunk"

```
Flight::chunk(200, function ($flights) {  
    foreach ($flights as $flight) {  
        //  
    }  
});
```

- ❖ Using Cursors: xử lý khối lượng dữ liệu lớn bằng "Cursors"

```
foreach (Flight::where('foo', 'bar')->cursor() as $flight) {  
    //  
}
```

## ❑ Tạo model

- ❖ Retrieving Single Models / Aggregates: lấy một kết quả sử dụng hàm `find` và `first`. Thay vì trả về một collection model, những hàm này trả về một model instance

```
// Retrieve a model by its primary key...  
$flight = App\Flight::find(1);  
  
// Retrieve the first model matching the query constraints...  
$flight = App\Flight::where('active', 1)->first();
```



DEMO

- Demo tạo modle, kết nối và lấy dữ liệu từ table







# LARAVEL FRAMEWORK

## BÀI 2 (PHẦN 2)

- ❑ Inserting models: Để thêm dữ liệu mới vào database, tạo một model instance mới, thiết lập các attributes vào model rồi gọi hàm save

```
<?php

namespace App\Http\Controllers;

use App\Flight;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class FlightController extends Controller
{
    /**
     * Create a new flight instance.
     *
     * @param Request $request
     * @return Response
     */
}
```

```
public function store(Request $request)
{
    // Validate the request...

    $flight = new Flight;

    $flight->name = $request->name;

    $flight->save();
}
```

## ❑ Updates:

- ❖ Hàm save cũng được dùng để cập nhật model đã tồn tại sẵn trong database
- ❖ Để update, cần lấy model instance ra trước, thay đổi các attribute bạn muốn, rồi gọi hàm save

```
$flight = App\Flight::find(1);  
  
$flight->name = 'New Flight Name';  
  
$flight->save();
```

## ❑ Updates:

❖ Cập nhật dựa trên điều kiện

```
Route::get('/update', function() {  
    $category = App\Category::find(16);  
    $category->name = 'HEAVY METAL';  
    $category->save();  
  
    $data = $category->all(array('name', 'id'));  
  
    foreach ($data as $list) {  
        echo $list->id . ' ' . $list->name . ' ';  
    }  
});
```

```
App\Flight::where('active', 1)  
    ->where('destination', 'San Diego')  
    ->update(['delayed' => 1]);
```

- ❑ Để thực hiện xóa model, gọi hàm delete trên model instance:

```
Route::get('/delete', function() {  
    $category = App\Category::find(5);  
    $category->delete();  
  
    $data = $category->all(array('name','id'));  
  
    foreach ($data as $list) {  
        echo $list->id . ' ' . $list->name . '  
    }  
});
```

- ❑ Xoá model tồn tại bằng một key: nếu đã biết primary key của model, có thể xoá model mà không cần lấy nó ra, chỉ cần gọi hàm destroy

```
App\Flight::destroy(1);
```

```
App\Flight::destroy([1, 2, 3]);
```

```
App\Flight::destroy(1, 2, 3);
```

- ❑ Xoá model bằng query

```
$deletedRows = App\Flight::where('active', 0)->delete();
```

## ❑ Soft Deleting

- ❖ Thay vì thực sự xóa các record khỏi database, Eloquent cũng cung cấp kiểu "soft delete" (xóa mềm) mode
- ❖ Khi model được soft deleted, chúng chưa thực sự bị xóa khỏi database
- ❖ Một trường là `deleted_at` sẽ được thiết lập trong model và chèn vào trong database. Nếu model có giá trị `deleted_at` khác NULL, tức là model đã bị soft deleted
- ❖ Để kích hoạt xóa mềm cho một model, sử dụng trait `Illuminate\Database\Eloquent\SoftDeletes` trên model và thêm vào column `deleted_at` vào trong thuộc tính `$dates` của model

- ❑ Thêm column `deleted_at` vào trong table

```
Schema::table('flights', function ($table) {  
    $table->softDeletes();  
});
```

- ❑ Lúc này, khi gọi hàm `delete` trên model, column `deleted_at` sẽ được set vào current date và time. Và, khi thực hiện query một model có sử dụng `soft delete`, thì model đó sẽ tự động bị loại khỏi tất cả các kết quả query.

```
if ($flight->trashed()) {  
    //  
}
```



## ❑ Soft Deleting

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

class Flight extends Model
{
    use SoftDeletes;

    /**
     * The attributes that should be mutated to dates.
     *
     * @var array
     */
    protected $dates = ['deleted_at'];
}
```

## ❑ Querying Soft Deleted Models

- ❖ Including Soft Deleted Models: có thể thêm các soft delete model vào kết quả sử dụng hàm withTrashed

```
$flights = App\Flight::withTrashed()  
            ->where('account_id', 1)  
            ->get();
```

- ❖ Chỉ lấy các soft delete model vào kết quả dùng onlyTrashed

```
$flights = App\Flight::onlyTrashed()  
            ->where('airline_id', 1)  
            ->get();
```

## ❑ Querying Soft Deleted Models

- ❖ Restoring Soft Deleted Models: Để khôi phục lại một soft delete model về trạng thái active, sử dụng hàm restore:

```
$flight->restore();
```

- ❖ Dùng hàm restore trên một query để nhanh chóng khôi phục nhiều model:

```
App\Flight::withTrashed()  
    ->where('airline_id', 1)  
    ->restore();
```

- ❖ Để xoá vĩnh viễn một soft delete model, hãy sử dụng hàm forceDelete:

```
// Force deleting a single model instance...  
$flight->forceDelete();  
  
// Force deleting all related models...  
$flight->history()->forceDelete();
```

- ❑ Eloquent model cung cấp các events cho phép có thể hook vào nhiều điểm của model lifecycle
- ❑ retrieved, creating, created, updating, updated, saving, saved, deleting, deleted, restoring, restored
- ❑ creating và created thực thi khi model được lưu lần đầu
- ❑ Nếu model đã tồn tại trong database và hàm save được gọi thì hai event updating / updated sẽ thực thi

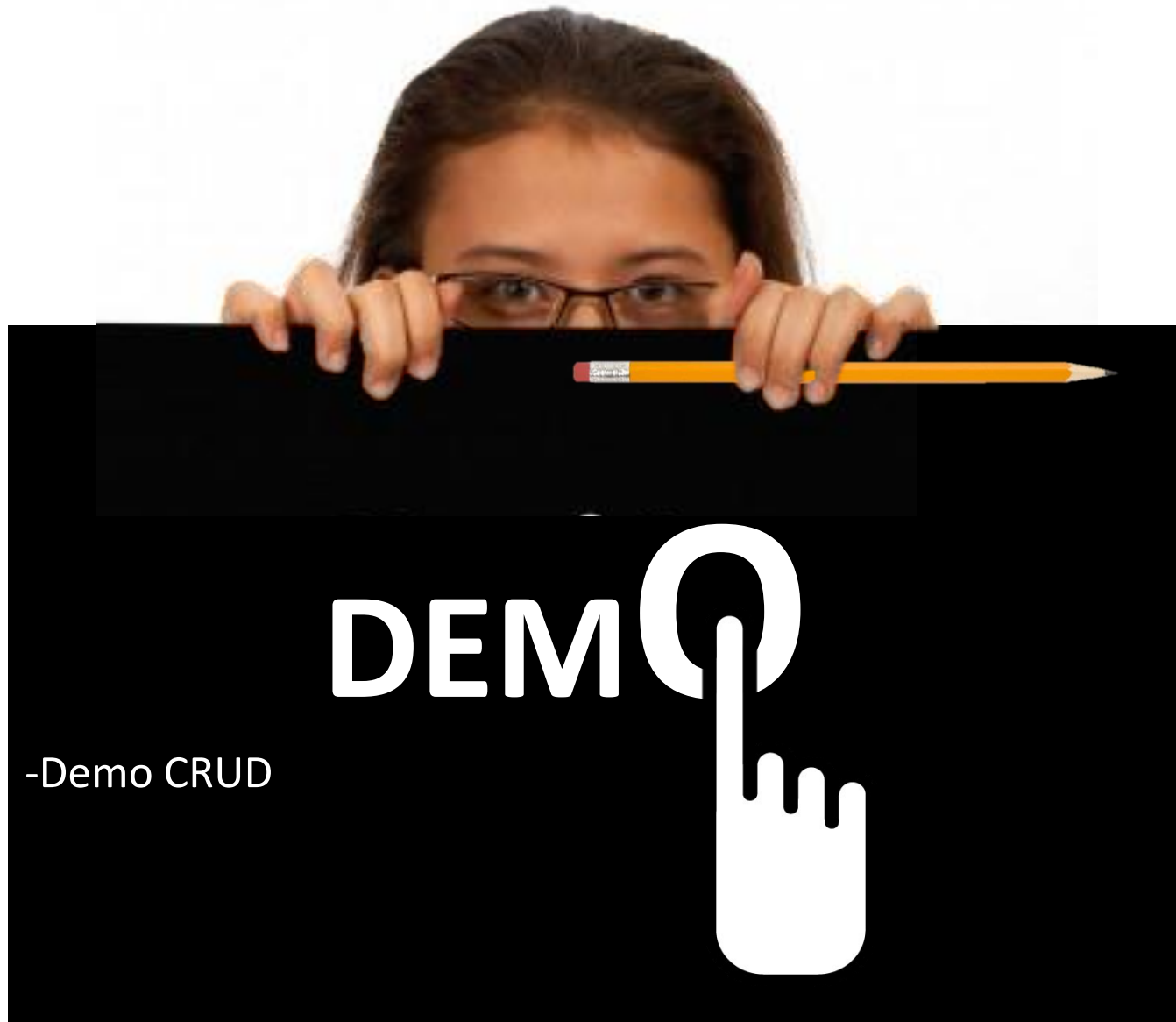
```
<?php

namespace App\Providers;


use App\User;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        User::creating(function ($user) {
            if ( ! $user->isValid()) {
                return false;
            }
        });
    }
}
```

```
/**
 * Register the service provider.
 *
 * @return void
 */
public function register()
{
    //
}
```



## Phần I: Tổng quan Eloquent

 Các thao tác giữa ứng dụng và database

 Giới thiệu Eloquent

 Tạo model với Eloquent

## Phần II: Các thao tác CRUD

 Inserting & Updating Models

 Deleting Models





**Cảm ơn**