Department of Information Technology
IT 350 - Web Technologies
Fall - 2021

Databases and Eloquent II

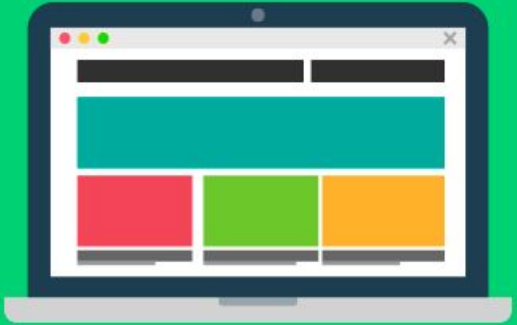
Lecture Seven

Rebin M. Ahmed
rebin.mohammed@tiu.edu.iq

Previous Lecture

- Introduction to Databases
- Databases Configuration
- Raw SQL
- Query Builder
- Eloquent

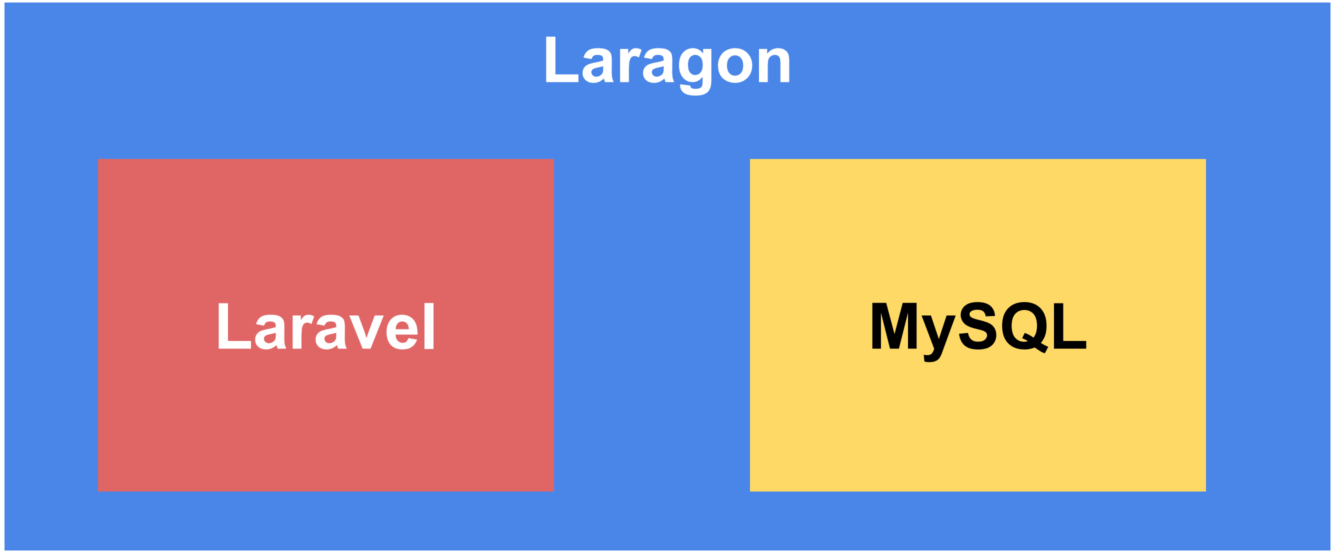
Content



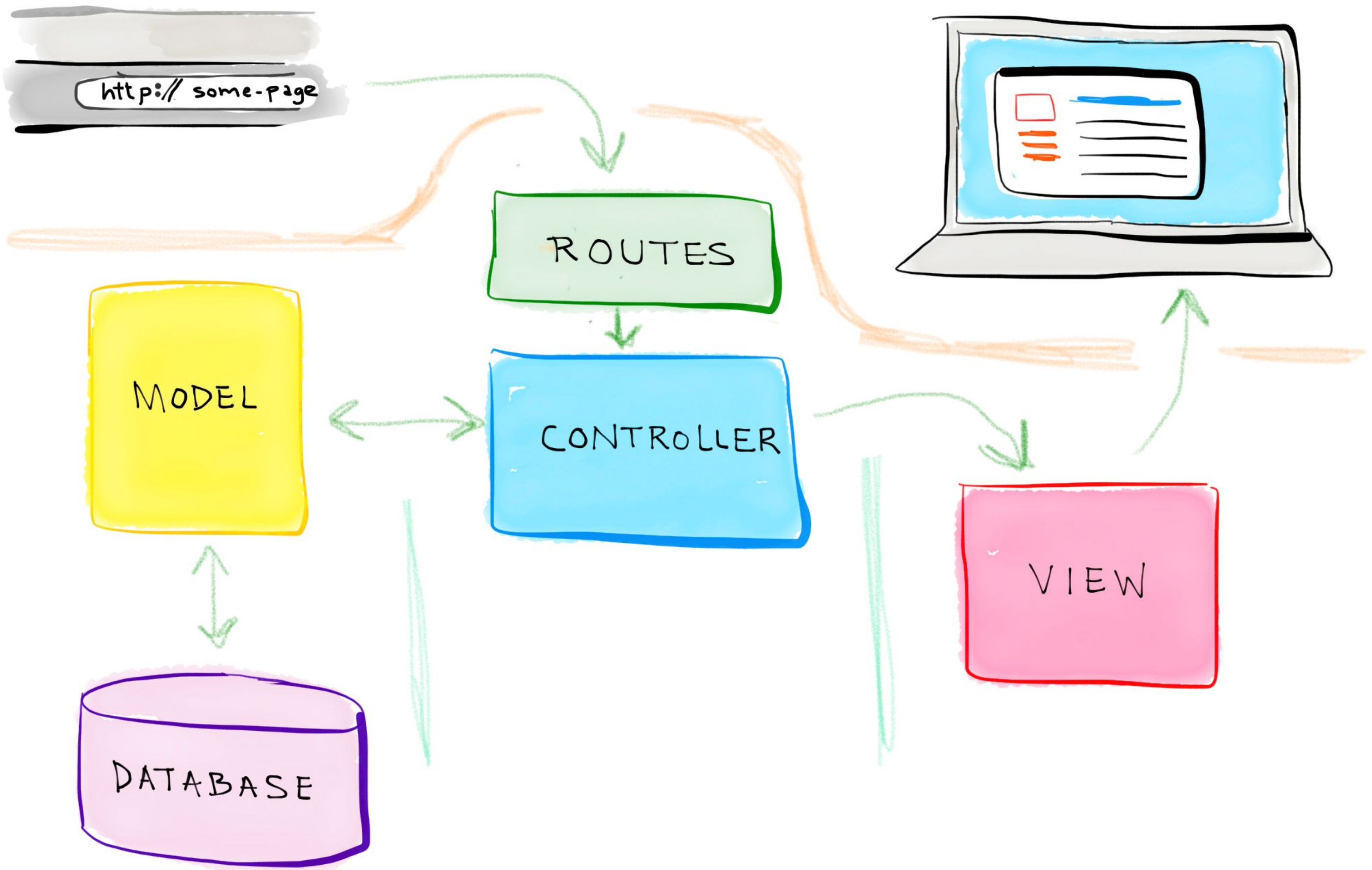
FRONTEND

BACKEND

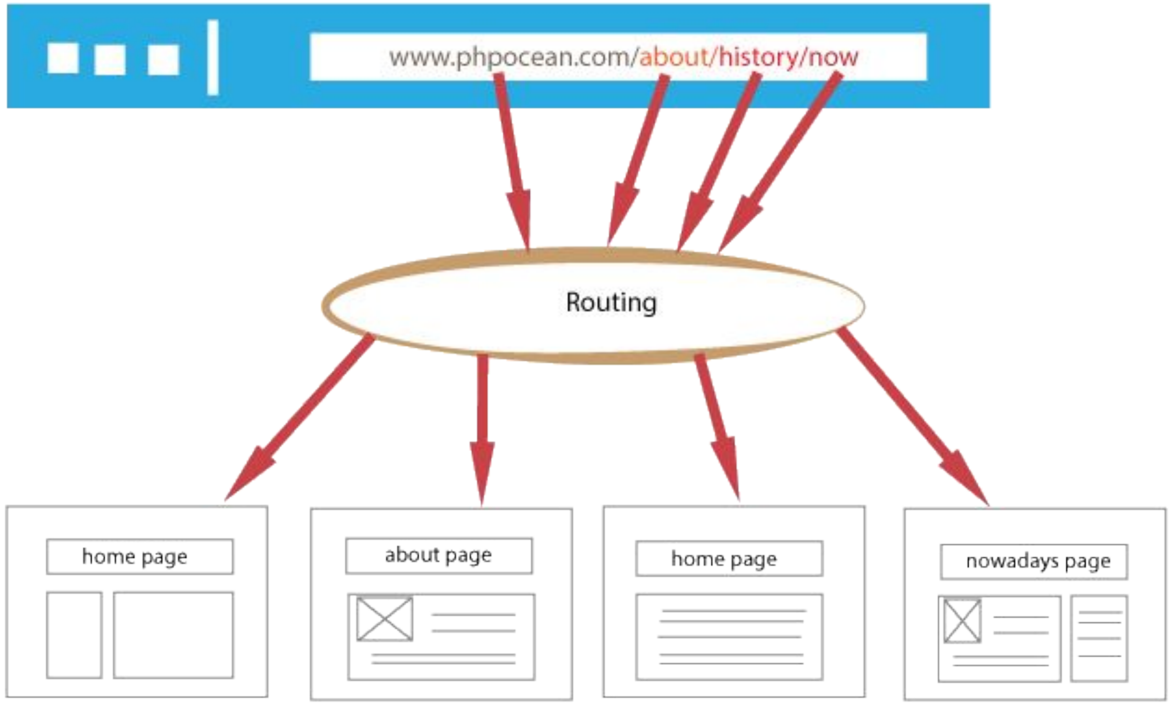
Bigger Picture



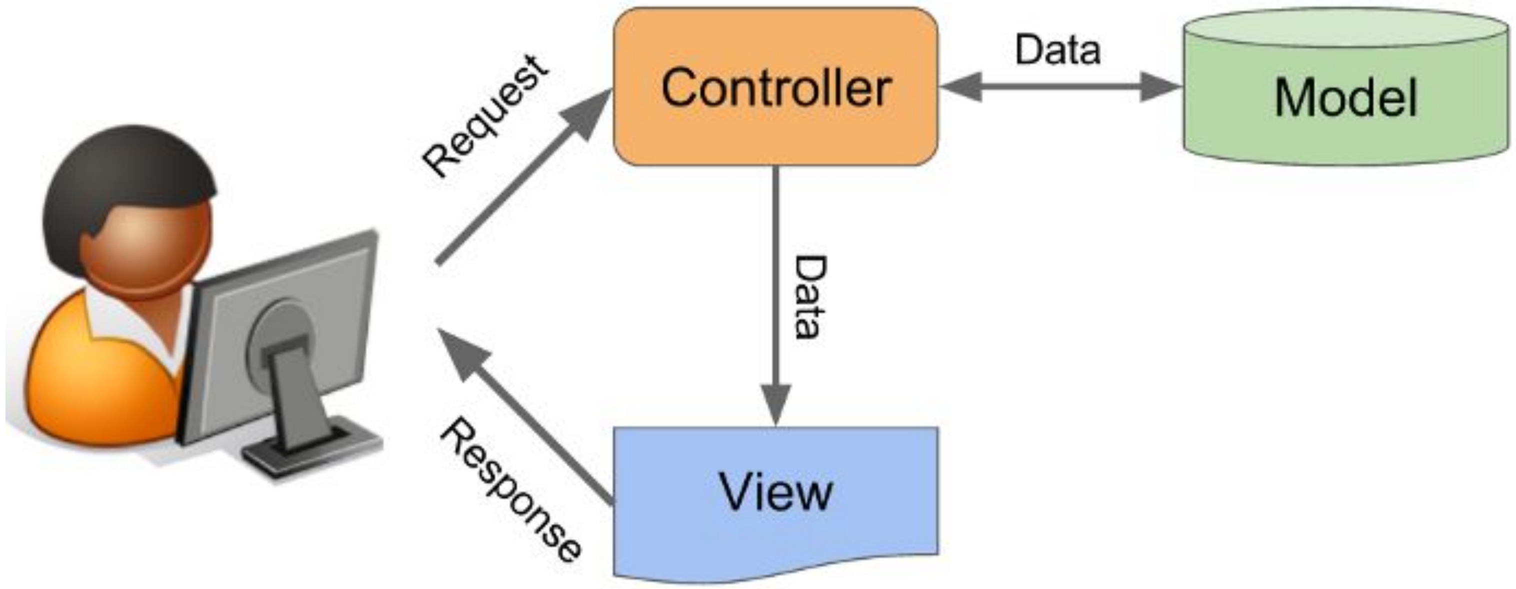
Larave



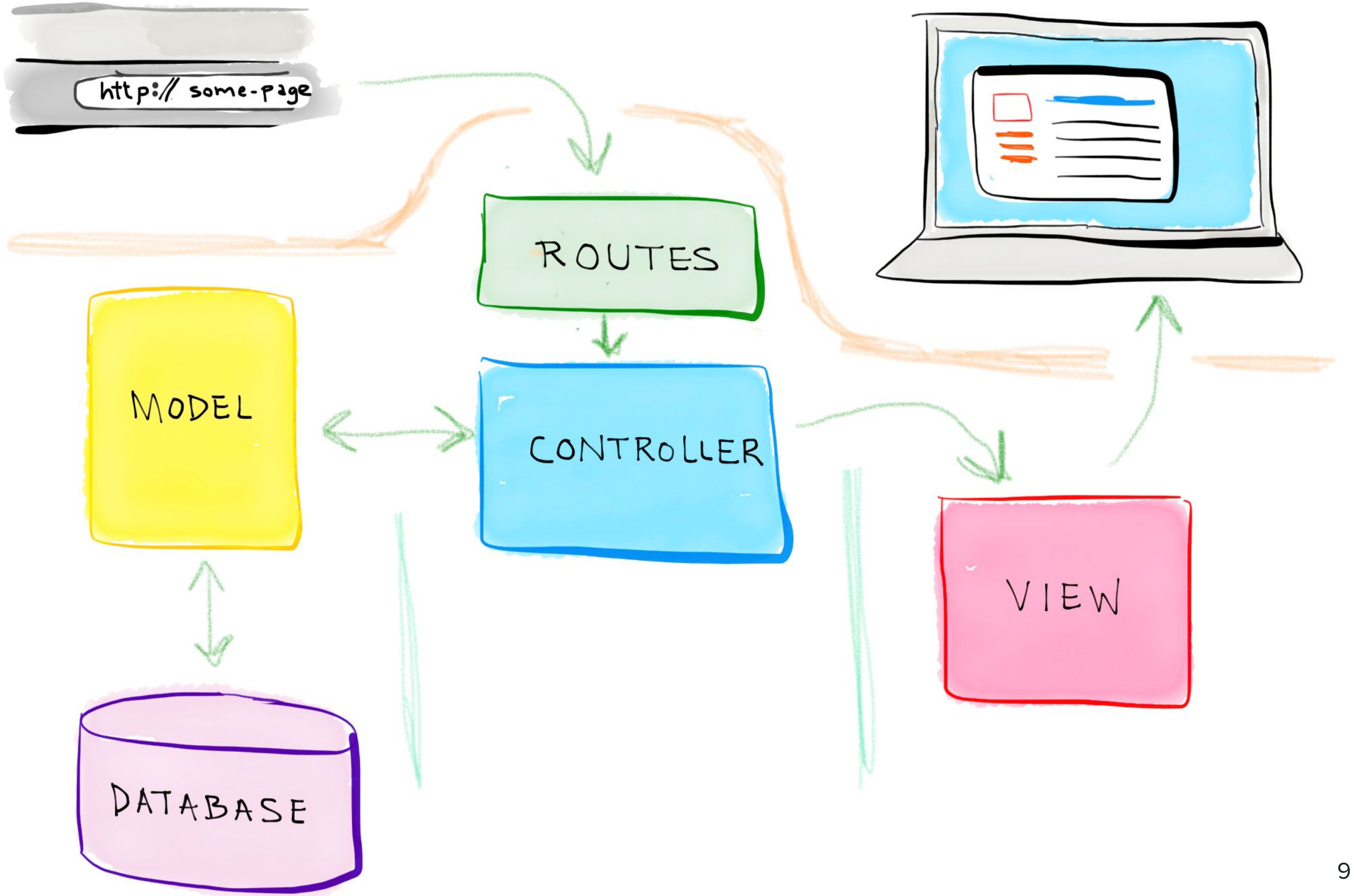
Route


www.phpocean.com/about/history/now
Routing
home page
about page
home page
nowadays page

Controller



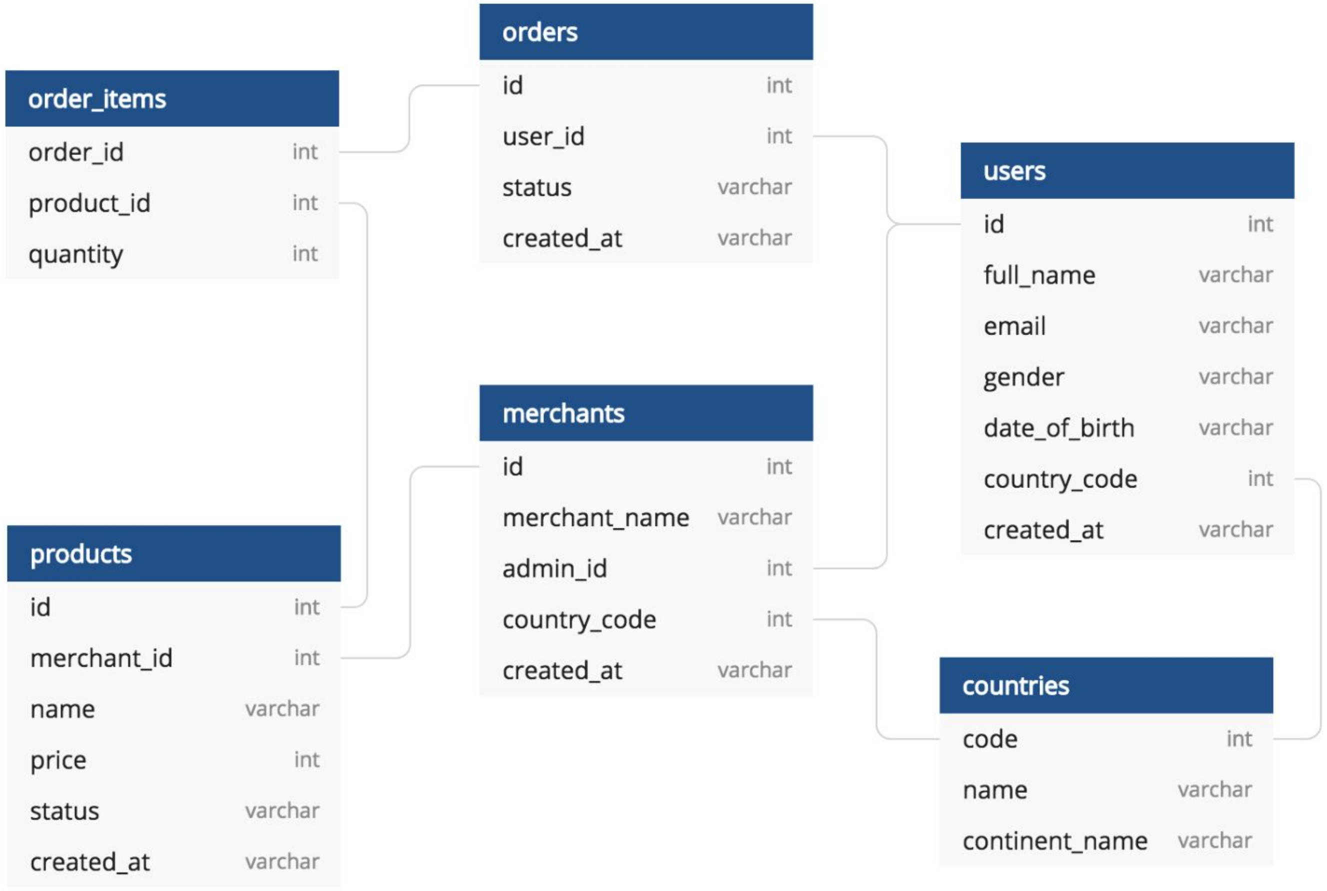
Model



Eloquent

The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database. Each database **table** has a corresponding "**Model**" which is used to interact with that table.

Models allow you to query for data in your tables, as well as insert new records into the table.



order_items	
order_id	int
product_id	int
quantity	int

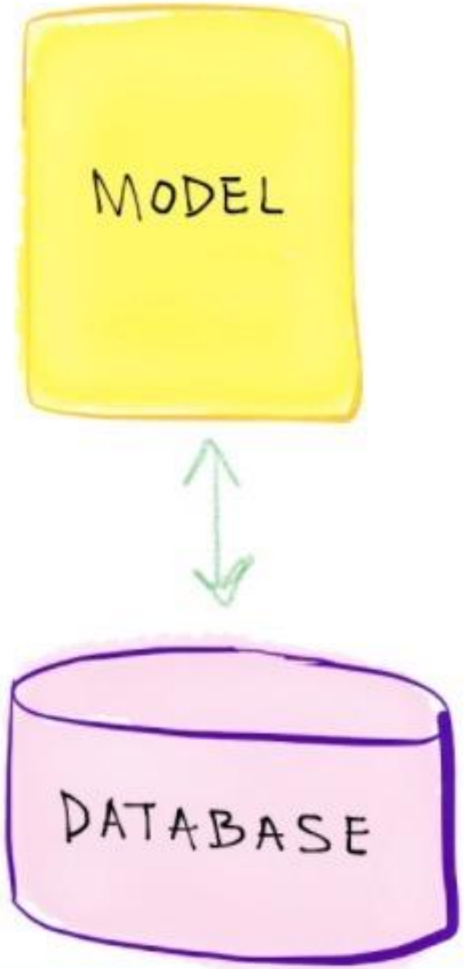
orders	
id	int
user_id	int
status	varchar
created_at	varchar

users	
id	int
full_name	varchar
email	varchar
gender	varchar
date_of_birth	varchar
country_code	int
created_at	varchar

merchants	
id	int
merchant_name	varchar
admin_id	int
country_code	int
created_at	varchar

products	
id	int
merchant_id	int
name	varchar
price	int
status	varchar
created_at	varchar

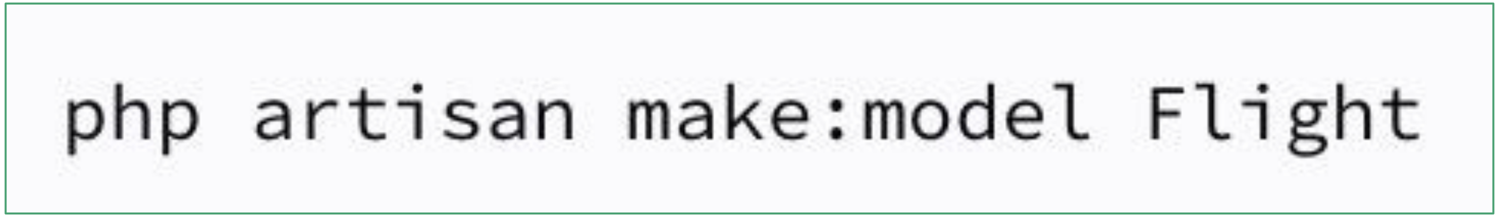
countries	
code	int
name	varchar
continent_name	varchar



Defining Models

To get started, let's create an Eloquent model. Models typically live in the app directory,

The easiest way to create a model instance is using the make:model Artisan command:



```
php artisan make:model Flight
```

Retrieving

In addition to retrieving all of the records for a given table, you may also retrieve single records using find, first, or firstWhere. Instead of returning a collection of models, these methods return a single 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();

// Shorthand for retrieving the first model matching the query constraints...
$flight = App\Flight::firstWhere('active', 1);
```

Inserts

To create a new record in the database, create a new model instance, set attributes on the model, then call the save method.

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

    $flight = new Flight;

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

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

Deletes

To delete a model, call the delete method on a model instance:

```php
$flight = App\Flight::find(1);

$flight->delete();
```

Deletes

To delete a model, call the delete method on a model instance:

```php
$flight = App\Flight::find(1);


$flight->delete();
```

For more details on topics of this lecture: Read Chapter 05

Defining Migrations

A migration is a single file that defines two things: the modifications desired when running this migration up and, optionally, the modifications desired when running this migration down

Creating a migration

Laravel provides a series of command-line tools you can use to interact with your app and generate boilerplate files. One of these commands allows you to create a migration file.

```
php artisan make:migration create_users_table
php artisan make:migration add_votes_to_users_table --table=users
php artisan make:migration create_users_table --create=users
```

Creating a migration

```php
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Creating a migration

```php
public function down()
{
    Schema::dropIfExists('users');
}
```

Running Migrations

Once you have your migrations defined, how do you run them? There's an Artisan command for that:

```
php artisan migrate
```

This command runs all "outstanding" migrations (by running the up() method on each). Laravel keeps track of which migrations you have run and which you haven't. Every time you run this command, it checks whether you've run all available migrations, and if you haven't, it'll run any that remain.

Running Migrations

migrate:refresh

Rolls back every database migration you've run on this instance, and then runs every migration available. It's the same as running migrate:reset and then migrate, one after the other.

migrate:fresh

Drops all of your tables and runs every migration again. It's the same as refresh but doesn't bother with the "down" migrations—it just deletes the tables and then runs the "up" migrations again.

Running Migrations

migrate:rollback

Rolls back just the migrations that ran the last time you ran migrate, or, with the added option --step=n, rolls back the number of migrations you specify.

migrate:status

Shows a table listing every migration, with a Y or N next to each showing whether or not it has run yet in this environment.

Activities and Next Week Topics

This Week:

- Read the rest of Chapter 05 of Laravel: Up & Running, for more information on Databases Migrations.
- Create all the tables shown in slide 11 using migrations.

Next Week:

- Apply what you learned to your final project.
- Retrieving Data, Inserts, Updates, and Deleting with Eloquent

References / Further Readings

- Laravel.com : Laravel's official Documentation.
- Matt Stauffer, 2019. Laravel: Up & Running: A Framework for Building Modern PHP Apps. O'Reilly Media.
- Dayle Rees, 2016. Laravel: Code Smart.