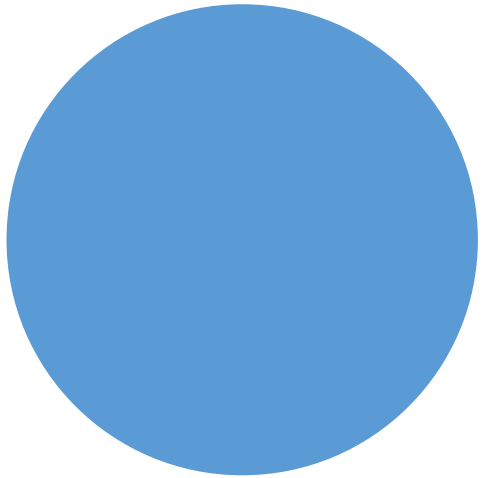
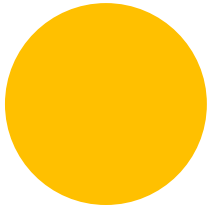



Models and Databases

Recap

- Laravel is a PHP framework which implements the MVC design pattern
- HTTP requests enter a Laravel application and are passed through the middleware pipeline
- One of the first middlewares it reaches is a routing engine, the routing engine inspects the path and decides what PHP function to call based on that
- We can pass data in a way that the routing system can recognize

Where to start?

- For some of you this is the first time you'll be developing a full application, front and back end
- One of the hardest things is to figure out where to start – there are lots of arguments for lots of different places
- My suggestion (actually Liam suggested this) is to start with your data and data relationships.
- Get the database working correctly then build a front end around that.

Database config – The .env file

Laravel stores configuration data in many places. Usernames, passwords, keys and data that may change between development and production servers are typically stored in the .env file.

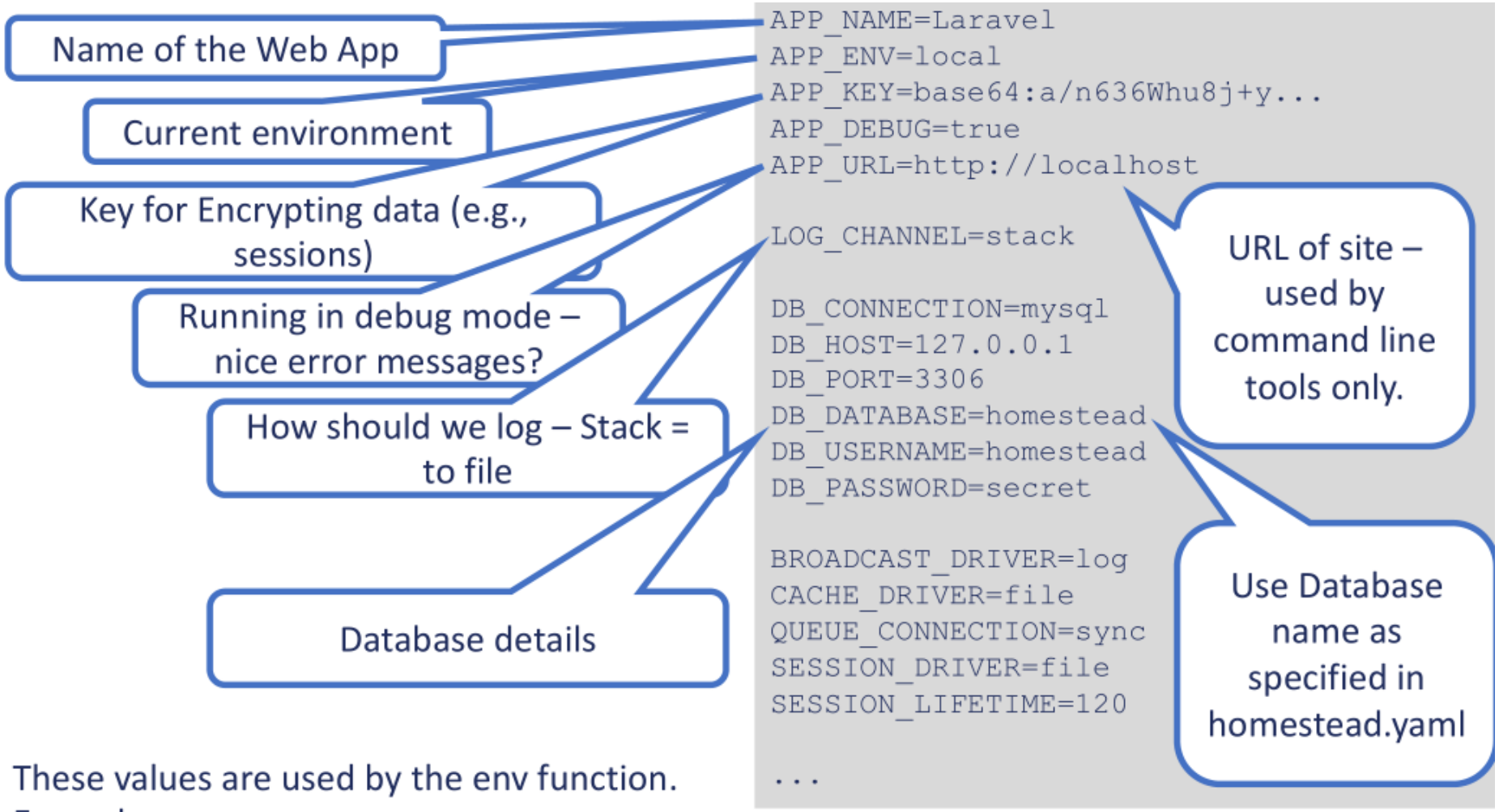
```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:a/n636Whu8j+y...
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

...
```



These values are used by the env function.
Example:

```
env('DB_DATABASE', 'forge')
```

will return the value in .env file for DB_DATABASE or the string '`forge`' if key is not found.

Other Configuration

Other configuration is stored in config folder, e.g.,

- config/database.php

```
...
'default' => env('DB_CONNECTION', 'mysql'),

...
'connections' => [

        'sqlite' => [
            'driver' => 'sqlite',
            ...
        ],

        'mysql' => [
            'driver' => 'mysql',
            'host' => env('DB_HOST', '127.0.0.1'),
            'port' => env('DB_PORT', '3306'),
            'database' => env('DB_DATABASE', 'forge'),
            'username' => env('DB_USERNAME', 'forge'),
            'password' => env('DB_PASSWORD', ''),
            ...
        ],
        ...
```

Default database connection also pulled from .env

List (array) of possible database connections

MySQL connection specified details. Uses env function to pull values from .env file

Configuration Files Best Practice

- The .env file contains machine specific configuration for your application – for example database username and password
- If you share your source code other developers will change this file for their development environment
- …or it will change when the application is deployed
- Therefore you don’t want to put the .env into your version control!

Default Homestead Database Configuration

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

Models and Databases in Laravel

- Lots of concepts heading your way – you'll need to play with this in practice to understand
- Models (in MVC) are responsible for managing the data
- In Laravel a model is a PHP class – for example an Animal
- These models are stored in tables in your database – for example the Animal model is stored in the Animals table
- In your code you'll be working with classes that construct query statements for you behind the scenes – for example Animal::get() will get a PHP collection of all the animals in the animals table

Creating Models

- To create a new model we use artisan

```
vagrant@homestead:~/Laravel/homestead1$ php artisan make:model Animal -m
Model created successfully.
Created Migration: 2019_06_06_085134_create_animals_table
```

This creates two files…

app\Animal.php

(The model class)

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Animal extends Model
{
    //
}

```

database\migrations\<datetime>_create_animals_table.php

(the database migration)

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAnimalsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('animals', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('animals');
    }
}

```

Database Migrations

- You might consider creating the database for your application using SQL commands
- That is a BAD idea!
- It could lock you into your initial design
- The way your database is setup won't be in your version control
- You can't easily wipe the database and start over (which you will need to do a lot)
- You can't easily pass the project to other people, if you ever find yourself emailing someone a database file you should feel bad

Database Migrations in Laravel

- Migrations are the files which live in the database/migrations folder
- They each contain two functions
  - up() which contains instructions for applying the change
  - down() which contains instructions for reverting the change
- These are run in the order of date time in the filenames
  - You might need to manually change the order, do this by editing the filename

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAnimalsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('animals', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('animals');
    }
}
```

Defining the Data in Your Model

- The migration files are where you define the properties of each model
- This is done using methods in the Blueprint class – see https://laravel.com/docs/5.8/migrations

```php
public function up()
{
    Schema::create('animals', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->double('weight', 8, 2);
        $table->dateTime('date_of_birth')->nullable();
        $table->timestamps();
    });
}
```

Running Migrations using Artisan

```
vagrant@homestead:~/Laravel/homestead1$ php artisan migrate
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2019_06_06_085134_create_animals_table
Migrated:  2019_06_06_085134_create_animals_table
vagrant@homestead:~/Laravel/homestead1$
```

Special Migration Table

There is a special migration table:”

```
mysql> select * from migrations;
+----+-----------------------------------------------+-------+
| id | migration                                     | batch |
+----+-----------------------------------------------+-------+
|  1 | 2014_10_12_000000_create_users_table          |     1 |
|  2 | 2014_10_12_100000_create_password_resets_table |     1 |
|  3 | 2018_10_09_185751_create_animals_table        |     1 |
+----+-----------------------------------------------+-------+
3 rows in set (0.00 sec)
```

This records the current migration level for each table.

Laravel will only top up the migrations with new ones.

- Extremely useful if you add more migrations that alter table.
- Especially in production. Allows you to add tables, or alter tables, and preserve data.

Rolling Back and Resetting

You can roll back the last migration operation

```
php artisan migrate:rollback
```

You can also reset back to a blank database:

```
php artisan migrate:reset
```

The Model Class

- The Animal class created is an example of a model, it extends the Eloquent Model class
- Eloquent is Laravels ORM (Object-Relational Mapping) system implementing the Active Record design pattern
- Essentially this design pattern means that a model class is responsible for:
  - interacting with the table as a whole,
  - Representing an individual row in the table, and
  - managing its own persistence

The Model Class – Eloquent examples

- $allAnimals = Animal::get() will get all animals from the animals table
- $cat = new Animal will create a new row
- $cat->save() will save that row to our database
- $cat->delete() will delete that row from the database

Tinker

- One of the other awesome tools which comes with Laravel
- Gives you a REPL (Read-Eval-Print Loop) for PHP/Laravel
- Essentially a command line where you can interact with your application
- Specifically, instantiating models and playing with them
- Great for testing!!

```
vagrant@homestead:~/Laravel/homestead1$ php artisan tinker
Psy Shell v0.9.9 (PHP 7.3.5-1+ubuntu18.04.1+deb.sury.org+1 — cli) by Justin Hileman
>>> use App\Animal;
>>> Animal::get();
=> Illuminate\Database\Eloquent\Collection {#3183
     all: [],
   }
>>>
```

Launching Tinker, importing the Animal class, and getting all Animals…

…empty as expected (make sure you perform the database migration first!)

Creating a model and saving it

```
vagrant@homestead:~/Laravel/homestead1$ php artisan tinker
Psy Shell v0.9.9 (PHP 7.3.5-1+ubuntu18.04.1+deb.sury.org+1 — cli) by Justin Hileman
>>> use App\Animal;
>>> $cat = new Animal;
=> App\Animal {#3176}
>>> $cat->name = "Garfield";
=> "Garfield"
>>> $cat->weight = 500.0;
=> 500.0
>>> $cat->save();
=> true
>>>
```

```
vagrant@homestead:~/Laravel/homestead1$ php artisan tinker
Psy Shell v0.9.9 (PHP 7.3.5-1+ubuntu18.04.1+deb.sury.org+1 — cli) by Justin
>>> use App\Animal;
>>> Animal::get();
=> Illuminate\Database\Eloquent\Collection {#3185
     all: [
       App\Animal {#3186
         id: 1,
         name: "Garfield",
         weight: 500.0,
         date_of_birth: null,
         created_at: "2019-06-06 10:41:41",
         updated_at: "2019-06-06 10:41:41",
       },
     ],
   }
>>>
```

The object we get is a collection and we can use it like an array…

```
>>> $allAnimals = Animal::get();
=> Illuminate\Database\Eloquent\Collection {#885
     all: [
       App\Animal {#3188
         id: 1,
         name: "Garfield",
         weight: 500.0,
         date_of_birth: null,
         created_at: "2019-06-06 10:41:41",
         updated_at: "2019-06-06 10:41:41",
       },
     ],
   }
>>> $allAnimals[0]
=> App\Animal {#3188
     id: 1,
     name: "Garfield",
     weight: 500.0,
     date_of_birth: null,
     created_at: "2019-06-06 10:41:41",
     updated_at: "2019-06-06 10:41:41",
   }
```

Next time we look at how to seed lots of test data…

Tasks

1. Create a new Laravel project and add a model and a database table for that model.
2. Define some properties in that model, then add and perform a database migration to create the table.
3. Using tinker add 3 or 4 objects to the database table.
4. Using tinker get all the objects you added to the database in the form of an array.