Creating Models

Whenever we have a table in the database that we want our app to interact with we need to make what is called a `Model`. Models interact with our database and gives us a layer to be able to add other methods and calls that return data from our database. By just creating the model alone there is already a connection made.

Normally we will have a table for example called `posts` and then we have a model that is tied to that table called `Post`. Notice the singular and capitalized name. Sometimes it may be hard to keep track of this. Luckily we can do so with `php artisan` command.

In our virtual machine we can run:

```
php artisan make:Model Post
```

Just like that we have file inside of our `app/Post.php` that has some boilerplate code that helps our file connect or extend the applications `Model` and tie it to the table `posts`.

Naming tables

Lets say your database table is not `posts` but something like `my_posts`. This break normal convention and although you probably shouldn't do this Laravel gives us a way to be explicit about our table names inside the model. You can add `protected $table = 'my_posts';` and that will take care of the connection.

Lets fire up tinker

In order to test some of the following we can do so with tinker. From our virtual machine and inside the project folder lets run `php artisan tinker`.

Creating a post

To create a post we can simply do the following and pass an array to `Post::create`:

```
Post::create(
    [
        'title' => $faker->name,
        'body' => $faker->paragraph
    ]
);
```

Returning all posts

If we want to grab all posts from the database instead of writing a query to do so we can just add the following code `$posts = Post::all();` which will normally go in our controller. We could then loop through all the posts in our view if we would like.

Finding one post

If we wanted to find one post we would run `$post = Post::find(1);` where 1 is the id of the post we are trying to find. Notice also how I am keeping the variable name plural or singular depending on what we are returning from the database.

Updating a post

To update a post we need to find that post first and then save it after we updated the records of that post.

```
$post = Post::find(1);
$post->title = 'New title';
$post->body = 'Lorem ipsum dolor';
$post->save();
```

Now our new record has been saved.

Deleting a post

To delete a post we need to find that post then pass the delete method to it.

```
$post = Post::find(2);
$post->delete();
```

Where

There are many ways for us to retrieve data from our Post model. Another way to do so is using the `where` clause. You can do `$post = Post::where('title', '=', 'New title');` and that will search the table posts to see where the title is equal to the third parameter passed.