LARAVEL DEVELOPMENT STEPS

# Create Laravel Project

Depends on the environment.

# Create models with everything

Models, migrations, controllers and factories can all be created in single line below:



* Better if I put all models in a single folder called models.
* We can remove edit and create from controllers because this is an api.

# Design database in migrations

* If database is complicated, try mind mapping it on paper first.

## ALWAYS USE “unsigendBigInteger” IN RELATIONSHIPS

I should always use unsigned big integers (UnsignedBigInteger) on relationships, because the auto increment id in all tables are unsigned and big integer. Therefore, if I use foreign() to for example cascade on delete or something, the foreign is not going to match and error happens.

## SOFT DELETE

* I once read an article that said we must not use soft delete on user model. However, it is very good to use soft delete in as many as models we can.

To use soft deletes, we add it to the model:



And we add it to the migration:



## CASCADE ON DELETE | CASCADE ON RESTORE

We must use cascade on delete (or restore if it is an option). To do it more eloquently, based on my experience we better use a package:





The “replies” is the relationship. For example, post has many “replies”.

This package is making this very EASY!

Now we use delete(), withTrashed(), restore() like before and the rest is done for us automatically.

# Code the relationships in Models

* Remember: Let’s say Post belongs to User. We know that Post is the child and User is the parent. then in Post model we say belongs to.
* And: Let’s say Post has many Replies. This time we know that Post is parent and replies are children, then in Reply model we say belongs to, and in Post model we say has many.

## USE MODEL CLASSES INSTEAD OF NAMES:



# Design Factories

I can always use bitfumes factory tutorial that is inside “Laravel spa forum pusher app” tutorial, until I can do it on my own. Its pretty complete and easy though.

# Code the Routes

* All routes inside “routes/api.php” will get “/api” prefix automatically

## USE “apiResource”

In order to create the routes for api, I use apiResource option:

For example, for Post routes:



## Use restful api FORMAT in relationships

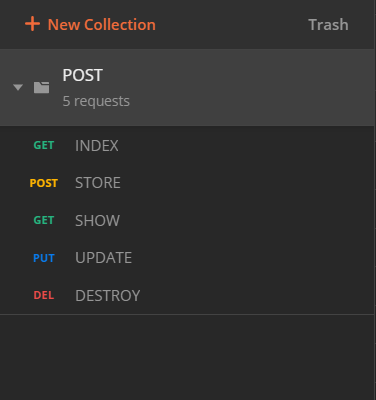
* I add /post/{post}/ before reply to have a restful api format.

For example, for Reply routes:



# Use Postman to test api

For every model I can create a new collection in postman, for example, Post. Inside post collection I can have the required api calls.



# Design resource templates

* The template structure is optional.

## CREATE

To create a resource, we run this command:



## USE

* To use a resource in controller, we must use collection in index, and “new ExampleResorce” in show.
* The example used in our controller section.

## DESIGN

Its optional, for example:

* In the result(postman) everything will be wrapped in a data object.
* We can change the name of particular data we send, for example instead of “body” bellow we can say “content”.
* We can use conditional situations for example if is admin (dev marketer in YouTube did that).
* Also, we can use an additional function called with to add more information along with data.

Example:



# Code the Controllers

* Double shift and type “response.php” to get every response statues text available!

## MASS ASSIGNMENT FIX IN MODEL

* Mass assignment error happens when we use create and update methods, in order to fix that, I use this in my models:



## CONTROLLER

### INDEX

My current template for index is like this:



* Let’s say, we have “hasMany” relationship, for example, question and replies. To get the replies of a particular question, we send the question to index controller, then our code would be something like this:



* Note that when we already sent the question from the route.

CAUTION: Because the route has question, then we have to get this question in all of our reply controller functions, otherwise we get an error.

### STORE



* Also, we can pass an associative array to create method. In this way we can modify data before saving it into database.



### SHOW



### UPDATE



* Also, we can pass an associative array to update method. In this way we can modify data before saving it into database.



### DESTROY



## DESIGN REQUEST VALIDATION RULES