Laravel API Boilerplate

How to build an API in a day

By Max Snow

COVER GENIUS

- PHP Developer since 2004-5 (2008 commercially)
- Worked to build countless projects...
- Working as Engineering Lead @ Cover Genius
- http://github.com/specialtactics
- https://maxsnow.me
- https://twitter.com/devopmax

The Why (of why every project should be API)

Good reasons to write your backend as a REST API

- Build once, use everywhere
  - Multiple frontends
  - Mobile apps
  - Third-party API consumers
- Easier to test & refactor
- Easier to manage
- Strong decoupling
- Rich frontends
- **Enables Microservices**

The How (of making it easier)

- Strong conventions and principles
  - Because REST APIs have strong conventions and principles, we can reuse code to implement their functionality
- Probably 70-80% of **API related** logic in an application is very similar
- We can use Laravel to great effect in order to generalise about the things that are different for each resource (**Model**)
- Laravel and REST actually have a lot of similarities
  - Simplicity, convention over configuration, interface oriented, resources as models

Quite often, the problem is that it takes time to invest in new tech

This is especially true in any company that wants to prioritise new features over anything perceived as risky.

What does this API Boilerplate provide you?

- Get moving **extremely quickly** with API development
- Almost no code necessary for bread-and-butter REST
- Extend and customise only when necessary for custom business logic
- Lots of conveniences and features
- Complimentary docker setup
- Minimalist & largely unopinionated
- Guidance for adding complex functionality
- Significant roadmap for some pretty awesome integrations and features

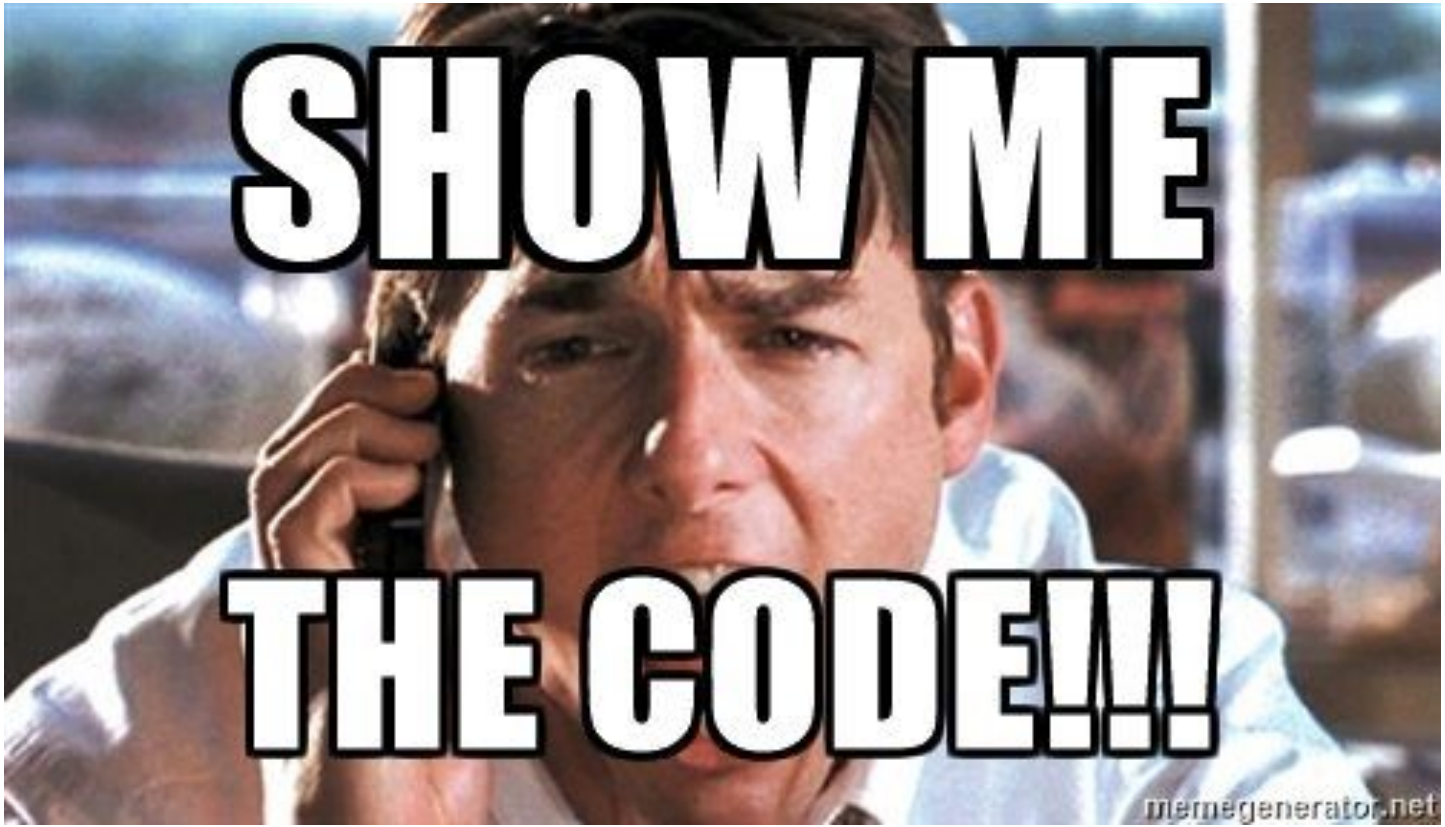
- Does not implement your business logic
- Does not magically know exactly how you want your API to be and behave
- Does not integrate with every one of the packages you might want to use
- Does not provide a frontend

The Benefits of using this boilerplate

- Shift the focus away from boring, repetitive code, and more towards business-specific logic and hence delivering real value
- Less code is better, a conventional approach to reducing complexity is good
  - Less code also means less tests to write!
- Uses normal OOP and existing Laravel conventions where possible
  - Easy to extend and customise logic
- All heavy lifting is done by it's own dependency package
  - Easy to update
- Preference is to be minimalist, will not try to redefine everything and does not require a great deal of learning on how to use it

Where can this be used?

- You need to build a large, enterprise scalable system from scratch
- You need to build a small microservice
- You are beginning to take apart the monolith
- You need a proof of concept / RAD
- You are working on a hobby project


SHOW ME
THE CODE!!!
memegenerator.net

Reference

- https://github.com/specialtactics/laravel-api-boilerplate
- https://github.com/specialtactics/laravel-api-boilerplate/wiki

Thank You!
Come chat to me @ after dark