Why ‘NO Framework’ (in PHP)?

Bikash Shrestha

Framework – In Brief

- A library which gives standard solution to typical problems.
- Generally, built for masses.
- For example, for an online shop, a framework built for that, is able to cover functionality like user log in (with session handling), shopping cart, placing orders etc.

Advantages (and why people fall for it)

- No need to reinvent the wheel.
- The code (usually) works and is tested.
- Fewer security leaks.

Reasons not to use a PHP Framework - 1

- PHP, itself is a pretty strong language.
- The Learning Curve of Frameworks.
- Limits your creativity (makes you DUMB).
- You will learn a lot by writing on your own.
- No framework is made for ‘your’ need.
- Requires more work than necessary (for small projects)

Reasons not to use a PHP Framework - 2

- Executes the same code repeatedly (even when not needed to do so)
- Complex and too many interdependent classes.
- Duplicates web server functionality.
- Large frameworks can hurt clients.
- Losing productivity while using outdated frameworks.

The Irony of Large Frameworks

- **Commonly heard - ‘use small framework for small project and robust framework for large projects’**
- **Problems with this approach:**
  - Difficult to migrate to newer versions.
  - Need to live on with the same version of framework….FOREVER!!
  - Limits features of your app, as it ages.

What you can (or should) do

- Pick a minimal/modular framework (example: Fat Free Framework).
- Establish a structure for your app.
- Adhere to a coding standard (example: PSR-2).
- Find common packages.
- Build your own library.

Conclusion

- For any Framework you choose, it has its pros and cons.
- Set your preference as such: core-language solutions – small abstractions – small helper libraries – general libraries – then, (maybe) framework.
- Use Frameworks only when absolutely necessary.