Fundamentals of Framework Design

1. Leave the Language Alone
2. Single Point of Entry
3. Simplify Common Tasks
4. Codify Patterns
5. Code to the Tools

# Leave the Language Alone

The purpose of a framework is not to “fix” the language. One of the criticisms of PHP is inconsistent parameters. Two functions with the same essential parameters have the parameters in a different order. If you are writing functions to simply reorder parameters of built in functions, then stop. A proper IDE will have code completion and hinting which will tell you what the parameters are when it comes up. It is important to know what the various built in functions do, not remember every parameter for every function.

The second “fix” is generally a template engine. PHP is a templating language. Shoving values into an array and then “str\_replace”ing them into a “template” is not helpful and only serves to break the connection between the view and the controller. A developer needs to know what variables are in the view as much as they do in the controller. Code completion and type hinting are just as critical when creating a view as it is when creating models and controllers.

The goal of a framework and coding practices is to minimize code so that by the time the developer gets to the view, there is little code to embed within the front-end languages. It is not the goal to invent a new language which solves no problems, either real or imagined.

# Single Point of Entry

While obviously you should not put all the code for your site into a single file, you should also not start in a different file for every page of your site. A key function of the framework is giving a single point of entry so that every page automatically has access to database connections and business logic without having to repeatedly include a large collection of files. In the case of QuickDRY, even cron jobs include index.php just as htaccess (and web.config) route every web request through index.php. This makes it trivial to give your pages and your crons access to the common business logic that drives both.

# Simplify Common Tasks

The key function of a framework is to simplify common tasks. As you develop multiple projects you start to see the same tasks pop up repeatedly. A simple example is how to display a date on the page. It then makes sense to create a function or few which format dates so that you do not have to keep retyping the same code repeatedly. Often the source information is not in a standard format and it is necessary to operate on the given value to put it into a form that can be formatted differently than given.

A more complex example is talking to the database. Especially with CRUD functions. QuickDRY implements a few ways to interact with data sources. You can write queries and read the results into associative arrays, you can read them directly into an object defined by the ORM, or you can write a custom PHP function that maps the result into an object. There is no “best” way to talk to the database because on complex projects it may be necessary to use multiple methods to get the job done. You may have custom reports that the ORM cannot be used to get the structure of. You may have ad hoc queries. You may have stored procedures.

The job of a framework is not to rigidly define how to do something, but simply to simplify the most common methods of doing things. If another method is found that works better, then the framework may adopt that method.