**Created 05/16/15**

**Updated**

**Gig Central Application**

**Basic Guidelines**

This document attempts describes some of the best practices and guidelines for coding PHP.  We also included information about the framework.

* Befriend the PHP Manual
* Don’t be Afraid to Ask for Help
* Avoid Obvious Comments
* Consistent Naming Scheme
* Capitalize SQL Special Words
* Use Meaningful, Consistent Naming Conventions
* Use Indentation
* Explain your code by providing comments

**GitHub**

**Branching**

Branches are used to develop features isolated from each other. The *master* branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion. When you're working on a project, you're going to have a bunch of different features or ideas in progress at any given time – some of which are ready to go, and others which are not. Branching exists to help you manage this workflow.

**Branching Naming Guidelines**

* Choose *short* and *descriptive* names
* Use *dashes* to separate words.
* When several people are working on the *same* feature, it might be convenient to have *personal* feature branches and a *team-wide* feature branch. Use the following naming convention:

$ git checkout -b feature-a/master # team-wide branch
$ git checkout -b feature-a/maria  # Maria's personal branch
$ git checkout -b feature-a/nick   # Nick's personal branch

**Pros of branching.**

* Keeps all of the work being done around a project in one place
* All collaborators can push to the same branch to collaborate on it

**Add commits**

Once your branch has been created, it's time to start making changes. Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch. This process of adding commits keeps track of your progress as you work on a feature branch.

Commit messages are important, especially since Git tracks your changes and then displays them as commits once they're pushed to the server. By writing clear commit messages, you can make it easier for other people to follow along and provide feedback.

Example: git\_commit\_message\_style\_guide

**Open a Pull Request**

Pull Requests initiate discussion about your commits. Because they're tightly integrated with the underlying Git repository, anyone can see exactly what changes would be merged if they accept your request.

**Discuss and review your code**

Once a Pull Request has been opened, the person or team reviewing your changes may have questions or comments. Perhaps the coding style doesn't match project guidelines, the change is missing unit tests, or maybe everything looks great and props are in order. Pull Requests are designed to encourage and capture this type of conversation.

**Deploy**

Once your pull request has been reviewed and the branch passes your tests, you can deploy your changes to verify them in production. If your branch causes issues, you can roll it back by deploying the existing master into production.

**CodeIgniter Basics**

****

* The **Model** represents your data structures. Typically, your model classes will contain functions that help you retrieve, insert and update information in your database.
* The **View** is information that is being presented to a user. A View will normally be a web page, but in CodeIgniter, a view can also be a page fragment like a header or footer. It can also be an RSS page, or any other type of “page”.
* The **Controller** serves as an intermediary between the Model, the View, and any other resources needed to process the HTTP request and generate a web page.