CodeSwatch

Pluggable App Development Framework

\_\_

**In Briefx**

CodeSwatch makes apps. This JavaScript framework organizes code in a manner great for collaboration. The vast majority of CodeSwatch-based apps will require less than 20% human-written code. This human-written code is leveraged to fully customize CodeSwatch functionality to bring forth almost limitless options in the kinds of apps created.

This is called a “pluggable” app development framework because both:

1. desired hybrid apps based on the functionality of popular technologies, acting as almost specifications for these CodeSwatch-based hybrid apps.
2. This framework is fully customizable and modular in a way that allows for easy removal and inclusion of functionality and design.

**Introduction**

The CodeSwatch Framework is based on Madhusudhan Srinivasa’s Node-Express-Mongoose project (MIT License), which to-date, along with its Demo application, has been forked over one thousand times, receiving over 3,000 stars. Mike De’Shazer worked from the core of Srinivasa’s code and added the JavasScript-based Ionic Framework (MIT License) to develop a framework for complete app development on mobile platforms, as well as the web, to develop both:

1. the Factory which uses the CodeSwatch framework to build apps
2. CodeSwatch-based apps (or “Flavor Homes”) developed via the Factory

**The CodeSwatch framework**

The CodeSwatch framework has a file structure such that at the root consists of:

**app**

The core functionality of the web and RESTful API accur here.

Controllers determine which models should used and how views should renders data.

Views contain the HTML templates, leveraging assets from the public folder.

Models contain the schema for data structure . Most data structures of extend generic CodeSwatch data schema, containing usually the attributes of “email”, “location”, “availabilitySlots” to allow communication between almost any object in the CodeSwatch framework.

**config**

Consists of the routes.js file, which determines which address endpoints lead to which controllers (which then link to a HTML or JSON-based) view, leveraging content from the models.

Additionally is the env folder, which declares environment variables on the server for security and protection of client IDs, API secrets, and other credentials. Middlewares which helps prevent race conditions for end-users accessing and manipulating content, mandating by default for example, users should be able to only edit content they can upload.

**lib**

The lib folder contains one file, which formats Mongo errors in a proper way.

**node\_modules**

Extensions to functionality is stored here in folders by the name of the modules acting as extenstions. One module is the Express framework, which we use within the CodeSwatch framework. Other modules are less notable, such as the imagemagick module, which is only used, say if we are uploading content to an Amazon, IBM, etc cloud.

**public**

Most css and javascript files are located here in the assets, js, and css folders.

The image uploads when working in a local development environment as save in uploads folder. Additionally, mobile app code for iOS and android are stored here, respectively in the android and ios folders, in public/mobileApp. (also see public/mobileApp/main).

**test**

Basic unit and functionality tests are written for ensuring the functionality of article and user functionality throughout the application for the *should* and *super\_test* modules.

(and hidden folders below)

**.openshift**

Contains runtime exectubales, as well as other configuration for a given project set to live on the portable OpenShift app.

**.git**

git repository folder, keeping track of each commit and branches within a project.

(with these folders notably, which contain front-end code and markup)

**public/mobileApp**

Here, all Ionic code is stored for the front-end of the Android and iOS applications. Blackberry, WindowsOS, and other mobile platform could be stored here, as well.

**public/templates**

Licensed and open-source code for the website for each application. Most templates contained are responsive for all screen dimensions, if not all. Event pages, calendar pages, homepages, messenger UIs, dashboard UIs, depending on the project are present here for use by app/views code.

**The Factory: Swatch & Sons**

The Factory, nicknamed “Swatch & Sons” is a set of scripts which previously exist in the config/generator directly which leverage code through to project and uses design/coding patterns to generate a specific application based on specificaitons generalized through the use of common, popular application (Such as Uber or AirBnB).

**The Store: CodeSwatch.com**

This is a web portal through which innovators can mash-up and integrate functionality between popular applications and for various types of users or objects, such as artists or photos. A specification might be an AirBnB & Youtube for Artists and Companies. The CodeSwatch Idea Generator can then give the innovator choices ranging from “video uploader for artists and companies with booking features” to “e-commerce platform for artists and companies with content upload features.”

**The Flavor Home**

CodeSwatch-based apps are usually designed via a specification of combinations of flavors in the CodeSwatch Store. Flavors refer to sets of functionality (or swatches) that combine to make flavors like Youtube, Facebook, Twitter or Netflix. A Twitter-Youtube mashup might resemble Vine in functionality. A CodeSwatch-based app is a Flavor Home, where one or more flavor types hav combined to host users that enter them in some way, shape or form.

**The Foundation: Core Datasets**

The Core Datasets that drive how a CodeSwatch Flavor Home is generated are:

1. Swatches dataset
2. Objects Dataset

**Structural Skeletons: Swatches**

Inside each application, in the app folder, in views, is layouts folder. Inside that folder, there is head.html file, which references angular and csApp.js. Included in every csApp.js contains all code that connects back-end services with the view (this is in public/asset/js) .This is auto-generated I the factory and can be edited to change how the DOM manipulates based on your logic. There is no need to put functionality in the DOM

, but to use Angular, as demonstrated by example. In default.html, the primary skeleton of each page includes the body tag of ng-app. The value of this attribute is “SwatchCtrl”. The csApp.js, has much of the data your app needs stored here.

**Nails, Beams and Plastic: Components**