Giv2Giv Development Team Programming Reference

Overview

The giv2giv system was built using CakePHP and a MySQL database backend to allow users and charities to make and receive micro-donations, respectively. It was built following the Model-View-Controller (MVC) paradigm. As such, the application logic falls in the controllers, the page layout code is in the views, and the data is encapsulated in the models. All of the MVCs are named following the CakePHP naming convention, and are directly tied to the database tables of similar names (i.e., UsersController is tied to the users database table). The code follows this structure and tries to separate application logic to ensure that functions relating to each different table are only located in its associated controller.

Also, to help augment this programming reference document, our code has been equipped with phpdoc-style comments at the beginning of functions to explain what the function does, describe any parameters, and clarify any return values. There are comments embedded various places within the code as well to explain more confusing portions of the system.

This document presents a more high-level overview of the different major portions of the system, while the embedded comments can help explain the inner-workings of the application on a more granular level.

Design Decisions

Charities

The parsing of the charities is based on the assumption that there are 6 data items associated with each charity. Between October 2012 and March 2013, this changed from 7 to 6, so I anticipate it will change again. The 501c3 upload to Amazon S3 makes use of a small library included in the code. Both charities and packages can be tagged; packages inherit any tags that belong to its member charities. There is currently no ranking algorithm to return more relevant or most used results upon a package search. Charity accounts cannot be created if the requested email is already a user account. Packages cannot be edited if anyone is following that package.

Donors

Donors must supply valid email addresses when logging in. These email addresses are verified through an email authentication process to ensure that people are not using fake emails to create users in our system. Donors must link a payment method (currently either Dwolla or PayPal) to use the system.

Funds Tracking

We have chosen to use a share-based approach to determine the fund value of donor's contributions. After a user donates money to a charity, but before it actually reaches the charity, it spends time in a giv2giv-held investment fund. Each night, the investment fund updates its values, and we alter the value of the shares in the system to reflect any appreciation or depreciation in the system. Exactly how this is accomplished can be seen in the funds tracking portion of the code.

Payment Methods

As previously described, users must choose a payment method, either Dwolla or PayPal to use the system. Although both can be linked, as a design decision, we have only chosen to implement the full funding and donation workflow with Dwolla. In the future PayPal should be integrated into the system as well.

On signup, we gather demographic information about the user from Dwolla in order to augment our knowledge of the user. Some of the information harnessed includes full name, current city, state, and zip code. We use Dwolla's RESTful API to handle linking of the accounts and submitting money requests to the users.

Dwolla provides webhook notifications that alert the system whenever a money request has been fulfilled or the status of a transaction changes. The notifications controller is set up to handle any incoming web requests, verify they are from Dwolla, parse them, and handle the resulting notification accordingly. The incoming web requests are JSON-encoded post requests. After parsing the request, each notification is examined to determine whether it is a request fulfilled or transaction status notification. The resulting information is the database is then updated to reflect the notification type as appropriate.

How data is stored

All data in the system is stored in a MySQL database. There is an assortment of tables in the database, each of which is named after a different entity in the system. For example, there is a users table to store information about a user, and there is a notifications table to store the incoming webhook notifications. Each database table has an associated, and similarly named, controller that handles the logic of saving data to the table. Specific examples of how this is done can be viewed in the source code; there are comments explaining the various procedures.

Other Components

Facebook, Twitter, and Google+ are all integrated into the giv2giv system. Users can link any of these accounts to their giv2giv accounts using the social networks' OAuth methods. Having these accounts allows us to gather demographic information about the users and to share giv2giv actions to their social networks.

We also use Amazon S3 to store 501c3 forms supplied by the charities. Our customer, Michael Blinn, provided this account.