Mango Punchbowl Documentation

Intent:

The intent of this application was threefold.

1. Establish a unified posting system to post messages to Facebook and Twitter that can be used for any desired function or integration in the future, keyed on either a supplied email or UUID. Handle initial authorization to both platforms as well as retention of tokens for posting on customers behalf in the background.
2. Build a front end interface to the posting flow.
    1. Mango Client integration - Give the client application a tool that can be loaded into their system that will handle posting to the posting flow for their customers based on UUID

Execution Point 1 - Facebook/Twitter:

Facebook and Twitter integration is a very difficult dance to get right. The Punchbowl application uses a number of tricks to make this as seamless as possible will remaining within the confines of the rules from both platforms for authorizations.

API keys:
Facebook and Twitter currently have temporary API keys assigned to our accounts and registered with the URLs of our test system at punchbowl.mangolanguages.com

Integration Method:

Facebook - We used Facebook’s new OpenGraph API with Oauth2 support when connecting from Punchbowl. Initial authorizations with Facebook are made via an http redirect to a special url. There the user will be presented two paths

1. If not logged into Facebook, a login request is made. Once logged in, move to #2
2. A screen asking to authorize mango's API account to access the users account offline and post stream

Once authorized, a session token is returned to punchbowl. PB will reconnect with FB to exchange that session token for a permanent authorization token that is stored with the customer record in the FacebookAccount model.

A customer record with a FacebookAccount that contains a token is considered Green Lit to post directly to facebook without reauthroization.

Twitter - We use Twitters Oauth1 support when connecting with Punchbowl. The steps are very similar to Facebook's when authorizing an API account to access the twitter user. Twitter uses a token and secret combination to exchange session information for a authorization token, but the idea is the same.

Once authorized, a TwitterAccount record is tied to the Customer record.

Error Handling - Both Facebook and Twitter return JSON for messaging. We have taken the approach of building an error handler into two models, TwitterApi and FacebookApi, which can recognize some of the more common errors returned by these platforms. Much of the time building the posting flow was in trapping and handling error events.

When an error happens, it is logged to the Customer's record, logged to the Rails log, and (optionally) can be emailed to a system account

One error to look out for is 'Duplicate Message' . Neither Facebook nor Twitter allow the same message to be posted back to back, so this error is an indication of someone using the back button or scripting posts to someones account

Modularize Flow:
In order to meet the requirements of the second execution point, we needed to build a modular approach to posting to social media that could

be used in many different controllers. A mixin called Postable was constructed to allow both html and json requests and responses to any controller result in a post to social media, along with the handling of redirects and messaging.

Execution Point #2 - Point of Entry

Mango Client:

The most complex integration task for this project was building a tool that could be used by the Mango Client. To this end we have constructed a jQuery plugin called Punchbowl that can be included as a javascript file. The plugin has all the needed code to query PB directly for information and send customers in different directions depending on the results of the query.

The example page uses a test and debug mode to show the various states of integration with PB from the client. Currently the initialization settings are

1. url - currently defaulted to 'http://punchbowl.mangolanguages.com'
2. debug - defaulted to false but turned on
3. testMode - defaulted to false but turned on

Once initialized, the client will pass the following information to the plugin on call

1. UUID - '1'
2. Message - 'I just completed lesson 2 of english'
3. badge_name - 'English'
4. language - 'English'
5. lesson_number - '2'

Naming is important with badge_names and language, as they are case sensitive and are capital in the demo

If a UUID is null, the customer is considered Anonymous and is shown the overlay without remember_me

If the UUID is not null, given the settings and the data, a call is made to PB using GET json to lookup the customer based on UUID. Response is evaluated

1. Customer is not known - show overlay
2. Customer is known but does not want to share - do nothing
3. Customer is known, wants to share. but wants to be asked - show overlay without remember_me
4. Customer is known, wants to share, doesn't want to be asked but is not green lit - show overlay
5. Customer is known, wants to share, doesnt want to be asked and is green lit - POST json to Accomplishments#create - **NOTE: Since this post is constructed by the jQuery plugin it lacks the authorization token created by Rails for forgery protection. As such forgery protection has been turned off at this time.**

Overlay:
The Mango Client will have a form that will be filled in

When the overlay is filled in and the post button is hit, a jQuery event needs to serialize the data and sends an does an html submit to a target = _blank to Posts#new to begin the authorization process.

NOTE: Currently the testing of this on the demo site is using a uuid of 1, associated with one of our customer accounts.