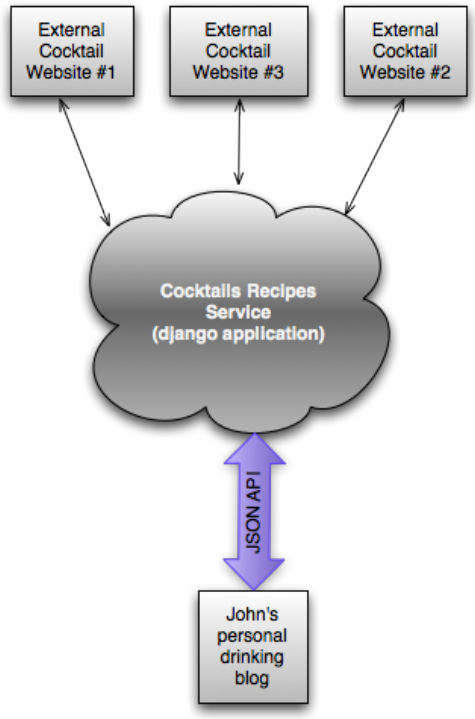
Project: Cocktails Recipes Service



Description:
Application is fairly simple web-service which offers website owners to put funny cocktails widget on their websites.
Widget will then show 1 random cocktail recipe to website visitors on each page load or by user request.
Application uses own cocktail drink recipes database and external websites as content sources in the same time.

Internal storage design:
Internal database can be represented by 3 models:
- Ingredient model (which holds titles of ingredients)
- Cocktail model (which has title, description, category and set of Ingredients)
- Category model (which holds set of Cocktails)

You need to provide simple administrator interface where one can:
- Add/Edit/Delete ingredient
- Add/Edit/Delete cocktail
- Add/Edit/Delete cocktail category
- View list of cocktails by ingredient
- View list of cocktails by category

Important notices:
Do not use django admin and do not bother yourself with authorization, this is still testing project! :)
Ask user what to do with depended entities before delete linked entity in other words:
What to do with cocktails when someone wants to delete Category or an Ingridient.

`*Add Cocktail*` should be a dynamic form, because one cocktail could have several ingridients attached.
This form should have "add ingridient" button which dynamically add additional ingridient field. Javascript based solution will fits best here.

Thats all about internal database. Please fill it with some sample data (few categories, ingridients and cocktails) via fixtures.
Sample data could be found here: http://www.1001cocktails.com/recipes/all-mixed-drinks.html

Plugins requirements:
In sum, application plays role of webservice and unified proxy to external content sources. Content sources are designed as plugins. Each plugin is designed for single external source and provides unified input and output signatures for it's users (internal api for developers). It should be fairly easy to add new external content source by creating a plugin.
There are 4 websites where you can get random recipe by accessing given URI. Each website outputs title, description and list of ingridients of a randomly picked cocktail. Plugin need to parse output and convert to internal structure for later use.

See the following websites:
http://www.cocktailbuilder.com/ - on each request website generates list of random cocktails.
http://www.randomcocktails.com/rc.py - on each post request website generates random cocktail.
http://www.1001cocktails.com/recipes/cocktails/recipe_cocktail.php?recette_cocktail=00 - on each request website outputs random cocktail
http://www.cocktail.uk.com/db/random.asp - on each request website redirects user to random cocktail page

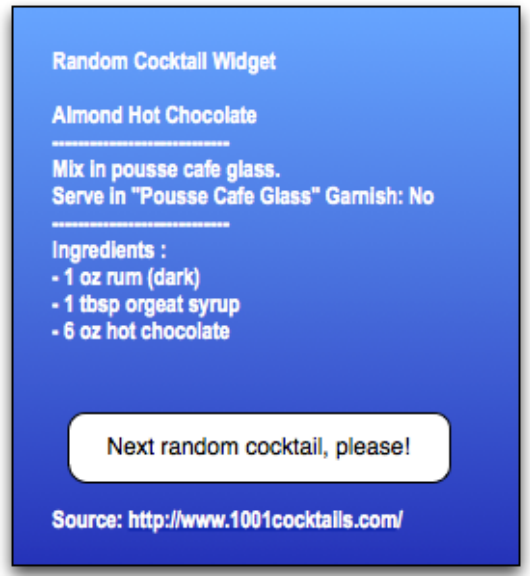
Prepare 4 plugins that could be used to parse cocktails. By calling plugin, application expects to receive dictionary with title (if available), description and list of ingridients of a random drink or an exception in case of error (connection, parsing error etc.).

You do not need to store retrived cocktail anywhere.
Thats all about external data source. Lets now define how our Widget should work.

Widget requirements:
Widget should be easy to install. User just grabs few lines of javascript code and puts on his website.
Please take a look how Google Analytics or Google AdSense javascript codes works.
Installation javascript code should not block user page loading even if target webservice (our server) is down. To simplify things, you are free to assume that everyone uses latest version of jQuery javascript library. So dont bother yourself and make use of this library.

Widget outputs 1 random coctail recipe in the following form:
Title
Description
List of ingridienst
<show me another coctail button>
Source (source domain name or 'local' in case of internal database usage)

Example:



Random coctail could be retrived from external data source (via plugin) or from internal database.
This should be random - either to use internal or external data source.

Widget should communicate with application via simple json api interface. In case of any errors (api level or http level) widget should correcly handle error and output apropriate message to enduser.

Please, create sample html file or django view with widget demo.

Final note(s):
- Create unit tests for critical parts of a system.
- Target python version is 2.7+
- Target django version: latest stable (1.3.1+)