Project 3 - Web APIs

Due Date: 5:00 p.m., Dec 4, 2015
Any changes made to the assignment after posting it will be in red.

Description

One of the primary programming skills you will need in today's software developer landscape is working with web APIs. There are a lot of fun, free API's out there for you to use. Here is a quick tutorial on how accessing API's works in Ruby, and I will also cover the topic in class.

For this project, you are going to build a program that uses at least 3 web API's to produce some output. For example, you may choose the following web API's:

- http://www.ipify.org/
  - returns the public ip address of the caller
- http://ip-api.com/
  - returns location information based on an ip address
- http://openweathermap.org/api
  - returns weather information based on city name

In the example API's, you would grab your current IP address, use it to get your city name, and pass that on to get the current weather forecast for your city.

Here is a link to a list of public API's: https://www.publicapis.com/. You are not limited to what is here, but you are looking for an API with the following attributes:

- Returns JSON
- RESTful
- Free, open
  - You may sign up for a key based API, such as a google service, but it is out of the scope of the assignment, so

you will need to work out authentication based API access on your own.

You must use the result from each API call as input for the next API call. Each API should have a proxy class built to encapsulate the API call. All API Proxy classes should also share the same interface so that you have the ability to combine the result of two API calls, and use it for the next call.

You may use as any and as many API's as you wish, but they must be fairly reliable. You also must create a fallback in your Proxy class in case the api call fails. Return a generic value with a message that the api call failed, however, your program as a whole should not fail.

Any changes made to the requirements to make program more sophisticated or interesting must be accompanied by an explanation in the Readme. The only rule is that the changes must not simplify the project.

Submission

- Create a readme file following the previous project README format, and add it to your project folder
  - Here is an example submission that your should project should match: example_submission.tar.gz

- All file inclusions must be case sensitive. If you are working on a Windows machine, you will need to double check this.
- Create a tar archive with the command ”tar -czvf project3.tar.gz *”, and then upload the archive to Blackboard before the deadline.
- Demo your project by downloading from blackboard and extracting your archive with the command "tar -xvf project3.tar.gz". Then run your code, show your source to to the TA, and answer any questions they may have.

Grading Guidelines

- **Project:**
    - Uses the proxy pattern to access each API (6 points)
    - Uses the composite or decorator pattern to organize the API calls (6 points)
    - Has fallback for each API in case of call failure (2 points)
    - Uses the ‘open_uri’ or Net::HTTP module from the Ruby library (1 points)
    - Uses the ‘JSON’ module from the Ruby library (1 points)
    - All classes in a separate file (2 points)
- **Submission (2 points):**
    - includes readme in plain text format (.txt)
    - submitted tar archive only files (not in a folder) to blackboard
    - Follows example submission and submission guidelines