ReachLocal Japan Programming Test Instructions

Instead of whiteboard coding or algorithm quizzes, we would like to see how you work with a small project that uses fundamentals similar to what we handle daily. While going beyond this spec is great, the purpose of this is primarily to see how you address these specific requirements. We place high value on maintainability and idiomatic use of Python; please see the second page for general expectations.

This project should take between 3 and 4 hours.

1. Create a new project, hosted on either BitBucket or GitHub, using git.
2. Include a complete requirements.txt file for dependency management.
3. Build a simple Django project.
    a. Consume from the /articles endpoint at newsapi.org.
    b. Use this API key, but do not include it in version control: af92652e76b745a6bde8dd2fc5739bfd
    c. Store news items locally using a database (SQLite is fine) and Django models.
    d. News item endpoint: Return the most recently-published 100 news items in JSON format.
    e. News item listing (HTML): Display a list of the most recent 20 news items, newest first.
4. Project should run normally with the development server (python manage.py runserver).
    a. Include a README file with any necessary setup steps.
    b. Do not include a fixture or database in your repository; your README should include steps for database migration and article population.
5. Either make your project public on GitHub/BitBucket and provide the link or share the private repository with us (username *reacheric* on either site)

While this obviously needs to be your code, don't hesitate to use libraries and code snippets where appropriate. Just make sure you leave a comment with the source if you choose to use snippets, and remember that the point of this exercise is for us to see your own work.

During the technical interview, we'll ask you to add a minor feature to this project so we can watch you work. The feature will be picked based on your particular project.

General project expectations

- It's up to you what to include on your listing and return values, but be sure to link the articles.
- Carefully consider what uniqueness constraints your models should use, and apply them at the model level. This is especially important if you choose to consume from more than one source.
- Avoid custom project templates or significant changes to Django's default project structure, as well as using additional unnecessary frameworks.

Expectations for Python and Django usage

- Use the latest stable release of Django 2.x
    - This implies using Python 3.x
- virtualenv
- PEP 8
- Basic HTML UI (JavaScript is entirely optional)
- git usage
  Ensure that you are correctly using the .gitignore file. While branching is unlikely to be useful for this kind of project, please be aware that we regularly use git flow for our repositories.
- staticfiles
  If you choose to use local assets in your HTML, use the staticfiles template tags to link them from your listing page (alternatively, you can link to a trustworthy CDN).
- Class-based views
  While it's still possible to write Django using view functions, we often need to work with forms, so for consistency, we always work with Django's generic class-based views. You won't need to write any forms, but please use these class-based views appropriately, rather than view functions.
- Ensure your models have Model migrations
- Queryset manipulation
  Your views should override the appropriate methods and use basic queryset manipulation to ensure your templates/responses receive the correct results.
- Meter API requests to avoid throttling
- Safe coding
  Be careful when handling external data, processing user input, applying regular expressions, or any other potential source of malicious or malformed data.