Signals Documentation

Release 0.1

Yeti

November 22, 2015

Contents

CHAPTER 1

Quickstart

1. `pip install yak-signals`
2. `signals --schema ~/path/to/schema.json --generator ios`

CHAPTER 2

What is Signals?

Signals is a code generation tool to take the grunt work out of communicating between your front-end and back-end applications. Simply describe your server endpoints in a JSON file and watch Signals automatically generate Objective-C data model files. When you update your API, all you have to do to update your front-end code is re-run the script. Signals is an open-source project initially built by the folks at Yeti.

CHAPTER 3

Contents

3.1 Get Started

Signals is a tool that will automatically generate front-end code for you based on a JSON schema file. This means that you can simply define your API endpoints, and if they change, you can just update the JSON schema, run signals, and have your front-end update to reflect the changes. No more need to write boilerplate code to define your models or how they talk to your server. Signals is backend-agnostic, and can be used for Django, Rails, Node, etc. apps.

1. Signals is written in Python. You'll need Python on your machine if you don't have it yet.
2. `pip install yak-signals`
3. To see if signals successfully installed, run `signals --help`
4. To run signals, you just need to specify the JSON schema file you want to use, what sort of code you want to generate (ObjC, Anguälar, etc.), and any options for that generator

See below for specific instructions for each generator.

3.1.1 iOS

Currently, our iOS generator assumes you are using Objective-C, Core Data, and Restkit. If you have a different setup (Alamofire, Swift, etc.), see the section on writing-your-own-template.

1. **If your API changes will affect Core Data (e.g., you are changing the fields you receive as a response to a call):**
    - Make sure you first quit Xcode
    - Run `signals` and pass the `--coredata` flag (e.g., `signals --schema ~/path/to/schema.json --generator ios --coredata`) (TODO: is this how you do it?)
    - If you changed the core data models, you'll need to have xcode auto generate the new model files (TODO: someone mention how to do this)
2. If you're not changing Core Data, just run signals: `signals --schema ~/path/to/schema.json --generator ios` (TODO: is this how you do it?)

3.2 Try the Demo Server

Signals comes with a demo server that you can test against just to see how this is all working.

1. Save the sample schema from the demo server docs.
2. `signals --schema ~/path/to/schema.json --generator ios`
3. TODO: what sort of thing can a user now do automatically?

3.3 How It Works

Signals has three main components:

- The parser
- The generators
- The templates

3.3.1 Parser

The parser reads your JSON schema file and translates those JSON objects into Python objects. It creates two main types of objects DataObject``s, which include the fields, relationships, and structure of the data your API sends, and ``URL``s with specific ``API endpoints associated.

3.3.2 Generators

By and large, each generator should correspond to a front-end platform (e.g., iOS, Android, AngularJS). TODO: Better explanation of what a generator does that is separate from what the templates do

3.3.3 Templates

We use Jinja, the popular Python templating engine, to produce the actual model definitions and code to make server requests. A single generator could have multiple template styles. For example, the iOS generator could have both Objective-C and Swift templates, an AngularJS generator could have templates for `$http`, `$resource`, and `restangular`.

3.4 Writing Your Own Generator

3.5 Writing Your Own Template

3.6 Contributing

Signals is an open-source project and we welcome suggestions, questions, and code.

Report bugs, issues, and feature requests to the Github issue tracker.

3.6.1 Recommended Setup

1. Fork and clone your own copy of the Signals Github repo
2. Create a new python virtual environment
3. `pip install -r requirements.txt`
4. `pip install -r dev-requirements.txt`

3.6.2 Run Tests

- To run the tests, just run `nosetests`.

3.6.3 Submitting Changes

- **Open a pull request from your Github repo to `/yeti/signals/master`**
  - Provide an overview of your changes
  - Specify any issues (by number) that your PR addresses
- PRs won't be merged if they break tests
- Feel free to ask for help or guidance!