django-wibses Documentation

Release 0.1

Wojciech Krzystek, Yaroslav Machkivskiy

January 31, 2014

Contents

django-wibses is a webapp simplifying creation and management of *semantic scripts* (it's basically a complex, structured JSON).

It comprises of a RESTful backend written in Django, which utilizes pydic and a rich-client frontend written in AngularJS.

django-wibses can be used as a standard django application, additionally it provides lightweight command-line execution wrapper.

btw: Wibses stands for **W**eb **I**nterface for **B**uilding **SE**mantic **S**cripts.

CHAPTER 1

Technologies used

- Django 1.6, Python 2.7
- AngularJS 1.2.X, Angular-UI, CoffeeScript
- Yeoman, Grunt

CHAPTER 2

Contents

2.1 Installation guide

2.1.1 2 available run modes

1 - Command line

Not available yet

starts a lightweight web server

2 - Django application

TODO vucalur: write about setting up a sample dajngo site

1. Run the script, which assembles the frontend and copies static resources to appropriate locations in django project:

   ```
   $ cd django-wibses
   $ ./prepare_dist.sh
   ```

2. Start the django server.

   The application, fully hosted by sole django server, will be available under http://localhost:8000/wibses (Change the port number if you don't use django's default `8000`)

2.2 Project setup guide - for Developers

2.2.1 Prerequisites

- Project is developed under GNU/Linux. All used tools work also on MacOS and Windows.
- Project is developed under PyCharm 3.X. (**Make sure you are using** JetBrains Codestyle **to indent your code.**)
- Here are packages for *buntu 13.10 64 bit. Install their equivalents on the OS of your choice:
  - **General**: `bash-completion git ubuntu-restricted-extras meld`
  - **Database**: `sqlite libsqlite3-dev`

- **Node.JS**: `npm nodejs (sudo add-apt-repository -y ppa:chris-lea/node.js)`
- **Python 2.7**: `python python-gpgme python-software-properties python-pip python-sphinx python-dev`
- **Other**: `ruby-compass ruby1.9.1`

2.2.2 Step-by-step setup guide

1. Get the source code from https://github.com/vucalur/django-wibses and navigate to the download directory

   ```
   $ git clone https://github.com/vucalur/django-wibses
   $ cd django-wibses
   ```

2. Install required python packages by running:

   ```
   $ (sudo) pip install -r requirements.txt
   ```

3. Prepare dictionary repository - TODO taipsedog

   https://pydic.readthedocs.org/en/latest/Introduction.html#preparing-a-pydic-dictionary

4. Add django-wibses to your django site:

   ```
   INSTALLED_APPS = (
      ...
      'wibses',
      'wibses.data_store',
      'wibses.py_dict'
   )
   ```

   TODO taipsedog: No 'wibses.data_store' and 'wibses.py_dict' - importing only 'wibses' shall do the trick

   rst reference: http://sphinx-doc.org/rest.html

5. Set wibses-related Django settings

   TODO taipsedog

   Sample - do this similarly to: http://django-getpaid.readthedocs.org/en/latest/installation.html#enabling-django-application http://django-getpaid.readthedocs.org/en/latest/settings.html

6. Run the backend server

   ```
   $ python manage.py runserver
   ```

   running from PyCharm is advised though

7. Navigate to `wibses/yo` and download dependencies:

   ```
   $ cd wibses/yo
   $ npm install
   $ bower install
   ```

7. Sitll inside `wibses/yo` run the frontend development server:

   ```
   $ grunt serve
   ```

   It should open the browser automatically.

2.3 Contributing guidelines

2.3.1 Git Workflow

- We use simplest possible rebase workflow based on this.
- Reading whole Chapter 3 is strongly encouraged.
- Do not even try invoking `$ git pull` or committing 3-way-merge crap like `Merge branch 'master' of github.com: blah blah blah` :-)

  3-way-merges obfuscate history and screw annotations in IDE - Existing code that you are merging in gets annotated with your name, even if you aren't the author.

Cheatsheet - Rebase Workflow

Make plain old local commmits of your work to the master branch:

```
$ # git pull    # !!!!!!!!!!!   DO NOT EVEN TRY   !!!!!!!!!!!
$ git fetch   # Keep up with recent changes before begining work.
...
$ git commit -m '[#123] Implemented a mechanism to make "blah blah blah" sound wise'  # commit your
```

Some advice:

- Use `git commit --amend`. It's more reliable and faster than local history in IDE.
- If you have a tendency to break down single unit of work into multiple commits locally, remember to squash them before submitting to repo.

Now, synchronize with repo:

```
$ git checkout master   # make sure you are on master branch
$ git fetch  # update origin/master with the latest changes from repo. It's safe = No conflicts here,
$ git rebase origin/master    # Place your local commits on top of commits from repo, that you just
   # Supposing you have a merge:
      # 1. Resolve conflicts by editing conflicted files
      $ git add <<conflicted_files_here__space_separated>>    # 2. Mark conflicted files as resolved
      $ git rebase --continue  # 3.

# At this point you have local history in-sync with repo
# Now you can submit your code with plain old push:
$ git push
```

Note: `fetch` & `rebase` can be replaced with `$ git pull --rebase`.

For more information what's happening here, refer to Rebasing subchapter of ProGit.

2.3.2 Indentation

- Project is developed under PyCharm 3.X.
- Make sure you are using JetBrains Codestyle to indent your code.
- Some files should not be formatted - check what you're committing.

- Warning: PyCharm's code formatter tends to leave parts of CoffeeScript code unindented or screw CS indentation at all. Beware.

2.3.3 Code Analysis

- lint your (CoffeelJava)Script. Linting is done in default grunt task:

```
$ grunt
```

- Feel free to ask for a code-review

2.3.4 CI

Unit tests

Unit tests are executed after each commit by Travis-CI.

They can be executed locally by running one of following commands:

- `$ grunt`
- `$ grunt test`
- `$ grunt test:unit`

E2E tests

End-to-end test can be executed only locally due to limitations of grunt-protractor-travis combination.

Historical note: Previously ngScenario was the framework used for e2e testing. Back then e2e test were also executed by Travis-CI. We have decided to switch to Protractor as advised by Angular documentation (ngScenario was becoming deprecated). Due to lack of good support for grunt-protractor-travis combination e2e test are executed only locally. We hope that good integration will be available soon.

In short: It's each developer's responsibility to make sure e2e tests pass before committing.

Running e2e tests

- Navigate to `yo` subdirectory
- Download the Protractor dependencies:

```
$ ./node_modules/protractor/bin/webdriver-manager update
```

- Start the Selenium server:

```
$ ./node_modules/protractor/bin/webdriver-manager start
```

- Start backend (django) server if your tests rely on backend and it's not being mocked
- Start the frontend server:

```
$ grunt serve
```

- Run Protractor:

```
$ ./node_modules/protractor/bin/protractor protractor-config.js
```

Debugging e2e tests

You may find this helpful

2.3.5 Commit messages

- Be precise, concise and meaningful
- Use Git Commit Guidelines from AngularJS project

  We use following *types* (Additional **concept** type compared to the original):

  - **feat** : A new feature
  - **fix** : A bug fix
  - **docs** : Documentation only changes
  - **style** : Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)
  - **refactor** : A code change that neither fixes a bug or adds a feature.
  - **perf** : A code change that improves performance
  - **concept** : Change of concept, both major and minor. Major ones shall be described in an issue: https://github.com/vucalur/django-wibses/issues.
  - **test** : Adding missing tests
  - **chore** : Changes to the build process or auxiliary tools and libraries such as documentation generation. Also bumping library version.
- Whenever there is an issue (aka ticket) created for what you are working on, reference it in a commit message, like:

  ```
  feat(blah): #123 Implemented a mechanism to make "blah blah blah" sound wise
  ```

2.3.6 Python

- Whenever introducing dependency on a new python module make sure you change `requirements.txt` accordingly

2.4 Developer's Corner - known work-arounds

2.4.1 Installing beta/RC dependency version with bower

```
$ bower install angular-cookies --save
```

It will in fact put the latest *stable* version in bower.json, even if you select otherwise, hence next steps:

1. open `bower.json`
2. manually change version of the new dependency to the beta/RC version

3. download the beta/RC version:

```
$ bower update  # to actually fetch manually changed version
$ grunt bower-install # to inject to index.html
```

The last one might not inject stuff properly, even if invoked a couple of times. In such case you will have to inject stuff manually to the `index.html`.

CHAPTER 3

Authors

Developers, Architects:

- Wojciech Krzystek (vucalur)
- Yaroslav Machkivskiy (taipsedog)

Customer, mentoring:

- Krzysztof Dorosz (cypreess)

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*