Cloudmeb

Cloudmeb is a django application designed to allow the management and configuration of the web application, its users, partners and offered solutions, etc.

Project architecture and structure

The django project directory is located at `cloudmeb`. Its applications/modules are located under the project at `cloudmeb/<app name>/`

> *This is not a default `Django` project structure, but makes sense as the project modules are mostly dependent on one another.*

The websites module

The website model class defines the configuration and integration information used to connect/identify various services.

The users module

The users model class inherits from the `AbstractBaseUser` and `PermissionsMixin` classes.

The partners module

The partners model class inherits both from the `User` and `Page`

> *Inheriting from the page class was done to allow exposing the partner on the web at a later time. This module is not extremely useful, as the client wished to be the partner of all solutions.*

The solutions module

This module is used to hold the common schema of both the `Products` and the `Services`, it also defines the intermediate tables for mapping to a collection of `Category`, `Value` and `Benefit`.

The benefits module

The benefit model class is used to express a collection of benefits tied to a `solution` object.

The products module

The product model class inherits from the `Solution`, `Price`, and `Page` classes.

The service module

The service model class inherits from the `Solution` and `Page` classes.

The partner_services module

The partner_services model class inherits from the `Price` class and maps the service and partner together.

The prices module

The price model class is used to hold a price evaluation formula which takes its context from a collection of `Input`.

> *Note: The formula is evaluated at runtime on the client side. The users input value is injected into the formula to evaluate the price.*

The inputs module

The input model class is a representation of a form input, this object is used to render a form component to capture the context of a price formula evaluation.

The values module

The value model class defines a relation to an `Input` object and can relate to another value instance, thus allowing the filtering of `Input Value` collection based on a user's selection/input.

> *Note: This mechenism is no longer being used by the client but was planned as part of the initial `SOW`.*

The pages module

The page model class defines a generic page object. The view is where the route functions are defined.

The seos module

The seo model class defines the common seo values that a `Page` inherits.

The testimonials module

The testimonial model defines a user or a partner testimonial and maps to a collection of `Solution`.

Technologies used

This is a very high level section about some of the technologies that were used to implement this solution.

- Python 3
- Django
- uWSGI
- Nginx
- Bootstrap
- jQuery

> *Explore the project to find out more.*

Development set-up

This section aims to provide information in regards to setting up your development environment.

System requirements

- Virtualbox
- Vagrant
- Git

Deploying locally under Vagrant

Install above-mentioned system requirements.

> *This setup leverages the virtualization of a near production environment ensuring the integrity of the environment, minimizing the host dependencies/requirements and so on. See vagrant's documentation for further information.*

Clone the repo locally

```
$ git clone https://bitbucket.org/mathieu.benard/cloudmeb.g
it
```

Bootstrapping the environment:

```
$ cd <repo path>/conf
$ vagrant up
```

> *A bash provisioning script will be ran on the vm which takes care of bootstraping the environment.*

> *Take a look at the `conf/Vagrantfile` and `conf/install.sh` for more information on the configuration. Be curious and follow the rainbow.*

Configure your host to listen to `cloudmeb.com` and `www.cloudmeb.com` and redirect the requests to `192.168.10.10`:

```
$ sudo vim /etc/hosts
```

> *Add the following entry:* `192.168.10.10 cloudmeb.com www.cloudmeb.com`

Developing the app

The host repository is mounted and synced with the guest allowing you to code directly on your host machine.

Considerations

This section aims at providing additional insight to any developer working on this project.

Python

The Python version used in this stack is `2.4`.

Django

The current django version used in this project is 1.8 which is an `LTS` release. We may want to consider upgrading before the next long time support release which is planned for `2018`.

Static assets

The static assets are located at the root of the project `assets` directory.

These assets are then collected and served from the `static` directory in production.

> *See the staticfiles for further information.*

Database engine

The current engine is SQLite.

> *As the application grows and the traffic increases, consider changing the database engine to postgreSQL.*

Sensitive information

Some sensitive information are stored in the database and accessible through the admin interface. Due to its sensitive nature, this information is being crypted using `AES` cyphering algorithm and stored `base64` encoded.

> *This information can then be decripted at runtime and used where required.*

When deploying the solution, the developer should consider generating a new secret key using the following code from a Python interpreter:

```
import os

BLOCK_SIZE = 32
SECRET = os.urandom(BLOCK_SIZE)
```

> *The secret should then be added to the following script:* `cloudmeb/utils/crypt.py`.

Adding a page

When adding a new page, the developer should also add an entry in the `sitemaps.py`. Additionally, it will be required to add the page name to the `PAGES` constant defined in `pages/models.py`.

> *Note: The page name should be exactly the same as the one defined in the routes, as it will be slugified to represent its url.*

> *Note: Don't forget to migrate your schema changes.*

```
$(venv) python manage.py makemigrations <app name>
$(venv) python manage.py migrate
```

API

Some work was started in regards to making some of the information accessible via a `REST API`. The API is currently not being leveraged and still requires development.

> *Head over to the tastypie documentation page and read the docs.*

Internationalization and localization

The web application is available to its users in both `French` and `English`. The core mechanism is implemented leveraging GNU's gettext which is first class in djnago's i18n.

> *The po/mo files are outputed at `locale/<language>/LC_MESSAGES/`.*

> *The English messages can be redefined in order to override the ones defined in the source code.*

> *Some of the internationalized strings are stored directly in the database for convenience of not having to restart the uWSGI process each time the strings require maintenance. Some utility functions are available in the following module: `cloudmeb/utils/i18n.py`.*

Caching

There is curently no database, or template caching in place.

Production

The current production instance is running on a `trusty64` rackspace cloud instance. Its ip is `104.130.7.106`. Additionally, a staging environment is alive at `104.130.26.146`.

Deploying to staging

Open an `SSH` connection to the environment `104.130.26.146`, pull your changes, run the `install.sh` script located under the `v1.1-dev` branch, and restart the `uWSGI` process.

> *This process should be revised. The `install.sh` script could accept arguments to customize the target env and perhaps rewrite some of the files. Additionally we may want to move to a more mature orchestration technique/framework such as ansible. Bash was a quick win at the time.*

> *See below section for more information, the process is similar. Use your best jugement.*

Deploying to production

Open an `SSH` connection to the environment `104.130.7.106`, pull your changes, run the `install.sh` script as `user:cloudmeb` located on the `master` branch, and restart the `uWSGI` process running in the virtual environment.

```
$ ssh cloudmeb@104.130.7.106
$ cd ~/cloudmeb
$ git pull
$ sudo bash install.sh
$ source venv/bin/activate
$(vnev) uwsgi --stop /tmp/cloudmeb.pid
$(venv) uwsgi --ini conf/cloudmeb.ini --socket :8001
```

> *You may need to forcefully kill this process:* `sudo killall -9 uwsgi`.

> *You may want to make a db dump first.*