Zeus install guide Documentation

Release 0.1b

Kostas Papadimitriou <kpap@grnet.gr>

Oct 01, 2018

Contents

1. Create a customized config file:

```
$ cp deploy/config.yaml.example deploy/config.yaml
$ vim deploy/config.yaml
```

2. Build docker image:

```
$ docker build -t grnet/zeus .
```

3. Create docker container:

```
$ docker create -P \
    -v `pwd`/deploy/config.yaml:/etc/puppet/hieraconf/common.yaml \
    -p 8000:8000 \
    --name zeus \
    grnet/zeus
```

4. Start docker container:

```
$ docker start zeus
$ docker logs -f zeus
```

5. Wait for a few seconds for initialization scripts to complete and login as administrator using credentials set in *config.yaml*:

```
$ open https://127.0.0.1:8000/zeus/auth/auth/login
```

CHAPTER 1

Development

1.1 I18N

Zeus is based on Django and thus makes use of Django i18n tools for the translation of messages across the user interface. The few cases where we deviate from the common Django i18n process are described below.

It is recommended to have a good read through Django documentation regarding i18n in advance of the following guidelines.

https://docs.djangoproject.com/en/1.7/topics/i18n/translation/

1.1.1 Application settings

To enable additional languages or change the default language used at your deployment you should modify the following settings in your *local_settings.py* module accordingly:

```
LANGUAGE_CODE = 'en-us'
LANGUAGES = (('en', 'English'), ('el', 'Greek'), ('<lang-code>', 'My Language'))
```

1.1.2 Add or update I18N messages

To add or update a new or existing language translation the following steps should be performed,

1. Create django.po mapping for the specified language. Notice that as Django keeps translation mappings separately for each django app you should run the following command once in the context of each app used by zeus. *zeus*, *zeus_forum*, *helios*, *heliosauth*, *server_ui*, *account_administration*:

   ```
   $ cd <appname>; django-admin makemessages --no-location -l <lang-code> -e .html -
   ↪e .txt
   ```

2. The command above scans for I18N messagess across the code of the app and creates proper *gettext* django.po catalogues in *<app_name>/locale/<lang-code>/LC_MESSAGES/django.po* files. It also handles updated/new/modified messages. Once the catalogue files are created you can edit them using your favourite text editor. Each message translation is formed in a <message-id>/<translation> two line tuple. *message-id* is usually the message in English or a string which denotes the context of the message (booth messages) and should not be manually modified.

3. Once messages are translated, you should compile catalogues and restart your zeus instance for updated messages to be displayed. Compile process also validates that translated messages are formated properly:

```
$ cd <appname>; django-admin compilemessages;
```

4. For booth messages to be updated a different command is required. Those messages are handled by javascript and therefore should be placed in gettext catalogues generated specifically for this case:

```
$ python manage.py makeboothmessages --no-location -l <lang-code> -e .html -e .js
$ cd zeus/static/booth && django-admin compilemessages
```

5. The commands above should cover the translation of messages displayed in admins, voters and trustees views. Additional localized content is exposed on site views such as */*, */faq*, */terms* etc. The content of these pages is not included in gettext i18n catalogues as it is *zeus.grnet.gr* service specific. You can modify the texts by overriding the corresponding templates used by Django to render the content of the page.

    - Home (index) view

      helios/templates/i18n/<lang-code>/zeus/home.html

    - FAQ

      helios/templates/i18n/<lang-code>/zeus/faqs.html

    - Contact

      helios/templates/i18n/<lang-code>/zeus/contact.html

    - Terms

      terms/terms_<lang-code>.html