DARIAH Contrib Tool - Deployment

2016-10-18

Dirk Roorda

Server

`tclarin11.dans.knaw.nl`

Url

`dariah-beta.dans.knaw.nl`

Database

Mongodb via pymongo.

Web-app

We use Bottle, a Python3 micro framework to route urls to functions that perform requests and return responses.
The webserver is httpd (Apache). Bottle connects to it through mod_wsgi (take care to use a version that speaks Python3).
See **Prerequisites** below.

The connection is defined in the default config file (for contents, see *default_example.conf* in the github repo):

```
/etc/httpd/config.d/ :
    default.conf
    shib.conf
    ...
```

Location

```
/opt
    shibboleth
    web-apps
        dariah
            server
                app.py:  routes and controllers
                data.py: json data from mongodb
                default_example.conf: example config file for Apache
httpd server
                serve.py: wsgi entry-point for apache
                serve.sh: local development server
            static  (static files, css, javascript, fonts, etc)
                css: stylesheets built from src
                js: javascript built from src
                favicons
```

```
                images
                fonts
                docs
                    deploy.pdf: notes on deploying this web app
                tools
                    update.sh: script deploy updates of the web app.
Pulls code from the github repo, restarts httpd.
                    Jupyter notebooks for legacy data conversion, not
used in web-scenario, except as documentation
            client
                node_modules: javascript dependencies
                package.json: npm config file
                README.md: short description for humans
                gulpfile.babel.js: config file for gulp, the build tool
                gulp_dev.sh: script for development builds
                gulp_prod.sh: script for production builds
                index.html: html entry-point for the client side app
                src
                    css
                        *.scss, *.css (plain CSS and SASS
stylesheets)
                    js
                        components
                            *.jsx: client-side code in jsx
                        helpers
                            *.js: client-side code and data in js
                        main.jsx: client-side entry-point for the
javascript
                    css
```

Prerequisites

Python can be installed by means of the package manager:

```
yum install python34
```

On a strict system, like SELinux, you can install Python3 and the extra modules needed by means of **yum install ...**
However, some of these modules end up in the Python2 framework, so I had to use **pip3**. On a strict system, you have to build pip3 first! On SELinux, this worked

```
sudo yum install python34-setuptools
sudo easy_install-3.4 pip
```

Then you can say

```
sudo pip3 install pymongo
```

```
sudo pip3 install bson
```

In order to run python3 in the webserver, I followed the mod_wsgi guide.
As preliminaries I had to install devel versions of apache and python3 first

```
yum install httpd-devel
yum install python34-devel
```

Then I downloaded the mod_wsgi source code (version 4.5.7), untarred it, and configured it with whatever python3 I found on the path.

```
cd mod_wsgi-4.5.7
./configure --with-python=/bin/python3
```

Then

```
make
sudo make install
```

After this, httpd works with python3.
The website runs with SELinux enforced, and also the updating process works.