Quiz Section Project: URL Validation and Testing

Introduction:

This document describe the I) what is implemented in the project, II) instructions to run the project, III) design choice and IV) exceptions that were made for the project. The project is located at "git@github.com:hunlan/mycsesortingoption.git"

I. What is in the Project:

This project is a django web application that takes in an input file, a sorting algorithm, and whether to validate URLs and then sort the urls.

In addition, this project included a django python script (checkURL.py) that reads from a file a list of URLs and displays 1) source URL, 2) is url valid, 3) canonical url, 4) is url source unique, and 5) is url canonical unique in read order. This script uses files located in "urlutils" package

II. Running the Project:

1) checkout the project via command line:

```
$ git clone git@github.com:hunlan/mycsesortingoption.git
```

2) enter the top-level directory of the project:

```
$ cd mycsesortingoption/
```

3) download virtualenv from (http://pypi.python.org/pypi/virtualenv) if you haven't already

4) create a virtual environment on top-level directory

```
$  virtualenv venv –distribute
```

5) source into the virtual environment and download django dependencies (download pip from http://pypi.python.org/pypi/pip), this will take a moment

```
$ source venv/bin/activate
$ pip install Django psycopg2 dj-database-url
```

6) at this point, you have django setup, and you can run the web application and the script via

```
$ python manage.py runserver
$ python checkURL.py <INPUT FILE>
```

7) to run the django unit test, it will require a postgres database, so download setup a postgres database.
Download linke (recommand v9.1): http://www.postgresql.org/download/
Installation instruction: http://www.postgresql.org/docs/9.1/static/installation.html

8) create a database with the name: 'postgres://postgres@localhost:5432/section'
basically on the last two lines of this: http://www.postgresql.org/docs/9.1/static/install-short.html
name the database 'section' instead of 'test'

9) run the database on command line:

```
$ sudo su – postgres
$ /usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data >logfile 2>&1 &
```

10) Now you are done, you can run the unit tests by invoking "python manage.py test" on top-level

<u>III. Design Choices</u>

<u>UrlValidator:</u>
UrlValidator (sectionproject>urlutils>urlvalidator>urlvalidator.py) is a class that validates whether a url is a valid url or not. In addition, this class will store the parts of the url into fields as it validates a url. In contrast, I could have decoupled the two functions by having one class that validates url and one class that tokenize url. However, I chose to tokenize a url during the validation process because the way I validate a url requires breaking the url into the various parts of a url, and I figure might as well store those parts into fields of the class instead of doing the same thing twice.

The validator validates the various parts of an url which are: scheme, username-password, host-name, port, path, query, and fragment. All parts of an url must be either 0-9, a-z, A-Z,"-", "_" unless specified (not including separators such as "?" for query and "#" for fragment). The specification of how each parts are validated are described below:

Scheme:
Scheme of an url is validated by the method _isSchemeNameValid. Scheme is optional. The only valid scheme are "http", "https", and "ftp". In addition, the scheme must follow by "://".

Username-Password:
Username-password of an url is validated by the method _isUserPasswordValid. Username-password is optional. A username-password is valid if 1) it ends with "@", 2) there exist a colon between username and password, and 3) username and password are not empty strings.

Host-Name:
Host-name of an url is validated by the method _isDomainNameValid or _checkAndRemoveIPv4. Currently, UrlValidator only accepts hostname or IPv4 format (and does not support IPv6). A host-name is required in an url. Firstly, the type of host-name (either domain name or IPv4) are determined by the number of "period" in the host-name.

Depending on which host-name type, different validation method is invoked. For the case of domain name, _isDomainNameValid is invoked. A valid domain name is one with one to two periods with an extension listed here "http://en.wikipedia.org/wiki/List_of_Internet_top-level_domains". The valid extension do *not* include countries extensions at this moment.

For the case of IPv4, _checkAndRemoveIPv4 is invoked. A valid IPv4 is one with three periods which integer values between 0-255 between each period. Currently, local mappings to IPv4 are not supported (e.g. localhost is not supported)

Port:
Port of an url is validated by the method _isPortNumberValid. Port number is optional. A port number is valid if there is a colon pior to the port number and the port number is a 16 bit integer.

Path:
Path of an url is validated by the method _isPathValid. Path is optional. A path is valid if it is a combination of 0-9, a-z, A-Z, "-", "_", "/", "./", "../". Inaddition, a path can include percent encoded symbols which is defined by a percent sign followed by hex number. Currently, all 256 ascii characters are supported as valid percent encoded symbols.

Query:
Query of an url is validated by the method _isQueryValid. Query is optional. A question mark must exist pior to a query. A query is valid if it has zero or more key-value pairs. A valid key-value pair has a key that does not start with a number and the key and value are separted by an equal sign. In addition, each key-value pair is separated among other key-value pairs by either a semi-colon or the and-percent sign.

Fragment:
Fragment of an url is validated by the method _isFragmentValid. Fragment is optional. A hash symbol must exist pior to a fragment. A fragment is valid if it is one of the following characters: a-z, A-Z, 0-9, “-”, “_”, “=”, “&”, “;”.

<u>UrlCanonicalizer:</u>
UrlCanonicalizer (sectionproject>urlutils>urlcanonicalizer>urlcanonicalizer.py) is a class that normalizes an url. UrlCanonicalizer takes in either a url string or a UrlValidator object and returns a normalized url string. The cononicalize rules are as follows:

1) Host-name is lower-cased
2) Percent symbols in a url path are decoded to ascii character
3) Port is removed
4) Username-password is removed
5) A trailing slash is added at the end of path
6) Dot-segments (such as “./” and “../”) are translated and removed (e.g. nba.com/./ → nba.com/ and nba.com/top-ten/../ → nba.com/)
7) Fragment is removed
8) Duplicated slashes in url path are removed
9) “www.” prefix of domain name is removed
10) Query key-value pairs are sorted by keys in alphabetical order (case sensitive)
11) Empty query is removed

<u>UrlComparator:</u>
UrlComparator (sectionproject>urlutils>urlcomparator>urlcomparator.py) is a class that compares two url in terms of source or canonical form or determines whether a url is source or cononical unique amoung other urls. Source url comparison compares two url base on the raw value of an url. Canonical url comparisons first converts a url to a canonical form then compares the canonical urls. Note that the latter comparison requires the url to be a valid url.

IV. Exceptions

Due to time and resource limitations, the url validation process is not perfect. The exceptions listed below are mostly edge cases that are not handled.

UrlValidator

1) UrlValidator only accepts url without scheme or url with scheme that are either http, https, or ftp
2) UrlValidator do not accept domain name with countries extension. (It is easy to add, but too many to type...)
3) UrlValidator only accepts domain name or IPv4 as host-names and do not handle the newer IPv6 host-names
4) UrlValidator accepts all types of percent encoding symbols where some of them might not be valid for an url

UrlCanonicalizer

1) Choice of normalization might not be the perfect standardize way to normalize an url.

UrlComparator

1) Implemented a method for comparison instead of overriding the <, >, and == operator