This file is just notes for now! TODO is turn this into a usable document…

A great web site for basic installation instructions for Selenium for Python is here:

<https://selenium-python.readthedocs.org/en/latest/installation.html>

I could not get the proxy configured (even with cntlm) to work right, but if you check the pip.log file, you can see where you can then just manually download the tar.gz files needed (tbd record versions used) and then use “pip install <path-to-tar.gz-file>”. There are only two packages needed, so not a lot of dependencies.

In firefox, install the Selenium plugin (version?).

Start the plugin and it will be by default be in record mode. Do something really simple and record it. Export it as a Python 2 test case.

This gives the example of what a Python test case looks like, and you can use this test case in the above installation instructions to test that you installed Selenium into Python correctly.

Some documentation on the Selenium python interface is here:

<http://selenium.googlecode.com/svn/trunk/docs/api/py/api.html>

So activate the pyenv environment

C:\seltest\selenv>Scripts\activate

Can then run tests via command line:

Scripts\python.exe ..\Test1.py

Or start up eclipse:

(selenv) C:\seltest\selenv>c:\Apps\eclipseJuno\pythonEclipse\eclipse.exe

* Open a shell and activate your CKAN pydev environment (e.g. . ~/pyenv/bin/activate).
* Now start eclipse.
* Add a Python Interpreter: Window/Preferences/PyDev/Intterpreter - Python:
* Create a New Interpreter entry: CKAN python as name and the executable should be: ~/pyenv/bin/python
* There are a bunch of libraries that have not yet been added to the pythonpath. Add all libraries.
* Create a new pydev project: File/New/ PyDev Project.
* Call your project "PyNGDSSeleniumTests".
* Uncheck the "Use default" box and enter the path to your project location
* Choose the newly created python interpreter for the project
* Click on "Add source folder" and add “..\ckanext\tests\ui” Click on one of the unit tests and run as Python Unit test

The main concept is that a set of functions are being written in Python for navigation and common actions in the web site. These actions are then combined into tests and verification is done by expecting certain text to appear and certain items to be available on the web page.

As the UI evolves, some of these core functions will of course have to be updated. The goal though is to write the core functions as robustly as possible (including adding ids/names, etc to the HTML to assist with this robustness), and then to have the tests use the core functions as must as possible, so that any changes to the UI result in very few places that the tests then have to be updated.