Summary

I have implemented a very simple Django-based web application that allows asynchronous maintenance of hardware status, and provides HTML access to this information, automatically refreshed with AJAX.

So this is asynchronous in two aspects: firstly, the hardware access to maintain up-to-date status in the DataBase is separate from the server process; and secondly, the refreshing of the specific status sections of the HTML page is asynchronous with AJAX. The first layer of asynchronous hardware access can be disabled by setting `use_direct_access=True` in `views.py`.

The asynchronous aspect of the hardware querying is important to avoid the need for any HTML request from a client blocking on server IO. It also allows the IO to be conducted efficiently, doing system checks and file opens only when required rather than on each request, while also allowing the possibility for the HTML interface to be RESTful (RESTfulness is not guaranteed here, but is made more easy by the separation of concerns (splitting hardware access and HTML serving).

The system implemented is much less clean than I'd like, and far from perfect, but unfortunately I have been travelling through England for the last 10 days and so had to implement this on a netbook with Vim and Bash inside a VirtualBox image of Fedora. I have endeavoured to make the code as portable as possible, and believe that the majority of it should work on all Linux systems offering the basic ACPI interface, but this has not been tested.

I ran out of time to implement the Wifi status information unfortunately, but once the hardware access layer is implemented (using a package such as `scapy`), the addition of this status to the `views.py` is straight-forward.

In implanting this, I cloned the python-acpi project and have added functionality to it.

Given more time, I would have liked to:

- Implement wifi status
- improve the graphical interface and define the schema for the HTML to allow not just for human readability but also for easy machine processing
- test properly on different systems (and implement low-level /proc access for older systems)
- implement graphical display

The task:

Build a web application (using django / or any other python web framework) which shows the battery status and the current available wifi networks.

Target platform: Ubuntu Linux 12.04 or higher

A) Battery view:

HTML page to show current capacity of laptop's battery as time and percentage.

Bonuses:

a. update view using ajax every 2 minutes.
b. graphical indicator showing battery status.

B) WiFi view:

HTML page to show currently available WiFi networks.

Bonuses:

a. update view using ajax every 5 seconds

Notes

Steps:

1. Establish web application capable of serving dynamically generated content from Python
    a. Get Django site up and running with a single battstat app [done].
    b. Add a view to battstat app for status [done]
    c. Add dynamic generation of content from Python for this view [done]
    d. Add asynchronous updating of database in background [done]
2. Code determination of battery status [done]
3. Code determination of WiFi network status
4. Add AJAX updating [done]
5. Add graphical indicator

The low-level system functions for retrieving battery and wifi status are likely to be distinct between distributions. The web application framework will not be. Therefore, it should be reasonable to begin the programming of the web framework in Fedora and then re-partition old laptop and install Ubuntu natively to test full functionality.

Working in Fedora 20 VirtualBox

Tested also on Fedora 19 native install.

Installed django 1.5.5-2.fc20 from system repositories

Ran server listening on all interfaces inside the VM:

```
python manage.py runserver 0.0.0.0:8000
```

Configure VirtualBox with port forwarding for 8000:

```
VBoxManage.exe modifyvm "fedora20kde_x64 Clone" --natpf1
"django,tcp,,8000,,8000"
```