Final CA Report

Tim O’ Mahony – t00152281

About

Script asks user to input for a list a fields. These fields and entries are saved to interoperable text files that the user is free to read, modify and delete with whatever application or system of they’re choosing. When launched in daemon mode the script runs a flask web server which hosts the status of the tickets in html with some css formatting.

Usage

akrt.py new - launch new ticket,

akrt.py update [woid] - update previous ticket

akrt.py about [woid] - read ticket file

akrt.py daemon - launch web server

Rationale

In my place of work I use a php/mysql repair tracker that overloads the underpowered server that hosts it with design features we never use. It’s also meant to host pages that inform the customer of the repair status but that feature has never worked.

I was then inspired by this https://joearms.github.io/2014/06/25/minimal-viable-program.html.

The writer discusses a very simple ticket system written in erlang by himself and another co-worker in 1985. So I tried to challenge myself to write a very simple repair tracker with the missing feature.

Hosting

Repository can be found at:

https://github.com/cythecylon/akrt/tree/submission

Late into project I abandoned the idea of checkouts and checked out into an earlier working version. From that I created a new branch called **submission** focusing on taking and updating tickets. Instead I chose just to show the status of the ticket and no personal details.

Modules Used

os.dir	Get current working directory for creating path to text files. Sys
Sys	Looks for arguments when programs are being launched
Pathlib	Used to create path to write text files if not already created
Time	For adding a time stamp to reports
Flask	For create a http server and display status in a html/css template

Methods Written

For clarification woid refers to work order id.

Main

If the module is not being imported. The module will be called __main__ by the interpreter. And if this case main() will be run with the user's arguments as parameters.

Main will try to check if these parameters are either new, update, daemon or about. If they're anything else or out of bounds the script will print it's usage.

Newwo(woid)

Creates a new text file using a path generated by genwoid(). Requires Python 3.6.1 for Pathlib.

Writes a time and date while asking user for input for each field on the script.

Genwoid()

Invoked by newwo().

Updatewo(woid)

Appends the text files. Asking the user to enter a status between 0 and 4 and a description of the device's current status along with the time the report was created.

Index()

Under @app.route("/"). Whenver localhost:5000 is accessed this method will run. Essentially the index page.

Returns string of text telling user to type woid in taskbar

Documentation Source: http://flask.pocoo.org/docs/0.12/quickstart/#routing

Webreport(woid)

Under @app.route("/report/<int:word>"). Whenever localhost:5000/reports/<woid> is accessed this method will be called.

Takes status number from getstatus to reference the statuses dictionary to explain the current status to the customer. Renders this string in a template under templates/report.html.

Documentation source:

http://flask.pocoo.org/docs/0.12/quickstart/#rendering-templates

getwo(woid)

Invoked by main("about <woid>") and getstatus(). This retrieves the values from a text file and returns a string called values.

Getstatus(woid)

Invoked by webreport(). This method returns a repair status number extracted from getwo's string using string splitting.

The method returns the status number of a ticket.

Flask Usage

To use flask I first had to create an instance of it with app=Flask(__name__). So now whenver I refer to app in the program it will be to that running instance of flask. In memory the instance will be called __main__ as that what the name of the script when it's not being run as a module.

When the program is run in daemon mode. As the parameter daemon is passed. It will then run the 'app' flask instance method app.run().

On the web browser if the user goes to the root directory it. The flask instance will search for the route @app.route("/") and find the index() method. Likewise if the user types @app.route("/report/woid") it will run webreport(woid) just under.

Webreport uses render_template. Another module inside flask that has to be import independently. Render_template("report.html",status) uses .html templates under the directory /template.

In this directory a html page contains the offollowing:

```
<h1>Status:</h1>

{% if status %}

<p> {{ status }} </p>

{% else %}

    <p> Unable to find status. Please contact store </p>

{% endif % }
```

If status has a value the page will render the variable passed as status. Other it will tell the user it was unable to find the status.

Conclusions

Flask is a very intuitive and simple micro web framework if you use it simply. Trying to manage my time for other projects. I was unable to get authentication to work as and as work around just displayed pseudonymous statuses. This would have been very interesting.

Additionally if I would have allocated more time. I might have used coder friendly json or human friendly YAML. Or used regular expressions to keep on eye on ticket timing. I may continue to work on this after college to see if I can convince my workplace to use it.