2023/4 AUTOMATION PROJECT

AUTOMATED .CSV REPORT FROM THE SYSTEM'S LOGS

by:
Francesco Manganelli

The Approach

The **procedure** I follow when coding is:

1. **Understand** the problem statement **inputs** and **outputs**
2. **Research** how to solve efficiently the problem. Consider using external modules or Python's existing libraries.
3. **Planning** the **data structures** and the **logic flow**. Create small problems to solve and link their results to complete the task.
4. **Write the code**

The Scenario

A server operates a service called **Ticky**, which assists in managing work and recording syslog events. The developer team wants you to comprehend the syslog and create **reports** that can aid in system improvements.

When Ticky runs properly, it generates an **INFO** log containing **details** about the completed task, along with the **username** and **ticket number** associated with the event.

In case of an issue, it generates an **ERROR** log that describes the problem and identifies the username that triggered it.

They request two reports in **.csv** format:

1. **Ranking of Errors Generated**: a list of the most common errors with the amount of times an error has been triggered. In descending order.
2. **Usage Statistics for the Service**: a list of all the users who have used the service with how much info and errors they generated. Sorted by username, alphabetically.

Visualize the data by converting the **.csv reports** into webpages with the **cv_to_html.py** script. The **.HTML** files need to be located in the server's directory for the webpages.

Planning

The **input** data in **syslog** is structured like:
[Date] [Server] [Service]: [TypeLog] [Event] [Ticket] [Username]

```
May 27 11:45:40 ubuntu.local ticky: INFO: Created ticket [#1234] (username)
Jun  1 11:06:48 ubuntu.local ticky: ERROR: Connection to DB failed (username)
```

The **desired output** data are two **.csv files** containing:

1. **ErrorName - Amount of times triggered**
2. **Username - Amount of Infos - Amount of Errors**

To solve the problem, I will perform the following logic:
1. Firstly, I will **access** the log file and use a for loop to **iterate** through each line.
2. For each line, I will check **if** it contains either "INFO" or "ERROR" using a regular expression (regex) expression.
3. With that information, I will capture and store the information in a dictionary with the format **Error : AmountTriggered**. This will be useful for the first report
4. Additionally, I will create a second dictionary with the format **Username : AmountErrorTriggered, AmountInfoGenerated** to generate the second report.

Worth mentioning that for each ERROR or INFO, the code will check **if it's already** in the dictionary. It will perform a different action depending on that condition, might initialize the value to 1 or increment.

When the dictionaries are correctly generated, they will be **sorted and formatted** in **.csv files**. Then, with the pre-existing script **csv_to_html.py** they will be converted into **.HTML** files.

Testing

Once understood the problem and planned a solution, I proceeded to write the code to **create the dictionaries** that would then be used to generate the .csv file.

```python
import re
import csv

logfile = "syslog.log"
with open(logfile, "r") as file:
    file_logs = file.readlines()

errors_generated = {}
user_info_error_generated = {}

for line in file_logs:
    usernameMatch = re.search(r"\((.+)\)", line)
    username = usernameMatch.group(1) if usernameMatch else None
    error_match = re.search(r"ticky: ERROR ([\w ]+)", line)
    info_match = re.search(r"ticky: INFO ([\w ]+)", line)

    if info_match:  # log line is an INFO
        # add user and increase the amount of infos he has generated
        if username in user_info_error_generated:
            user_info_error_generated[username]["info"] += 1
        else:
            user_info_error_generated[username] = {"info": 1, "error": 0}
    elif error_match:  # log line is an error
        # add the error and increase the occurred times, also increase the errors generated by the user
        error = error_match.group(1)
        if error in errors_generated:
            errors_generated[error] += 1
        else:
            errors_generated[error] = 1

        if username: # In case the user's first log is an error
            if username in user_info_error_generated:
                user_info_error_generated[username]["error"] += 1
            else:
                user_info_error_generated[username] = {"info": 0, "error": 1}
```

To test the code, I printed on the screen the contents of the dictionaries.

```
student-02-a57e3f518e6f@linux-instance:~$ sudo chmod +x csv_to_html.py
student-02-a57e3f518e6f@linux-instance:~$ nano ticky_check.py
student-02-a57e3f518e6f@linux-instance:~$ chmod +x ticky_check.py
student-02-a57e3f518e6f@linux-instance:~$ ./ticky_check.py
I will print the errors generated: {'The ticket was modified while updating ': 9, 'Permission denied while closing tic
': 7, 'Connection to DB failed ': 13}
now I will print the user info and errors: {'mdouglas': {'info': 2, 'error': 3}, 'noel': {'info': 6, 'error': 3}, 'bre
: 2, 'error': 1}, 'mcintosh': {'info': 4, 'error': 3}, 'jackowens': {'info': 2, 'error': 4}, 'oren': {'info': 2, 'erro
enim.non': {'info': 2, 'error': 3}, 'flavia': {'info': 0, 'error': 5}, 'sri': {'info': 2, 'error': 2}, 'nonummy': {'in
0, 'error': 3}, 'kirknixon': {'info': 2, 'error': 1}}
```

I then wrote the code to **sort** and **generate the .csv** file

```python
# Sorting the dictionaries: errors_generated will be in descending order, user_info_error_generated in alphabetic order
user_info_error_generated = dict(sorted(user_info_error_generated.items())) #Alphabetically sorted by default!

errors_generated = dict(sorted(errors_generated.items(), key=lambda item: item[1], reverse=True))

# Writing error_message.csv
with open("error_message.csv", "w", newline="") as error_file:
    fieldnames = ["Error", "Count"]
    writer = csv.DictWriter(error_file, fieldnames=fieldnames)
    writer.writeheader()
    for error, count in errors_generated.items():
        writer.writerow({"Error": error, "Count": count})

# Writing user_statistics.csv
with open("user_statistics.csv", "w", newline="") as user_file:
    fieldnames = ["Username", "INFO", "ERROR"]
    writer = csv.DictWriter(user_file, fieldnames=fieldnames)
    writer.writeheader()
    for username, stats in user_info_error_generated.items():
        writer.writerow({"Username": username, "INFO": stats["info"], "ERROR": stats["error"]})

print("CSV files generated successfully.")
```

To make sure the .csv files are poperly formatted, I checked their contents with the **cat** command.

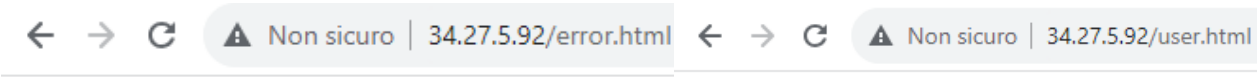
```
student-02-a57e3f518e6f@linux-instance:~$ ls
csv_to_html.py  error_message.csv  syslog.log  ticky_check.py  user_statistics.csv
student-02-a57e3f518e6f@linux-instance:~$ cat error_message.csv
Error,Count
Timeout while retrieving information ,15
Connection to DB failed ,13
Tried to add information to closed ticket ,12
Permission denied while closing ticket ,10
The ticket was modified while updating ,9
Ticket doesn,7
```

```
student-02-a57e3f518e6f@linux-instance:~$ cat user_statistics.csv
Username,INFO Count,ERROR Count
ac,2,2
ahmed.miller,2,4
blossom,2,6
bpacheco,0,2
breee,1,5
britanni,1,1
enim.non,2,3
flavia,0,5
jackowens,2,4
kirknixon,2,1
mai.hendrix,0,3
mcintosh,4,3
mdouglas,2,3
montanap,0,4
noel,6,3
nonummy,2,3
oren,2,7
rr.robinson,2,1
sri,2,2
xlg,0,4
```

With an already existing script, I converted the **.csv files into .HTML**

```
student-02-a57e3f518e6f@linux-instance:~$ sudo chmod  o+w /var/www/html
student-02-a57e3f518e6f@linux-instance:~$ ./csv_to_html.py error_message.csv /var/www/html/error.html
Processing error_message.csv
Table succesfully written to /var/www/html/error.html
student-02-a57e3f518e6f@linux-instance:~$ ./csv_to_html.py user_statistics.csv /var/www/html/user.html
Processing user_statistics.csv
Table succesfully written to /var/www/html/user.html
student-02-a57e3f518e6f@linux-instance:~$ ls /var/www/html
error.html  index.nginx-debian.html  user.html
```

It’s now possible to access and visualize the reports **via browser**.



Error Message

Error	Count
Timeout while retrieving information	15
Connection to DB failed	13
Tried to add information to closed ticket	12
Permission denied while closing ticket	10
The ticket was modified while updating	9
Ticket doesn	7

User Statistics

Username	INFO Count	ERROR Count
ac	2	2
ahmed.miller	2	4
blossom	2	6
bpacheco	0	2
breee	1	5
britanni	1	1
enim.non	2	3
flavia	0	5
jackowens	2	4
kirknixon	2	1
mai.hendrix	0	3
mcintosh	4	3
mdouglas	2	3
montanap	0	4
noel	6	3
nonummy	2	3
oren	2	7
rr.robinson	2	1
sri	2	2
xlg	0	4