Perfbot - Robot Framework Performance Analyser

Perfbot ermittelt Performance-Veränderungen anhand von bestehenden automatisierten UI-Tests. Es erweitert dabei das Robot Framework um die Möglichkeit, Test-Laufzeiten in einer Datenbank zu speichern und mit den archivierten Laufzeiten der Vergangenheit zu vergleichen. Das Ergebnisse der Performance-Analyse werden in die Robot-Testresults (`log.html` / `report.html`) integriert.

Installation

Voraussetzung: `python` und `pip` ist installiert (Mindestversion 3.8 (getestet auf 3.8.10 und 3.10.9))

Repo klonen und folgenden Befehl ausführen:

```
python setup.py install
python3 -m robot --prerebotmodifier
perfbot.perfbot:db_path="example/robot-exec-times.db:"keywordstats="True"
example/test
```

Quickstart

Starten der Robot-Testfälle inkl. Perfbot:

```
robot robot --prerebotmodifier perfbot.perfbot [path to tests]
```

Funktionsweise

Perfbot nutzt den `ResultVisitor` der Robot-API, um über die Tests bzw. deren Ergebnisse zu iterieren und diese in einer Datenbank abzuspeichern. Basierend auf den vergangen Testläufen aus der Datenbank werden die aktuellen Laufzeit der Tests Suite-weise analysiert und das Ergebnis als Metadaten in die Report- bzw. Log-Datei geschrieben. Folgende weitere Funktionen stehen zur Verfügung:

- **Box-Plot** (standardmäßig aktiviert): Zu jedem Testfall wird ein Box-Plot generiert, der die statistische Verteilung der Laufzeiten in Quartile grafisch aufbereit. Die aktuelle Ausführungszeit des Tests wird mit dem Punkt makiert. Die Box-Plot-Erstellung kann aufgrund ihrer teils langlaufenden Erstellung deaktiviert werden (siehe Konfiguration).
- **Testbreaker** (standardmäßig deaktiviert, Aktivierung siehe Konfiguration): Der Testbreaker vergleicht die Testdauer jedes Testfalls mit einer Maximalwert der prozentualen Abweichungen vom Durchschnitt der letzten Läufe. Lässt sich daraus ein Performanzproblem erkennen, so wir der Testfall auf FAIL gesetzt.

- **Keyword-Analyse** (standmäßig aktiviert, in Entwicklung): Neben den Laufzeiten der Testfälle, sind auch die Laufzeiten der darunter liegenden Keywords interessant. Dafür werden auch diese Laufzeiten in der Datenbank gespeichert. Zur Betrachtung der Keyword-Laufzeiten ist die Erzeugung einer seperaten HTML-Datei basierend auf robotmetrics in Entwicklung.

Konfiguration

Starten der Robot-Testfälle inkl. perfbot (standardmäßig mit Box-Plot-Modus und ohne Testbreaker-Modus):

```
robot --prerebotmodifier perfbot.perfbot [path to tests]
```

Angabe, welche Sqlite3-Datenbank mit archiverten Testlaufzeiten genutzt werden soll (standardmäßig wird eine Datenbank mit dem Namen `robot-exec-times.db` erzeugt bzw. verwendet):

```
robot --prerebotmodifier perfbot.perfbot:db_path=[path to sqlite3 file]
[path to tests]
```

Aktivierung des Testbreaker-Modus:
Beispiel: Bei einer Abweichung (`devn`) der Testlaufzeit von 10% vom Durchschnitt der vergangen Testläufe soll der Testfall auf FAIL gesetzt werden. (Hinweis: Perfbot läuft im Rebot-Schritt, die `output.xml` und CLI-Ausgaben werden durch den Testbreaker deshalb nicht verändert)

```
robot --prerebotmodifier perfbot.perfbot:devn=0.1:testbreaker=True [path
to tests]
```

Deaktivierung und Konfiguration des Box-Plot-Modus:
Die Generierung des Box-Plots benötigt weitere Datenbank-Zugriffe und zudem die Funktionen der Python-Module `pandas` und `matplotlib`. Zur Beschleunigung der Erstellung der Log- und Report-Dateien und zur Dependency-Reduzierung lässt sich der Box-Plot-Modus deaktivieren. Bei aktivierten Box-Plot-Modus kann der Ablageort mit dem Parameter `boxplot_folder` angegeben werden. Sofern der robot-Parameter `--outputdir` verwendet wird, muss der Ablageort als absoluter Pfad eingetragen werden.

```
robot --prerebotmodifier
perfbot.perfbot:boxplot="False":boxplot_folder="perfbot-graphics/" [path
to tests]
```

Lesender Zugriff auf Datenbank: Um lediglich die Performance zu analyisieren, jedoch nicht den aktuellen Testlauf in die Datenbank zu schreiben, so kann folgende Konfiguration genutzt werden:

```
robot --prerebotmodifier perfbot.perfbot:readonly=True [path to tests]
```

Deaktivierung der Keyword-Speicherung zur späteren Analyse: Die Keyword-Analyse erfolgt nach gelagert (siehe oben). Sofern die Betrachtung der Keywords nicht relevant ist, kann zugunsten eines schnellern Perfbots die Speicherung der Keyword-Laufzeiten deaktiviert werden:

```
robot --prerebotmodifier perfbot.perfbot:keywordstats=False [path to
tests]
```

Ausführen von Perfbot mittels `rebot`: Die `log.html` und `report.html` von Robot-Testfällen können auch ohne Testausführung basierend auf der `output.xml` generiert werden. D. h. Perfbot kann nachträglich ohne Ausführung der Tests gestartet werden. Dazu ist eine bestehende `output.xml` nötig. Bei der Ausführung von Perfbot mittes `rebot` kann neben den HTML-Dokumenten auch eine neue `output.xml` erzeugt werden, die dann auch den fehlgeschlagene Tests des Testbreaker enthält. Hinweis: Standardmäßig führt jede Ausführung von Perfbot zu neuen Datensätzen, doppelte Ausführungen zu gleichen Testdurchläufen sollte deshalb vermieden werden bzw. dann der Readonly-Modus genutzt werden.

```
# Vgl. untenstehendes Beispiel
rebot --prerebotmodifier perfbot.perfbot:devn=0.1:db_path="example/robot-
exec-times.db":testbreaker=True --output example/newoutput.xml
example/output.xml
```

Ausführung von Perfbot mit allen möglichen Parameter: Hinweis zum Entwicklungsstand: Nicht zu alle Parameter sind andere Werte als die Defaults auswählbar.

```
robot --prerebotmodifier
perfbot.perfbot:stat_func='avg':devn=0.1:db_path="example/robot-exec-
times.db":boxplot=True:boxplot_folder="perfbot-
graphics/":testbreaker=True:keywordstats="True":readonly="False" [path to
tests]
```

Beispiel Login-Page

Im Ordner `./example` sind Beispiel-Testfälle aus der Repo der Selenium-Library (entnommen aus https://github.com/robotframework/SeleniumLibrary) abgelegt. Die Tests wurden mehrmals mit Perfbot ausgeführt. Ebenfalls dort sind die dazugehörige Datenbank `robot-exec-times.db` und die Robot-Testresults (`log.html` / `report.html`), in denen die Performance-Analyse berichtet wurde, zu finden. Mit folgenden Befehl lassen sich das Beispiel starten (Installation der SeleniumLibrary vorausgesetzt):

```
# 1. Starten des System-under-Test (Login-Page)
python example/sut/server.py
# 2. Ausführung der Tests inkl. Perfbot
robot --prerebotmodifier perfbot.perfbot:devn=0.1:db_path="example/robot-
exec-times.db":testbreaker=True example/tests
```

Screenshot 1: Einbindung der Performance-Analyse in die Log-Dateiund die Funktionsweise der Testbreakers aussehen sollen:

SUITE **Invalid Login** 00:00:06.448

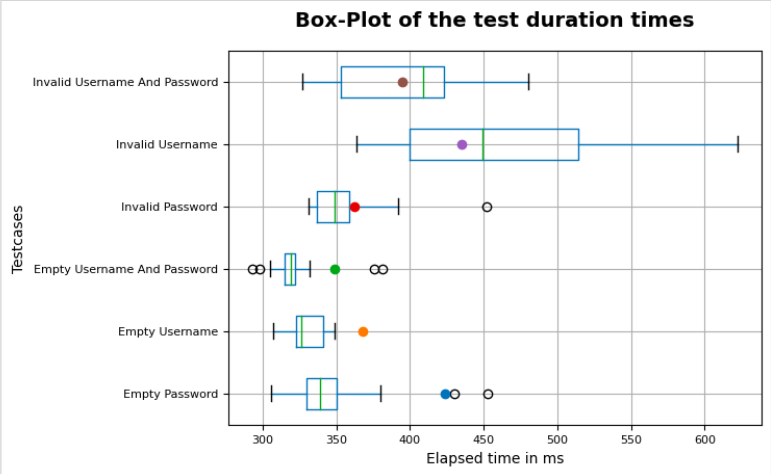
Full Name: Tests.Invalid Login

Documentation: A test suite containing tests related to invalid login.
These tests are data-driven by their nature. They use a single keyword, specified with Test Template setting, that is called with different arguments to cover different scenarios.
This suite also demonstrates using setups and teardowns in different levels.

Performance Analysis: **Summary of Tests Performance**

Testcase	Elapsed	Avg	Min	Max	Evaluated test runs	Deviation from avg
Invalid Username	00:00.435 ms	00:00.467 ms	00:00.364 ms	00:00.622 ms	17	-7.36 %
Invalid Password	00:00.362 ms	00:00.358 ms	00:00.331 ms	00:00.452 ms	17	0.86 %
Invalid Username And Password	00:00.395 ms	00:00.395 ms	00:00.327 ms	00:00.480 ms	17	-0.13 %
Empty Username	00:00.368 ms	00:00.328 ms	00:00.307 ms	00:00.349 ms	17	10.74 %
Empty Password	00:00.424 ms	00:00.351 ms	00:00.306 ms	00:00.453 ms	17	17.11 %
Empty Username And Password	00:00.349 ms	00:00.322 ms	00:00.293 ms	00:00.381 ms	17	7.58 %

Box-Plot



Screenshot 2: Testbreaker in Aktion am Beispiel des Beispiels Login-Page

TEST **Valid Login** 00:00:23.305

Full Name: Testcases.Example-Tests.Valid Login.Valid Login
Start / End / Elapsed: 20230202 20:12:00.936 / 20230202 20:12:24.241 / 00:00:23.305
Status: FAIL
Message: PerfError: Test run lasted 51.00 % than the average runs in the past and is thus above the maximum threshold of 10.00 % (original test status was PASS).

KEYWORD resource. **Open Browser To Login Page**	00:00:02.824
KEYWORD resource. **Input Username** demo	00:00:00.166
KEYWORD resource. **Input Password** mode	00:00:00.083
KEYWORD resource. **Submit Credentials**	00:00:00.127
KEYWORD BuiltIn. **Sleep** 20	00:00:20.002
KEYWORD resource. **Welcome Page Should Be Open**	00:00:00.015
TEARDOWN SeleniumLibrary. **Close Browser**	00:00:00.084

Technische Dokumentation

Für weitere Details u. a. den architektonischen Aufbau siehe ARC42_DOC.md.

Quellen

Für den Aufbau dieses Repositories wurde auf die Docs der entsprechenden Technologien zurückgeriffen und Codeschnipsel aus Implementierungsbeispielen übernommen. Im Folgendenen eine Auflistung der entsprechenden Docs, Tutorials und Implementierungsbeispielen:

- Robot Framework User Guide Version 6.0.2, insbesondere:
  - Kapitel *4.3 Listener interface* und zugehöriger Unterpunkt *Modifying execution and results*
  - Kapitel *3.6.9 Programmatic modification of results* inkl. Listing *ExecutionTimeChecker*
- Robot Framework API documentation
- SeleniumLibrary bzw. deren Beispielprojekt zum Ausprobieren dieser Implementieren (siehe `example`)
- Erweiterung von robotmetrics

- verwendete Python-Module inkl. Links zu den Docs:
  - Robot Framework
  - Pandas
  - Matplotlib
  - Sqlite3
  - Python3 (allgemein): PythonDocs,Python3 - Ein umfassende Handbuch,W3Schools, Pythonbuch
- das Logo von Perfbot basiert auf dem Robot Framework logo und ist damit unter Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0) lizensiert

Lizenz

Schlussbemerkung

Perfbot wurde im Rahmen einer Masterthesis erstellt:
Titel der Masterthesis: Automatisierte Performance-Analyse von IT-Anwendungen mit dem Testautomatisierungswerkzeug Robot Framework und Evaluation bei einem Versicherungsunternehmen
Student: Lennart Potthoff
Studiengang: M. Sc. Wirtschaftsinformatik (in Teilzeit)
Semester: Sommersemester 2023
Hochschule: FH Münster
Praxispartner: Provinzial Versicherung AG