Perfbot: Perfbot (Starter-Klasse)

```
from .PerfEvalResultModifier import PerfEvalResultModifier

"""Hier ist der Einstiegspunkt von perfbot:

Perfbot ermittelt Performance-Veränderungen anhand von bestehenden
automatisierten UI-Tests. Es erweitert dabei das
[Robot Framework](http://www.robotframework.org)
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.
"""

class perfbot(PerfEvalResultModifier):
    """Dies ist nur ein Wrapper, damit der Aufruf mit dem Parameter --
prerebotmodifier perfbot/perfbot.py aufgerufen werden kann.

    :param PerfEvalResultModifier: Basisklasse in der die eigentliche
Logik stattfindet.
    """
    pass

def main():
    print("Please start with --prerobotmodifier Option of rebot oder
robot")
```

Perfbot: PerfEvalResultModifier

```
# Documentation for ResultVisistor see
https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.h
tml#toc-entry-532
from robot.api import ResultVisitor
from robot.api.logger import info, debug, trace, console
import json, os
import sqlite3
import time
from datetime import datetime
from robot.result.model import TestCase, TestSuite, Body, Keyword
from .PersistenceService import PersistenceService
from .Sqlite3PersistenceService import Sqlite3PersistenceService
from .PerfEvalVisualizer import PerfEvalVisualizer
from .model import JoinedPerfTestResult, Keywordrun
from typing import List

# Constants
DEFAULT_MAX_DEVIATION_FROM_LAST_RUNS = 1.0
```

```
DEFAULT_LAST_N_RUNS = None
DEFAULT_DATABASE_TECHNOLOGY = "sqlite3"
DEFAULT_DATABASE_PATH = "robot-exec-times.db"
DEFAULT_BOXPLOT_FOLDER_REL_PATH = "perfbot-graphics/"
DEFAULT_STAT_FUNCTION = "avg"
TEXT_PERF_ANALYSIS_TABLE_HEADING = "*Summary of Tests Performance*\n\n|
=Testcase= |  =Elapsed=  | =Avg= | =Min= | =Max= | =Evaluated test runs= |
=Deviation from avg= |\n"
TEXT_PERF_ANALYSIS_TABLE_ROW =  "| {name}  | {elapsedtime}  | {avg} |
{min} | {max} | {count} | {devn} % |\n"
TEXT_PERF_ANALYSIS_BOXPLOT = ""
TEXT_PERF_ANALYSIS_FOOTNOTE = ""
TEXT_PERF_ERROR_MESSAGE = "PerfError: Test run lasted {calced_devn:.2f} %
than the average runs in the past and is thus above the maximum threshold
of {max_devn:.2f} % (original test status was {old_test_status})."


class PerfEvalResultModifier(ResultVisitor):
    """Diese Klasse übernimmt die eigentliche Verarbeitungslogik nach dem
Aufruf durch rebot oder von robot mit der Option prerebotmodifier.

    :class ResultVisitor: Basisklasse aus der robot.api von der diese
Klasse erbt, welche das Iterieren über die Testergebnisse ermöglicht.
    :raises NotImplementedError: Einige Parameter sind nur mit default-
Werten zulässig.
    """
    ROBOT_LISTENER_API_VERSION = 2

    perf_results_list_of_testsuite: List[JoinedPerfTestResult] = []

    body_items_of_testsuite = []

    #TODO: Globales und Suite-Timeout aus Testfällen berücksichtigen
    def __init__(self, stat_func: str=DEFAULT_STAT_FUNCTION,
        devn: float=DEFAULT_MAX_DEVIATION_FROM_LAST_RUNS, last_n_runs:
int=DEFAULT_LAST_N_RUNS, db: str=DEFAULT_DATABASE_TECHNOLOGY,
       db_path: str=DEFAULT_DATABASE_PATH, boxplot: bool=True,
boxplot_folder: str=DEFAULT_BOXPLOT_FOLDER_REL_PATH,
testbreaker:bool=False, readonly=False, keywordstats:bool=True):
        """Es sind keine Parameter für den Aufruf nötig. Es lässt sich
aber eine Vielzahl von Einstellung über folgende Parameter vornehmen:

        :param stat_func: Angabe, welche statistische Funktion zur
Auswertung genutzt wird, defaults to DEFAULT_STAT_FUNCTION
        :type stat_func: str, optional
        :param devn: Angabe, ab welcher prozentualen Abweichung der
Testbreaker auslösen soll, defaults to
DEFAULT_MAX_DEVIATION_FROM_LAST_RUNS
        :type devn: float, optional
        :param last_n_runs: Angabe, wie viele letzten Testergebnisse
analysierte werden, defaults to DEFAULT_LAST_N_RUNS
        :type last_n_runs: int, optional
        :param db: Angabe, welches Persistenz-Variante bzw. Datenbank
genutzt wird, defaults to DEFAULT_DATABASE_TECHNOLOGY
```

```
        :type db: str, optional
        :param db_path: Angabe, wo die Datenbank gespeichert ist, defaults
to DEFAULT_DATABASE_PATH
        :type db_path: str, optional
        :param boxplot: Angabe, ob die Historie der Testlaufzeiten in
einem Boxplot grafisch aufbereitet werden soll, defaults to True
        :type boxplot: bool, optional
        :param testbreaker: Angabe, ob Testfälle bei schlechter Performanz
(abhängig von devn) auf FAIL gesetzt werden sollen, defaults to False
        :type testbreaker: bool, optional
        :raises NotImplementedError: Einige Parameter (stat_func,
last_n_runs, db) sind nur mit default-Werten zulässig und somit nicht
veränderbar.
        """
        self.stat_func = stat_func
        if not self.stat_func == DEFAULT_STAT_FUNCTION:
            raise NotImplementedError("Only Avg as statistical function
supported yet.")

        self.max_deviation= devn

        self.last_n_runs = last_n_runs
        if not self.last_n_runs == DEFAULT_LAST_N_RUNS:
            raise NotImplementedError("No limit supported yet.")

        self.db_technology = db
        if not self.db_technology == DEFAULT_DATABASE_TECHNOLOGY:
            raise NotImplementedError("Only Sqlite3 as database technology
supported yet.")
        self.db_path = db_path
        self.persistenceService: PersistenceService =
Sqlite3PersistenceService(db_path)

        self.boxplot_activated = boxplot
        if  self.boxplot_activated:
            self.visualizer = PerfEvalVisualizer(boxplot_folder)
        else:
            self.visualizer = None

        self.testbreaker_activated = testbreaker
        self.readonly = readonly
        self.keywordstats = keywordstats

        if not self.readonly:
            self.persistenceService.insert_test_execution(os.uname()[1])



    def start_suite(self, suite: TestSuite):
        """Geerbte Methode aus robot.api.ResultVisitor wird an dieser
Stelle überschrieben,
        um folgende Aktionen beim Aufruf jeder Testsuite durchzuführen:

        - Wegschreiben der Ausführungsergebnisse aller Tests der Testsuite
```

```
        - Performanzstatiskten abrufen und für HTML aufbereiten
        -  optional: weitere Daten für Boxplot holen und Boxplot genieren


        :param suite: übergebene TestSuite inkl. aller Tests
        :type suite: TestSuite (siehe robot.api)
        """
        if  not suite.suites:
            testcase_perf_stats =
self.persistenceService.select_testcase_stats_filtered_by_suitename(suite.
longname)

            joined_test_results: List[JoinedPerfTestResult] =
self._eval_perf_of_tests(suite.tests, testcase_perf_stats)
            text: str = self._get_perf_result_table(joined_test_results)

            self.perf_results_list_of_testsuite = joined_test_results

            if self.boxplot_activated:
                testruns =
self.persistenceService.select_testcase_runs_filtered_by_suitename(suite.l
ongname)

                if len(testruns) == 0:
                    text+= "\n *Box-Plot* \n\n No historical data to
generate the Boxplot"
                else:
                    rel_path_boxplot =
self.visualizer.generate_boxplot_of_tests(testruns,suite.tests)
                    text+= "\n *Box-Plot* \n\n  ["+ rel_path_boxplot + "|
Boxplot ]"


            suite.metadata["Performance Analysis"] = text

            if  not suite.suites and not self.readonly:

self.persistenceService.insert_multiple_testcase_runs(suite.tests)


    def visit_test(self, test):
        """Geerbte Methode aus robot.api.ResultVisitor wird an dieser
Stelle überschrieben,
        um im Testbreaker-Modus die Testfälle bei schlechter Performanz
auf FAIL zu setzen.

        :param test: übergebener Testfall
        :type test: TestCase (siehe robot.api)
        """
        if self.testbreaker_activated:
            for perf_result in self.perf_results_list_of_testsuite:
                if perf_result.longname == test.longname:
                    calced_devn = perf_result.devn
                    break
```

```
            if calced_devn:
                if calced_devn >self.max_deviation*100:
                    old_test_status = test.status
                    test.status = 'FAIL'
                    test.message = "PerfError: Test run lasted " +
f'{calced_devn:.2f}' + " % than the average runs in the past and is thus
above the maximum threshold of " + f'{self.max_deviation*100:.2f}' + " %
(original test status was "+ str(old_test_status) + ")."

        if not self.readonly and self.keywordstats:
            self.body_items_of_test= []
            counter = 0
            if test.setup:
                counter =
self._recursive_keywords_traversal(test.setup,test.longname,0, counter)

            for bodyItem in test.body:
                if isinstance(bodyItem,Keyword):
                    counter =
self._recursive_keywords_traversal(bodyItem,test.longname,0, counter)
            if test.teardown:
                counter =
self._recursive_keywords_traversal(test.teardown,test.longname,0, counter)



self.persistenceService.insert_multiple_keyword_runs(self.body_items_of_te
st)


    def _recursive_keywords_traversal(self, bodyItem: Body,
testcase_longname: str, level: int, counter: int, stoplevel=None):

        if isinstance(bodyItem,Keyword):
            level+=1
            counter+=1
            if isinstance(bodyItem.parent, Keyword):
                parentname = bodyItem.parent.kwname
            else:
                parentname = "NO KEYWORD"


self.body_items_of_test.append(Keywordrun(bodyItem.kwname,bodyItem.name,te
stcase_longname,
parentname,bodyItem.libname,str(bodyItem.starttime),str(bodyItem.elapsedti
me),bodyItem.status,level,counter))
            for children in bodyItem.body:
                counter =
self._recursive_keywords_traversal(children,testcase_longname,level,
counter)
        return counter
```

Perfbot: PerfEvalVisualizer

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import io
from pathlib import Path
import seaborn as sns

class PerfEvalVisualizer:
    """Diese Klasse übernimmt die visuelle Aufbereitung von
Performanzdaten der Testfälle.

    :return: _description_
    :rtype: _type_
    """

    def __init__(self, boxplot_folder):
        self.boxplot_folder = boxplot_folder

    def generate_boxplot_of_tests(self, hist_tests, act_tests):
        hist = pd.DataFrame(hist_tests, columns =["id" ,"name",
"longname", "starttime" , "elapsedtime" , "status"], copy=True)

        t_list = []
        for t in act_tests:
            t_json = {
                "name": t.name,
                "longname": t.longname,
                "elapsedtime": t.elapsedtime
            }
            t_list.append(t_json)
        act = pd.DataFrame(t_list, columns =["name", "longname",
"elapsedtime"], copy=True)

        return self.generate_boxplot(hist, act, format="png")

    def generate_boxplot(self, hist_results: pd.DataFrame, act_results:
pd.DataFrame, x="elapsedtime", y='name', xlabel="Duration
(s)",ylabel="Testcase", heading='Box-Plot of the test duration times',
format="svg"):
        sns.set_theme(style="whitegrid", context="notebook")
        hist = pd.DataFrame(hist_results, copy=True)
        hist[x] = hist[x].astype(int) / 1000
        boxplot = sns.boxplot(x=x, y=y, data=hist)

        boxplot.set_xlabel(xlabel)
        boxplot.set_ylabel(ylabel)
        boxplot.figure.suptitle(heading, fontsize=14, fontweight='bold')
        boxplot.set_title("")

        sns.stripplot(ax=boxplot,x=x, y=y, data=hist, color="grey")
```

```
        if True:
            act = pd.DataFrame(act_results, copy=True)
            act[x] = act[x].astype(int) / 1000
            plt.plot(act[x], act[y],'o', color='orange', zorder=10)

        match format:
            case "svg":
                f = io.StringIO()
                boxplot.figure.savefig(f, format = "svg",
bbox_inches="tight")
                plt.clf()
                return f.getvalue()

            case "png":
                Path(self.boxplot_folder).mkdir(parents=True,
exist_ok=True)
                pathname  = self.boxplot_folder + "boxplot" +
datetime.now().strftime("-%m-%d-%Y-%H-%M-%S-%f") + ".png"
                try:
                    plt.savefig(pathname, bbox_inches="tight")
                    plt.clf()
                except:
                    print("An execption occured")
                print("Boxplot generiert: " + pathname)
                return pathname
            case _:
                raise KeyError("Wrong Format of Boxplot generation.")
```

Datenbankschema

```
CREATE TABLE IF NOT EXISTS test_execution (
    id integer PRIMARY KEY AUTOINCREMENT,
    imported_at text DEFAULT CURRENT_TIMESTAMP,
    hostname text
);

CREATE TABLE IF NOT EXISTS testcase (
    id integer PRIMARY KEY AUTOINCREMENT,
    name text,
    longname text,
    suitename text,
    UNIQUE(longname)
);

CREATE TABLE IF NOT EXISTS keyword (
    id integer PRIMARY KEY AUTOINCREMENT,
    name text,
    longname text,
    libname text,
```

```
    UNIQUE(longname)
);

CREATE TABLE IF NOT EXISTS testcase_run (
    id integer PRIMARY KEY AUTOINCREMENT,
    testcase_id integer REFERENCES testcase(id) ON DELETE CASCADE NOT
NULL,
    test_execution_id integer REFERENCES test_execution(id) ON DELETE
CASCADE NOT NULL,
    starttime text NOT NULL,
    elapsedtime text NOT NULL,
    status text NOT NULL
);

CREATE TABLE IF NOT EXISTS keyword_run (
    id integer PRIMARY KEY AUTOINCREMENT,
    testcase_run_id integer REFERENCES testcase_run(id) ON DELETE CASCADE
NOT NULL,
    keyword_id integer REFERENCES keyword(id) ON DELETE CASCADE NOT NULL,
    starttime text NOT NULL,
    elapsedtime text NOT NULL,
    status text NOT NULL,
    keyword_level integer,
    stepcounter integer,
    parent_keyword_longname text
);

CREATE VIEW IF NOT EXISTS testcase_run_view AS
SELECT testcase.name, testcase.longname, testcase_run.starttime,
testcase_run.elapsedtime, testcase_run.status, test_execution.id,
test_execution.hostname
FROM testcase_run
INNER JOIN testcase ON testcase_run.testcase_id = testcase.id
INNER JOIN test_execution ON testcase_run.test_execution_id =
test_execution.id;


CREATE VIEW IF NOT EXISTS keyword_run_view AS
SELECT testcase.name as testcase_name, testcase.longname as
testcase_longname, testcase.suitename, keyword.name as kw_name,
keyword.longname as kw_longname, keyword.libname, keyword_run.starttime,
keyword_run.elapsedtime, keyword_run.status, keyword_run.keyword_level,
keyword_run.stepcounter, keyword_run.parent_keyword_longname,
test_execution.id, test_execution.hostname
FROM keyword_run
INNER JOIN keyword ON keyword_run.keyword_id = keyword.id
INNER JOIN testcase_run ON keyword_run.testcase_run_id = testcase_run.id
INNER JOIN testcase ON testcase_run.testcase_id = testcase.id
INNER JOIN test_execution ON testcase_run.test_execution_id =
test_execution.id;
```

Integrationstest (definiert im Robot Framework)

```
*** Settings ***
Suite Setup       Vorbereiten
Documentation     Integrationstest zur Perfbot. Startet die Beispieltests
und prüft die log.html und report.html. Vorm Starten den Ablageort der
LOG_HTML anpassen.
Library           Process
Library           SeleniumLibrary
Library           OperatingSystem
Suite Teardown    Aufraeumen

*** Variables ***
${BROWSER}      Chrome
${START_SUT}    python3 example/sut/server.py
${RUN_ROBOT}    python3 -m robot --prerebotmodifier
perfbot.perfbot:devn=0.1:db_path="tests/itests/temp/test.db":boxplot=True:
testbreaker=True:boxplot_folder="tests/itests/temp/" -o tempoutput.xml -l
templog.html -r tempreport.html example/tests

*** Test Cases ***
Perfbot im ersten Durchlauf testen
    Beispiel mit Robot testen
    Vorhandensein der Dateien pruefen
    Log-Datei pruefen    testflauf_anzahl=NO STATS
    # beim ersten Durchlauf gibt es noch keinen Boxplot
    Close Browser
Perfbot im zweiten Durchlauf testen
    Beispiel mit Robot testen
    Vorhandensein der Dateien pruefen
    Log-Datei pruefen    testflauf_anzahl=1
    Boxplot in Log-Datei und lokal pruefen
    Close Browser
Perfbot im dritten Durchlauf testen
    Beispiel mit Robot testen
    Vorhandensein der Dateien pruefen
    Log-Datei pruefen    testflauf_anzahl=2
    Boxplot in Log-Datei und lokal pruefen
    Close Browser

*** Keywords ***
Vorbereiten
    Remove Directory    tests/itests/temp    recursive=True
    Create Directory    tests/itests/temp
    Beispiel SUT starten
    ${pwd}= Run Process    pwd    shell=yes
    Log    pwd: ${pwd.stdout}
    Set Global Variable    ${LOG_HTML}
file://${pwd.stdout}/templog.html
    Log    Ablageort der LOG-HTML ermittelt: ${LOG_HTML}
Beispiel SUT starten
    Start Process    ${START_SUT}    shell=yes    alias=sut
Beispiel mit Robot testen
    ${result}=  Run Process    ${RUN_ROBOT}    shell=yes
```

```
Vorhandensein der Dateien pruefen
    File Should Exist    tests/itests/temp/test.db
    File Should Exist    tempoutput.xml
    File Should Exist    templog.html
    File Should Exist    tempreport.html
Log-Datei pruefen
    [Arguments]    ${logdatei}=${LOG_HTML}    ${metadata_feld}=Performance
Analysis:    ${titel_in_tabelle}= Deviation from avg
${erster_testfall}=Invalid Username    ${testflauf_anzahl}=1
    Open Browser    ${logdatei}   ${BROWSER}
    Title Should Be    Tests Log
    Click Element    css:div#s1-s1
    Page Should Contain Element    css:div#s1-s1 table
    Table Should Contain    locator=css:div#s1-s1 table
expected=${metadata_feld}
    ${element}=    GetWebElement    locator=xpath://*[@id="s1-
s1"]/div[2]/table/tbody/tr[3]/td/table
    Element Should Contain    ${element}    ${titel_in_tabelle}
    Table Cell Should Contain    locator=${element}   row=2    column=1
expected=${erster_testfall}
    Table Cell Should Contain    locator=${element}   row=2    column=6
expected=${testflauf_anzahl}

Boxplot in Log-Datei und lokal pruefen
        Click Image    //*[@id="s1-
s1"]/div[2]/table/tbody/tr[3]/td/p[3]/img
        ${pic}=    Get Element Attribute    //*[@id="s1-
s1"]/div[2]/table/tbody/tr[3]/td/p[3]/img    src
        Log        Boxplot saved under: ${pic}
        ${file}=   Evaluate    '${pic}'.replace('file://','')
        File Should Exist    ${file}
Aufraeumen
    Terminate All Processes    kill=True
    Remove Directory    tests/itests/temp    recursive=True
    Remove File         tempoutput.xml
    Remove File         templog.html
    Remove File         tempreport.html
```