



# Performance Teststraat



Werking, onderhoud en uitbreiding



**Van**


**Versie**

**Datum**

Marcel Wigman, Erik Post

1.5

23-5-2019




## Revisietabel

Datum	Versie	Auteur	Opmerking
1-5-2018	0.1	M.Wigman	Initiële opzet
7-5-2018	1.0	M.Wigman	Rapportgenerator deel klaar
11-10-2018	1.1	E.Post	Machines ingevuld en teststraat + reporttemplategenerator aangevuld
16-1-2019	1.2	M.Wigman	Details sub tools reportgenerator naar bijlage, installatiebeschrijving toegevoegd. Eerste oplevering aan ICTU.
24-1-2019	1.3	M.Wigman	Review ICTU (Frank Niessink) verwerkt, definitief
26-2-2019	1.4	M.Wigman	Aanpassingen tekst stijl
23-5-2019	1.5	M.Wigman	Documentatie rond rapportgenerator bijgewerkt (.NETCore, shell scripts, nieuwe functionaliteit)

## Inhoudsopgave

<b>1. Inleiding.....</b>	<b>5</b>
<b>2. Architectuur .....</b>	<b>6</b>
2.1. Onderdelen.....	6
2.1.1. Testcluster .....	6
2.1.2. Jenkins centraal en cluster .....	7
2.1.3. Teststraat.....	7
2.1.4. Loadgeneratoren .....	7
2.1.5. Rapport template generator .....	7
2.1.6. Rapportgenerator .....	7
2.1.7. Versiebeheer (GIT) .....	7
2.2. Begrippen .....	8
2.3. Proces.....	8
<b>3. Werking .....</b>	<b>9</b>
3.1. Teststraat.....	9
3.1.1. Schematische weergave .....	9
3.1.2. Aanroep .....	10
3.1.3. Scripts .....	10
3.1.4. Installatie.....	10
3.2. ReportTemplateGenerator .....	12
3.2.1. Aanroep .....	12
3.2.2. Installatie.....	12
3.3. Rapportgenerator.....	13
3.3.1. Architectuur .....	13
3.3.2. Samenhang en functie van onderdelen .....	14
3.3.3. Installatie.....	16
3.3.4. Details.....	16
<b>4. Usecases .....</b>	<b>22</b>
4.1. Toevoegen van een nieuw project .....	22
4.2. Eisen aan een testscript .....	22
4.2.1. Technologie: JMeter of Silkperformer .....	22
4.2.2. Gebruik van loadgeneratoren.....	22
4.2.3. Configuratie van workload.....	22
4.3. Toevoegen, wijzigen of wijzigen van een transactienaam.....	23
4.4. Toevoegen of wijzigen van thresholds.....	24

4.5. Disable/enable van een testrun .....	24
4.6. Toevoegen/verwijderen van een testrun .....	25
4.7. Toevoegen/verwijderen van een project.....	25
4.8. Wijzigen van bestandslocaties .....	25
4.9. Wijzigen database locatie .....	26
<b>5. Bijlage 1: Report Generator tools .....</b>	<b>27</b>

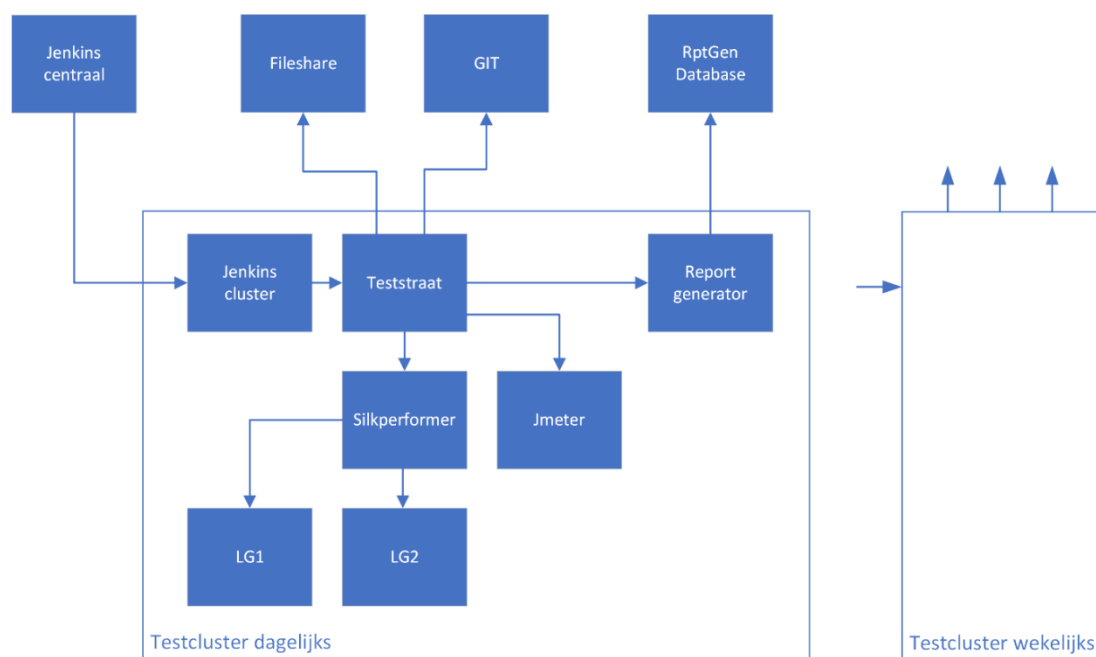
## 1. Inleiding

Voor het dagelijks geautomatiseerd uitvoeren van performancetesten is voor ICTU een 'teststraat' ontwikkeld. Vanuit de teststraat worden zonder menselijke interactie performancetesten uitgevoerd en geëvalueerd naar standaard rapportages. De testrapporten worden op hoofdlijnen geëvalueerd door het centraal kwaliteitssysteem waarin meerdere kwaliteit metrieken samen komen: security, functionele testen, code kwaliteit en performancetesten.

Voor elke applicatie worden drie testen uitgevoerd: een performancetest, een stresstest en een duurtest. De doorlooptijd van een testserie (1+1+6=8h) legt een te grote claim op een testcluster om 10-20 projecten te kunnen dekken. De testen zijn daarom verdeeld over twee testclusters: dagelijkse performancetesten (loadtest) vanaf testcluster 1 en wekelijkse performancetesten (duurtest, stresstest) vanaf testcluster 2. De testclusters zijn autonoom, er zijn geen beperkingen voor schaling..

Dit document is bedoeld om de lezer inzicht te geven in de werking van het geheel en te voorzien van handvatten voor de meest voorkomende bewerkingen op een geïmplementeerde teststraat zoals: toevoegen en wijzigen van een project, eisen aan een project, wijzigen van rapporten, wijzigen van thresholds (non-functional requirements) enz. Er is ook een beperkte installatiebeschrijving opgenomen.

## 2. Architectuur



### 2.1. Onderdelen

In dit hoofdstuk worden de onderdelen waaruit de teststraat bestaat globaal beschreven: wat is het en wat is zijn/haar rol.

#### 2.1.1. Testcluster

De teststraat bevat momenteel een tweetal testclusters, elk bestaande uit vier machines, en een ontwikkelmachine:

	Aanwezig
Jenkins Centraal	JA
Fileshare	JA
Git	JA
Rptgen database	JA

	Ontwikkelmachine	Testcluster dagelijks*	Testcluster wekelijks**
Jenkins	JA	JA	JA
Controller (teststraat, Silk, Jmeter, Rptgen)	JA	JA	JA
LG1: loadgenerator	NEE	JA	JA
LG2 loadgenerator	NEE	JA	JA

\* dagelijks: dagelijkse testen (productie)

\*\* wekelijks: wekelijkse testen (stresstest, duurttest)

#### 2.1.2. Jenkins centraal en cluster

De centrale Jenkinsinstantie bevat jobs voor de testcluster Jenkinsinstanties en initieert bijvoorbeeld: testscript updates, Jenkins job updates, distributie van laatste versies van teststraat scripts en tools over de testclusters. Een concurrency plug-in zorgt ervoor dat maar één test tegelijk draait vanaf elk cluster en dat maar één test tegelijk draait op een omgeving. Er zijn dus twee concurrency mechanismen actief: (1) concurrency per applicatie omgeving en (2) concurrency per cluster. Op termijn moet in elk cluster een eigen queuemanager worden opgenomen zodat een cluster zijn eigen concurrency kan bewaken.

#### 2.1.3. Teststraat

De 'teststraat' bestaat uit een aantal (shell) scripts die de opdracht voor het uitvoeren van een test aannemen van een queue manager, bijvoorbeeld Jenkins. De regie voor het uitvoeren van metingen, starten en stoppen van een test en genereren van een rapport ligt bij de teststraat.

#### 2.1.4. Loadgeneratoren

Momenteel zijn twee loadgeneratoren aanwezig: Silkperformer (commercieel) en Jmeter (open source). Beide draaien op dezelfde machine als waarop de teststraat draait. Voor Silkperformer zijn twee ondersteunende machines ingericht als loadgenerator. Niet afgebeeld is een licentieserver waarop Silkperformer licenties staan die worden gebruikt door alle Silkperformer instanties.

#### 2.1.5. Rapport template generator

De rapportgenerator voegt testdata samen met een HTML-template. Deze template is voor alle projecten hetzelfde, uitgezonderd transactienaam specifieke delen. De templategenerator voegt de statische templateonderdelen samen met een project specifieke lijst transactienamen. Het resultaat is een project specifieke HTML-template welke na afloop van elke testrun wordt gegenereerd voor gebruik door de rapportgenerator.

#### 2.1.6. Rapportgenerator

Custom applicatie die testresultaten van de loadgeneratoren (Silkperformer of Jmeter) omzet in een HTML-rapport. Deze rapportgenerator is voor deze toepassing ontwikkeld in .NET/C#. De testresultaten worden door de rapportgenerator opgeslagen in een centrale database voor gebruik voor trend-weergave.

#### 2.1.7. Versiebeheer (GIT)

Alle onderdelen van de teststraat worden geversioneerd in GIT en worden via GIT gedistribueerd over de testclusters. Dit geldt voor scripts en executables. Testresultaten (HTML-rapporten) worden in GIT opgeslagen en opgeslagen op een fileshare voor gebruik door het centrale kwaliteitssysteem. Deze dubbele opslag wordt verzorgd door de teststraat.

## 2.2. Begrippen

Script	Technisch klikpad waarmee een loadgenerator gevoed wordt. Een script is loadgenerator specifiek: lpt (Silkperformer) of jmx (Jmeter). Daarbij kan testdata gebruikt worden in de vorm van een CSV-bestand of query's rechtstreeks op de database.
Workload	Voor gedefinieerde configuratieset in/bij het script waarmee de samenstelling en intensiteit van belasting wordt geregeld: Productie, Duurtest, Stresstest.
Soorten testen	Loadtest (workload=Productie): toets responsetijden tegen eisen Duurtest (workload=Duurtest): toets stabiliteit over langere tijd Stresstest (workload=Stresstest): toets van belastbaarheid/marge
Testclusters	Samenstel van controller en één of meer loadgeneratoren die als geheel autonoom testen kunnen uitvoeren. Er worden twee clusters gebruikt: een voor het uitvoeren van dagelijkse loadtesten (workload=Productie) en een voor wekelijkse stress- en duurtesten.
Loadgenerator	Software die verantwoordelijk is voor het genereren van belasting. Gebruikte varianten: Silkperformer en JMeter.

## 2.3. Proces

Het proces dat gevolgd wordt bij het doorlopen van een test op één van de testclusters, is zoals hieronder opgesomd.

Jenkins: reserveren omgeving+testcluster

Jenkins: start een test (trigger → teststraat)

Teststraat: pre-test acties

Teststraat: uitvoeren URI-check omgeving (controle omgeving)

Teststraat: uitvoeren verkorte test (controle applicatie)

Teststraat: uitvoeren test (start loadgenerator)

Teststraat: post-test acties

Teststraat: start rapporttemplategenerator voor <project>

Rapporttemplategenerator: genereren van een rapport template

Teststraat: start rapportgenerator <project>

Rapportgenerator: interpreteren van testresultaat (parse)

Rapportgenerator: inlezen van data in database

Rapportgenerator: genereren rapport (merge)

Teststraat: publiceren van het rapport

Jenkins: opslag van rapport in workspace

Jenkins: vrijgeven omgeving+testcluster



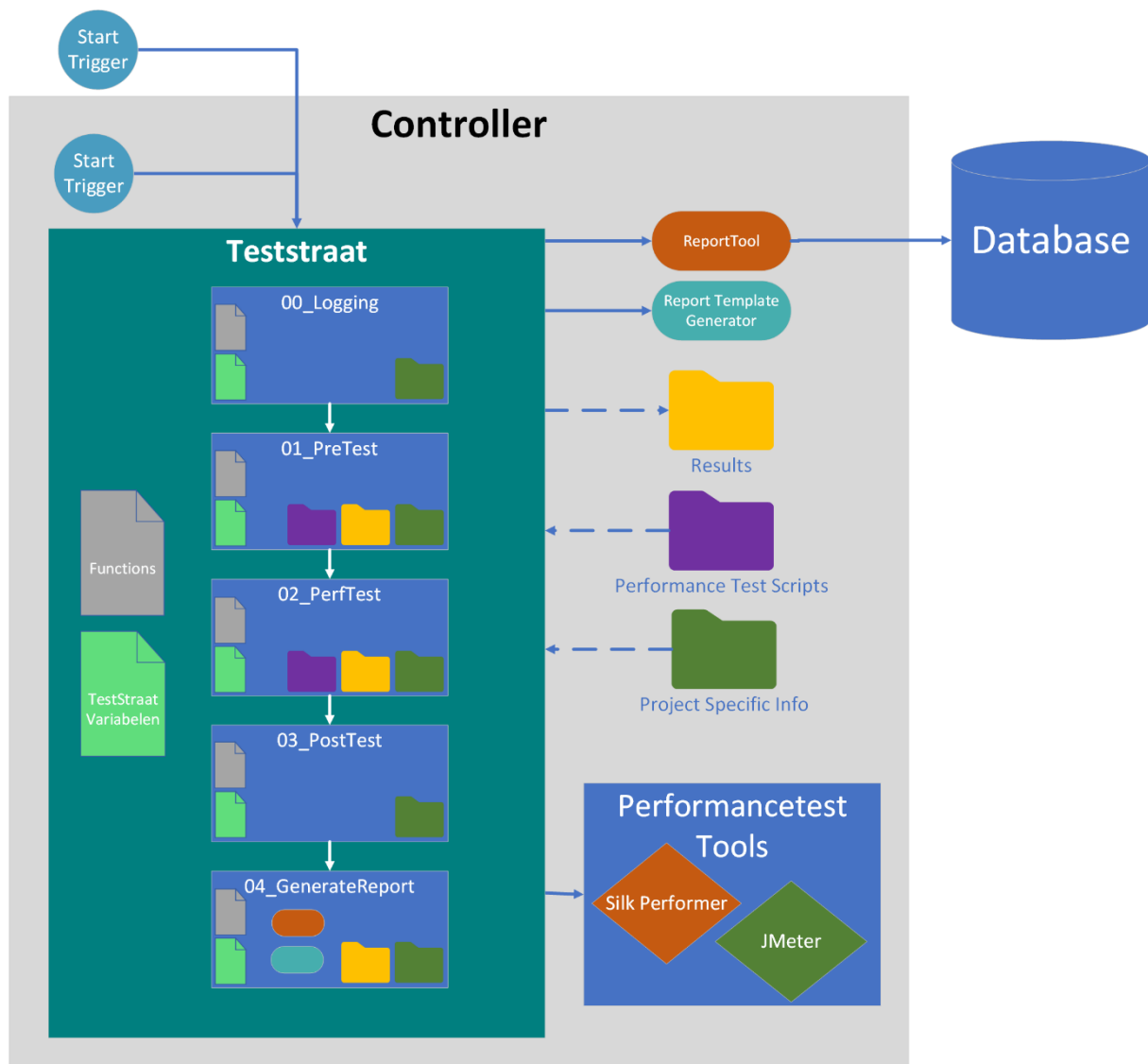
### 3. Werking

#### 3.1. Teststraat

De teststraat bestaat uit een serie van shell(bin/bash) scripts die ervoor zorgen dat de omgeving opgeschoond wordt, er checks gedaan worden, de test wordt uitgevoerd, en er een rapport gegenereerd wordt. Daarnaast is er een functies bestand waar alle functies in te vinden zijn, en een teststraatvars bestand waar alle omgeving parameters van de teststraat zijn vastgelegd.

De teststraat is verder zo ingericht dat er een projectfolder per project benodigd is waar projectspecifieke informatie in te vinden is zoals welke testtool er gebruikt moet worden en de naam van het script. Verder staan hier een aantal bestanden waar eventuele project specifieke handelingen in uitgevoerd kunnen worden zoals het ophalen van de applicatie versie, uitlezen van een DB, starten/stoppen monitoring zoals NMON, etc.

##### 3.1.1. Schematische weergave



### 3.1.2. Aanroep

De teststraat dient met de volgende parameters gestart te worden:

`./00_logging.sh $buildnumber $workload $baseline $project $runverification`

- buildnumber
  - Parameter vanuit Jenkins is overbodig en kan uit de scripts gehaald worden
- prodload
  - Welke scenario er gestart moet worden, keuze is nu tussen productie, stresstest, duurttest.
- baseline
  - Welke baseline (datum/tijd testrun id) er gebruikt moet worden. Is optioneel, als deze niet gebruikt moet worden dient er wel wat ingevuld te worden zoals bijvoorbeeld [leeg].
- project
  - Welk project er gestart moet worden
- runverification
  - De teststraat draait ook een verificatie run voordat de echte te gestart wordt om te weten of alle transacties het gewenste resultaat opleveren. Waarden: true | false.

### 3.1.3. Scripts

De teststraat is verdeeld over 5 scripts met ieder een eigen doel.

- 00\_Logging
  - Dit script is het startpunt en bekijkt of de omgeving klaar is om een test te starten
- 01\_Pretest
  - In dit script wordt een validatie run gedaan waarbij het de bedoeling is dat alle handelingen in de scripts eenmalig worden uitgevoerd. Dit om te verifiëren dat het script voor alle stappen het gewenste resultaat krijgt. En eventueel projectspecifieke pretest handelingen uit te voeren
- 02\_Perftest
  - Het uitvoeren van de daadwerkelijke test
- 03\_Posttest
  - Het verzamelen van de gegevens, eventueel een backup maken en project specifieke posttest stappen uitvoeren
- 04\_GenerateReport
  - Genereren van een rapport door middel van de ReportTool

### 3.1.4. Installatie

Deze installatiebeschrijving gaat er vanuit dat de teststraat vanuit de GitHub repository wordt geïnstalleerd.

1. Configureer een Windows machine: 64 bit, 16GB memory met 'extra' schijf >20GB met recente versie van het .NETCore framework (zie voor versie eis releasenotes van rapportgenerator).
2. Zet op de 'extra schijf' (bijv. D- of E-schijf) ofwel in de root, ofwel in een folder, de hoofdfolder van de teststraat (fig. 1)

3. Vul vanuit de GitHub repository [ICTU/performancetest-runner]:  
git clone <https://github.com/ICTU/performancetest-runner.git> .
4. Maak projectfolders aan naar voorbeeld van het Test voorbeeldproject onder **Fout!**  
**Verwijzingsbron niet gevonden.**
5. Installeer Cygwin
6. Installeer PostgreSQL en creëer een lege database 'Teststraat' mbv een lege dump,  
opgenomen in folder [.\08\_Database\ExportImport]
7. Configureer connect gegevens naar de database in de Report Generator  
[..\06\_Tools\ReportTool\tools\\*.config]
8. Installeer loadgenerator
  - a. Optioneel: Jmeter (+Java) (tip: folder 06\_Tools\jmeter)
  - b. Optioneel: Silkperformer default Windows <Program Files> dir
9. Configureer locatie specifieke inrichting in  
[00\_Globals\testautomation\_globals.incl]
  - a. root\_sh
  - b. root\_cygwin
10. Configureer transactienamen per project in  
[09\_ProjectFolders\<projectnaam>\Transactions.csv]
11. Installeer en configureer Jenkins tbv scheduling en queuing (alternatief: GIT/CI)  
Aanroep: [00\_Logging.sh <buildnummer> <belastingprofiel> <baseline> <project>  
<runverification>]
12. Richt het overnemen van het gegenereerde rapport naar een centrale plaats in  
Bron: [E:\03\_Resultaten\<project>\report]  
of laat de teststraat het resultaat direct op de juiste plaats zetten  
Variabele single: [reportsfolder\_root]  
Variabele alles incl history: [reporthistory\_root]

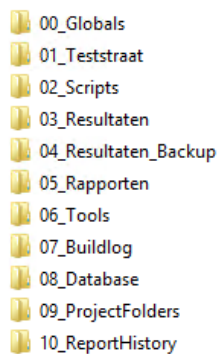


Fig 1: folderstructuur teststraat

### 3.2. ReportTemplateGenerator

De ReportTemplateGenerator is ontwikkeld om het maken van de project specifieke HTML-templates te automatiseren. Elk project moet voordat de ReportTool aangeroepen wordt een template beschikbaar hebben waar de te meten transacties zich in bevinden. De ReportTemplateGenerator creëert deze template op basis van een generiek template dat gevuld wordt op basis van een Transactions.csv file welke te vinden is in de project specifieke folders.

#### 3.2.1. Aanroep

# Onderstaande commando uitvoeren vanuit de root van de ReportTemplateGenerator (waar ook dit bestand staat)

# parameter 1 = Project naam zoals deze ook door de teststraat gebruikt wordt

# parameter 2 = Transactions file

# Voorbeeld 1: Niet ingevuld

`./generateTemplate.sh [project] ../../[project]/[transactieFile]`

# Voorbeeld 2: Ingevuld

`./generateTemplate.sh cs ../../cs/Transactions.csv`

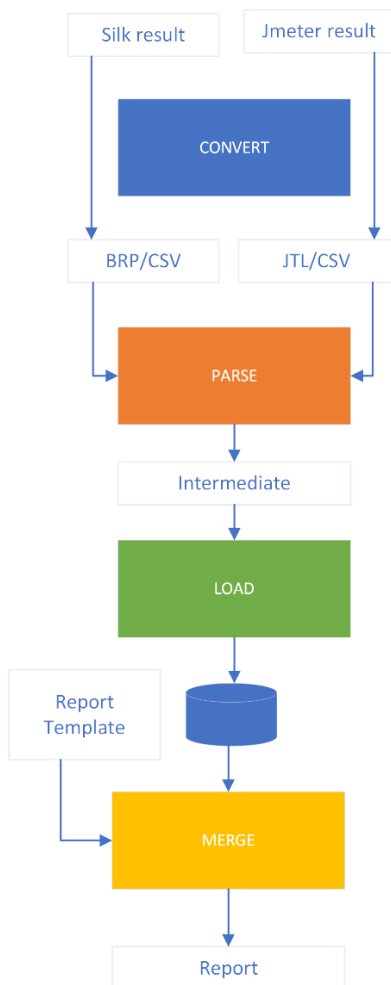
#### 3.2.2. Installatie

Zie 3.1.4

### 3.3. Rapportgenerator

De rapportgenerator bestaat uit een aantal losse command-line tools (.NETCore) en scripts (shell), deze staan in dir `..\06_Tools\ReportTool`. Het .NETCore framework moet op de server geïnstalleerd zijn, zie voor versie eisen: `..\06_Tools\ReportTool\tools\rpg.releasenotes.txt`

#### 3.3.1. Architectuur

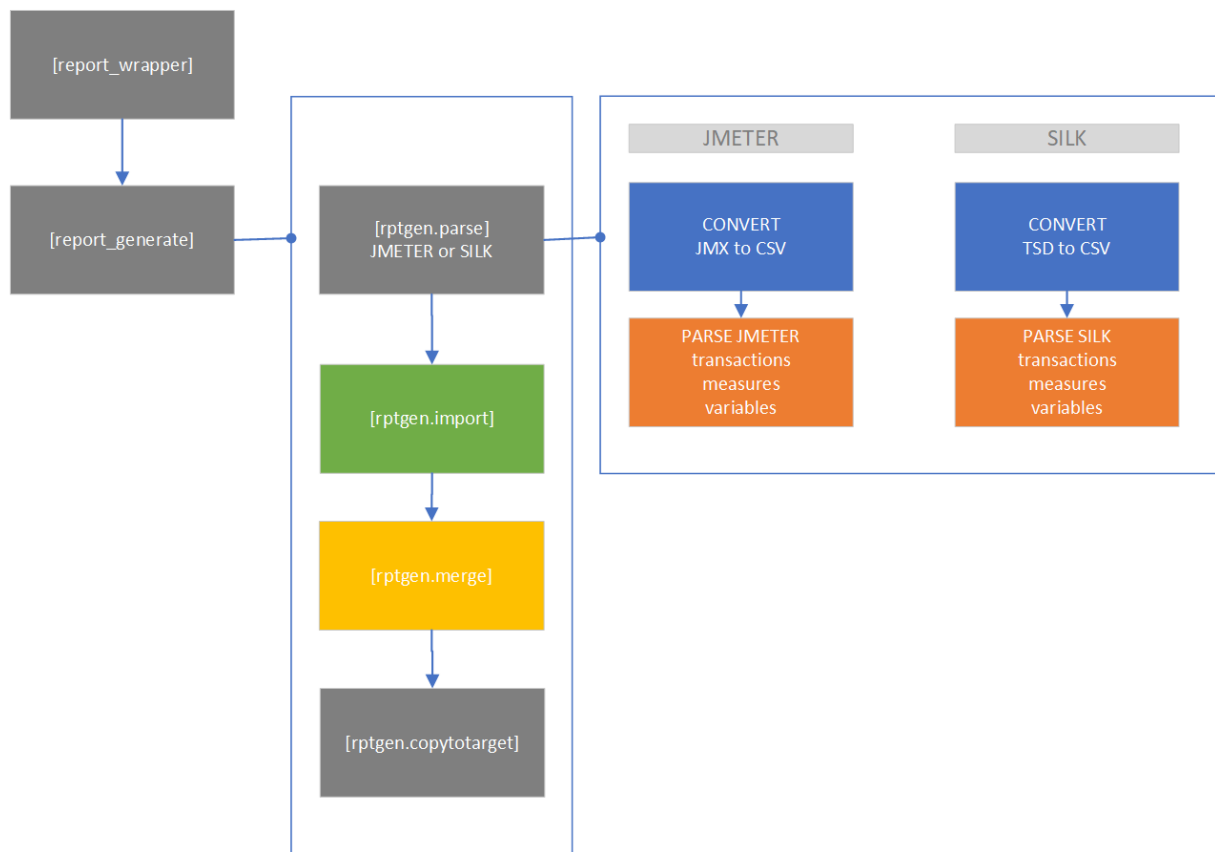


Bij het genereren van een rapport wordt een aantal stappen/fases doorlopen:

- 1) **Verzamelen** van testresultaten (niet in de figuur), technologie specifiek
- 2) **Convert** van binair- naar CSV-formaat, technologie specifiek
- 3) **Parse** naar generiek 'intermediate' formaat
- 4) **Load** intermediate data in database
- 5) **Merge** van intermediate data en HTML-template naar HTML-rapport
- 6) **Kopieren** HTML-rapport naar doel locatie

Stappen kunnen overigens optioneel worden overgeslagen via parameter 'partial' (report\_wrapper.sh).

### 3.3.2. Samenhang en functie van onderdelen



Boven: samenhang van scripts en bijbehorende fases (kleur)

De rapportgenerator wordt gestart met script report\_wrapper.

#### 3.3.2.1. Report-wrapper (report\_wrapper.sh)

Hoogste niveau aanroep/start van de reportgenerator. Variabelen worden geïnitieerd en het genereren van het rapport wordt gestart. Variabelen worden opgebouwd op basis van globale teststraat variabelen in ..\00\_Golbals\. Aan de rapportgenerator hoeft niets te worden geconfigureerd behalve connectiegegevens naar de database.

Workflow:

1. Laden van variabelen [vars.incl]
2. Genereren van het rapport (report\_generate.sh)

Gebruik:

```
..\ReportTool\report_wrapper.sh <project> [jmeter|silk] <comment> <baseline>  
testrun> <optional:partial>
```

Voorbeeld:

```
report_wrapper.sh lrk jmeter "Gegenereerd door teststraat" "2018.05.01.03.34.36"  
123456
```

Partial:

De zes fases die het onderliggend script `report_generate.sh` doorloopt kunnen indien nodig aan of uit gezet worden met parameter 'partial'. Dit is bijvoorbeeld handig wanneer testdata alleen geparsed moet worden, om bijvoorbeeld aan transactienamen te komen. Partial is dan '123'. Data wordt dan niet ingelezen in de database (4), er wordt geen rapport gegenereerd (5) en het rapport wordt niet gekopieerd naar target folder (6).

Inlezen van generieke teststraatvariabelen in `..\00_globals` en zetten van eigen variabelen t.b.v. rapportgenerator.

#### 3.3.2.2. Report Generate (`report_generate.sh`)

De aanroep van de stappen voor het genereren van het rapport: parsen van ruwe testresultaten naar 'intermediate' formaat, import van meetgegevens naar de database, samenvoegen (merge) van data en rapporttemplate en tot slot het kopiëren van het gegenereerde rapport naar de juiste doeldirectory. De temp folder van de rapportgenerator functioneert als 'workdir'.

Workflow in 6 fases:

1. Voorbereiding, creëren van de directory structuur
2. Kopiëren van test output naar temp directory
3. Start parse testresultaten [`tools\rptgen.parse.<jmeter/silk>.sh`]
4. Start import testresultaten [`tools\rptgen.import.sh`]
5. Start merge testresultaten+template [`tools\rptgen.merge.sh`]
6. Afronding: kopiëren van output naar doeldirectory [`tools\rptgen.copytotarget.sh`]

#### 3.3.2.3. Parse [`rptgen.parse.jmeter.sh` / `rptgen.parse.silk.sh`]

Conversie van ruwe logs naar csv. Deze stap is voor elk type loadgenerator anders, daarom bestaat per loadgenerator een apart script dat het parsen afhandelt. Voor nu worden resultaten van loadgeneratoren Jmeter en Silkperformer ondersteund. Er wordt geconverteerd naar key=value pairs. Binnen de context van de report generator wordt dit 'intermediate' format genoemd. Dit wordt 'as is' ingelezen in de database. De rapporten worden gegenereerd vanuit de database data.

Zie 3.3.4.1 *Intermediate format (parsing)*

#### 3.3.2.4. Import [`rptgen.import.sh`]

Importeren van key-value sets uit het parse resultaat (intermediate format bestanden) in de (PostgreSQL) database. De connectiegegevens naar de database zijn geconfigureerd in het global variable script.

#### 3.3.2.5. Merge [`rptgen.merge.sh`]

Samenvoegen van data in de HTML-rapporttemplate. In feite gebeurt er een zoek-en-ervang van variabele namen en hun waarden. Variabelen waarvan geen data in de database gevonden worden, worden leeg gemaakt. Het mergen gebeurt in fases, van elke fase wordt een tussenversie van de merge bewaard (`report.00.html`, `report.01.html...`), handig voor debugging mocht het nodig zijn. Na afloop van het merge proces staat er een volledig ingevulde werkversie van het rapport in de temp folder (`work.html`).

Zie 3.3.4.3 *HTML Template* (merge van variabelen).

#### 3.3.2.6. *CopyToTarget [rptgen.copytotarget.sh]*

De werkversie van het rapport (temp folder\work.html) wordt samen met de benodigde javascript, plaatjes en de log gekopieerd naar de doeldirectory.

### 3.3.3. Installatie

Zie 3.1.4

### 3.3.4. Details

#### 3.3.4.1. *Intermediate format (parsing)*

De rapportgenerator is grotendeels loadgenerator-technologie onafhankelijk. Loadgenerator output (Silkperformer, Jmeter, ...) wordt door de rapportgenerator parser vertaald naar 'intermediate format'. Dit formaat bestaat in essentie uit key=value combinaties waarvan de 'value' een enkele waarde of een ';' gescheiden reeks kan bevatten. De value waarden worden 1:1 overgenomen in de corresponderende \${key} placeholders in de HTML-template.

De intermediate bestanden zijn te herkennen aan de naam \_intermediate.<type informatie>.csv. In principe wordt uit elke tool dezelfde set intermediate format bestanden gegenereerd. Conversie, parsing en merge wordt uitgevoerd in folder:

```
..ReportTool\temp
```

### Bestanden

<i>Input voor het parse proces</i>	<i>Geconverteerd naar intermediate</i>
Jmeter:	_intermediate.trs.csv
_transactions.jtl	_intermediate.trs.csv.definitions
_transactions_all.csv (jtl -> csv)	_intermediate.trs.csv.transactionnames
_transactions_success.csv (jtl->csv)	_intermediate.msr.csv
	_intermediate.var.trs.csv
Silkperformer:	_intermediate.var.msr.csv
_transactions.brp	_intermediate.var.csv
_transactions.tsd	
_transactions.csv (tsd -> csv)	
Teststraat:	
_intermediate.var.runinfo.csv	_intermediate.var.runinfo.csv

### Typen data

Er worden verschillende typen waarden geëxtraheerd:

TRS      transactiewaarden (count;minimum;average...)\*



MSR    measure waarden (tijdseries: meetwaarden, gescheiden door ';')  
VAR    variabelen (variabele=waarde)

\* serie statistische waarden, zie gegenereerde \_intermediate.trs.csv.definitions:

0. Count
1. Minimum
2. Average
3. Maximum
4. 90 percentile
5. Failed
6. Canceled
7. Median
8. 95 percentile
9. Std deviation

Voorbeelden intermediate format inhoud:

```
#Overall Response Time#=1223;0;;1,31465658;60,735;0,108999997;65;236;0,046999  
Beheerder_03_ZoekenHouder=37;0,297;0,52108108;1,109;0,703000009;;;0,532000005  
TInit=12;0;;0,04166667;0,187;0,001;;;0,001;0,001;0,06452433  
Workloadexit=190;0;;0;;0,001;;;0,001;0,001;0,  
Beheerder_04_PersoonInschrijven=22;0,063;0,10154545;0,219;0,140000001;12;;0,0  
Beheerder_05_Zoekenpersoon=22;0,046;1,98004545;42,079;0,360000014;;;12;0,06199  
Beheerder_06_Uitschrijven=22;0,031;0,99727273;21,046;0,141000003;;12;0,034000
```

Intermediate transactie data (statistieken) \_intermediate.trs.csv

Het 'value' deel van transactiedata (trs) is een opsomming van statistische getallen

```
Summary_General---Active_users=1.00000000,1.00000000,1.00000000,1.00000000,1.00000000,2.00000000,2.00000000,  
Summary_General---Active_users_-_Web=1.00000000,1.00000000,1.00000000,1.00000000,1.00000000,2.00000000,2.000  
Summary_General---Transactions=2.00000000,0.00000000,0.00000000,0.00000000,0.00000000,1.00000000,6.00000000,  
Summary_General---Errors=0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,0.00000000,1.00000000,0.000  
Summary_Internet---Request_data_sent[kB]=null,15.00000000,0.00000000,0.00000000,0.00000000,0.00000000,1.000  
Summary_Internet---Response_data_received[kB]=null,234.00000000,0.00000000,0.00000000,0.00000000,0.00000000,  
Summary_Internet---Requests_sent=null,18.00000000,0.00000000,0.00000000,0.00000000,0.00000000,5.00000000,17  
Summary_Internet---Responses_received=null,17.00000000,0.00000000,0.00000000,0.00000000,0.00000000,5.0000000  
Summary_Internet---Connects_successful=null,18.00000000,0.00000000,0.00000000,0.00000000,0.00000000,5.000000
```

Intermediate measure data (timeseries) \_intermediate.msr.csv

Het 'value' deel van measure data (msr) is een serie met meetgegevens, één getal (avg) per iteratie.

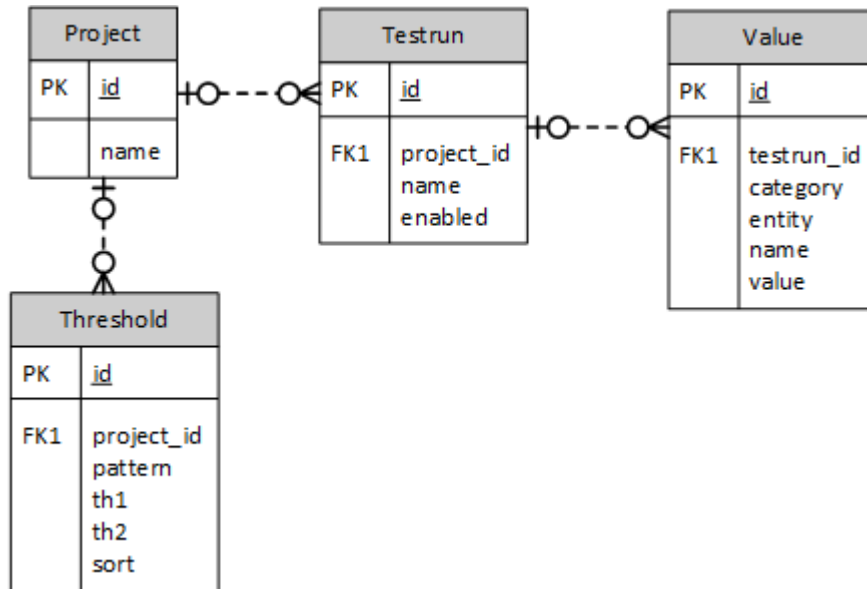
```
printableTime=2017.06.29.13.01.29  
testrundatetime=2017.06.29.13.01.29  
testduration=00:41:00  
Merged users=12  
loadgeneratortype=Silk Performer
```

Intermediate variabelen data \_intermediate.var.csv

Het key=value format is het meest duidelijk zichtbaar in de variabelen intermediate csv.

### 3.3.4.2. Database

De intermediate format data wordt opgeslagen in een PostgreSQL database, deze bevat vier tabellen.



Datamodel database

Project	per project een row name=project afkorting (moet uniek zijn)
Testrun	Alle testruns behorende bij een project (1:n) name=datetime van de test (moet uniek zijn) enabled=zichtbaar of niet
Value	Alle key-value paren behorende bij een testrun (1:n) key=value pairs per testrun category=trs, msr, var entity=servernaam e.d. name=value combinaties moeten uniek zijn
Threshold	Threshold waarden die gelden voor een project (1:n) pattern=transactienaam pattern (regex) th1=yellow threshold th2=red threshold sort=sorteerhulp, default=10

Alleen voor de threshold entiteit moet direct op de database gewerkt worden, alle overige entiteiten worden toegevoegd/gemuteerd/verwijderd via de programmatuur.

#### Logica

1. Het toevoegen van dubbele rijen wordt voorkomen door de database (unique constraint), dus dubbel laden van een testrun met bijbehorende key-value pairs kan (geeft geen fatale fouten) maar wordt op database geblokkeerd en heeft dus geen effect op de data.

2. Relaties tussen tabellen worden bewaakt door foreign key relaties. Het verwijderen van testruns waar nog value waarden aan hangen of projecten waar nog testruns voor bestaan wordt geblokkeerd door de database.

#### 3.3.4.3. HTML Template

De HTML-templates bevatten variabelenamen die met een zoek-vervang methode uit de value tabellen gehaald worden. De layout en keuze van af te drukken data wordt bepaald door de HTML-template, dus tijdens het merge-proces.

#### Variabelen uit het parse proces (var)

```
<h3>Specification</h3>
<table class="config">
  <tr><td class="name">Comment</td><td>${label}</td></tr>
  <tr><td class="name">Workload</td><td>${workload}</td></tr>
  ..
</table>
```

#### Vluchtige variabelen, bestaan alleen tijdens het mergen (\_var)

```
<h3>Specification</h3>
<table class="config">
  ...
  <tr><td class="name">Baseline used</td><td>${_baselineref}</td></tr>
  <tr><td class="name">Baseline reason</td><td>${_baselinereason}</td></tr>
  <tr><td class="name">Max trend items</td><td>${maxtrendcount}</td></tr>
</table>
```

#### Transactievariabelen met statistische reeks (var:2), kleurcode (var\_c) en getalformat (:0.000)

```
</tr>
<tr valign="top" class="">
  <td class="name">ISZW_01_OpenStartPagina</td>
  <td>${ISZW_01_OpenStartPagina:0}</td>
  <td class="${ISZW_01_OpenStartPagina_c:1}">${ISZW_01_OpenStartPagina:1:0.000}</td>
  <td class="${ISZW_01_OpenStartPagina_c:2}">${ISZW_01_OpenStartPagina:2:0.000}</td>
```

\${variabelenaam:0} = variabele index 0 (count\*)

\${variabelenaam\_c:1} = variabele index 1 (minimum\*) color code (groen, geel, rood)

\${variabelenaam:1:0.000} = variabele index 1\* met decimaal format 0.000

\* zie typen data

#### Herhaling bij datareeksen \${var:#}

```
...
  <h2>Trend (ms)</h2>
<table cellpadding="5" cellspacing="2" border="0" class="details">
  <tr valign="top">
    <th>Transaction</th>${<th>${printableTime:#}</th>}
  </tr>
  <tr valign="top" class="">
    <td class="name">ISZW_01_OpenStartPagina</td>
    ${<td class="${ISZW_01_OpenStartPagina_c:#}">${ISZW_01_OpenStartPagina:#:0.000}</td>}
  </tr>
  <tr valign="top" class="">
    <td class="name">ISZW_01_OpenStartPagina2</td>
    ${<td class="${ISZW_01_OpenStartPagina2_c:#}">${ISZW_01_OpenStartPagina2:#:0.000}</td>}
```

```
...</tr>
```

Wanneer voor een serie metingen dezelfde waarde moet worden afgedrukt, bijvoorbeeld het 90 percentiel van een transactie voor elke x laatste metingen, wordt herhaling gebruikt `$(dit wordt herhaald)`. De # is de waarde uit de meting reeks (0=1<sup>e</sup> meting, 1=2<sup>e</sup> meting...), de # wordt door de merge tool zelf vervangen door volgnummers. Welke waarde uit de statistische set wordt herhaald, wordt bepaald op command-line in de merge tools `srt.mergetransactions.exe`, `srt.mergevariables.exe` ... Bijvoorbeeld 90 percentile: **indexvalue=4**, zie Typen Data.

Let op: de open- en sluitkarakters van de variabelen {} en de herhaling [] wordt gezocht op dezelfde regel. Er wordt niet over regels heen gezocht naar een sluit-tag. Deze restrictie geldt omwille van eenvoud en voldoet voor nu. Deze logica kan worden verstoord door het gebruik van HTML-formattering tools.

#### Default waarde bij 'geen waarde'

```
...
    name: '00200_opstartpagina_in',
    yAxis: 1,
    data: [null$[, ${00200_opstartpagina_in:#:0.000:null}]],
    tooltip: {
        valueSuffix: ' s'
    }
}
```

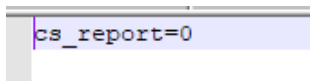
Er is een derde formattering parameter mogelijk bij variabelen (hier: null), dit is een default waarde. Deze string wordt gebruikt wanneer de data serie op dit punt geen data bevat, met name handig bij measures (timeseries) voor het afdrukken van 'niet gemeten' waarden.

#### 3.3.4.4. Foutopsporing

Wanneer bij het genereren van een rapport fouten zijn opgetreden, kijk dan eerst naar de inhoud van bestand:

```
..\03_Resultaten\<project>\report\<project>_report.exitcode
```

Voorbeeld:



```
cs_report=0
```

Boven: inhoud `cs_report.exitcode`

Exit codes:

- 0 = goed
- 1 = fout tijdens verzamelen van data (missing file error)
- 2 = fout tijdens parsen van data (parse error)
- 3 = fout tijdens het mergen van database data en template (merge error)
- 4 = fout tijdens het kopiëren van resultaat naar target directory (wrap-up error)

Bij het genereren van een rapport wordt in de html target directory een log aangemaakt met daarin de logregels die alle sub-tools hebben gegenereerd naar de console. Hierin is de progressie van de diverse fases te volgen en hierin worden ook exceptions gelogd wanneer ze optreden.

```
..\ReportTool\tools\report\<projectcode>.report.log
```

```
### srt.loadintermediate <project> <testrun> <category> <entity> <int
args: project=cs testrun=2017.06.29.13.01.29 category=var entity=runi
number of args: 5
function option = NULL
function switch = NULL
project=cs
testrun=2017.06.29.13.01.29
category=var
entity=runinfo
intermediatefile=C:\Data\Projecten\ICTU\TestStraat\ReportTool\temp\_i
Read intermediate...
C:\Data\Projecten\ICTU\TestStraat\ReportTool\temp\_intermediate.var.:
2 lines
use database on localhost:5432
connect to database teststraat...
storing cs|2017.06.29.13.01.29|var|svntag...
double row blocked by database
storing cs|2017.06.29.13.01.29|var|workload...
double row blocked by database
### srt.loadintermediate finished
```

Boven: voorbeeld cs\_report.log

## 4. Usecases

### 4.1. Toevoegen van een nieuw project

1. Toevoegen van een script directory (voorbeeld testproject)
  - a. `..\02_Scripts\<projectnaam>`
  - b. Plaatsen van de scripts in deze folder
2. Toevoegen van een project directory met inhoud (voorbeeld van testproject)
  - a. `..\09_ProjectFolders\<projectnaam>`
  - b. `checks.sh`, `getparams.sh`, `posttest.sh`, `Transactions.csv`, `vars.incl`
3. Configuratie van het project
  - a. Vullen van transactienamen in `<projectfolder>\Transactions.csv`, zie par. 4.3.
  - b. Configuratie van loadgenerator en project specifieke parameters in `<projectfolder>\vars.incl`. Globale variabelen kunnen hierin per project en optioneel worden overschreven.
4. Eventueel: toevoegen of aanpassen van requirements aan responstijden
  - a. Open Postgres database admin tool, voeg per transactie in het juiste project specifieke requirements toe aan tabel 'threshold', zie par. 4.4.

### 4.2. Eisen aan een testscript

Het testscript moet aan een beperkt aantal voorwaarden voldoen om te kunnen functioneren in de teststraat.

#### 4.2.1. Technologie: JMeter of Silkperformer

De teststraat ondersteunt momenteel JMeter en Silkperformer. Voor Silkperformer testen is een beperkt aantal licenties on-premise beschikbaar. De maximaal in te stellen belasting is afhankelijk van het beschikbaar aantal licenties. Het uitgangspunt is dat nieuwe projecten worden gemaakt met JMeter (afbouwen Silkperformer), gebruik alleen Silkperformer als JMeter niet voldoet. Op die manier hopen we in de toekomst volledig over te gaan naar JMeter (open source). Jmeter wordt gebruikt in combinatie met Google Jmeterplugins.

#### 4.2.2. Gebruik van loadgeneratoren

##### JMeter

Jmeter draait vanaf de controller en maakt (nog) geen gebruik van aparte loadgeneratoren voor gedistribueerd testen.

##### Silkperformer

De Silkperformer 'controller' component draait op de teststraat machine, het verkeer wordt gegenereerd vanaf een aantal loadgeneratoren. Het gebruik van deze loadgeneratoren moet geconfigureerd worden in het testscript (workload). Let erop dat verschillende typen testen draaien vanaf andere testclusters. Het gebruik van loadgeneratoren voor specifieke workloads wordt geconfigureerd in het Silkperformer project.

#### 4.2.3. Configuratie van workload

Voor elk script worden vier workloads voorgedefinieerd: Productie, Duurtest, Stresstest, Verificatie.

- Productie: belastingniveau is productie piekbelasting, doorlooptijd: zo lang nodig is om voldoende samples te verzamelen, uitgangspunt is 1 uur
- Duurtest: belastingniveau 'Productie', doorlooptijd 6 uur
- Stresstest: ramp-up over een heel uur tot 5x of 10x belastingniveau 'productie', doorlooptijd 1 uur
- Verificatie: korte test, elk script 1x doorlopen single user tbv compatibiliteit script-applicatie

### JMeter

Maak een kopie van het testplan voor elke workload, dus per project vier jmx bestanden volgens template:

<project>\_<workloadnaam>.jmx

lrk\_productie.jmx + lrk\_duurtest.jmx + lrk\_stresstest.jmx + lrk\_verificatie.jmx

Maak het jezelf makkelijk door in het script een set parameters te definiëren in een configuratieset die enabled of disabled kan worden. Enable dan voor elke kopie een andere workload configuratieset.

### Silkperformer

Maak in de Silkperformer test vier voorgedefinieerde workload configuraties: productie, stresstest, duurtest en verificatie en definieer dus vier configuratieset van aantal vusers, orkestratie en iteratieduur van de scripts en gebruikte loadgeneratoren. Het resultaat is één testscript met daarin vier workload definities.

## 4.3. Toevoegen, wijzigen of wijzigen van een transactienaam

Voeg de nieuwe of gewijzigde transactienaam toe in projectbestand 'transactions.csv':

E:\09\_ProjectFolders\<project>\transactions.csv

Voorbeeld:

```
00100_opstartpagina_op
00110_btnlogin
HEADER
01000_hoofdpagina_pp
01010_linkzoekscher
01020_kiessoortopvang
01030_btnzoeken
```

De test zal een nieuwe of gewijzigde transactienaam gaan opleveren, de rapportgenerator haalt alle transacties uit het testresultaat, maar welke transactie moet worden opgenomen in het gegenereerde rapport hangt af van de lijst in deze .csv.

Transacties die niet opgenomen hoeven worden in het rapport kunnen hier weggelaten worden. Alle transacties wegen echter mee in de berekening van grand-totals.

#### 4.4. Toevoegen of wijzigen van thresholds

Standaard worden generieke drempelwaarden toegepast (generic). Deze gelden als basis thresholds voor transacties waarvoor geen threshold geformuleerd is. Per transactie of groep transacties kan een aparte threshold gemaakt worden op basis van reguliere expressies.

Wanneer voor de eerste keer een rapport gedraaid heeft, is een nieuw project, testrun en 'generic' threshold regel toegevoegd met standaard thresholds van 1 en 3 seconden. Let op: threshold 1 = geel = waarschuwing; threshold 2 = rood = nonfunctional. Voeg nieuwe threshold regels toe direct op de PostgreSQL database met admin tool (pgAdmin). Open de data viewer/browser van tabel 'threshold'.

Aandachtspunten:

- 1) Zorg dat referentie project\_id klopt bij het toevoegen van een nieuwe threshold regel.
- 2) De volgorde waarin de regels worden gesorteerd (sort, id) is bepalend voor de volgorde waarin de thresholds worden toegepast. Voor elke transactie (naam) wordt de thresholdlijst van boven naar onder doorlopen, de laatste match wordt toegepast op de transactie. Dus zet algemene patterns bovenaan, specifieke onderaan.

id [PK] serial	project_id integer	pattern text	th1 real	th2 real	sort numeric
9	1	generic	1	3	10
13	1	RB01	1	1	10
15	1	DV01	1	1	10
16	1	BelastControle_04_Zoeken	1	3	10
17	1	Brief_03_Zoeken	1	3	10
18	1	Hand_03_Zoeken	1	3	10
19	1	Loket_18_ZoekFilter	1	6	10
20	1	LoketControle_03_ZoekenDossier	1	3	10
21	2	generic	2	4	10
22	3	generic	1	3	10
23	4	generic	1	3	10
24	2	RWS_01_OpenStartPagina	1	3	10
25	2	RWS_02_Login	2	4	10
26	2	RWS_03_vts_gebiedslijst	1	3	10
27	2	RWS_04_Logout	1	3	10
28	3	TSZW_02_OpenTaakScherm_Motities	3	4	10

Tabel 'threshold' in database 'teststraat'

#### 4.5. Disable/enable van een testrun

Wanneer een testrun mislukt is en de historie of baselining verstoort kan deze worden gedisabled zodat hij niet meer getoond wordt. Dit is een enabled vlag in de testrun tabel, deze kan worden gezet met de console tool.

Locatie van de reporttool:

E:\06\_Tools\ReportTool

Optioneel:

```
..\ReportTool\report_console.sh listtestruns  
project=<projectcode>
```

Kopieer de testruncode



Disable:

```
..\ReportTool\report_console.sh disabletestrun  
project=<projectcode>  
testrun=<testruncode>
```

Enable:

```
..\ReportTool\report_console.sh enabletestrun  
project=<projectcode>  
testrun=<testruncode>
```

#### 4.6. Toevoegen/verwijderen van een testrun

Het kan nodig zijn om een specifieke testrun met alle verbonden meetwaarden fysiek te verwijderen uit de database, dit kan worden uitgevoerd met de console tool:

Optioneel:

```
..\ReportTool\report_console.sh listtestruns  
project=<projectcode>  
Kopieer de testruncode
```

Verwijderen:

```
..\ReportTool\report_console.sh deletetestrun  
project=<projectcode>  
testrun=<testruncode>
```

#### 4.7. Toevoegen/verwijderen van een project

Verwijderen van een project met alle verbonden testruns en meetwaarden kan met de console tool:

Optioneel:

```
..\ReportTool\report_console.sh listprojects  
Selecteer de projectcode
```

Toevoegen:

```
..\ReportTool\report_console.sh createproject  
project=<projectcode>
```

Verwijderen:

```
..\ReportTool\report_console.sh deleteproject  
project=<projectcode>
```

#### 4.8. Wijzigen van bestandslocaties

De rapportgenerator gaat uit van bestandslocaties die geconfigureerd zijn in de teststraat globals. Dit is een standaard locatie binnen de teststraat, maar kan eventueel gewijzigd worden. De rapportgenerator heeft geen eigen configuratie, maar haalt zijn variabelen uit deze globale variabelenlijst.

```
..\00_Globals\testautomation_globals.incl
```

reporttemplatedestinationfolder	bronlocatie template
loadtest_logdir	bronlocatie meetwaarden
loadtest_measures	bronlocatie meetwaarde bestanden
loadtest_report	doellocatie rapport

#### 4.9. Wijzigen database locatie

Een aantal reportgenerator tools maken gebruik van de PostgreSQL database, vooralsnog alleen merge, loadintermediate en console. De database naam is 'teststraat', de machine, poort, username en password wordt gehaald uit de globals. Poort nummer is optioneel. Default is 5432, maar als een afwijkend nummer is gebruikt kan deze als tweede parameter worden toegevoegd. Als de default poort gebruikt is kan met drie parameters worden volstaan.

```
..\00_Globals\testautomation_globals.incl
```

```
reportdbconnectstring=<machine>:<port>:<username>:<password>
of, wanneer default port 5432 gebruikt wordt:
reportdbconnectstring=<machine><username>:<password>
```

## 5. Bijlage 1: Report Generator tools

De rapportgenerator bestaat uit een aantal command-line tools die worden aangeroepen vanuit shell scripts. Zij maken deel uit van het van project Report Generator (rpg). De tools zijn gebaseerd op .Net Core en draaien dus zowel op Windows als op Linux, zolang de platform specifieke versie van .NetCore geïnstalleerd is. Alle extra voorzieningen die nodig zijn (Entity Framework, Application tools, PostgreSQL drivers) zijn meegeleverd met de applicatie. De tools worden gestart met "dotnet <tool> <parameters>"

### Parameters

Voor de aanroep van de tools geldt dat de benodigde command-line parameters kunnen worden meegegeven bij aanroep van de tool, of interactief met vraag-en-antwoord interactie. Wanneer parameters verplicht zijn maar niet zijn meegegeven, wordt erom gevraagd in vraag-antwoord interactie.

### rpg.common.dll

library, bevat gedeelde functionaliteit

### rpg.console.dll (gebruikt vanuit report\_console.sh)

rpg.console.dll <functie> <parameters> database=<database connect string>

Voor speciale bewerkingen aan database data:

1. **listprojects** = overzicht van projecten in de database
2. **createproject/deleteproject** = creëren / verwijderen\* van een project
3. **listtestruns** = overzicht van testruns onder een project
4. **enabletestrun/disabletestrun** = enable/disable testrun (tijdelijk onzichtbaar maken)
5. **deletetestrun** = verwijderen van een specifieke testrun\* (verwijderen van mislukte test)

\* bij het verwijderen van data worden ook alle onderliggende gegevens (cascading) verwijderd.

Voorbeeld weergegeven overzicht van testruns onder een project

```
$ . report_console.sh listtestruns
project=lrk
read testruns into reference list...
Enabled:
2018.02.14.14.36.36
2018.03.28.15.52.33
2019.04.30.11.43.34
2019.05.07.03.49.44
2019.05.10.04.26.25
2019.05.21.03.14.51
Disabled:
2017.11.01.16.42.42
2018.02.13.11.46.55
2018.03.07.18.18.42
2018.03.28.13.40.26
```

Voorbeeld disable van een testrun

```
$ . report_console.sh disabletestrun
project=lrk
testrun=2019.01.12.19.54.57
read testruns into reference list...
disable testrun 2019.01.12.19.54.57
```

### **rpg.parsemeasures.dll**

Parsen van 'measures' uit brondata naar 'intermediate' format. Measures zijn timeseries gegevens die kunnen worden afgedrukt in een tijd-waarde grafiek. Measures zijn bijvoorbeeld gemiddelde responstijd, aantal momentane threads/vusers, errors, als functie van tijd. Alle parsers zijn beschikbaar in deze ene tool, het type loadgenerator wordt opgegeven met parameter 'parser'. Voor het parsen van Silkperformer data, gebruik parser=silk.

#### Voorbeeld

```
dotnet rpg.parsemeasures.dll parser=jmeter  
transactionfilejtl=_transactions.jtl intermediatefile=_intermediate.msr.csv
```

#### Parameters

Parser = 'jmeter' of 'silkperformer'

Transactionfilejtl = input file (jmeter log)

Intermediatefile = output file met intermediate format data

### **rpg.parsetransactions.dll**

Parsen van 'transactiedata' uit brondata naar 'intermediate' format. Transactiedata bestaat uit een set statistische metrieken per transactie: aantal, min, max, avg... Een regel per transactie.

#### Voorbeeld

```
dotnet rpg.parsetransactions.dll parser=jmeter  
transactionfilecsv_success=_transactions_success.csv  
transactionfilecsv_all=_transactions_all.csv  
intermediatefile=_intermediate.trs.csv
```

#### Parameters Jmeter

Parser=jmeter

Transactionfilecsv\_all = CSV export (door Jmeter) met alle transactiedata

Transacitonfilecsv\_success = CSV export (door jmeter) met alle transactiedata, excl fouten

Intermediatefile = output file met intermediate format

#### Parameters Silkperformer

Parser=silkperformer

Tranasactionfilebrp= BRP file, testresultaat

Transactionfilecsv = CSV export (door Silkperformer) van de samengevoegde TSD

Intermediatefile = output file met intermediate format

### **rpg.parsevariables.dll**

Parsen van variabelen (testduur, startdatum/tijd, tijdinterval, aantal users...) uit brondata naar 'intermediate' format.

Voorbeeld:

```
dotnet rpg.parsevariables.dll parser=jmeter  
transactionfilejtl=_transactions.jtl  
transactionfilecsv=_transactions_all.csv  
intermediatefile=_intermediate.var.csv
```

Parameters Jmeter

Parser=jmeter

Transactionfilejtl = ruwe (xml !) output van jmeter

Transactionfilecsv = geconverteerde output van jmeter (door jmeter)

Intermediatefile = output file met intermediate format

Parameters Silkperformer

Parser = silkperformer

Transactionfilebrp= BRP file, testresultaat

Transactionfilecsv = CSV export (door Silkperformer) van de samengevoegde TSD

Intermediatefile = output file met intermediate format

### **rpg.loadintermediate.dll**

Laden van intermediate .csv bestanden naar de database.

Voorbeeld

```
dotnet rpg.loadintermediate.dll project=vastgoed  
testrun=2018.12.17.01.15.17 category=var entity=runinfo  
intermediatefile=_intermediate.var.csv
```

Bovenstaande regel leest de key=value pairs uit \_intermediate.var.csv en slaat ze op in de database onder het juiste project/testrun, in variabele categorie 'var' en entiteit 'runinfo'. De categorie en entiteit hebben inhoudelijk geen waarde, maar worden later bij het mergen van data in de HTML gebruikt voor het selectief invoegen van data (filter).

### **rpg.merge.dll**

Samenvoegen van data uit de database in variabelen in de HTML-template. Variabelen van een specifieke categorie worden ingevoegd in de template en opgeslagen onder dezelfde naam.

Deze tool bevat van alle rpg tools de meeste flexibiliteit, deze zijn te beïnvloeden door verschillende 'modi' waarin de tool wordt gestart. Dit is de eerste command-line parameter, in onderstaand voorbeeld 'i', dit staat voor 'intermediate'. Verschillende soorten data worden namelijk op verschillende manieren behandeld. Het grootste verschil is het onderscheid in 'eenvoudige' waarden en 'timeseries' waarden. Een waarde kan 'eenvoudig' zijn, een set statistische waarden bevatten (transactie: count, min, max...), of een timeseries reeks (meetwaarden).

Voorbeeld

```
dotnet rpg.merge.dll i project=vastgoed testrun=2018.12.17.01.15.17  
category=var templatefile=work.html
```

#### Functie switches

i = intermediate (enkelvoudig type)

it = intermediate van het type transactie (met statistische set: count, min, max...)

ib = intermediate baseline waarden voor 'it' waarden

t = threshold waarden

v = enkelvoudige variabele command-line parameter

jt = joined transactie waarden (historisch overzicht uit transactie statistische set)

j = joined enkelvoudige waarden

#### Overige parameters

category = het type veld (filter) uit de teststraat database