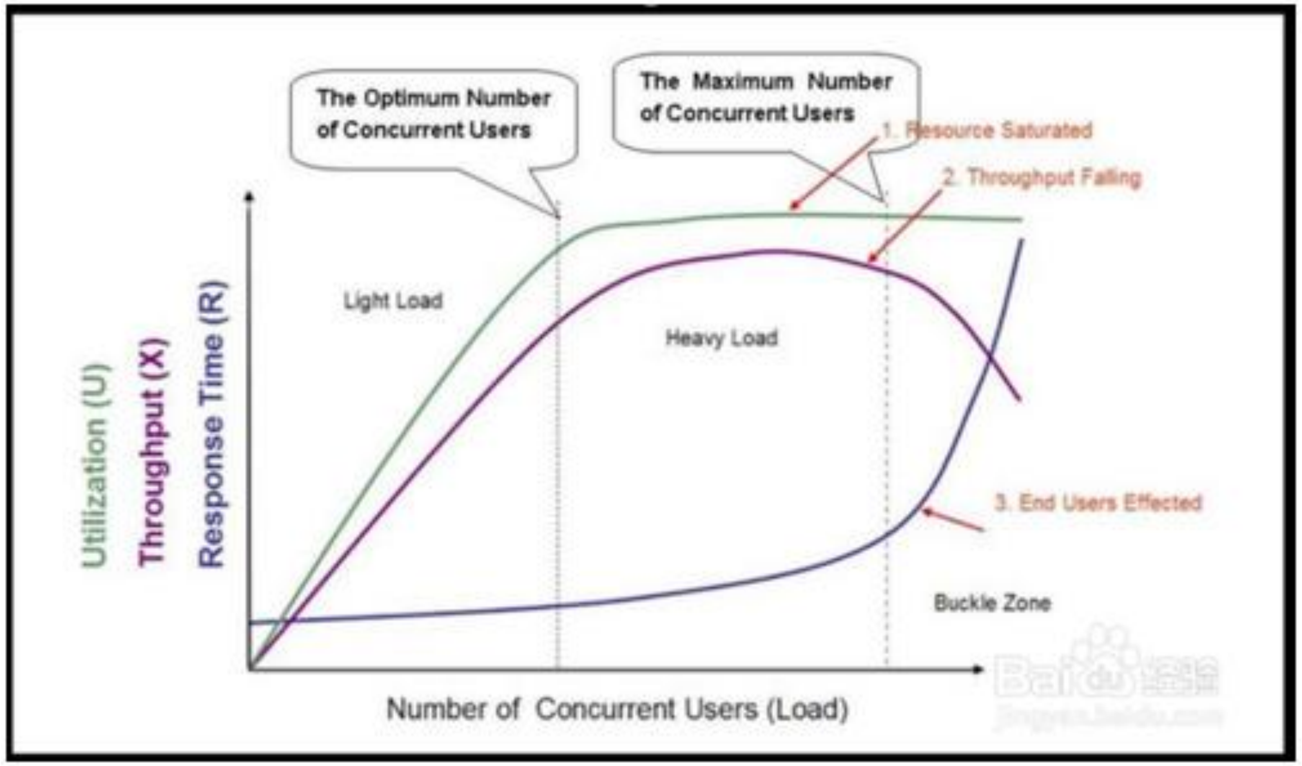
Performance Test

David LI

什么是性能自动化测试?

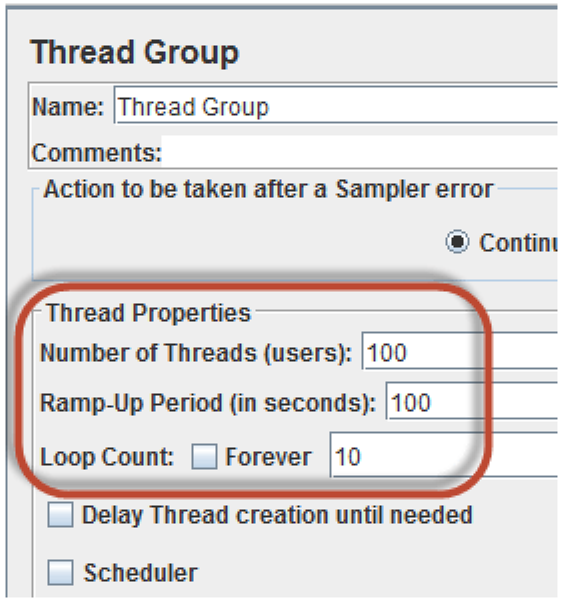
- 目标
  - Performance test(speed and stability)
  - Pressure test(run large data with low resource)
  - Loading test(run large data for long time)
  - Volume test(to identify maximum users)
  - Scalability test
  - Configuration test(to identify optimal configuration)
- 指标
  - Throughput
  - Response time
  - CPU/Memory/Disk usage/Network/Database
- 工具
  - JMeter/loadrunner

通常需要一个团队的协作，包括QA、Developer、CM


The Optimum Number of Concurrent Users
The Maximum Number of Concurrent Users
1. Resource Saturated
2. Throughput Falling
3. End Users Effected
Utilization (U)
Throughput (X)
Response Time (R)
Light Load
Heavy Load
Buckle Zone
Number of Concurrent Users (Load)

Thread Group

- **Number of Threads**: 100 (Number of users connects to the target website: 100)
- **Loop Count**: 10 (Number of time to execute testing)
- **Ramp-Up Period**: 100



Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

(●) Continu

Thread Properties

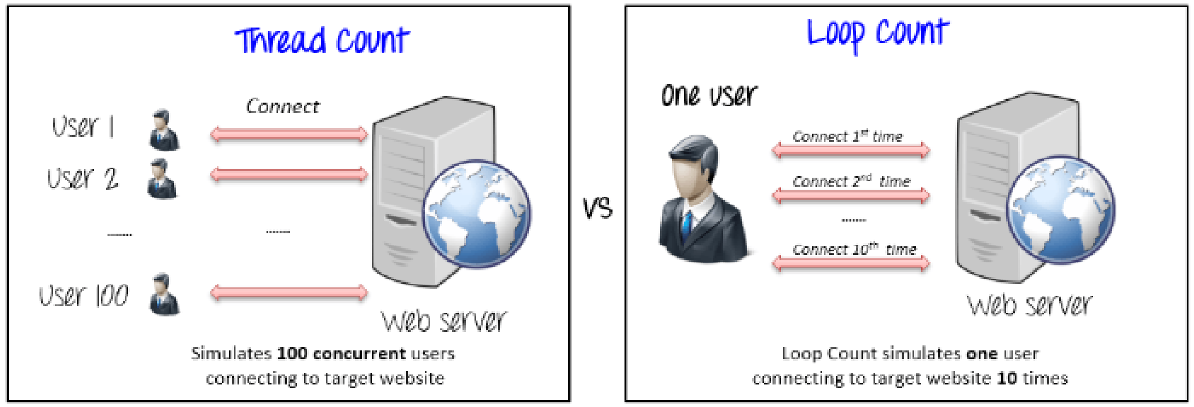
Number of Threads (users): 100

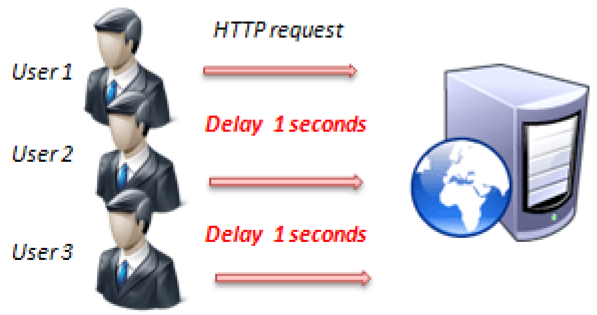
Ramp-Up Period (in seconds): 100

Loop Count: ☐ Forever 10

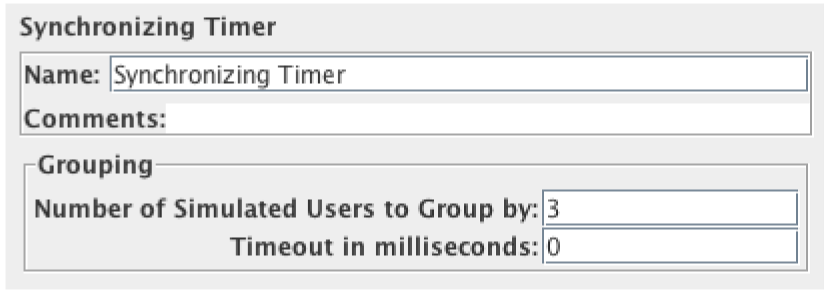
☐ Delay Thread creation until needed

☐ Scheduler





Synchronizing Timer



Synchronizing Timer

Name: Synchronizing Timer

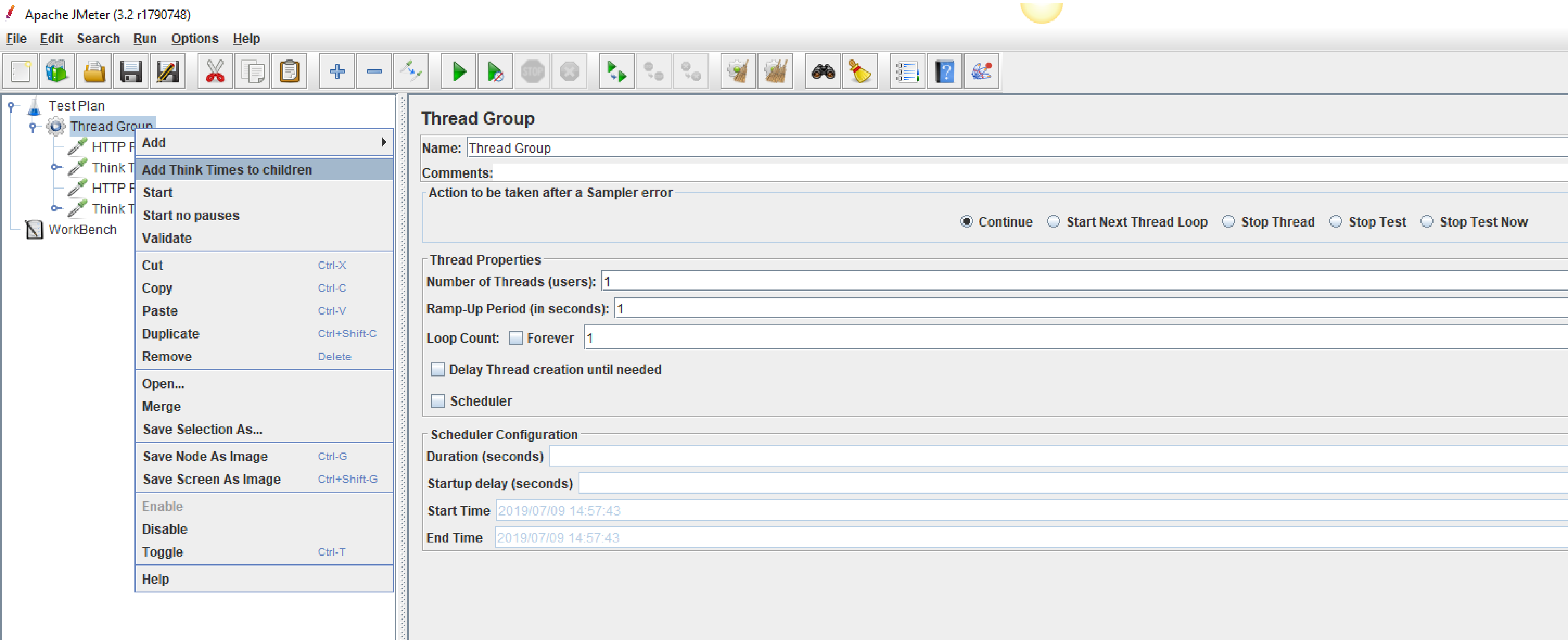
Comments:

Grouping

Number of Simulated Users to Group by: 3

Timeout in milliseconds: 0

Think Time



Apache JMeter (3.2 r1790748)

File Edit Search Run Options Help

Test Plan
Thread Group
HTTP R
Think T
HTTP R
Think T
WorkBench

Add	
Add Think Times to children	
Start	
Start no pauses	
Validate	
Cut	Ctrl-X
Copy	Ctrl-C
Paste	Ctrl-V
Duplicate	Ctrl+Shift-C
Remove	Delete
Open...	
Merge	
Save Selection As...	
Save Node As Image	Ctrl-G
Save Screen As Image	Ctrl+Shift-G
Enable	
Disable	
Toggle	Ctrl-T
Help	

Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

[✓ Continue] Start Next Thread Loop Stop Thread Stop Test Stop Test Now

Thread Properties

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: ☐ Forever 1

☐ Delay Thread creation until needed

☐ Scheduler

Scheduler Configuration

Duration (seconds)

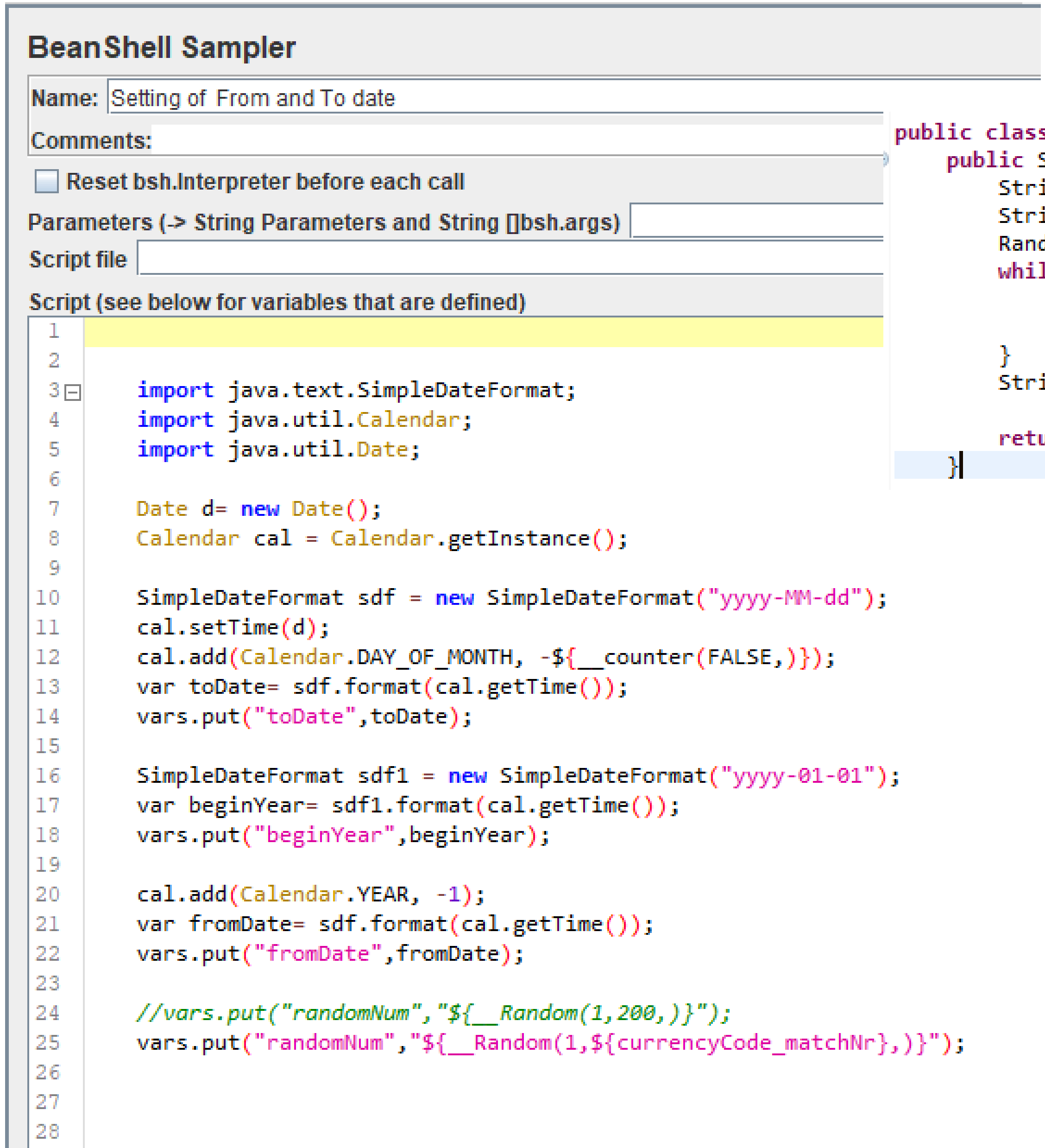
Startup delay (seconds)

Start Time 2019/07/09 14:57:43

End Time 2019/07/09 14:57:43

Dynamic generated test data

- Random
- Counter
- Thread Number
- Time



BeanShell Sampler

Name: Setting of From and To date

Comments:

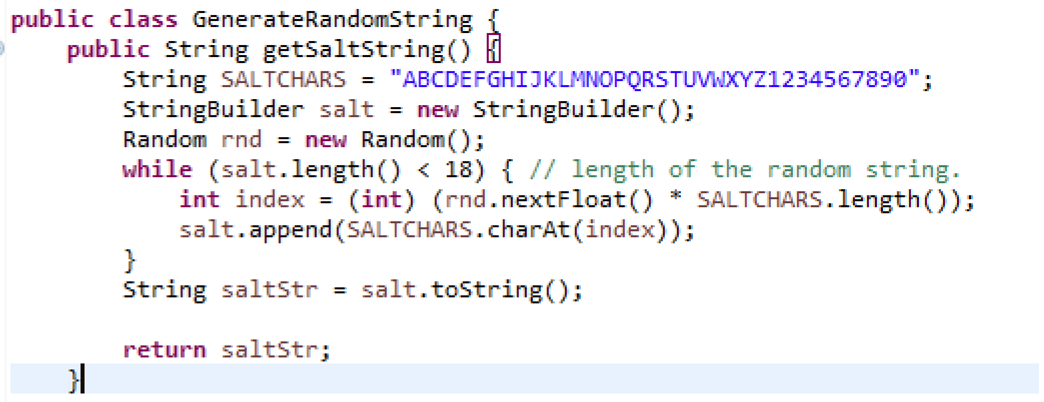
☐ Reset bsh.Interpreter before each call

Parameters (-> String Parameters and String []bsh.args)

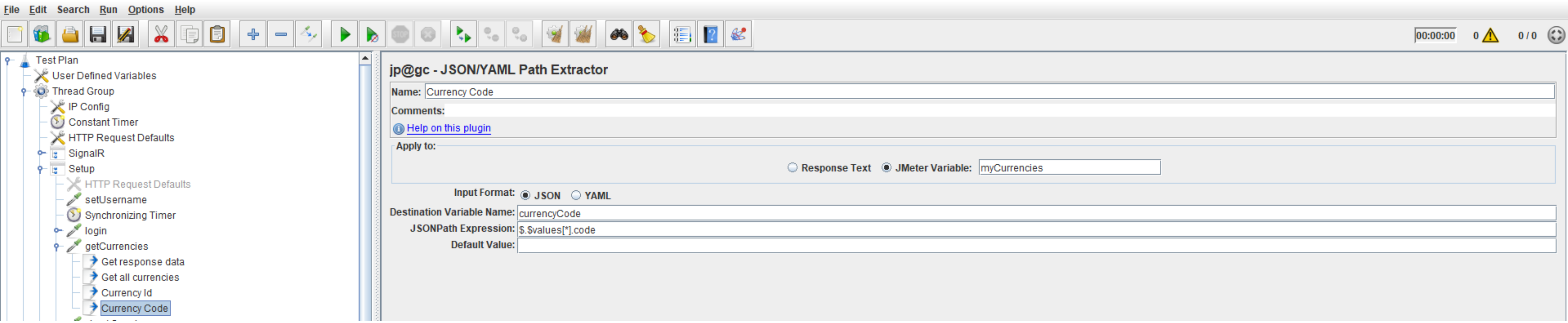
Script file

Script (see below for variables that are defined)

```


import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

Date d= new Date();
Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
cal.setTime(d);
cal.add(Calendar.DAY_OF_MONTH, -${__counter(FALSE,)});
var toDate= sdf.format(cal.getTime());
vars.put("toDate",toDate);

SimpleDateFormat sdf1 = new SimpleDateFormat("yyyy-01-01");
var beginYear= sdf1.format(cal.getTime());
vars.put("beginYear",beginYear);

cal.add(Calendar.YEAR, -1);
var fromDate= sdf.format(cal.getTime());
vars.put("fromDate",fromDate);

//vars.put("randomNum","${__Random(1,200,)}");
vars.put("randomNum","${__Random(1,${currencyCode_matchNr},)}");
```



```
public class GenerateRandomString {
    public String getSaltString() {
        String SALTCHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
        StringBuilder salt = new StringBuilder();
        Random rnd = new Random();
        while (salt.length() < 18) { // length of the random string.
            int index = (int) (rnd.nextFloat() * SALTCHARS.length());
            salt.append(SALTCHARS.charAt(index));
        }
        String saltStr = salt.toString();

        return saltStr;
    }
```



- Test Plan
  - User Defined Variables
  - Thread Group
    - IP Config
    - Constant Timer
    - HTTP Request Defaults
    - SignalR
    - Setup
      - HTTP Request Defaults
      - setUsername
      - Synchronizing Timer
      - login
      - getCurrencies
        - Get response data
        - Get all currencies
        - Currency Id
        - Currency Code

jp@gc - JSON/YAML Path Extractor

Name: Currency Code

Comments:

Help on this plugin

Apply to:

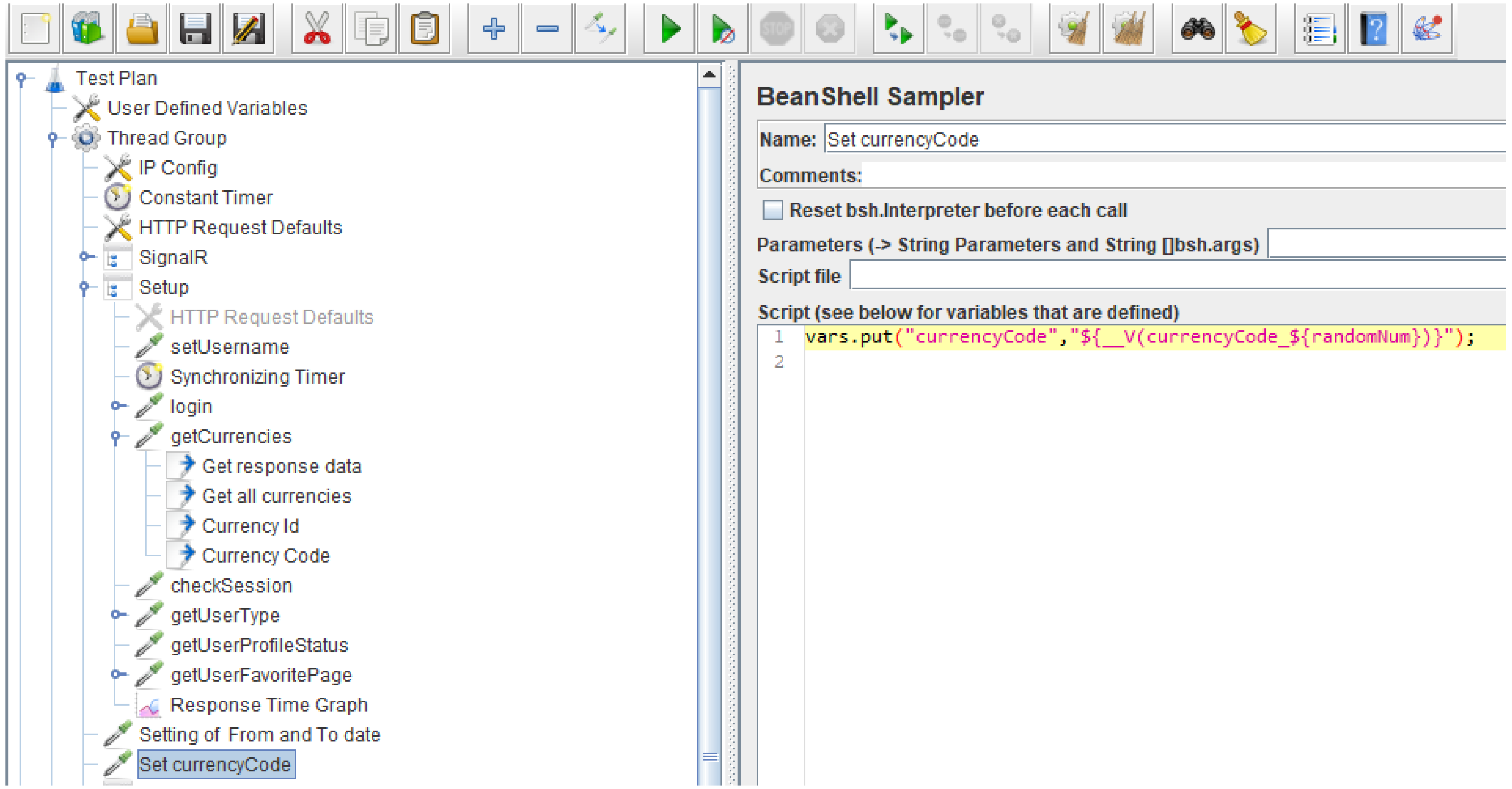
Response Text [✓ JMeter Variable:] myCurrencies

Input Format: [✓ JSON] YAML

Destination Variable Name: currencyCode

JSONPath Expression: $.$values[*].code

Default Value:



- Test Plan
  - User Defined Variables
  - Thread Group
    - IP Config
    - Constant Timer
    - HTTP Request Defaults
    - SignalR
    - Setup
      - HTTP Request Defaults
      - setUsername
      - Synchronizing Timer
      - login
      - getCurrencies
        - Get response data
        - Get all currencies
        - Currency Id
        - Currency Code
      - checkSession
      - getUserType
      - getUserProfileStatus
      - getUserFavoritePage
      - Response Time Graph
    - Setting of From and To date
    - Set currencyCode

BeanShell Sampler

Name: Set currencyCode

Comments:

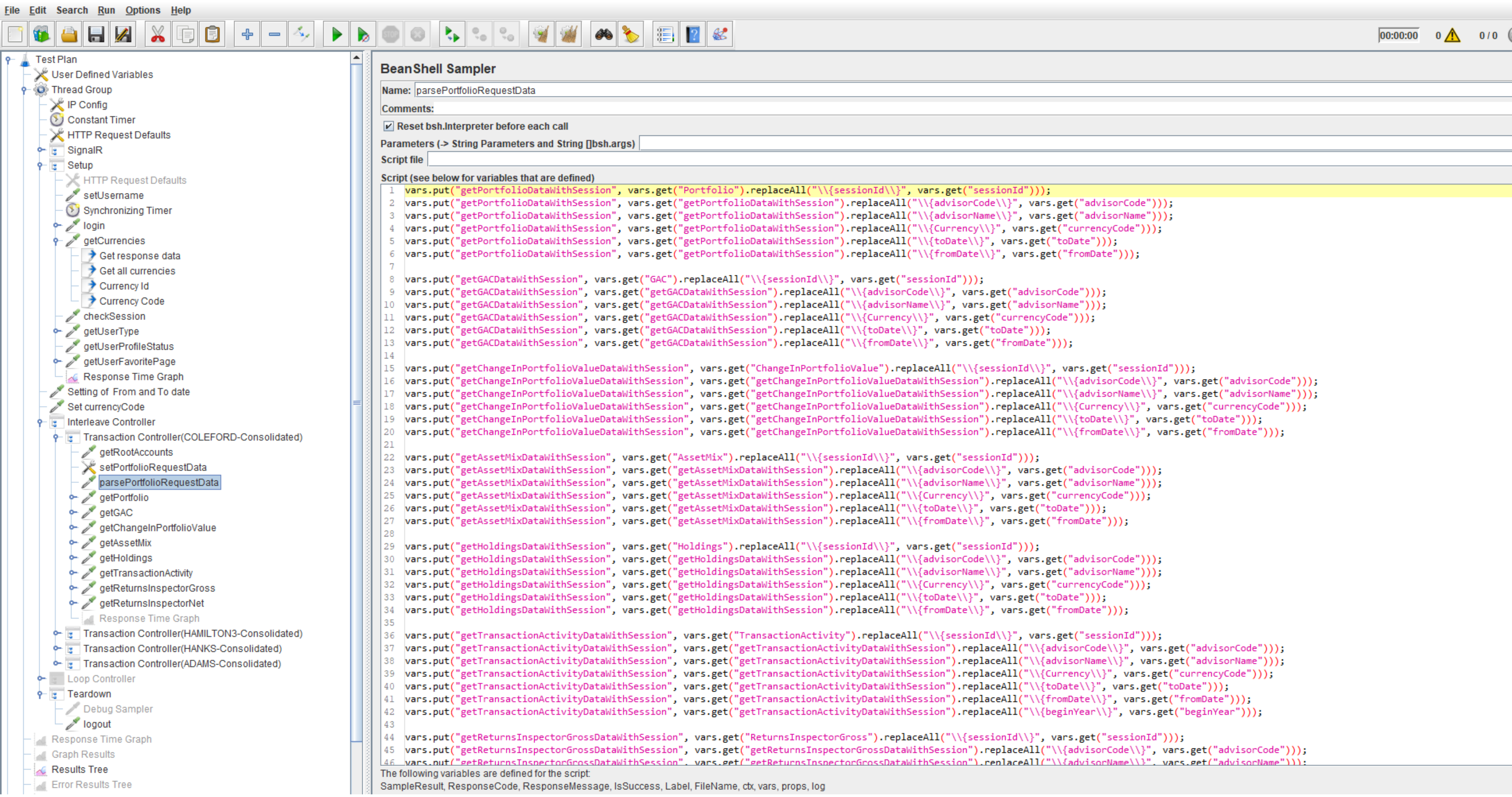
☐ Reset bsh.Interpreter before each call

Parameters (-> String Parameters and String []bsh.args)

Script file

Script (see below for variables that are defined)

```
vars.put("currencyCode","${__V(currencyCode_${randomNum})}");
```



BeanShell Sampler

Name: parsePortfolioRequestData

Comments:

☑ Reset bsh.Interpreter before each call

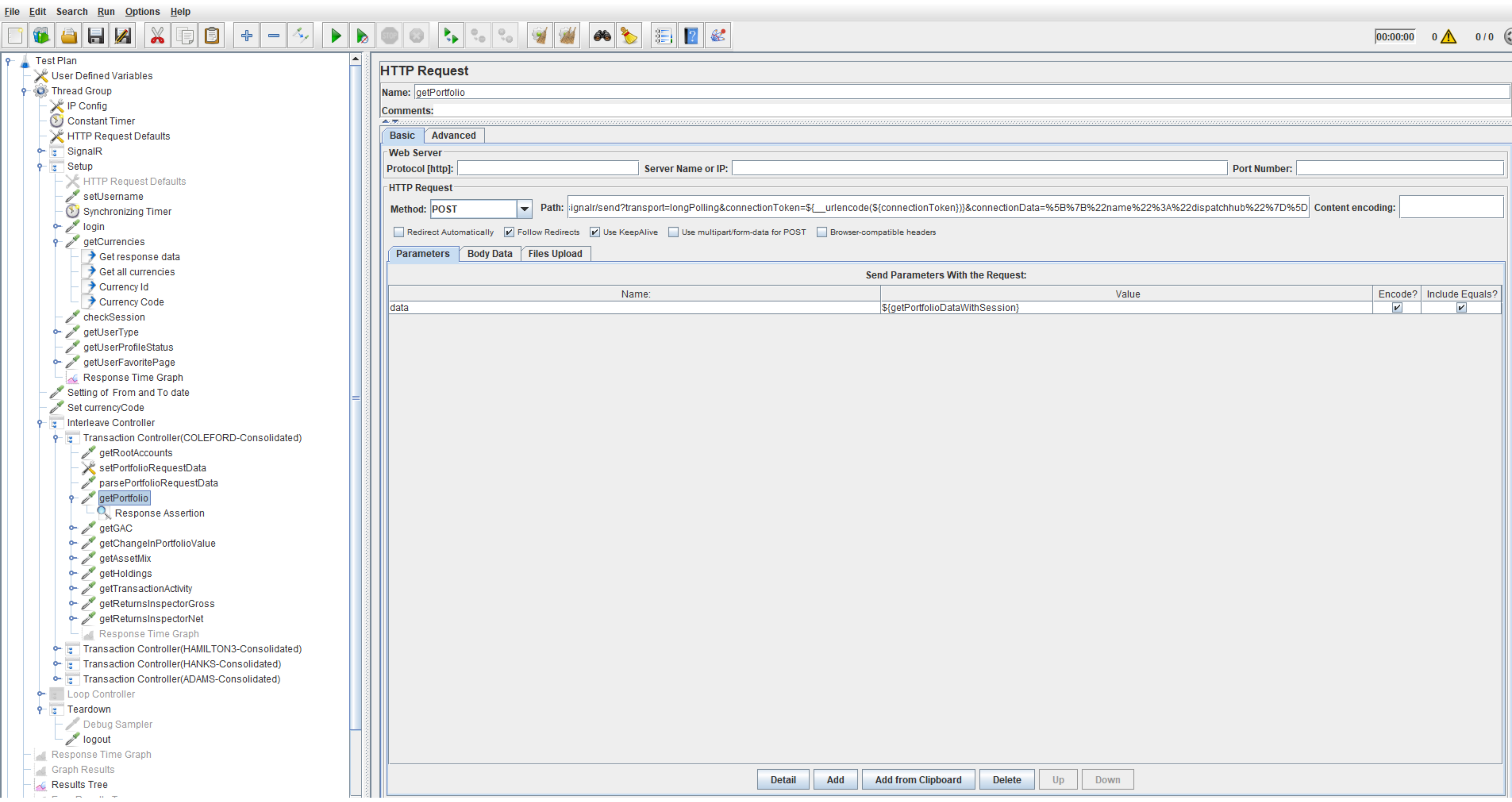
Parameters (-> String Parameters and String []bsh.args)

Script file

Script (see below for variables that are defined)

```
vars.put("getPortfolioDataWithSession", vars.get("Portfolio").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getPortfolioDataWithSession", vars.get("getPortfolioDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getPortfolioDataWithSession", vars.get("getPortfolioDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getPortfolioDataWithSession", vars.get("getPortfolioDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getPortfolioDataWithSession", vars.get("getPortfolioDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getPortfolioDataWithSession", vars.get("getPortfolioDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));

vars.put("getGACDataWithSession", vars.get("GAC").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getGACDataWithSession", vars.get("getGACDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getGACDataWithSession", vars.get("getGACDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getGACDataWithSession", vars.get("getGACDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getGACDataWithSession", vars.get("getGACDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getGACDataWithSession", vars.get("getGACDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));

vars.put("getChangeInPortfolioValueDataWithSession", vars.get("ChangeInPortfolioValue").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getChangeInPortfolioValueDataWithSession", vars.get("getChangeInPortfolioValueDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getChangeInPortfolioValueDataWithSession", vars.get("getChangeInPortfolioValueDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getChangeInPortfolioValueDataWithSession", vars.get("getChangeInPortfolioValueDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getChangeInPortfolioValueDataWithSession", vars.get("getChangeInPortfolioValueDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getChangeInPortfolioValueDataWithSession", vars.get("getChangeInPortfolioValueDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));

vars.put("getAssetMixDataWithSession", vars.get("AssetMix").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getAssetMixDataWithSession", vars.get("getAssetMixDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getAssetMixDataWithSession", vars.get("getAssetMixDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getAssetMixDataWithSession", vars.get("getAssetMixDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getAssetMixDataWithSession", vars.get("getAssetMixDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getAssetMixDataWithSession", vars.get("getAssetMixDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));

vars.put("getHoldingsDataWithSession", vars.get("Holdings").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getHoldingsDataWithSession", vars.get("getHoldingsDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getHoldingsDataWithSession", vars.get("getHoldingsDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getHoldingsDataWithSession", vars.get("getHoldingsDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getHoldingsDataWithSession", vars.get("getHoldingsDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getHoldingsDataWithSession", vars.get("getHoldingsDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));

vars.put("getTransactionActivityDataWithSession", vars.get("TransactionActivity").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{Currency\\}", vars.get("currencyCode")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{toDate\\}", vars.get("toDate")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{fromDate\\}", vars.get("fromDate")));
vars.put("getTransactionActivityDataWithSession", vars.get("getTransactionActivityDataWithSession").replaceAll("\\{beginYear\\}", vars.get("beginYear")));

vars.put("getReturnsInspectorGrossDataWithSession", vars.get("ReturnsInspectorGross").replaceAll("\\{sessionId\\}", vars.get("sessionId")));
vars.put("getReturnsInspectorGrossDataWithSession", vars.get("getReturnsInspectorGrossDataWithSession").replaceAll("\\{advisorCode\\}", vars.get("advisorCode")));
vars.put("getReturnsInspectorGrossDataWithSession", vars.get("getReturnsInspectorGrossDataWithSession").replaceAll("\\{advisorName\\}", vars.get("advisorName")));
```

The following variables are defined for the script:
SampleResult, ResponseCode, ResponseMessage, IsSuccess, Label, FileName, ctx, vars, props, log



Test Plan
- User Defined Variables
- Thread Group
  - IP Config
  - Constant Timer
  - HTTP Request Defaults
  - SignalR
  - Setup
    - HTTP Request Defaults
    - setUsername
    - Synchronizing Timer
    - login
    - getCurrencies
      - Get response data
      - Get all currencies
      - Currency Id
      - Currency Code
    - checkSession
    - getUserType
    - getUserProfileStatus
    - getUserFavoritePage
    - Response Time Graph
  - Setting of From and To date
  - Set currencyCode
  - Interleave Controller
    - Transaction Controller(COLEFORD-Consolidated)
      - getRootAccounts
      - setPortfolioRequestData
      - parsePortfolioRequestData
      - getPortfolio
        - Response Assertion
      - getGAC
      - getChangeInPortfolioValue
      - getAssetMix
      - getHoldings
      - getTransactionActivity
      - getReturnsInspectorGross
      - getReturnsInspectorNet
      - Response Time Graph
    - Transaction Controller(HAMILTON3-Consolidated)
    - Transaction Controller(HANKS-Consolidated)
    - Transaction Controller(ADAMS-Consolidated)
  - Loop Controller
  - Teardown
    - Debug Sampler
    - logout
- Response Time Graph
- Graph Results
- Results Tree

HTTP Request

Name: getPortfolio

Comments:

Basic | Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

Method: POST Path: :ignalr/send?transport=longPolling&connectionToken=${__urlencode(${connectionToken})}&connectionData=%5B%7B%22name%22%3A%22dispatchhub%22%7D%5D Content encoding:

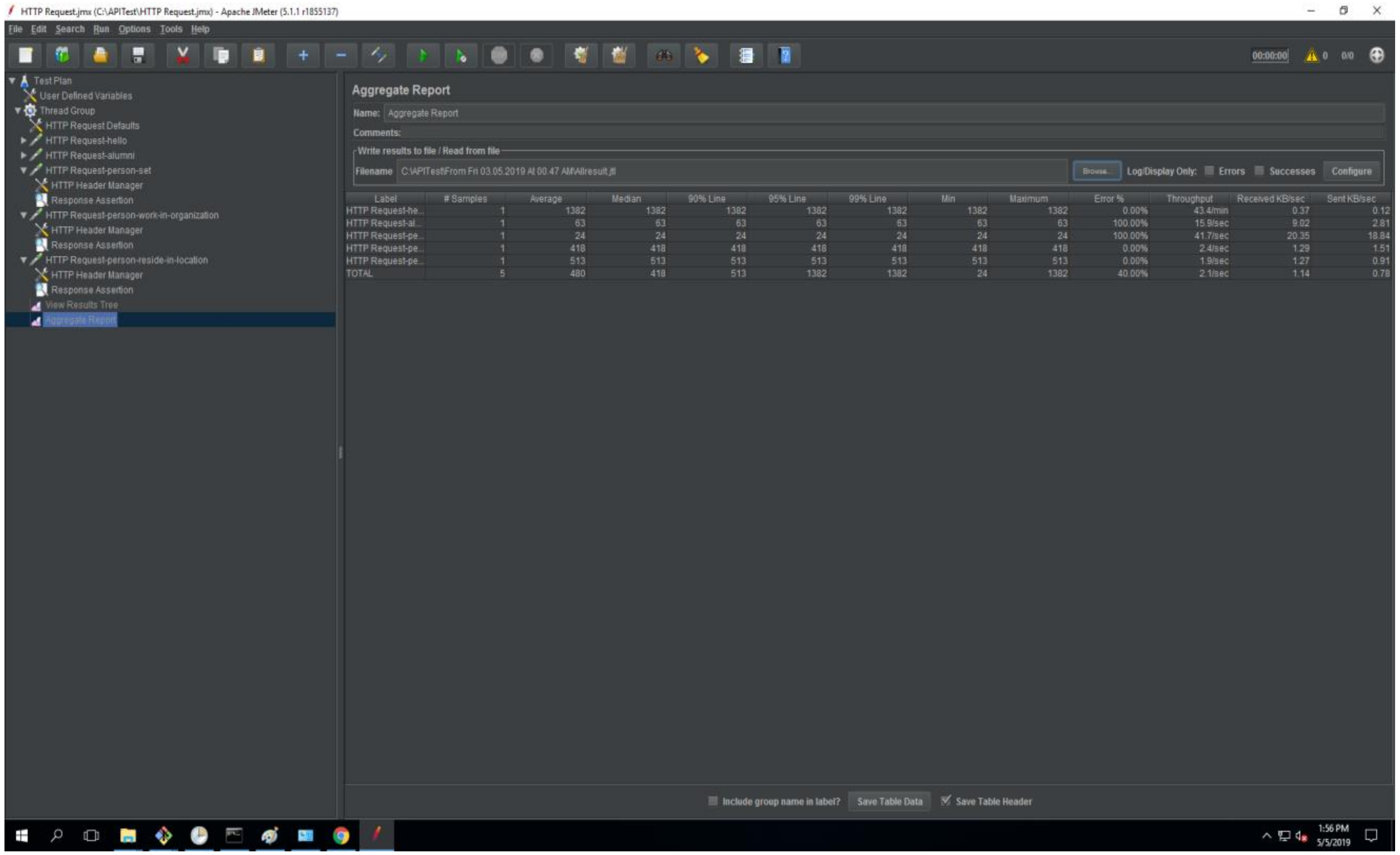
☐ Redirect Automatically ☑ Follow Redirects ☑ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameters | Body Data | Files Upload

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?
data	${getPortfolioDataWithSession}	✓	✓

Test result



HTTP Request.jmx (C:\APITest\HTTP Request.jmx) - Apache JMeter (5.1.1 r1855137)

- Test Plan
  - User Defined Variables
  - Thread Group
    - HTTP Request Defaults
    - HTTP Request-hello
    - HTTP Request-alumni
    - HTTP Request-person-set
      - HTTP Header Manager
      - Response Assertion
    - HTTP Request-person-work-in-organization
      - HTTP Header Manager
      - Response Assertion
    - HTTP Request-person-reside-in-location
      - HTTP Header Manager
      - Response Assertion
    - View Results Tree
    - Aggregate Report

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename: C:\APITest\From Fri 03.05.2019 At 00.47 AM\Allresult.jtl

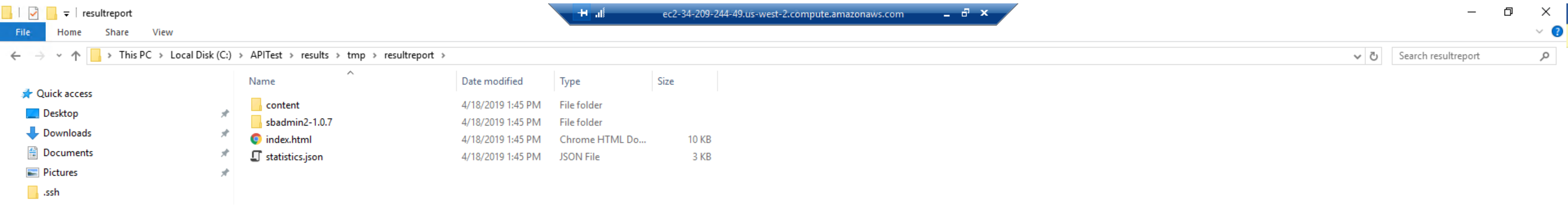
Log/Display Only: Errors Successes

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request-he...	1	1382	1382	1382	1382	1382	1382	1382	0.00%	43.4/min	0.37	0.12
HTTP Request-al...	1	63	63	63	63	63	63	63	100.00%	15.9/sec	9.02	2.81
HTTP Request-pe...	1	24	24	24	24	24	24	24	100.00%	41.7/sec	20.35	18.84
HTTP Request-pe...	1	418	418	418	418	418	418	418	0.00%	2.4/sec	1.29	1.51
HTTP Request-pe...	1	513	513	513	513	513	513	513	0.00%	1.9/sec	1.27	0.91
TOTAL	5	480	418	513	1382	1382	24	1382	40.00%	2.1/sec	1.14	0.78

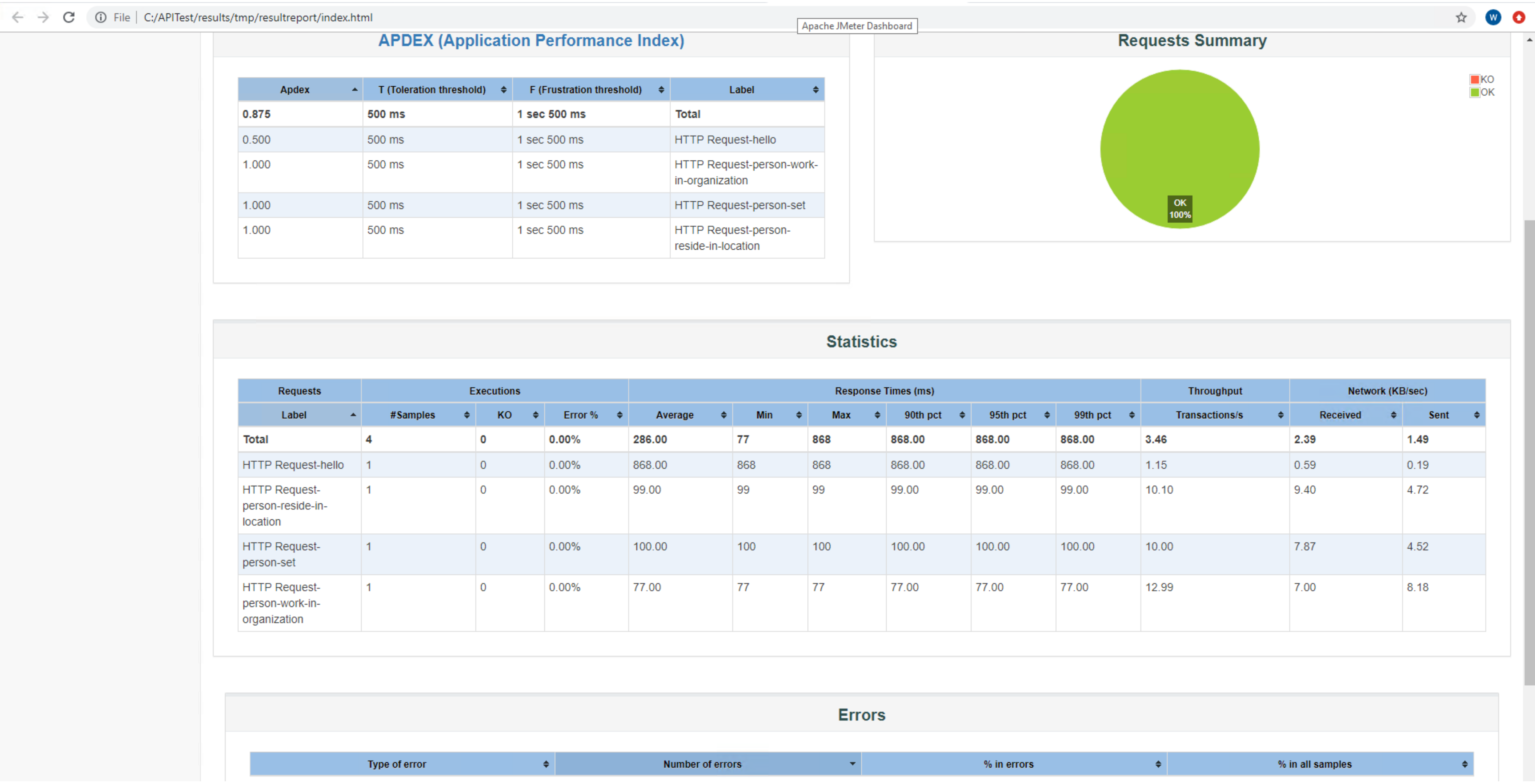
Include group name in label? Save Table Data ✓ Save Table Header

Console command

- jmeter -n -t "c:\APITest\HTTP Request.jmx" -l "C:\Users\harryliu\.jenkins\workspace\API test\results\Allresult.jtl" -e -o "C:\Users\harryliu\.jenkins\workspace\API test\results\tmp\resultreport"


resultreport
ec2-34-209-244-49.us-west-2.compute.amazonaws.com
File
Home
Share
View
This PC › Local Disk (C:) › APITest › results › tmp › resultreport ›
Search resultreport
Quick access
Desktop
Downloads
Documents
Pictures
.ssh

Name	Date modified	Type	Size
content	4/18/2019 1:45 PM	File folder	
sbadmin2-1.0.7	4/18/2019 1:45 PM	File folder	
index.html	4/18/2019 1:45 PM	Chrome HTML Do...	10 KB
statistics.json	4/18/2019 1:45 PM	JSON File	3 KB



File | C:/APITest/results/tmp/resultreport/index.html

Apache JMeter Dashboard

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.875	**500 ms**	**1 sec 500 ms**	**Total**
0.500	500 ms	1 sec 500 ms	HTTP Request-hello
1.000	500 ms	1 sec 500 ms	HTTP Request-person-work-in-organization
1.000	500 ms	1 sec 500 ms	HTTP Request-person-set
1.000	500 ms	1 sec 500 ms	HTTP Request-person-reside-in-location

Requests Summary

Statistics

Requests	Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**4**	**0**	**0.00%**	**286.00**	**77**	**868**	**868.00**	**868.00**	**868.00**	**3.46**	**2.39**	**1.49**
HTTP Request-hello	1	0	0.00%	868.00	868	868	868.00	868.00	868.00	1.15	0.59	0.19
HTTP Request-person-reside-in-location	1	0	0.00%	99.00	99	99	99.00	99.00	99.00	10.10	9.40	4.72
HTTP Request-person-set	1	0	0.00%	100.00	100	100	100.00	100.00	100.00	10.00	7.87	4.52
HTTP Request-person-work-in-organization	1	0	0.00%	77.00	77	77	77.00	77.00	77.00	12.99	7.00	8.18

Errors

Type of error	Number of errors	% in errors	% in all samples

性能测试（Performance Test）

- 模拟用户负载来测试系统在负载情况下，系统的响应时间，吞吐量等。
- 通常收集所有和测试有关的所有性能
- 关注点：how much和how fast

负载测试（Load Test）

- 在一定的软硬件环境上，通过不断的加大负载来确定在满足性能指标情况下所能够承受的最大用户数。
- 负载测试是一种性能测试，指数据在超负荷环境中运行，程序是否能够承担。
- 一般不超过80%cpu，正常情况工作下最大用户数数据。
- 关注点：how much

强度测试（Stress Test）

- 在一定的软件硬件环境下，通过高负载的手段来使服务器资源处于极限的状态，测试该系统在极限状态长时间运行是否稳定。
- 强度测试是一种性能测试，他在系统资源特别低的情况下软件系统运行情况，目的是找到系统在哪里失效以及如何失效的地方。
- Spike testing：短时间的极端负载测试
  Extreme testing：在过量用户下的负载测试
  Hammer testing：连续执行所有能做的操作

容量测试(Volume Test)

- 确定系统可处理同时在线的最大用户数
- 关注点：how much（而不是how fast）
- 容量测试，通常和数据库有关
- 容量和负载的区别在于：容量关注的是大容量，而不需要表现实际的使用。

Others

- Baseline
- Configuration test
- Concurrency test
- Recovery test

Any Question?