APACHE JMETER™ 3.0 WARSZTATY – SKRYPT

Adrian Bala

adrian.bala@gft.com

Spis treści

Macierz ćwiczeń

level / exercise	E1	E2	E3	E4	E5	Extras
basic	hello-world	multi-hello-world	multi-function-hello-world	setup-multi-function-hello-world	assert-setup-multi-function-hello-world	rec-play
medium	simple-get-post	firebase-rest-api	google-translate-api	config-google-maps-api	regression-suite	JSR223
advanced	sqlite-jdbc	stress-test	ftp-test	jms-queue	chess-game	blaze-meter

Poziom basic

Ćwiczenie 1 – hello-world

Cel ćwiczenia

Zapoznanie z ogólną budową Test Planu i zasadą działania Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
3. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
4. Do Thread Group dodaj sampler – **Java Request** (Add > Sampler > Java Request).
5. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (Hello JMeter World!).
6. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
7. Wykonaj Test Plan (Ctr ^ r).
8. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Posługiwanie się podstawowymi skrótami klawiaturowymi w Apache JMeter™ 3.0.
2. Dodawanie komentarzy w obrębie wykonywanego Test Planu.

Wskazówki

1. Hints and Tips – http://jmeter.apache.org/usermanual/hints_and_tips.html
2. Java Request – http://jmeter.apache.org/usermanual/component_reference.html#Java_Request

Rozwiązanie

hello-world.jmx

Ćwiczenie 2 – multi-hello-world

Cel ćwiczenia

Zapoznanie z wielowątkowością i współbieżnym wykonaniem Test Planu w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
3. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
4. W ramach Thread Group ustaw **Number of Threads**, np. 10 (Thread Group > Number of Threads).
5. W ramach Thread Group wpisz **Loop Count**, np. 2 (Thread Group > Loop Count).
6. Do Thread Group dodaj sampler – **Java Request** (Add > Sampler > Java Request).
7. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (Hello JMeter World!).
8. W polu **Label** samplera – Java Request – dodaj odwołanie do przykładowej zmiennej, np. **${__threadNum}**.
9. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
10. Wykonaj Test Plan (Ctr ^ r).
11. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Projektowanie wielowątkowego Test Planu w Apache JMeter™ 3.0.
2. Posługiwanie się zmiennymi w obrębie Test Planu.

Wskazówki

1. Thread Group – http://jmeter.apache.org/usermanual/test_plan.html
2. Functions and Variables – http://jmeter.apache.org/usermanual/functions.html

Rozwiązanie

multi-hello-world.jmx

multi-hello-world.jmx

Ćwiczenie 3 – multi-function-hello-world

Cel ćwiczenia

Zapoznanie z funkcjami oraz zmiennymi współdzielonymi pomiędzy wątkami Test Planu w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
3. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
4. W ramach Thread Group ustaw **Number of Threads**, np. 10 (Thread Group > Number of Threads).
5. W ramach Thread Group wpisz **Loop Count**, np. 2 (Thread Group > Loop Count).
6. Do Thread Group dodaj **Counter** (Add > Config Element > Counter).
7. W liczniku ustaw wartość początkową – **Start**, np. 1.
8. W liczniku ustaw wartość przyrostu – **Increment**, np. 1.
9. W liczniku ustaw nazwę referencyjną – **Reference Name**, np. c.
10. Do Thread Group dodaj sampler – **Java Request** (Add > Sampler > Java Request).
11. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (Hello JMeter World!).
12. W polu **Label** samplera – Java Request – dodaj odwołanie do przykładowej zmiennej, np. **${__threadNum}**.
13. W polu **Label** samplera – Java Request – dodaj odwołanie do licznika, np. **${c}**.
14. W polu **Label** samplera – Java Request – dodaj odwołanie do funkcji, np. **${__RandomString(${c},abcdefghijklmnopqrstuvwxyz,)}**.
15. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
16. Wykonaj Test Plan (Ctr ^ r).
17. Sprawdź szczegółowy wynik w **View Results Tree**.
18. Powtórz to ćwiczenie z zaznaczoną opcją w liczniku – **Track counter independently for each user**.

Nabyte umiejętności

1. Współdzielenie licznika pomiędzy wieloma wątkami Test Planu w Apache JMeter™ 3.0.
2. Posługiwanie się funkcjami wbudowanymi w Apache JMeter™ 3.0.

Wskazówki

1. Counter – http://jmeter.apache.org/usermanual/component_reference.html#Counter
2. How use counter in jmeter test – https://www.blazemeter.com/blog/how-use-counter-jmeter-test

Rozwiązanie

multi-function-hello-world.jmx

multi-function-hell
o-world.jmx

Ćwiczenie 4 – setup-multi-function-hello-world

Cel ćwiczenia

Zapoznanie się ze specjalnymi grupami wątków, tj. SetUp i TearDown oraz pomiarem czasu w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
4. Do Test Planu dodaj grupę wątków – SetUp (Add > Threads > SetUp).
5. W grupie wątków – SetUp – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
6. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
7. W ramach Thread Group ustaw **Number of Threads**, np. 10 (Thread Group > Number of Threads).
8. W ramach Thread Group wpisz **Loop Count**, np. 2 (Thread Group > Loop Count).
9. Do Thread Group dodaj **Counter** (Add > Config Element > Counter).
10. W liczniku ustaw wartość początkową – **Start**, np. 1.
11. W liczniku ustaw wartość przyrostu – **Increment**, np. 1.
12. W liczniku ustaw nazwę referencyjną – **Reference Name**, np. c.
13. Do Thread Group dodaj sampler – **Java Request** (Add > Sampler > Java Request).
14. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (Hello JMeter World!).
15. W polu **Label** samplera – Java Request – dodaj odwołanie do przykładowej zmiennej, np. **${__threadNum}**.
16. W polu **Label** samplera – Java Request – dodaj odwołanie do licznika, np. **${c}**.
17. W polu **Label** samplera – Java Request – dodaj odwołanie do funkcji, np. **${__RandomString(${c},abcdefghijklmnopqrstuvwxyz,)}**.
18. Do Test Planu dodaj grupę wątków – TearDown (Add > Threads > TearDown).
19. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
20. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na
    Total testing time is ${__longSum(${__time()},-${timestamp})} ms.
21. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
22. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
23. Wykonaj Test Plan (Ctr ^ r).
24. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Użycie specjalnych grup wątków, tj. SetUp i TearDown w Apache JMeter™ 3.0.
2. Wyliczanie czasu trwania Test Planu w oparciu o User Defined Variables i dostępne funkcje.

Wskazówki

1. A setUp Thread Group – http://jmeter.apache.org/usermanual/component_reference.html#setUp_Thread_Group
2. A tearDown Thread Group – http://jmeter.apache.org/usermanual/component_reference.html#tearDown_Thread_Group

Rozwiązanie

setup-multi-function-hello-world.jmx

Ćwiczenie 5 – assert-setup-multi-function-hello-world

Cel ćwiczenia

Zapoznanie się z asercjami oraz typowymi kodami odpowiedzi HTTP, tj. 200, 404, 500.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
4. Do Test Planu dodaj grupę wątków – SetUp (Add > Threads > SetUp).
5. W grupie wątków – SetUp – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
6. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
7. W ramach Thread Group ustaw **Number of Threads**, np. 10 (Thread Group > Number of Threads).
8. W ramach Thread Group wpisz **Loop Count**, np. 2 (Thread Group > Loop Count).
9. Do Thread Group dodaj **Counter** (Add > Config Element > Counter).
10. W liczniku ustaw wartość początkową – **Start**, np. 1.
11. W liczniku ustaw wartość przyrostu – **Increment**, np. 1.
12. W liczniku ustaw nazwę referencyjną – **Reference Name**, np. c.
13. Do Thread Group dodaj sampler – **Java Request** (Add > Sampler > Java Request).
14. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (Hello JMeter World!).
15. W polu **Label** samplera – Java Request – dodaj odwołanie do przykładowej zmiennej, np. **${__threadNum}**.
16. W polu **Label** samplera – Java Request – dodaj odwołanie do licznika, np. **${c}**.
17. W polu **Label** samplera – Java Request – dodaj odwołanie do funkcji, np. **${__RandomString(${counter},abcdefghijklmnopqrstuvwxyz,)}**.
18. W polu **ResponseCode** samplera – Java Request – ustaw wartość, np. 200.
19. Do samplera – Java Request – dodaj **Response Assertion** (Ctr ^ 3).
20. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
21. Do Test Planu dodaj grupę wątków – TearDown (Add > Threads > TearDown).
22. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
23. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na
    Total testing time is ${__longSum(${__time()},-${timestamp})} ms.
24. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).

25. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
26. Wykonaj Test Plan (Ctr ^ r).
27. Sprawdź szczegółowy wynik w **View Results Tree**.
28. Powtórz to ćwiczenie dla **Response Code** – 404 lub 500.

Nabyte umiejętności

1. Użycie Response Assertion w Apache JMeter™ 3.0.
2. Sprawdzenie pozytywnych, np. 200 oraz negatywnych, np. 404 lub 500 kodów odpowiedzi.

Wskazówki

1. Assertions – http://jmeter.apache.org/usermanual/component_reference.html#assertions
2. Hypertext Transfer Protocol -- HTTP/1.1 – https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Rozwiązanie

assert-setup-multi-function-hello-world.jmx

assert-setup-multi-f
unction-hello-world

Poziom medium

Ćwiczenie 1 – simple-get-post

Cel ćwiczenia

Zapoznanie z najprostszym i NIE zalecanym podejściem do wykonywania zapytań typu GET i POST w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
3. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
4. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
5. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
6. Ustaw w Response Assertion następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
7. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).
8. W polu Path samplera – **HTTP Request** – wklej przykładowy link (**http://chem-calc.appspot.com/api/v1/get?chemform=TW16**).
9. Do samplera – HTTP Request dodaj **Response Assertion** (Ctr ^ 3).
10. Ustaw w Response Assertion następujące pola: **Text Response**, **Equals** i dodaj **Pattern to Test** – null.
11. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).
12. W polu **Path** samplera – HTTP Request – wklej przykładowy link (**http://chem-calc.appspot.com/api/v1/calc**).
13. W samplerze – HTTP Request – dodaj parametr: **chemform** z wartością H2O.
14. W samplerze – HTTP Request – wybierz metodę **POST**.
15. W polu **Path** samplera – HTTP Request – wklej przykładowy link (**http://chem-calc.appspot.com/api/v1/get?chemform=H2O**).
16. Do samplera – HTTP Request dodaj **Response Assertion** (Ctr ^ 3).
17. Ustaw w Response Assertion następujące pola: **Text Response**, **Equals**, **Not** i dodaj **Pattern to Test** – null.
18. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
19. Wykonaj Test Plan (Ctr ^ r).
20. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Poznanie ZŁEJ praktyki bezpośredniego wpisywania i wywoływania URL w Apache JMeter™ 3.0.
2. Wywoływanie zapytań typu GET i POST z parametrem w obrębie wykonywanego Test Planu.

Wskazówki

1. REST – https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
2. StackOverflow RESTful API – https://api.stackexchange.com/docs/answers
3. Postman plug-in – https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop

Rozwiązanie

simple-get-post.jmx

Ćwiczenie 2 – firebase-rest-api

Cel ćwiczenia

Zapoznanie z DOBRYM i zalecanym podejściem do wykonywania zapytań typu GET, POST, PUT i DELETE w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port**.
4. W zmiennych: server, protocol, encoding, port uzupełnij wartości:
   jmeter-61c17.firebaseio.com, **https**, **utf-8**, **443**.
5. Do User Defined Variables dodaj zmienne: **user_id**, **text**, **name**, **surname**, **psurname**, **Fname**, **Lname**.
6. W zmiennych: user_id, text, name, surname, psurname, Fname, Lname uzupełnij wartości, np.:
   anba, TestWarez, AdB, GFT, Poland, Test, Item.
7. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
8. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
9. W HTTP Request Defaults ustaw wartości:
   Server Name or IP: ${server}, **Port Number: ${port}**, **Protocol [http]: ${protocol}** i **Content encoding: ${encoding}**.
10. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
11. Do Test Planu dodaj grupę wątków – **SetUp** (Add > Threads > SetUp).
12. W grupie wątków – **SetUp** – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
13. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
14. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
15. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
16. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
17. Do Thread Group dodaj **4 x Simple Controller** (Add > Logic Controller > Simple Controller).
18. Nazwij każdy Simple Controller odpowiednio: **GET**, **POST**, **PUT** i **DELETE**.
19. Do każdego Simple Controller dodaj sampler – **HTTP Request** (Ctr ^ 1) – typu: **GET**, **POST**, **PUT** i **DELETE**.
20. Przykładowo, w polu **Path** samplera – **HTTP Request GET** – wklej (**/users.json**).
21. Przykładowo, w polu **Path** samplera – **HTTP Request POST** – wklej (**/message_list.json**).
22. Przykładowo, w zakładce **Body Data** samplera – **HTTP Request POST** – wklej JSON (**{"user_id" : "${user_id}", "text" : "${text}"}**).

23. Przykładowo, do samplera – HTTP Request POST – dodaj **JSON Path PostProcessor**.
24. W JSON Path PostProcessor ustaw **Variable names: jsonHash**, **JSON Path expressions: $..name**, **Match numbers: 1** i **Default values: null**.
25. Do Simple Controllera typu POST dodaj **HTTP Request GET** i w polu **Path** wklej (**/message_list/${jsonHash}.json**).
26. Przykładowo, w polu Path samplera – **HTTP Request PUT** – wklej (**/users/${name}/name.json**).
27. Przykładowo, w zakładce **Body Data** samplera – **HTTP Request PUT** – wklej JSON (**{ "first": "${name}", "last": "${surname}" }**).
28. Do Simple Controllera typu PUT dodaj **HTTP Request GET** i w polu **Path** wklej (**/users/${name}.json**).
29. Przykładowo, w polu **Path** samplera – **HTTP Request DELETE** – wklej (**/users/${jsonHash}.json**).
30. Do Simple Controllera typu DELETE dodaj **HTTP Request GET** i w polu **Path** wklej (**/users/${jsonHash}.json**).
31. Do samplera – HTTP Request GET dodaj **Response Assertion** (Ctr ^ 3).
32. Ustaw w Response Assertion następujące pola: **Text Response**, **Equals** i dodaj **Pattern to Test** – null.
33. Do Test Planu dodaj grupę wątków – **TearDown** (Add > Threads > TearDown).
34. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
35. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na
    Total testing time is ${__longSum(${__time()},-${timestamp})} ms.
36. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
37. Wykonaj Test Plan (Ctr ^ r).
38. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Poznanie DOBREJ praktyki definiowania zmiennych w obrębie wykonywanego Test Planu.
2. Wykonywania zapytań typu GET, POST, PUT i DELETE w Apache JMeter™ 3.0.

Wskazówki

1. JSONPath – http://goessner.net/articles/JsonPath/
2. REST – https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
3. FireBase – https://firebase.googleblog.com/2016/05/firebase-expands-to-become-unified-app-platform.html
4. FireBase RESTful API – https://firebase.google.com/docs/reference/rest/database/
5. FireBase REST API – https://www.firebase.com/docs/rest/api/

Rozwiązanie

firebase-rest-api.jmx

firebase-rest-api.jmx

Ćwiczenie 3 – google-translate-api

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu opartego o dane wczytywane z pliku CSV w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port**, **API_KEY**.
4. W zmiennych: server, protocol, encoding, port oraz API_KEY uzupełnij wartości:
5. **www.googleapis.com**, **https**, **utf-8**, **443**, **API_KEY**.
6. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
7. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
8. W HTTP Request Defaults ustaw wartości:
   Server Name or IP: ${server}, **Port Number: ${port}**, **Protocol [http]: ${protocol}** i **Content encoding: ${encoding}**.
9. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
10. Do Test Planu dodaj grupę wątków – **SetUp** (Add > Threads > SetUp).
11. W grupie wątków – **SetUp** – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
12. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
13. W ramach Thread Group ustaw **Number of Threads**, np. 4 (Thread Group > Number of Threads).
14. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
15. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
16. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
17. Utwórz plik **words.csv** i zapisz go w wybranej lokalizacji na dysku, np. **C:\Temp**.
18. W pliku **words.csv** zdefiniuj strukturę:
    q,source,target
    "Hello","en",”de”
    "Dziękuję","pl",”en”
    "Hello","en",”pl”
    "Dziękuję","pl",”de”
19. Do Thread Group dodaj – **CSV Data Set Config** (Add > Config > CSV Data Set Config).

20. W **CSV Data Set Config** podaj, np.: **Filename: C:\Temp\words.csv**, **File encoding: ${encoding}**, **Allow quoted data?: True**.
21. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).
22. W polu **Path** samplera – HTTP Request – wklej (**/language/translate/v2?key=${API_KEY}**).
23. Do samplera – **HTTP Request** – dodaj **parametry** – **q=${q}**, **source=${source}** i **target=${target}**.
24. Do parametru **q** samplera – HTTP Request zaznacz pole **Encode?**.
25. Do samplera – HTTP Request – dodaj **JSON Path PostProcessor**.
26. W JSON Path PostProcessor ustaw **Variable names: t**, **JSON Path expressions: $..translatedText**, **Match numbers: 1** i **Default values: null**.
27. Do Thread Group dodaj – **Java Request** (Add > Sampler > Java Request).
28. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (**Translated: ${json}**).
29. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
30. Wykonaj Test Plan (Ctr ^ r).
31. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Zbudowanie Test Planu opartego o dane testowe wczytywane z pliku CSV.
2. Zrównoleglone przyspieszenie wykonania Test Planu w Apache JMeter™ 3.0.

Wskazówki

1. Google Translate API – https://cloud.google.com/translate/v2/quickstart
2. Google Cloud Platform – https://cloud.google.com/

Rozwiązanie

google-translate-api.jmx , words.csv

Ćwiczenie 4 – config-google-maps-api

Cel ćwiczenia

Zapoznanie z konfigurowalnym podejściem tworzenia Test Planów w Apache JMeter™ 3.0.

Instrukcja

1. Zbuduj następującą strukturę plików i katalogów:

```
config-google-maps-api
|_fixtures
  |_env
    |_dev
      |_places.csv
    |_pro
      |_places.csv
|_properties
  |_misc.properties
  |_env
    |_dev.properties
    |_pro.properties
|_tests
  |_config-google-maps-api.jmx
|_dev.bat
|_pro.bat
```

2. Włącz Apache JMeter™ 3.0.
3. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
4. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port** i **dir**.
5. W zmiennych: server, protocol, encoding, port, dir uzupełnij wartości:
   ${__P(serverNameOrIP)}, ${__P(defaultProtocol)}, ${__P(defaultContentEncoding)}, ${__P(defaultPort)}, ${__P(dataDir)}.
6. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
7. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
8. W HTTP Request Defaults ustaw wartości:

Server Name or IP: ${server}, Port Number: ${port}, Protocol [http]: ${protocol} i **Content encoding: ${encoding}**.

9. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
10. Do Test Planu dodaj grupę wątków – **SetUp** (Add > Threads > SetUp).
11. W grupie wątków – **SetUp** – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
12. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
13. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
14. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
15. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
16. W plikach **places.csv** umieszczonych odpowiednio w **dev** oraz **pro** zdefiniuj strukturę, np.:

dev

```
origins,destinations
"Poland|Legnica","Poland|Opole"
"Poland|Warszawa","Poland|Leszno"
```

pro

```
origins,destinations
"Poland|Gniezno","Poland|Karpacz"
"Poland|Szczecin","Poland|Zakopane"
```

17. W pliku **misc.properties** zdefiniuj:
    defaultContentEncoding=utf-8
18. W pliku **dev.properties** zdefiniuj:
    serverNameOrIP=maps.googleapis.com
    defaultProtocol=http
    defaultPort=80
    dataDir=.\\fixtures\\env\\dev
19. W pliku **pro.properties** zdefiniuj:
    defaultContentEncoding=utf-8
20. W pliku **pro.properties** zdefiniuj:
    serverNameOrIP=maps.googleapis.com
    defaultProtocol=https
    defaultPort=443
    dataDir=.\\fixtures\\env\\pro

21. W pliku **dev.bat** zdefiniuj:
    jmeter -t .\tests\config-google-maps-api.jmx -q .\properties\misc.properties -q .\properties\env\dev.properties
22. W pliku pro.bat zdefiniuj:
    jmeter -t .\tests\config-google-maps-api.jmx -q .\properties\misc.properties -q .\properties\env\pro.properties
23. Do Thread Group dodaj – **CSV Data Set Config** (Add > Config > CSV Data Set Config).
24. W CSV Data Set Config podaj, np.: **Filename: ${dir}\places.csv**, **File encoding: ${encoding}**, **Allow quoted data?: True**.
25. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).
26. W polu **Path** samplera – HTTP Request – wklej (**maps/api/distancematrix/json**).
27. Do samplera – HTTP Request – dodaj **parametry** – **origins: ${origins}** i **destinations: ${destinations}**.
28. Do parametrów **origins** i **destinations** samplera – HTTP Request zaznacz pole **Encode?**.
29. Do samplera – HTTP Request – dodaj **Regular Expression Extractor** (Ctr ^ 2).
30. W **Regular Expression Extractor** ustaw:
    Reference Name: json,
    Regular Expression: "distance"\s*:\s*{\s*"text"\s*:\s*"(.+)",\s*"value"\s*:\s*(\d+),
    Template: $1$$2$,
    Match No.: 4,
    Default Value: null
31. Do Thread Group dodaj – **IF Controller** (Add > Logic Controller > If Controller).
32. W If Controller ustaw **Condition: "${json}" != "null"**.
33. W If Controller dodaj sampler – **Java Request** (Add > Sampler > Java Request).
34. W polu **Label** samplera – **Java Request** – dodaj przykładowy tekst (**Distance: ${json_g2} m ~ ${json_g1} km**).
35. Do Test Planu dodaj grupę wątków – **TearDown** (Add > Threads > TearDown).
36. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
37. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na
    Total testing time is ${__longSum(${__time()},-${timestamp})} ms.
38. Zapisz **Test Plan** w **tests** (Ctr ^ s).
39. Wykonaj Test Plan (Ctr ^ r).
40. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Poznanie BARDZO DOBREJ praktyki rozdziału na: Test Plan, Konfigurację i Dane testowe.
2. Użycie instrukcji warunkowej IF w Apache JMeter™ 3.0 oraz odwołanie do grupy dopasowań ekstraktora.

Wskazówki

1. Regular expressions – http://jmeter.apache.org/usermanual/regular_expressions.html
2. Google Maps API – https://developers.google.com/maps/get-started/
3. JMeter – testing tool – http://www.testwarez.pl/jmeter-narzedzie-testera/
4. Jakarta ORO 2.0.6 – http://archimedes.fas.harvard.edu/scrapbook/jakarta-oro-2.0.6/docs/api/org/apache/oro/text/regex/package-summary.html

Rozwiązanie

config-google-maps-api.zip

config-google-map
s-api.zip

Ćwiczenie 5 – regression-suite

Cel ćwiczenia

Zapoznanie z wykonaniem liniowym lub współbieżnym Test Planu opartego o dane wczytywane z pliku CSV w Apache JMeter™ 3.0.

Instrukcja

1. Utwórz plik **urls.csv** i zapisz go w wybranej lokalizacji na dysku, np. **C:\Temp**.
2. W pliku **urls.csv** zdefiniuj strukturę:
   url,respcode
   "/users.json","200"
   "/users/jack.json","200"
   "/userss/jack.json","404"
3. Włącz Apache JMeter™ 3.0.
4. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
5. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port** i **dir**.
6. W zmiennych: server, protocol, encoding, port, dir uzupełnij wartości:
   jmeter-61c17.firebaseio.com, **https**, **utf-8**, **443**, **C:\Temp**.
7. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
8. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
9. W HTTP Request Defaults ustaw wartości:
   Server Name or IP: ${server}, **Port Number: ${port}**, **Protocol [http]: ${protocol} i Content encoding: ${encoding}**.
10. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
11. Do Test Planu dodaj grupę wątków – **SetUp** (Add > Threads > SetUp).
12. W grupie wątków – **SetUp** – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
13. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
14. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
15. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
16. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – ${respcode}.
17. Do Thread Group dodaj – **CSV Data Set Config** (Add > Config > CSV Data Set Config).
18. W **CSV Data Set Config** podaj, np.: **Filename: C:\Temp\urls.csv**, **File encoding: ${encoding}**, **Allow quoted data?: True**.
19. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).

20. W polu **Path** samplera – HTTP Request – wklej (**/${url}**).
39. Do Test Planu dodaj grupę wątków – **TearDown** (Add > Threads > TearDown).
40. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
41. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Total testing time is ${__longSum(${__time()},-${timestamp})} ms**.
42. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
43. Wykonaj Test Plan (Ctr ^ r).
44. Sprawdź szczegółowy wynik w **View Results Tree**.
45. Zapamiętaj lub zapisz czas liniowego wykonania Test Planu.
46. Powtórz to ćwiczenie dla zrównoleglonego wykonania Test Planu wpisując w **Thread Group** ich liczbę, np. 3 (Thread Group > Number of Threads).
47. Porównaj czas liniowego wykonania Test Planu z czasem zrównoleglonego wykonania Test Planu.

Nabyte umiejętności

1. Zbudowanie Test Planu opartego o dane testowe wczytywane z pliku CSV.
2. Zrównoleglone przyspieszenie wykonania Test Planu w Apache JMeter™ 3.0.

Wskazówki

1. Hypertext Transfer Protocol -- HTTP/1.1 – https://www.w3.org/Protocols/HTTP/1.1/rfc2616.pdf

Rozwiązanie

regression-suite.jmx , urls.csv

Poziom Advanced

Ćwiczenie 1 – sqlite-jdbc

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu wykorzystującego interfejs JDBC oraz BeanShell Sampler w Apache JMeter™ 3.0.

Instrukcja

1. Do katalogu **lib** instancji Apache JMeter™ 3.0 dodaj plik **sqlite-jdbc-3.7.2.jar**.
2. Włącz Apache JMeter™ 3.0 z prawami administratora (Run as administrator).
3. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
4. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
5. Do Thread Group dodaj sampler – **JDBC Connection Configuration** (Add > Config Element > JDBC Connection Configuration).
6. W polu **Variable Name** samplera – JDBC Connection Configuration – wpisz **SQLite**
7. W polu **Database URL** samplera – JDBC Connection Configuration – wpisz **jdbc:sqlite:../carsdb.sqlite**
8. W polu **JDBC Driver class** samplera – JDBC Connection Configuration – wpisz **org.sqlite.JDBC**
9. Do Thread Group dodaj sampler – **BeanShell Sampler** (Add > Sampler > BeanShell Sampler).
10. W polu **Script** samplera – BeanShell Sampler – wpisz skrypt tworzący bazę danych SQLite:
    File mydb = new File("../carsdb.sqlite");
    vars.put("exists", Boolean.toString(mydb.exists()));
    SampleResult.setSuccessful(mydb.exists());
11. Do Thread Group dodaj – **IF Controller** (Add > Logic Controller > If Controller).
12. W If Controller ustaw **Condition: ${exists} == false**
13. W If Controller dodaj sampler – **JDBC Request** (Add > Sampler > JDBC Request).
14. W polu **SQL Variable Name** samplera – JDBC Request – wpisz **SQLite**.
15. W polu **Query Type** samplera – JDBC Request – wybierz: **Update statement**.
16. W polu **SQL Query samplera** – JDBC Request – dodaj: **create table car (id integer, brand string);**
17. Do Thread Group dodaj **10 x JDBC Request** (Add > Sampler > JDBC Request).
18. W polu **SQL Variable Name** każdego z dziesięciu samplerów – JDBC Request – wpisz **SQLite**.
19. W polu **SQL Query** każdego kolejnego samplera – JDBC Request – wpisz:
    insert into car values(1, 'BMW');

insert into car values(3, 'Honda');
insert into car values(2, 'Audi');
select * from car;
select * from car where id < 3;
delete from car where brand = 'BMW';
select * from car where id < 3;
select * from car;
delete from car;
select * from car;

20. W polu **Query Type** samplera – JDBC Request – innego niż **select**, np. **insert** lub **delete** wybierz **Update statement**.
21. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
22. Wykonaj Test Plan (Ctr ^ r).

Nabyte umiejętności

1. Zbudowanie Test Planu wykorzystującego BeanShell Sampler.
2. Wykonanie Test Planu wykorzystującego interfejs JDBC bazy SQLite w Apache JMeter™ 3.0.

Wskazówki

1. Building a Database Test Plan – http://jmeter.apache.org/usermanual/build-db-test-plan.html
2. JDBC Request – http://jmeter.apache.org/usermanual/component_reference.html#JDBC_Request
3. JDBC Connection Configuration – http://jmeter.apache.org/usermanual/component_reference.html#JDBC_Connection_Configuration
4. SQLite JDBC – https://github.com/djangofan/jmeter-jdbc-sqllite-example/
5. SQLite – https://www.sqlite.org/
6. SQLite Manager – https://addons.mozilla.org/pl/firefox/addon/sqlite-manager/

Rozwiązanie

sqlite-jdbc.jmx , carsdb.sqlite , sqlite-jdbc-3.7.2.jar

Ćwiczenie 2 – stress-test

Cel ćwiczenia

Zapoznanie z podejściem Non-GUI wykonywania testów obciążeniowych w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port**, **timestamp**, **dir**.
4. W zmiennych: server, protocol, encoding, port uzupełnij wartości:
   chem-calc.appspot.com, **http**, **utf-8**, **80**, **${__time()}**, **.** .
5. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
6. W HTTP Request Defaults ustaw wartości:
   Server Name or IP: ${server}, **Port Number: ${port}**, **Protocol [http]: ${protocol} i Content encoding: ${encoding}**.
7. Do Test Planu dodaj **Response Time Graph** (Add > Listener > Response Time Graph).
8. W polu **Filename** listenera – Response Time Graph – wpisz lokalizację i nazwę pliku z wynikami, np.:
   C:\Temp\stress-test.csv.
9. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
10. W ramach Thread Group ustaw **Number of Threads**, np. 100 (Thread Group > Number of Threads).
11. W ramach Thread Group wpisz **Loop Count**, np. 10 (Thread Group > Loop Count).
12. Do Thread Group dodaj sampler – **HTTP Request** (Ctr ^ 1).
13. W polu **Path** samplera – HTTP Request – wklej przykładowy link (**/api/v1/get?chemform=H2O**).
14. Do samplera – HTTP Request dodaj **Response Assertion** (Ctr ^ 3).
15. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
16. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
17. Zamknij Apache JMeter™ 3.0.
18. Otwórz **Windows Command Line** (Win ^ r + cmd).
19. Wykonaj Test Plan w trybie **Non-GUI** wykonując komendę: **jmeter -n -t stress-test.jmx**
20. Włącz Apache JMeter™ 3.0.
21. Wczytaj dane z pliku **C:\Temp\stress-test.csv**.

Nabyte umiejętności

1. Zbudowanie lekkiego Test Planu wykonywanego w trybie Non-GUI.
2. Zapisywanie danych z testów obciążeniowych do pliku CSV i ich analiza w Apache JMeter™ 3.0.

Wskazówki

1. Non-GUI Mode (Command Line mode) – http://jmeter.apache.org/usermanual/get-started.html
2. Five Ways To Launch a JMeter Test – https://www.blazemeter.com/blog/5-ways-launch-jmeter-test-without-using-jmeter-gui

Rozwiązanie

stress-test.jmx , stress-test.csv


stress-test.jmx


stress-test.csv

Ćwiczenie 3 – ftp-test

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu wymagającego połączenia ftp w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Wybierz opcję **Templates** (File > Templates).
3. Z opcji **Select Template** wybierz **Building an FTP Test Plan** (Create).
4. Rozwiń pełne drzewo **Test Planu** (Ctr + Shift + -).
5. Przeanalizuj każdy element Test Planu.
6. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
7. Wykonaj Test Plan (Ctr ^ r).

Nabyte umiejętności

1. Zbudowanie Test Planu wymagającego połączenia ftp.
2. Poznanie opcji Templates w Apache JMeter™ 3.0.

Wskazówki

1. Building an FTP Test Plan – http://jmeter.apache.org/usermanual/build-ftp-test-plan.html
2. SSH Sampler for Jakarta JMeter – https://code.google.com/archive/p/jmeter-ssh-sampler/
3. JMeter SFTP Request – https://linkeshkannavelu.com/2015/10/07/jmeter-sftp-request/
4. FTP on-line tester – https://ftptest.net/
5. Public (s)ftp for tests – http://test.rebex.net/

Rozwiązanie

ftp-test.jmx

Ćwiczenie 4 – jms-queue

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu wymagającego kolejek JMS w Apache JMeter™ 3.0.

Instrukcja

1. Uruchom z linii komend terminal Apache ActiveMQ™ (activemq.bat start).
2. Sprawdź czy Apache ActiveMQ™ działa na **http://127.0.0.1:8161/admin/** używając **admin** jako login oraz password.
3. Skopiuj **activemq-all-5.14.0.jar** do katalogu lib w Apache JMeter™ 3.0.
4. Włącz Apache JMeter™ 3.0.
5. Do Test Planu dodaj **Graph Results** (Add > Listener > Graph Results).
6. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
7. W ramach Thread Group ustaw **Number of Threads**, np. 100 (Thread Group > Number of Threads).
8. W ramach Thread Group ustaw **Ramp-Up Period**, np. 10 (Thread Group > Loop Count).
9. W ramach Thread Group wpisz **Loop Count**, np. 10 (Thread Group > Loop Count).
10. Do Thread Group dodaj sampler – **JMS Point-to-Point** (Add > Sampler > JMS Point-to-Point).
11. Wypełnij odpowiednio pola w **JMS Point-to-Point**
    QueueConnectionFactory: ConnectionFactory,
    JNDI Name Request Queue: Q.REQ,
    JNDI Name Reply Queue: Q.RPL,
    Communication Style: Request Response,
    Content: TestWarez2016,
    InitialContextFactory: org.apache.activemq.jndi.ActiveMQInitialContextFactory,
    queue.Q.REQ example.A,
    queue.Q.RPL example.B,
    Provider URL: tcp://localhost:61616.
12. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
13. Wykonaj Test Plan (Ctr ^ r).

Nabyte umiejętności

1. Zbudowanie Test Planu wymagającego kolejek JMS w Apache JMeter™ 3.0.
2. Poznanie dodatkowego narzędzia Open Source – Apache ActiveMQ™.

Wskazówki

1. JMS point to point test plan – http://jmeter.apache.org/usermanual/build-jms-point-to-point-test-plan.html
2. Building JMS testing plan – https://www.blazemeter.com/blog/building-jms-testing-plan-apache-jmeter
3. Apache ActiveMQ™ – http://activemq.apache.org/

Rozwiązanie

jms-queue.jmx , activemq-all-5.14.0.jar

Ćwiczenie 5 – chess-game

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu grającego w szachy z użyciem Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienne: **server**, **protocol**, **encoding**, **port**, **fen1**, **fen2**.
4. W zmiennych: server, protocol, encoding, port, fen1 i fen2 uzupełnij wartości:
   syzygy-tables.info, https, utf-8, 443,
   rnbqkbnr/pppp1ppp/4p3/8/6P1/5P2/PPPPP2P/RNBQKBNR_b_KQkq_-_0_1
   7k/Q7/5K2/8/8/8/8/8_w_-_-_0_1
5. Do User Defined Variables dodaj zmienną **timestamp**, której wartością będzie **${__time()}**.
6. Do Test Planu dodaj **HTTP Request Defaults** (Test Plan > Add > Config Element > HTTP Request Defaults).
7. W HTTP Request Defaults ustaw wartości:
   Server Name or IP: ${server}, Port Number: ${port}, Protocol [http]: ${protocol} i Content encoding: ${encoding}.
8. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
9. Do Test Planu dodaj grupę wątków – **SetUp** (Add > Threads > SetUp).
10. W grupie wątków – **SetUp** – dodaj **Java Request** z **Label** ustawionym na **Tests started at ${__time(HH:mm:ss,${timestamp})}**.
11. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
12. Do Thread Group dodaj timer – **Constant Timer** (Ctr ^ 4).
13. Do Thread Group dodaj **Response Assertion** (Ctr ^ 3).
14. Ustaw w **Response Assertion** następujące pola: **Response Code**, **Equals** i dodaj **Pattern to Test** – 200.
15. Do Thread Group dodaj 2 samplery – **HTTP Request** (Ctr ^ 1).
16. W polu **Path** pierwszego samplera – HTTP Request – wklej (**/api/v1**).
17. Do pierwszego samplera – HTTP Request – dodaj parametry – **fen=${fen1}**.
18. W polu **Path** drugiego samplera – HTTP Request – wklej (**/api/v1**).
19. Do drugiego samplera – HTTP Request – dodaj parametry – **fen=${fen2}**.
20. Do drugiego samplera – HTTP Request – dodaj **JSON Path PostProcessor**.
21. W JSON Path PostProcessor ustaw **Variable names: json1, JSON Path expressions: $..bestmove, Match numbers: 1 i Default values: null**.

22. Do Thread Group dodaj 2 samplery – **HTTP Request** (Ctr ^ 1).
23. W polu **Path** pierwszego samplera – HTTP Request – wklej (**/api/v2**).
24. Do pierwszego samplera – HTTP Request – dodaj parametry – **fen=${fen1}**.
25. W polu **Path** drugiego samplera – HTTP Request – wklej (**/api/v2**).
26. Do drugiego samplera – HTTP Request – dodaj parametry – **fen=${fen2}**.
27. Do drugiego samplera – HTTP Request – dodaj **JSON Path PostProcessor**.
28. W JSON Path PostProcessor ustaw **Variable names: json2, JSON Path expressions: $..bestmove, Match numbers: 1 i Default values: null**.
29. Do Thread Group dodaj – **IF Controller** (Add > Logic Controller > If Controller).
30. W If Controller ustaw **Condition: "${json1}" == "${json2}"**.
31. W If Controller dodaj sampler – **Java Request** (Add > Sampler > Java Request).
32. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (**Best move: ${json1} : ${json2}**).
33. Do Test Planu dodaj grupę wątków – **TearDown** (Add > Threads > TearDown).
34. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na **Tests finished at ${__time(HH:mm:ss)}**.
35. W grupie wątków – TearDown – dodaj **Java Request** z **Label** ustawionym na
    Total testing time is ${__longSum(${__time()},-${timestamp})} ms.
36. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
37. Wykonaj Test Plan (Ctr ^ r).
38. Sprawdź i porównaj szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Oparcie wykonania Test Planu w Apache JMeter™ 3.0 O szachowy silnik i web API.
2. Zbudowanie Test Planu służącego rozrywce.

Wskazówki

1. GNU Chess Engine – http://www.net-chess.com/gnu/
2. Syzygy endgame tablebases – https://syzygy-tables.info/
3. Chess RESTful API – https://github.com/ornicar/lila#http-api
4. Analysis tool of lichess.org – https://pl.lichess.org/analysis
5. Notacja Forsytha-Edwardsa – FEN – https://pl.wikipedia.org/wiki/Notacja_Forsytha-Edwardsa
6. Szachowa notacja algebraiczna – PGN – https://pl.wikipedia.org/wiki/Szachowa_notacja_algebraiczna

Rozwiązanie

chess-game.jmx

Dodatki

Poziom basic – rec-play

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu wymagającego nagrania ruchu sieciowego w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Wybierz opcję **Templates** (File > Templates).
3. Z opcji **Select Template** wybierz **Recording** (Create).
4. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
5. Rozwiń węzeł **WorkBench**.
6. W elemencie **HTTP(S) Test Script Recorder** sprawdź element **Port**, np. **8888**.
7. Skonfiguruj **proxy** w przeglądarce **FireFox** (Narzędzia > Opcje > Zaawansowane > Sieć > Ustawienia > Ręczna konfiguracja proxy > 8888).
8. W elemencie **HTTP(S) Test Script Recorder** rozpocznij nagrywanie ruchu sieciowego – **Start** i zatwierdź certyfikat.
9. W przeglądarce **FireFox** odwiedź jakąś stronę, np. **chip.pl**
10. W elemencie **HTTP(S) Test Script Recorder** zakończ nagrywanie ruchu sieciowego – **Stop**.
11. Wykonaj Test Plan (Ctr ^ r).
12. Przeanalizuj nagrany ruch sieciowy.

Nabyte umiejętności

1. Zbudowanie Test Planu wymagającego nagrania ruchu sieciowego w Apache JMeter™ 3.0.
2. Poznanie opcji konfigurowania serwera proxy w przeglądarce FireFox.

Wskazówki

1. HTTP(S) Test Script Recorder – http://jmeter.apache.org/usermanual/component_reference.html#HTTP_Proxy_Server
2. Configure your browser to use the JMeter Proxy – http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.pdf

Rozwiązanie

rec-play.jmx

rec-play.jmx

Poziom medium – JSR223

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu opartego o sampler JSR223 w Apache JMeter™ 3.0.

Instrukcja

1. Włącz Apache JMeter™ 3.0.
2. Do Test Planu dodaj **User Defined Variables** (Add > Config Element > User Defined Variables).
3. Do User Defined Variables dodaj zmienne: **mult**, **num**, **iter**.
4. W zmiennych: mult, num uzupełnij wartości:
   10, **1**, **5**.
5. Do Test Planu dodaj **View Results Tree** (Ctr ^ 9).
6. Do Test Planu dodaj **Thread Group** (Ctr ^ 0).
7. W ramach Thread Group dodaj **Loop Controller** (Add > Logic Controller > Loop Controller).
8. Ustaw w Loop Controller jego **Loop Count:** np. na **${iter}**.
9. W Loop Controller dodaj sampler – **Java Request** (Add > Sampler > Java Request).
10. W polu **Label** samplera – Java Request – dodaj przykładowy tekst (num: ${num}).
11. W Loop Controller dodaj PostProcessor – **JSR223 PostProcessor** (Add > PostProcessors > JSR223 PostProcessor).
12. W JSR223 PostProcessor ustaw język **groovy**.
13. W polu **Script** PostProcessora JSR223 wpisz:
    m = ${mult}
    n = ${num}
    n = n * m
    vars.put("num", String.valueOf(n)).
14. Zapisz **Test Plan** w wybranej lokalizacji na dysku (Ctr ^ s).
15. Wykonaj Test Plan (Ctr ^ r).
16. Sprawdź szczegółowy wynik w **View Results Tree**.

Nabyte umiejętności

1. Zbudowanie Test Planu opartego o sampler JSR223 w Apache JMeter™ 3.0.
2. Zapoznanie z pętlą sterującą – Loop Controller oraz dynamicznym podstawianiem zmiennych w języku groovy.

Wskazówki

1. Scripting for the Java Platform – http://docs.oracle.com/javase/6/docs/technotes/guides/scripting/
2. JSR 223: Scripting for the Java™ Platform – https://www.jcp.org/en/jsr/detail?id=223
3. Groovy – http://www.groovy-lang.org/

Rozwiązanie

JSR223.jmx

Poziom advanced – blaze-meter

Cel ćwiczenia

Zapoznanie z wykonaniem Test Planu w chmurowej wersji Apache JMeter™ 3.0, tj. BlazeMeter.

Instrukcja

1. Załóż konto BlazeMeter na **https://www.blazemeter.com/**, np. używając G+.
2. W polu przykładowy URL wstaw: **http://chem-calc.appspot.com/api/v1/get?chemform=H2O**.
3. Ustal **liczbę wątków** (max 20), tj. ilość użytkowników generujących ruch sieciowy na wskazanym adresie.
4. Wykonaj **Test Plan**.
5. Przeanalizuj wyniki.

Nabyte umiejętności

1. Wykonanie Test Planu w BlazeMeter.
2. Poznanie metryk i analiz stosowanych w stres testach.

Wskazówki

1. Creating A Jmeter Test – https://guide.blazemeter.com/hc/en-us/articles/206732819-Creating-A-Jmeter-Test
2. Running Your Test On BlazeMeter – https://guide.blazemeter.com/hc/en-us/articles/206732449-Running-Your-Test-On-BlazeMeter
3. Timeline Report – https://guide.blazemeter.com/hc/en-us/articles/206733919-Timeline-Report

Rozwiązanie

My first BlazeMeter test – https://a.blazemeter.com/app/#projects/42579/masters/15591047/summary

data.csv

Wskazówki i podpowiedzi

1. Ctr ^ SER – jako przydatny skrót od Save, Erase i Run.
2. Ctr ^ - oraz Ctr + Shift ^ - – zwija i rozwija drzewo Test Planu.
3. Ctr ^ . – kończy natychmiastowo wykonanie wątku / pętli, itp.
4. Ctr ^ , – kończy wykonanie wątku / pętli, itp.
5. Ctr ^ T – włącza lub wyłącza dany element Test Planu.
6. Ctr ^ 0, Ctr ^ 1, …, Ctr ^ 9 – odnoszą się do najpopularniejszych komponentów JMetera.
7. Nazwy zmiennych są tekstowe, a odwołania do nich poprzez ${ }.
8. JMeter pozwala na KONFIGURACJĘ (kod, dane, konfiguracja).
9. Przykładowe wywołanie z linii komend w trybie Non-GUI: jmeter -n -t testplan.jmx -l logfile.log
10. Dobrze jest, gdy na jeden rdzeń procesora przypada, co najwyżej 2 instancje JMetera.
11. Jedna instancja JMetera powinna być w stanie obsłużyć około 1000 wątków.
12. Proces JMetera powinien zajmować zasoby procesora, co najwyżej w 75 %.
13. Każdy element zużywa zasoby dlatego warto ograniczać ilość monitorów, asercji czy procesorów.
14. Przydatne są setUp Thread Group oraz tearDown Thread Group.
15. Bardzo użyteczne są funkcje, np. __time[1] do wyznaczania czasu trwania testów.
16. Warto korzystać z wyrażeń regularnych[2], np. do weryfikowania odpowiedzi.
17. Efektywnie korzysta się z selektorów JSONpath.
18. W roli komentarza można użyć Java Request[3].
19. Dobry opis pozwala na samodokumentowanie kodu testów.
20. Przydaje się odróżniać testy pozytywne (+) od negatywnych (-).
21. Warto przyjmować i konsekwentnie stosować konwencje nazewnicze zmiennych.
22. Zamiast wstawiać kilka jednakowych samplerów lepiej jest w pętli użyć pojedynczego.
23. Funkcja przeszukiwania ułatwia odnajdywanie elementu w Test Planie.
24. Walidacja ułatwia sprawdzanie wielowątkowego Test Planu.

[1] Opis funkcji - http://jmeter.apache.org/usermanual/functions.html
[2] Opis wyrażeń regularnych - http://jmeter.apache.org/usermanual/regular_expressions.html oraz http://archimedes.fas.harvard.edu/scrapbook/jakarta-oro-2.0.6/docs/api/org/apache/oro/text/regex/package-summary.html
[3] Java Request zamiast Comment Sampler - https://bz.apache.org/bugzilla/show_bug.cgi?id=51824

Odnośniki

[1]. Bayo Erinle – JMeter Cookbook

[2]. Bayo Erinle – Performance Testing with JMeter

[3]. Apache JMeter – http://jmeter.apache.org/index.html

[4]. Apache JMeter Manual – http://jmeter.apache.org/usermanual/index.html

[5]. Apache JMeter Wiki – http://wiki.apache.org/jmeter/

[6]. Apache JMeter Group – https://www.linkedin.com/groups?gid=2017104

[7]. ASF Bugzilla – https://bz.apache.org/bugzilla/describecomponents.cgi?product=JMeter

[8]. Custom Plugins for Apache JMeter™ – http://jmeter-plugins.org/

[9]. Firebase RESTful API – https://www.firebase.com/docs/rest/api/

[10]. Google Cloud Platform – https://cloud.google.com/

[11]. Google Translate API – https://cloud.google.com/translate/docs/

[12]. BlazeMeter – https://www.blazemeter.com/

[13]. BlazeMeter Docs – https://guide.blazemeter.com/

[14]. KatieHome – Learn Jmeter – https://play.google.com/store/apps/details?id=com.tdt.learnjmeter

[15]. Apache JMeter - Quale 3 / 2015 TW (Edycja specjalna) - Adrian Bala – http://quale.pl/pl/pobierz-magazyn/

[16]. Apache JMeter 3.0 - PTaQ XXXIII - Adrian Bala – opracowanie własne

[17]. Apache JMeter - automatyzacja testowania - Adrian Bala – opracowanie własne

[18]. Apache JMeter workshop by Piotr Sobieraj – materiały szkoleniowe GFT