Legacy - JMeter: Integration and Load Testing

- Parameters and Variables
- Creating Templates
- Creating Tests
- Running Tests
- Load Testing
- Simple Overview
- Test Modules
- Load Test Results
- User Simulation
- Throughput Stress Test
- Integration Tests (Endpoint Test)
- Manual Running Endpoint Test
- JMeter Visual Studio Plugin

Parameters and Variables

Fundamentals:

When performing load testing, it will be common to pass test specific information into other steps of the test. To do this, you will be using Parameters and Variables. The basic difference between the two can be summarized as follows:

Parameters:

Are treated mostly like constants and will exist on a global scope that can be referenced universally by all testing threads.

Examples:

```
NODE_CABINET_TYPE = 4
NODE_DRAWER_TYPE = 5
NODE_FOLDER_TYPE = 6
NODE_FILE_TYPE = 7
```

Variables:

Variables are local to the thread running the tests and will be used for passing ids and the like between test steps. Variables declared on the Test Plan will be copied to a local variable configuration by each testing group. Variables can be assigned to using post processors within JMeter.

Assigning and Referencing Variables and Parameters:

JSON Extractor PostProcessor:

This will be used in general to pull key information out of responses from the server. General information regarding both basic and more advanced use of this post processor can be found here:

https://octoperf.com/blog/2017/03/09/how-to-extract-data-from-json-response-using-jmeter/

Our naming conventions for variables will be in all caps, underscore separated, using the object and property as the name.

Example:

```
TARGET_NODE_ID
TARGET_NODE_TYPE
TARGET_ROLE_ID
```

While TARGET will not be used in all cases, in general the TARGET ID is the singular ID we intend to perform the next test step on. In most steps, the PostProcessors will adjust information to the TARGET IDs based on the server response. Following that, PreProcessor logic that exists outside of the individual tests will use that information to configure certain other variables that may be used in the following step.

To reference a variable that has been stored, you can use the following syntax almost universally throughout JMeter:

```
${TARGET_NODE_ID}
${TARGET_NODE_TYPE}
```

If additional scripting is required, it is also possible to use the BeanShell Processors to access the information and perform logic on it. This is Java, but not too far off from c# and most logic should be easily implemented.

Creating Templates

We've divided up the *Templates* folder into subfolders. There is one subfolder for each api controller in Utopia, and there should be, at least, one template for each endpoint in Utopia.

Templates are essentially small components that can be used to create a test.

A 'template' should at least consist of an HTTP Request with the url and body of the request configured to expect variable data. They may also have other required items like (a post processor to extract data from the response, query paramaters, special http headers, etc).

When making a template, please remove duplicated headers and all unneeded parts from a request (i.e. most of the managers and random timers can go). Most templates will consist solely of the HTTPRequest and occasionally some unique headers.

see the Parameters and Variables page for help on configuring http requests to contain dynamic data in the url
or body

BlazeMeter Recording

BlazeMeter is a chrome extension that can be used to record your http traffic in a file that can be imported to JMeter to quickly create tests.

It can be installed by visiting https://chrome.google.com/webstore/detail/blazemeter-the-continuous/mbopgmdnpcbohhpnfglgohlbhfongabi?hl=en

Make sure to sign into BlazeMeter, accounts are free.

Edits after Recording

Delete HTTP Header Manager:

- Left Menu (Test Plan) -> TestName (GetFileUploads) -> HTTP Header Manager -> Right Click -> Remove

image.png
Image not found or type unknown

Fields:

- Protocol [http] = ${REQUEST_PROTOCOL}

- Server Name or IO = ${BASE_URL_1}
- Port Number = ${SERVER_PORT_NUMBER}

image.png
Image not found or type unknown

- Path = ${TARGET_NODE_ID} (Could also be TARGET_ROLE_ID etc)

image.png
Image not found or type unknown

Parameters:

- URL Encoding = True (Checked)

image.png
Image not found or type unknown

Body Data:

- "id": or "nodeID": = "${TARGET_NODE_ID}"

image.png
Image not found or type unknown

- "parentID": = "${NEW_PARENT_ID}"
- "name": = "${NEW_NODE_NAME}"

image.png
Image not found or type unknown

Video Aids

The following a video on how to use the BlazeMeter chrome extension:

https://www.youtube.com/embed/x0RlSnDBlR0?app=Azure%2520DevOps

Creating a Basic Template in JMeter

The following video is a walkthrough on how to create a basic template:

https://www.youtube.com/embed/ZZE8FqYlwbw?app=Azure%2520DevOps

Creating Tests

The 'Tests' folder is divided into subfolders to group different types of tests.

Tests are different from templates because they are designed to actually be run.

Within each subfolder there should be a 'LoadTests' folder, and an 'IntegrationTests' folder. These folders should contain the same tests as their parent folders but they should be configured slightly differently. Integration tests should do some validation to make sure the operation completed successfully and accurately. Load tests should be configured to be easily scaled up and down so we can test different load sizes.

Running Tests

Tests should be run in **CLI mode**, see https://blog.e-zest.com/how-to-run-jmeter-in-non-gui-mode/ for details on running a test in CLI mode

Reporting Dashboard: https://www.ubik-ingenierie.com/blog/reporting-feature-in-jmeter/

Load Testing

We use JMeter to create and run load tests in Utopia. JMeter docs can be found at https://jmeter.apache.org/ .

See simple overview for a little more information.

Recommended Downloads

BlazeMeter Chrome Extension: https://chrome.google.com/webstore/detail/blazemeter-the-continuous/mbopgmdnpcbohhpnfglgohlbhfongabi?hl=en

File Structure

All things load testing related can be found in the Utopia Repository at Utopia\UtopiaTesting.

3rdParty

JavaSetup8u221.exe: Java 8+ is required to run JMeter. Running this file will ensure it is installed on your machine.

apache-jmeter-5.1.1.zip: This is a compressed version of all the files required to run JMeter. Unzip this zip file somewhere on your machine. To run JMeter, use the 'apache-jmeter-5.1.1\bin\jmeter.bat' file. (Running the 'ApacheJMeter.jar' does not include all need files for scripting, and possibly other things)

Templates

Templates are essentially small components that can be used to create a test.

see Creating Templates for more information.

Tests

Tests contain the actual tests that can be run to test different components of Utopia.

see Creating Tests or Running Tests for more information.

Resources

This folder is intended to contain data needed for tests. (for example, login credentials for test users).

Simple Overview

General Process followed

1. We created individual templates of every endpoint.
2. We Created Modules for each Controllers set of endpoints, these should be self contained tests, with their own setup and tear down to allow for the endpoints to be ran. Some Modules use its own endpoints as part of the test, where one endpoint creates and later will delete.
3. Full System tests. These tests are the User Simulation test and the Throughput Stress test. These tests incorporate many (if not all) modules/templates, and condenses them down to do proper setup and tear down for the entire test and not for individual modules/endpoints, with a few exceptions. The User Simulation test took the more "popular" endpoints or all the ones used in a given interval (I think it was within a months time) and used all those endpoints to simulate a regular load we would see in a given interval of time. The Throughput Stress test takes all known endpoints and calls them as many times as possible in a small interval window.
4. We ran the tests against local, Dev and Staging (full production snapshot) environments. We ran them using both the JMeter GUI (considerably long run times) and the CLI methods with a few options also being set. We also used the CLI to generate a report or dashboard and had to create a manual filter on all the endpoints (removing all the setup and tear down pieces). So make sure to gather all the options (destinations folders, reporting files, test locations etc) before using the CLI to run the tests.

Steps to utilize tests

1. Make sure you've downloaded all relevant software (JMeter etc) to load the tests in the GUI of JMeter or to use the CLI. This is also in our code C:...\Utopia\UtopiaTesting\3rdParty
2. Find or create the test you wish to run. Can also be found C:...\Utopia\UtopiaTesting\Tests
3. Make sure all configuration items have the proper path (some actual tests may also need the proper path, however those should of all been made to a relevant path)
   image.png
   Image not found or type unknown
   or
   image.png
   Image not found or type unknown
   Also make sure you have proper Username and Passwords in the files C:...\Utopia\UtopiaTesting\Resources\Test Data\CSV
4. Make sure the test has the proper test Turned on/Enabled.
   image.png
   Image not found or type unknown
   This image shows "Delete User Settings" as disabled and "Delete Account Items" as

enabled.

5. a) If in the GUI make sure the test has "Result Items" like these:
   image.png
   Image not found or type unknown
   so that when the test is ran, data is collected and able to be viewed. These result pages can also be saved off to a file regardless of the test being ran through the GUI or CLI.
   b) If using the CLI a statement like:
   `jmeter -n -t C:\...\Utopia\UtopiaTesting\Tests\LoadTests\ThroughputStressTest.jmx -l C:\...\StressTestResport.csv -j C:\...\StressTestLog.log -e -o C:\...\ComprehensiveTest`
   this statement has several options, a few of these options are: where the test is, whether to have a log file, where the log file should save to, and where the dashboard should be saved to. Look up JMeter CLI run Options to find out more details.

Test Modules

General Information

The purpose of our test modules is to create a completely finished, plug and play style test template that can be merged into virtually any JMeter test. The goal is for NO additional configuration to be required to get these modules to work. To achieve this goal, we have a design philosophy behind every module that can be broken down into the following categories:

Modular Set Up
Endpoint Set Up
Endpoint Execution
Endpoint Tear Down
Modular Tear Down

Each module will contain any module specific information that it requires to run within it's own scope. Examples of this would be that the FileUploadModule will contain references to files that will be uploaded. Then, if a portion of the FileUploadModule will be required within a test, the module can be imported with any of the excess endpoints being disabled or removed altogether.

Structure

All of the Test Modules will consist primarily of Transaction Controllers with the "Generate Parent Sample" checked so that reports generated will be in a more convenient format.

The naming convention for the structure is as follows:

> {ControllerName}Controller
>
> - Example: RoleController

> > {ControllerName}_SetUp

- If any setup for this controller module to run across all steps needs to be done, it should be done in this Transaction Controller

> {ControllerName}_{EndpointAction}
>
> - Example: Role_Add
>
> > {ControllerName}_{EndpointAction}_SetUp
> > Example: Role_Add_SetUp
> >
> > - If any setup is needed for this action to be able to succeed independently
>
> > {ControllerName}_{EndpointAction}_Execute
> >
> > - This is where the httpRequest to perform the action is run
>
> >

{ControllerName}_{EndpointAction}_Tear Down

- This is where any clean up specific to this action is performed

> {ControllerName}_TearDown
>
> - If there was any setup done that still needs to be cleaned up, it should be run in this step after all the Controller Actions have been performed.

Load Test Results

The following are links for load test results for our different environments

Staging: https://loadtestingresults.blob.core.windows.net/%24web/staging/testResults.html

User Simulation

I (Steve) am going to leave my thoughts on how to set up a user simulation test. If I am not present when the time comes to implement what would have been my favorite part of the testing process. To set this up I would have relied upon the core functionality provided in JMeter to randomize actions that comes in the form of the 'Random Controller' and the ${__Random(0000,9999, (optional)varToStoreValue)}.

I would recommend doing the following:

> General Setup:
> Here I would set up a cabinet to be used for all tests to follow.

Using the modules we have created, I would set up a few general actions that users frequently do as tests. Some of the tests I have thought of would be as follows:

> Create new Drawer for Client
> Upload a File
> Share Drawer with user

> Create Template

4/22/2020 - This approach was altered some. The test can be found at:
src\Utopia\UtopiaTesting\Tests\IntegrationTests\Weighted Random Switch Controller.jmx
This is the test that Steve had prepared following the above idea and as testing was being done, changes and alterations were made.

Throughput Stress Test

Much like Steve's User Simulation test, we also created a "Stress" test to run each endpoint as many times in a given interval. The test that was created can be found: \src\Utopia\UtopiaTesting\Tests\LoadTests\ThroughputStressTest.jmx

This test takes every endpoint we have (as of like Dec 2019) and runs them through a throughput controller. Pretty much takes the endpoint and runs it as many times as it can in a 15 second interval, with a 5 second cool down. The cool down is needed, otherwise the results will be skewed, as call times will cut off (stop listening for the return) and cause the return to timeout, and the results to not display accurate return times.

Integration Tests (Endpoint Test)

This Test can be found "\src\Utopia\UtopiaTesting\Tests\LoadTests\LocalFullEndpointTest.jmx". As of 1/25/2021 This test was used for general purpose localized testing. No effort was used to make this an Integration test. A few of the modules were updated in the process of Converting to Postgres and switching from .Net Framework to .Net Core 5.0. Not all modules were tested or revised with current Utopia changes/fixes to Endpoint logic.

Manual Running Endpoint Test

Prerequisites

1. Need Java installed (I have Java 8 update 261)
   - Just download the latest.
   - If you only want to run the CLI, then Java may not be needed.
2. JMeter unpacked or installed
   - We have an unpacked version "\src\Utopia\UtopiaTesting\3rdParty".
   - Might be able to just download, but our zipped version has additional plugins already installed.
3. A JMeter (.jmx) test file
   - All Tests are here "\src\Utopia\UtopiaTesting"
   - Templates - "\src\Utopia\UtopiaTesting\TestTemplates"
   - Modules - "\src\Utopia\UtopiaTesting\TestModules"
   - "Full" Tests - "\src\Utopia\UtopiaTesting\Tests"
   - Resource Files - "C:\Coding\Work\src\Utopia\UtopiaTesting\Resources"

JMeter Icon Simple Overview

image.png - Main Test being ran, can only have one test plan at a time loaded into JMeter.
image.png - A Results page after running a test.
image.png - Configuration file used to create JMeter "Properties"/variable
image.png - "Thread Group" but used to contain all controllers and relevant test operations.
image.png - A Controller that in the results page will group things for better readability
image.png - HTTP Request, the actual creation of the information to be sent.
image.png - PreProcessor, will run script code before execution of parent (or used as a script before a set of operations)
image.png - PostProcessor, how we set or create JMeter "Properties"/variable for later use (typically return data)

Manual Running of Endpoint Test

1. Load into JMeter
   image.png

2. Open "\src\Utopia\UtopiaTesting\Tests\LoadTests\LocalFullEndpointTest.jmx".
3. Fix configurations for your local instance
   image.png
   Image not found or type unknown
   a. Make sure your protocol and Port are correctly lined up with Utopia (Might need to run and check your IIS in your menu tray).
   b. Make sure to use a valid **Admin User** in your Local DB.
   c. Some tests have used the Names.CSV file to generate "User names" or folder names...
4. This should ALWAYS be the first thing called (http sent to the server)
   image.png
   Image not found or type unknown
5. Make sure the test and any controllers or child items are **Turned on/Enabled**.
   image.png
   Image not found or type unknown
   This image shows "Delete User Settings" as disabled and "Delete Account Items" as enabled.
6. Run the Test and check the result pages to verify if the test passed.
   image.png
   Image not found or type unknown
   a. Green arrow is to run the test
   b. Stop sign is to stop the test mid run
   c. Broom is to clear test results, otherwise each run will just be appended to result pages.
7. (**Optional**) CLI run
   `jmeter -n -t C:\...\Utopia\UtopiaTesting\Tests\LoadTests\LocalFullEndpointTest.jmx -l C:\...\EndPointTestResport.csv -j C:\...\EndPointTestLog.log -e -o C:\...\ComprehensiveTest`
   this statement has several options:
   a. First path is Test location
   b. Second path is file to save results,
   c. Third path is file to save log information,
   d. Fourth path is the Dashboard creation directory.

Basic Steps to Creating new Endpoints

1. Make sure to do this in a new Thread group, or in another JMeter Window (can have several instances open of the same test). Or do the hard thing, and disable all tests in the endpoint test and create a new controller structure for new end point.
2. Structure your Module (new Endpoint) correctly, follow already made examples in EndPoint Test.

   https://dev.azure.com/eFileCabinet/Utopia/_wiki/wikis/Utopia.wiki/39/Test-Module
3. Made sure to "Create" any data needed a head of time in the Setup parts of the structure.
4. Create the HTTP Request "sample" and fill in all the data needed to hit the new end point server side. Also include any "Mock" or fake data in the body as a full JSON structure. Use other end points as an example.
5. If HTTP request requires additional data Parameters or Files Upload can be used. Typically will need a Resource File if it can't be "Mocked" with simple data.

JMeter Visual Studio Plugin

There is a Visual Studio Plugin for JMeter

https://marketplace.visualstudio.com/items?itemName=EmtecInc.JMeterTestExplorer&ssr=false#overview

It was created in 2018, and hasn't been updates since 2019. So some JMeter 5.1.1 features seem to be missing or not able to recognize. It has no concept of Controllers. Which our tests are wrapped in a lot of controllers. We did that for Result grouping in the generated Results pages, and the Dashboard that can be generated as well.

Another caveat this plugin has is that it needs the .jmx Test file to be in the solution you have open. Which all our testing files and resource files are outside of the Solution, but in the repository...

Anyways the download and install of the Plugin is extremely simple. Once installed all you need to do is go to Tools and down to JMeter Test Explorer

image.png
Image not found or type unknown

It'll prompt you with a file explorer asking for the JMeter.Bat file, which is found in the unpacked directory for JMeter/Bin folder.

Should open a window like this, may need to hit the refresh window

image.png
Image not found or type unknown

When expanding the tree for "Utopia" you'll see this (minus the green check circle)

image.png
Image not found or type unknown

If we restructure the Test so that we have no controllers, then the "Thread Group" should be expandable and able to see User Defined Variables and Samplers. The Samplers will be our HTTP Requests, and the User Defined Variables are configuration Properteries/Variables we can use instead of hardcoding everything into the Samplers. If we do hit the Blue arrow and run the test, it'll Save a .CSV file to the Test's location with the results, along with load up a window in Visual Studio showing the results in a simple styled tabled.

If we don't restructure the Test, then the only benefit we have here with this Plugin is a quick link to open the test to run in JMeter GUI. Which is the little "Red Gradient" feather icon.