FREE eBook

LEARNING Apache JMeter

Free unaffiliated eBook created from
Stack Overflow contributors.

#jmeter

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: apache-jmeter

It is an unofficial and free Apache JMeter ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Apache JMeter.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with Apache JMeter

Remarks

JMeter is a Load-Testing Tool used for Performance Testing. A Performance Tester can record actions in a web browser or manually build a *script* which can then be run with hundreds or thousands of users.

JMeter can be used to create incredibly dynamic users and scenarios using its various elements. For instance, the `CSV Data Set Config` can be used to specify a set of users to log into a web application. The `Regular Expression Extractor` or the CSS/JQuery Extractor can be used to save session ids to be used in future requests. The `JSR223 PreProcessor` coupled to Groovy language can be used to create dynamic unique data for each user to be sent as part of a `POST` body.

Versions

Version	Java Version	Release Date
3.2	Java 8+	2017-04-14
3.1	Java 7+	2016-11-20
3.0	Java 7+	2016-05-17
2.13	Java 6+	2015-03-13
2.12	Java 6+	2014-11-10
2.11	Java 6+	2014-01-05
2.10	Java 6+	2013-10-21
2.9	Java 6+	2013-01-28
2.8	Java 5+	2012-10-06
2.7	Java 5+	2012-05-27
2.6	Java 5+	2012-02-01
2.5.1	Java 5+	2011-10-03
2.5	Java 5+	2011-08-17
2.4	Java 5+	2010-07-12

Version	Java Version	Release Date
2.3.4	Java 1.4+	2009-06-21

Examples

Installation or Setup

1. Download a distributed archive from Binaries section of JMeter from Download Apache JMeter page.
2. Depending on the version you downloaded, check minimal Java version requirements and install Java if needed. Ensure the `JAVA_HOME` environment variable is set and points to a correct version.
3. Extract the distribution archive in the directory of your choice.
4. Open JMeter UI:
    - **On Windows**: navigate to `<jmeter_location>\bin` directory and run `jmeterw.bat` or `jmeter.bat`
    - **On Linux/Mac**: navigate to `<jmeter_location>/bin` directory and run `jmeter` or 'jmeter.sh`.

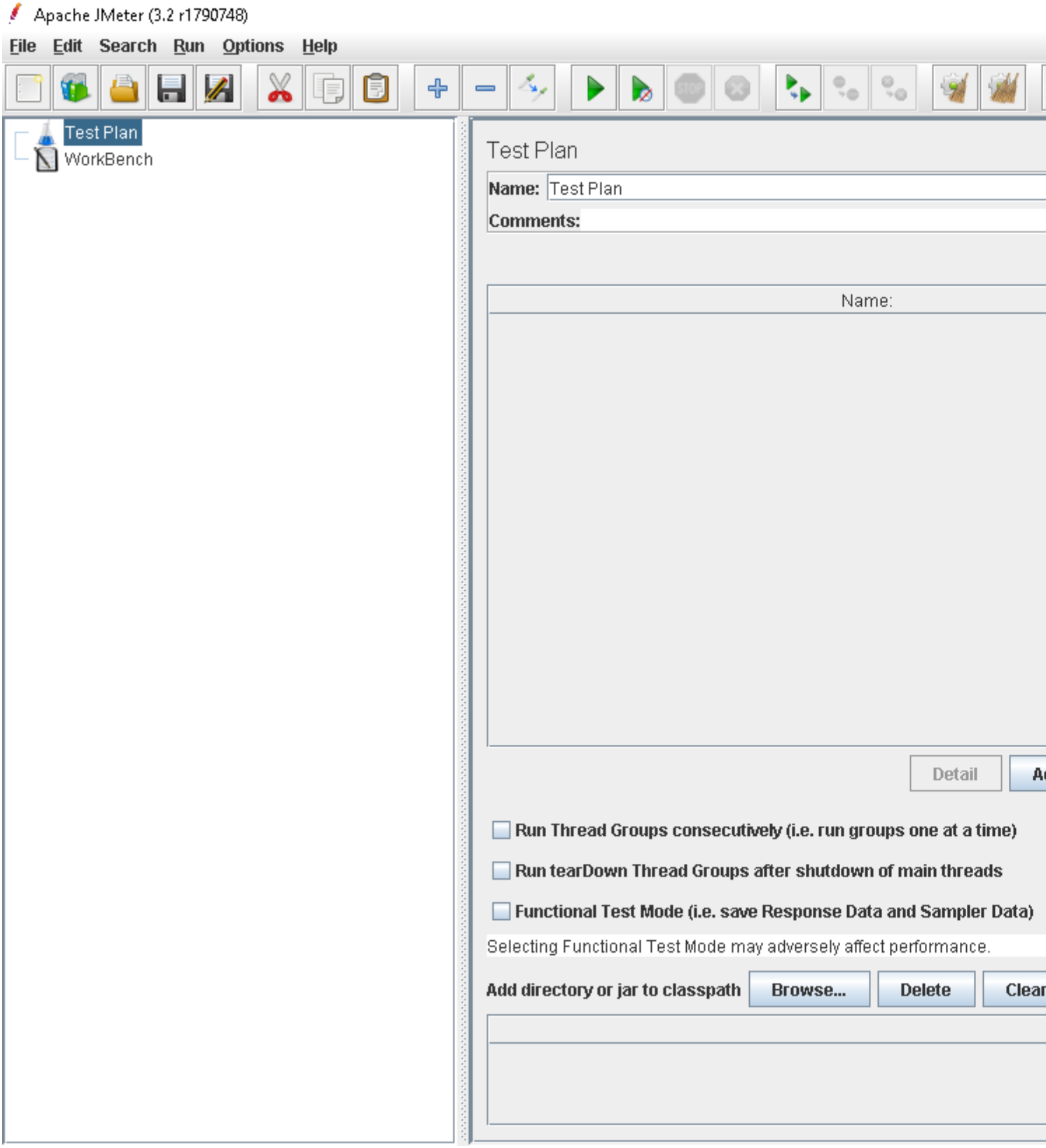
      For example:

      ```
      cd /Users/me/apache-jmeter/bin
      ./jmeter
      ```

      Note: if the above command fails with `Permission denied` error, set execute permission on `jmeter` file:

      ```
      cd /Users/me/apache-jmeter/bin
      chmod u+x ./jmeter
      ```

If you are able to see JMeter UI, basic setup was successful.



Overview of Apache JMeter components at high level

Apache JMeter segregated all the components into following groups based on their functionality:

1. `Test Plan`: Starting point for scripting. JMeter saves the Test Plan in .jmx format. You add components to the Test Plan by Right Click on the Test Pand and navigating to the component you want to add.

2. `Workbench`: Is a temporary place to start scripting. Along with all the components available in Test Plan, you get `HTTP(s) Test Script Recorder` in order to `record` the browser actions. Scripts can be saved in the Workbench provided you check the "Save Workbench" checkbox, otherwise they are no.
3. `Threads (Users)`: you can define a number of (virtual) users to run, ramp-up time and loop count. you can also define on Test Plan whether Thread Groups need to run in sequential or parallel in the case of multiple Thread Groups. some examples are `Thread Group, setUp Thread Group, and tearDown Thread Group`
4. `Logic Controller`: Allows you define the flow of execution and grouping of the samplers. one of the useful examples is Transaction Controller, where you combine all the samplers of Login page (all resources including images, .css, and .js files) so that combined response time can be retrieved.
5. `Sampler`: Sampler is the core of the JMeter. It gives components to simulate requests of various protocols such as HTTP, JDBC, FTP, SMTP etc. for example, HTTP sampler allows you simulate an HTTP packet (of GET, POST or any supported methods). Main stream protocols are supported, for others you can use Free or Commercial plugins.
6. `Config Element`: Configuration elements can be used to set up defaults and variables for later use by samplers. Note that these elements are usually processed at the start of the scope in which they are found, i.e. before any samplers in the same scope. `CSV Dataset Config` allows you to provide test data like usernames, passwords of Login scenario `from a file`. `User Defined variables` config element allows you define variables which can be used across the Test Plan but where each Thread has its own copy.
7. `Timer`: By default, a JMeter thread executes samplers in sequence without pausing. Components presented here provide the functionality to introduce `User Think Time` in various forms among samplers. some examples are `Constant Timer, Constant Throughput Timer.`
8. `Pre Processors`: allow you to perform operations/actions before sampler gets executed. `JSR223 Pre Processor` with Apache Groovy (similar to java coding style) allows you to make changes to the sampler before sending it.
9. `Post Processors`: allow you perform operations/actions after sampler get executed. some useful examples are retrieving dynamic value such as Session ID, using `Regular Expression Extractor` post processor for any type of text, `CSS/JQuery Extractor` for HTML, `JSON Extractor` for JSON, `XPath Extractor` for XML.
10. `Assertions`: As the name suggests, you can assert the response of samplers in different ways like searching for some text, the size of the response, and duration to receive the response etc. For example, you can use `Response Assertion` to search for some text in the response. If Assertion fails, JMeter marks the sampler, to which Assertion is applied, as Failure.
11. Listeners: Listeners allow you to save the test results, see the test execution etc. for example, using `View Results Tree`, you can see the samplers request/response and whether they marked as PASS (green colour)/FAIL (red colour) by JMeter. using Aggregate Report, you can save the test results in CSV format. Important note is that, you use listeners either before the test run (for test script debug) or after the test run (to view results in graphs or summary) but not during the run. we must remove the listeners during the test as it consume a lot of system resources. So, we run the test in non-GUI mode and save the results using `-l` option in `.csv/.jtl` formats. Post the test, you can load this saved files into any of the listeners in the JMeter to view graphs/summary.

Following is the general syntax (`you add any component on need basis`):

```
Test Plan
    Thread Group
        Config Element
        Logic Controller
            Pre Processor
            Sampler
            Timer
            Post Processor
            Assertion
        Listener
```

References:

1. Test Plan and Components
2. Execution Order
3. Scoping Rules

Read Getting started with Apache JMeter online: https://riptutorial.com/jmeter/topic/1941/getting-started-with-apache-jmeter

Chapter 2: Apache JMeter Correlations

Introduction

In JMeter performance testing, Correlations means the ability to fetch dynamic data from the server response and to post it to the subsequent requests. This feature is critical for many aspects of testing, like token-based protected applications.

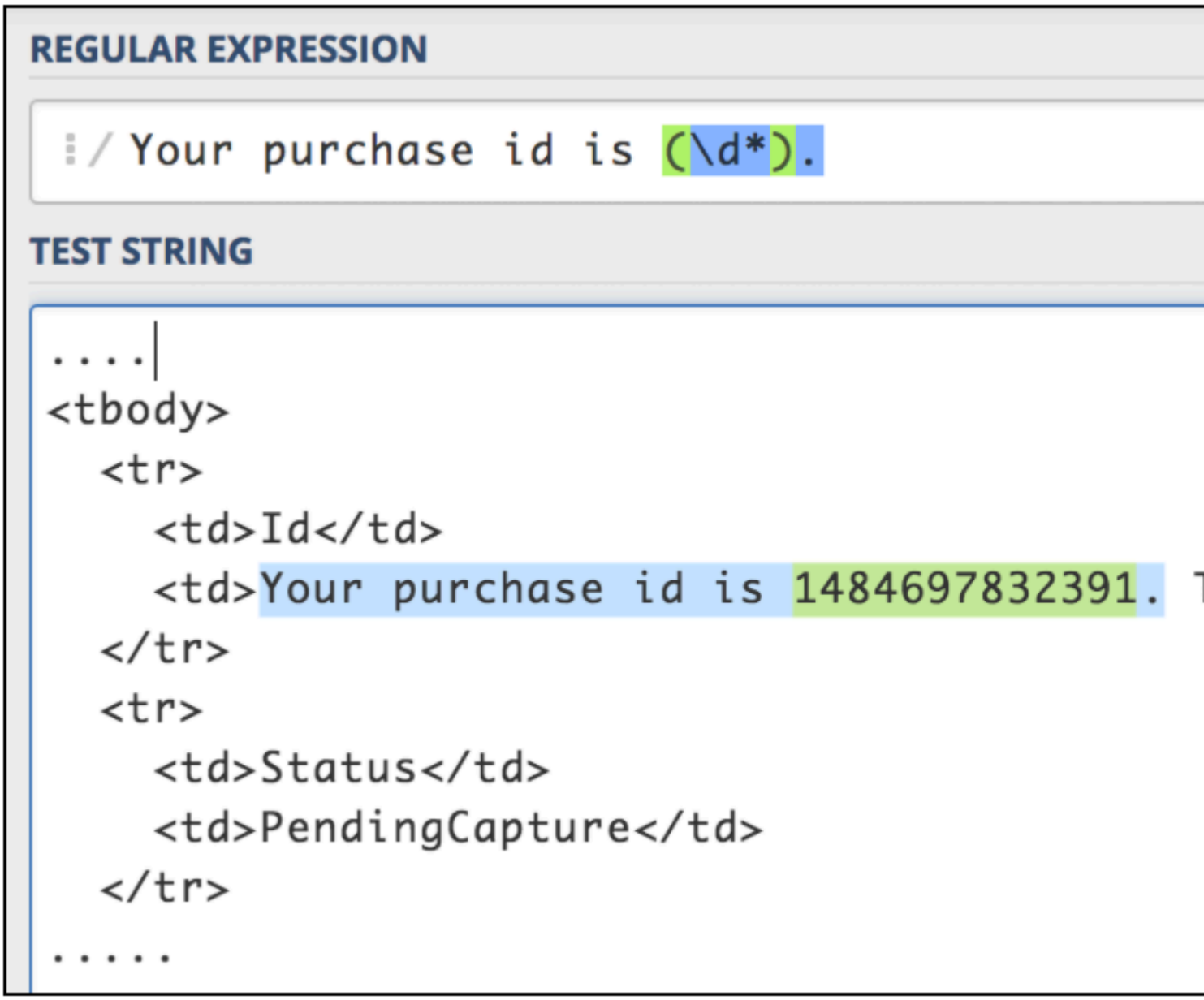
Examples

Correlation Using the Regular Expression Extractor in Apache JMeter

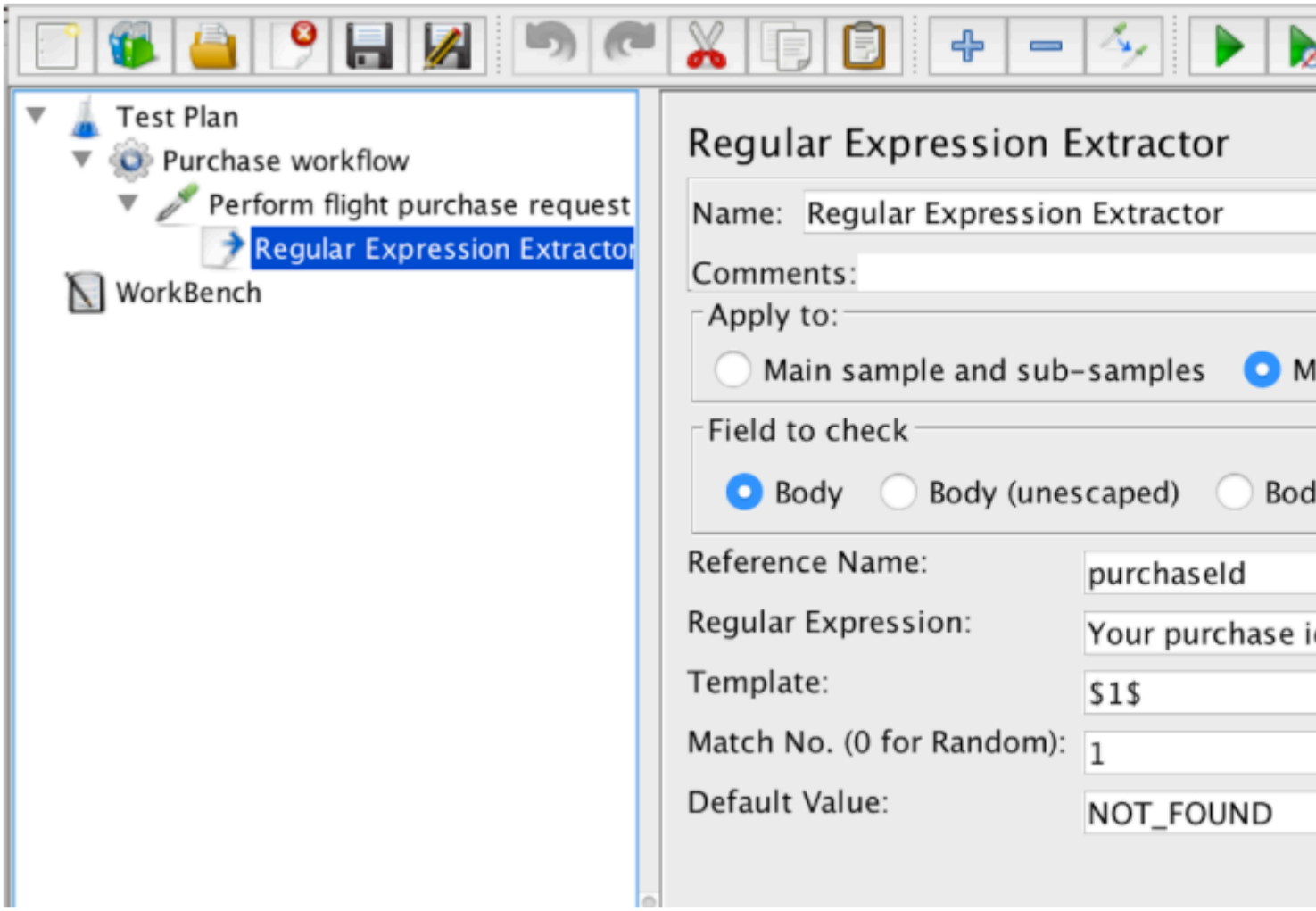
If you need to extract information from a text response, the easiest way is to use Regular Expressions. The matching pattern is very similar to the one used in Perl. Let's assume we want to test a flight ticket purchase workflow. The first step is to submit the purchase operation. The next step is to ensure we are able to verify all the details by using the purchase ID, which should be returned for the first request. Let's imagine the first request returns a html page with this type of ID that we need to extract:

```
<div class="container">
  <div class="container hero-unit">
    <h1>Thank you for you purchse today!</h1>
       <table class="table">
           <tr>
             <td>Id</td>
             <td>Your purchase id is 1484697832391</td>
           </tr>
           <tr>
             <td>Status</td>
             <td>Pending</td>
           </tr>
           <tr>
             <td>Amount</td>
             <td>120 USD</td>
            </tr>
         </table>
    </div>
</div>
```

This kind of situation is the best candidate for using the JMeter Regular Expression extractor. Regular Expression is a special text string for describing a search pattern. There are lots of online resources that help writing and testing Regular Expressions. One of them is https://regex101.com/.



To use this component, open the JMeter menu and: *Add -> Post Processors -> Regular Expression Extractor*



The Regular Expression Extractor contains these fields:

- Reference Name - the name of the variable that can be used after extraction
- Regular Expression - a sequence of symbols and characters expressing a string (pattern) that will be searched for within the text
- Template - contains references to the groups. As a regex may have more than one group, it allows to specify which group value to extract by specifying the group number as 1 or 2 or $1$$2$ (extract both groups)
- Match No. - specifies which match will be used (0 value matches random values / any positive number N means to select the Nth match / negative value needs to be used with the ForEach Controller)
- Default - the default value which will be stored into the variable in case no matches are found, is stored in the variable.

The “Apply to” checkbox deals with samples that make requests for embedded resources. This parameter defines whether Regular Expression will be applied to the main sample results or to all requests, including embedded resources. There are several options for this param:

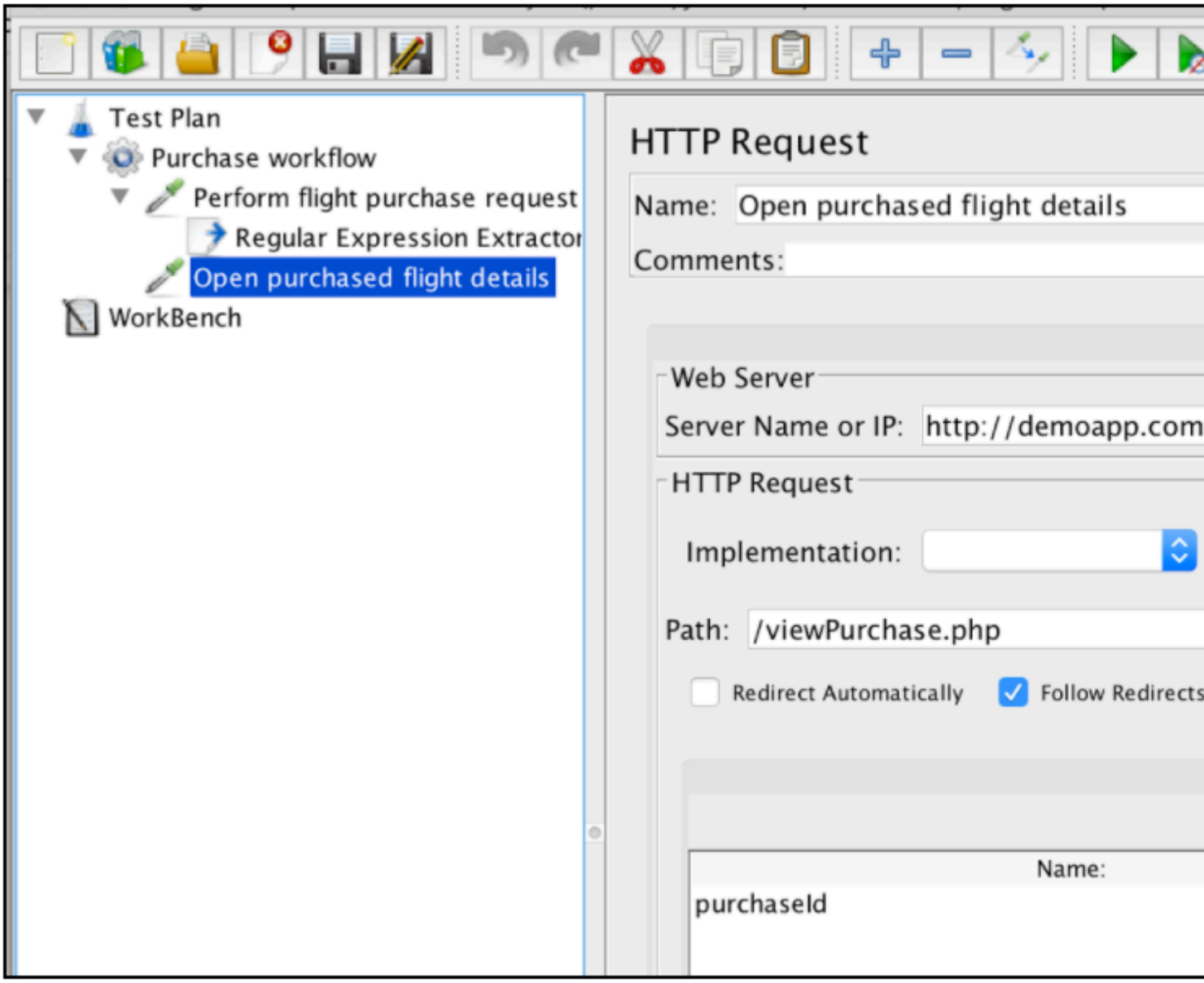
- Main sample and sub-samples
- Main sample only
- Sub-samples only
- JMeter Variable - the assertion is applied to the contents of the named variable, which can

be filled by another request

The “Field to check” checkbox enables choosing which field the Regular Expression should be applied to. Almost all parameters are self descriptive:

- Body - the body of the response, e.g. the content of a web-page (excluding headers)
- Body (unescaped) - the body of the response, with all HTML escape codes replaced. Note that HTML escapes are processed without regard to context, so some incorrect substitutions may be made (*this option highly impacts performance)
- Body - Body as a Document - the extract text from various type of documents via Apache Tika (*also might impact performance)
- Body - Request Headers - may not be present for non-HTTP samples
- Body - Response Headers - may not be present for non-HTTP samples
- Body - URL
- Response Code - e.g. 200
- Body - Response Message - e.g. OK

After the expression is extracted, it can be used in subsequent requests by using the ${purchaseId} variable.



This table contains all the contractions that are supported by JMeter Regular Expressions:

chars	action
abc...	Match that character (metacharacte
\\\.*...	Match that metacharacter following
""	De-meta any chars inside quotes
\t,\n,\r,\f	tab, newline, return, form feed
.	Match any character
[]	Character class
[^]	Inverse Character class
[-]	Character ranges
\w	Match a "word" character (alphanu
\W	Match a non-word character
\s	Match a whitespace character
\S	Match a non-whitespace character
\d	Match a digit character
\D	Match a non-digit character
anchors	**action**
^	Match the beginning of the line
$	Match the end of the line
\b	Match a word boundary
\B	Match a non-(word boundary)

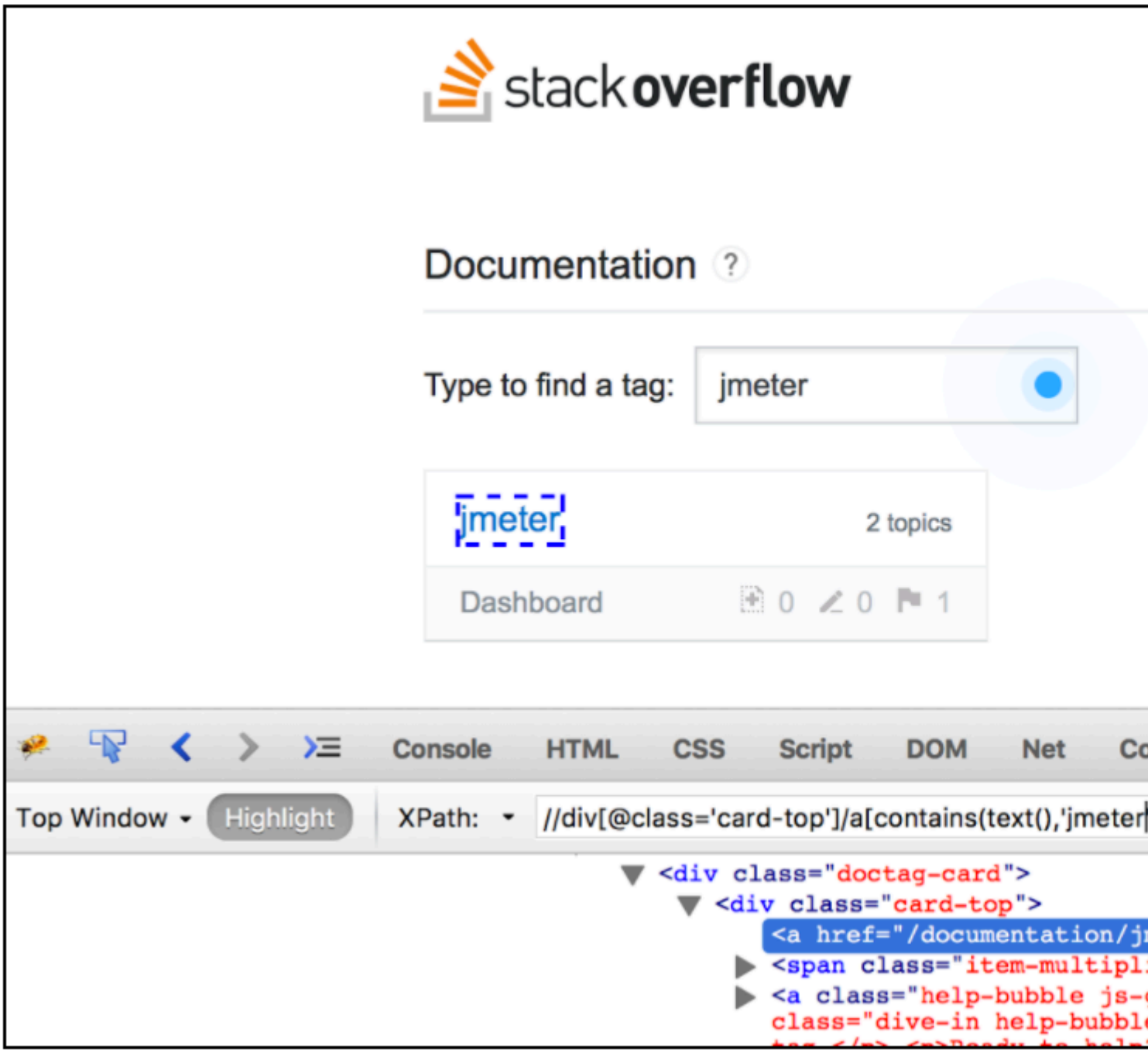
- Report Errors - if a Tidy error occurs, set the Assertion accordingly
- Show Warnings - sets the Tidy show warnings option

If ‘Use Tidy’ is unchecked:

- Use Namespaces - if checked the XML parser will use the namespace resolution
- Validate XML - check the document against its specified schema
- Ignore Whitespace - ignore Element Whitespace
- Fetch External DTDs - if selected, external DTDs are fetched

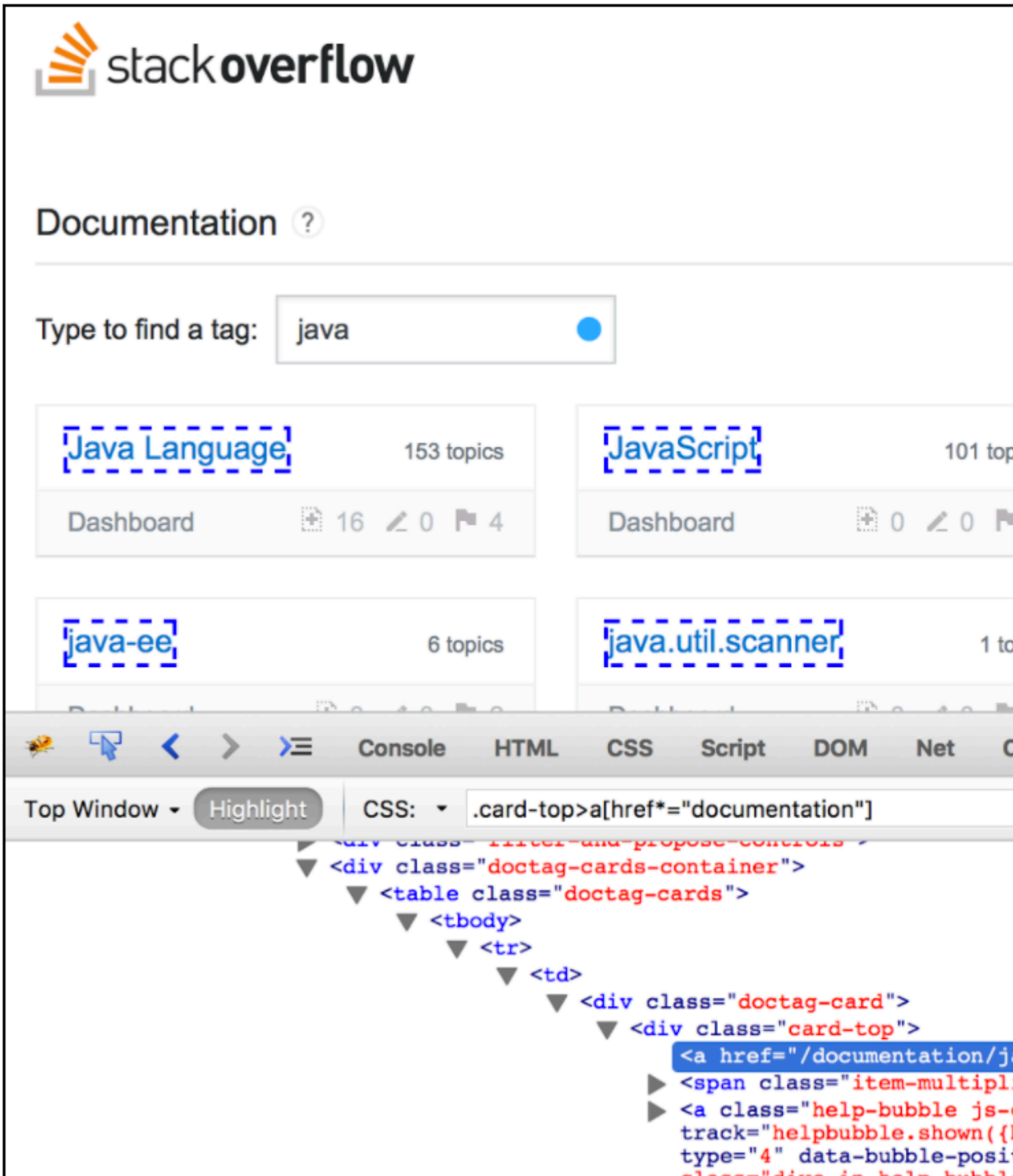
‘Return entire XPath fragment instead of text content’ is self descriptive and should be used if you want to return not only the xpath value, but also the value within its xpath locator. It might be useful for debugging needs.

It is also worth mentioning there are list of very convenient browser plugins for testing XPath locators. For Firefox you can use the ‘Firebug’ plugin while for Chrome the ‘XPath Helper’ is the most convenient tool.

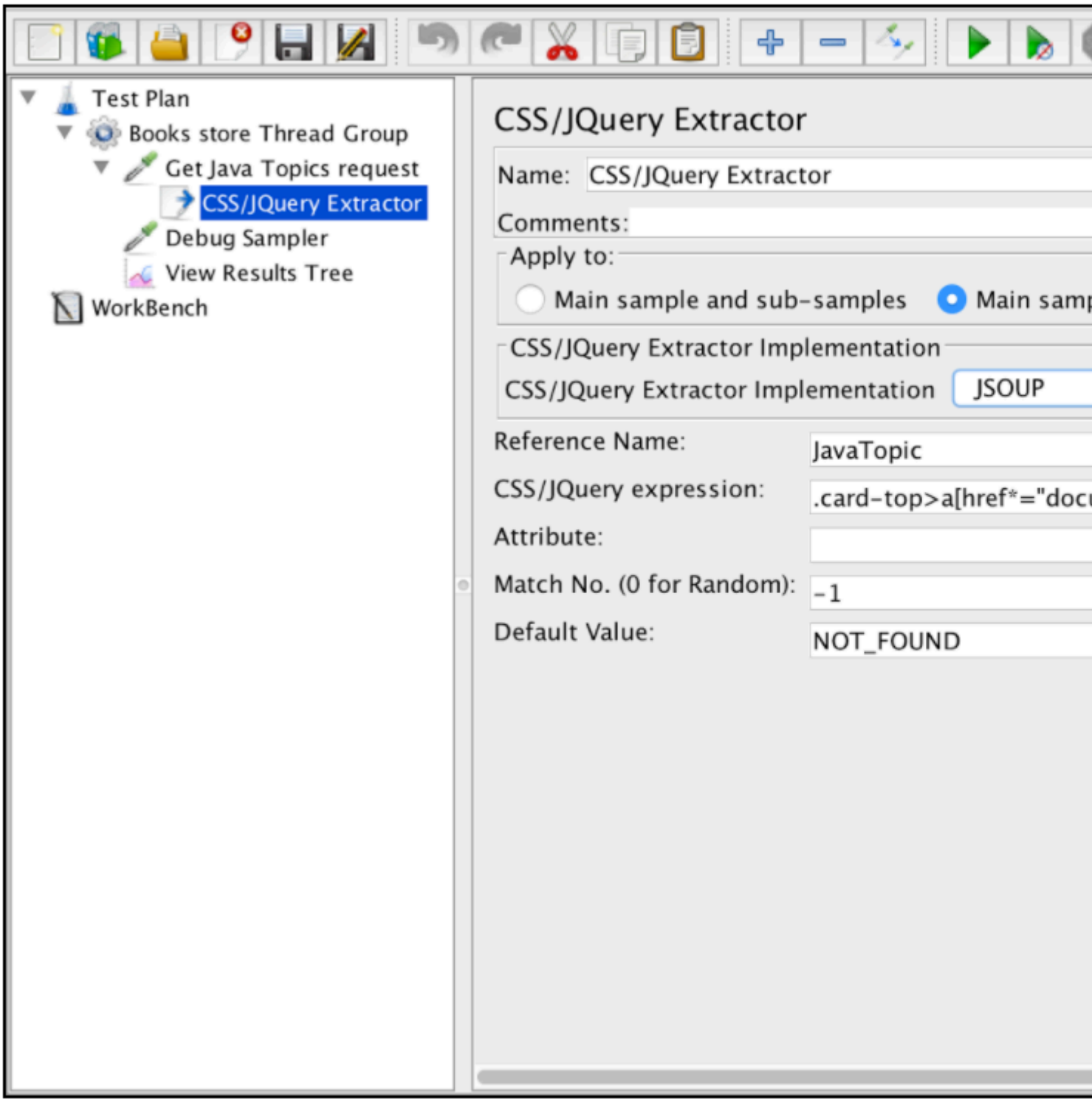


Correlation Using the CSS/JQuery Extractor in JMeter

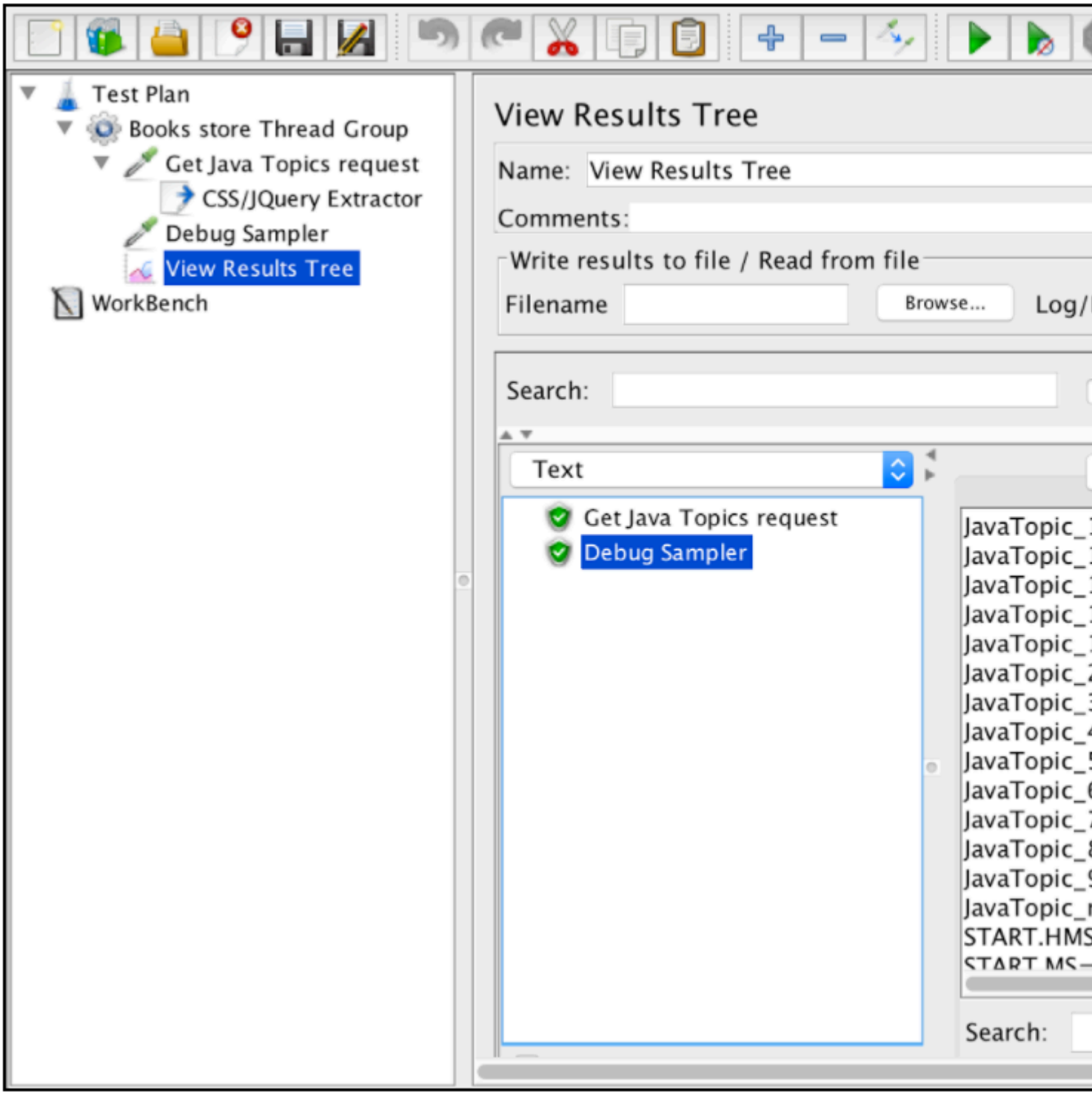
The CSS/JQuery extractor enables extracting values from a server response by using a CSS/JQuery selector syntax, which might have otherwise been difficult to write using Regular Expression. As a post-processor, this element should be executed to extract the requested nodes, text or attribute values from a request sampler, and to store the result into the given variable. This component is very similar to the XPath Extractor. The choice between CSS, JQuery or XPath usually depends on user preference, but it's worth mentioning that XPath or JQuery can traverse down and also traverse up the DOM, while CSS can not walk up the DOM. Let's assume that we want to extract all the topics from the Stack Overflow documentation that are related to Java. You can use the Firebug plugin to test your CSS/JQuery selectors in Firefox, or the CSS Selector Tester in Chrome.



To use this component, open the JMeter menu and: *Add -> Post Processors -> CSS/JQuery Extractor*



Almost all of this extractor's fields are similar to the Regular Expression extractor fields, so you can get their description from that example. One difference however is the "CSS/JQuery Extractor implementation" field. Since JMeter 2.9 you can use the CSS/JQuery extractor based on two different implementations: the jsoup implementation (detailed description of its syntax here) or the JODD Lagarto (detailed syntax can be found here). Both implementations are almost the same and have only small syntax differences. The choice between them is based on user's preference.

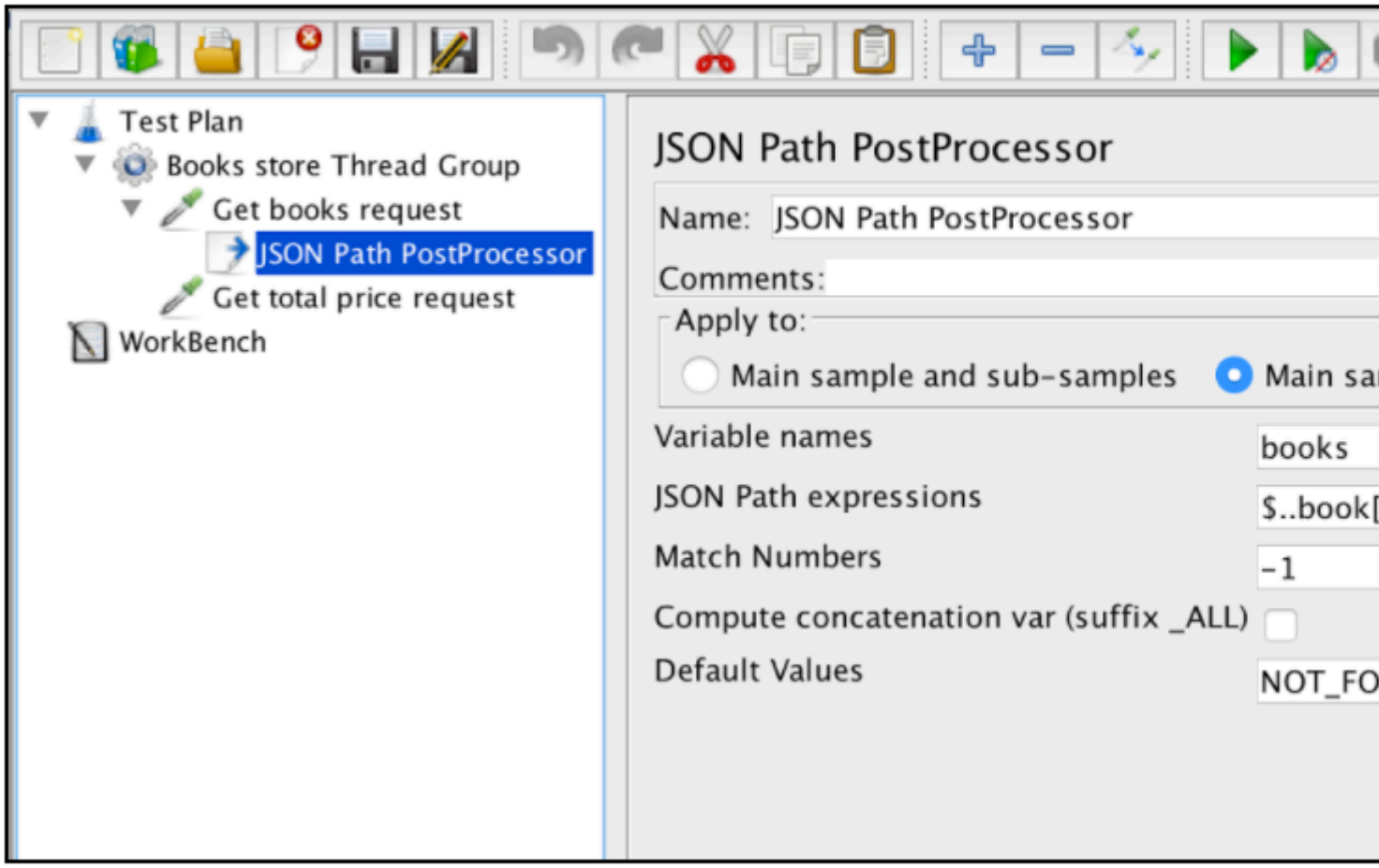


Based on the above mentioned configuration, we can extract all the topics from the requested page and verify the extracted results by using the “Debug Sampler” and the “View Results Tree” listener.

Correlation Using the JSON Extractor

JSON is a commonly used data format that is used in web based applications. The JMeter JSON Extractor provides a way to use JSON Path expressions for extracting values from JSON-based responses in JMeter. This post processor must be placed as a child of the HTTP Sampler or for any other sampler that has responses.

To use this component, open the JMeter menu and: *Add -> Post Processors -> JSON Extractor.*



The JSON Extractor is very similar to the Regular Expression Extractor. Almost all the main fields are mentioned in that example. There is only one specific JSON Extractor parameter: 'Compute concatenation var'. In case many results are found, this extractor will concatenate them by using the ',' separator and storing it in a var named _ALL.

Let's assume this server response with JSON:

```
{
    "store": {
        "book": [
          { "category": "reference",
            "author": "Nigel Rees",
            "title": "Sayings of the Century",
            "price": 8.95
          },
          { "category": "fiction",
            "author": "Evelyn Waugh",
            "title": "Sword of Honour",
            "price": 12.99
          },
          { "category": "fiction",
            "author": "Herman Melville",
            "title": "Moby Dick",
            "isbn": "0-553-21311-3",
            "price": 8.99
          },
```

```
          { "category": "fiction",
            "author": "J. R. R. Tolkien",
            "title": "The Lord of the Rings",
            "isbn": "0-395-19395-8",
            "price": 22.99
          }
        ],
        "bicycle": {
        "color": "red",
        "price": 19.95
    }
  }
}
```

The table below provides a great example of different ways to extract data from a specified JSON:

$.store.book[*].author	the auth
$..author	all autho
$.store.*	all things red bicy
$.store..price	the price
$..book[2]	the third
$..book[(@.length-1)] $..book[-1:]	the last
$..book[0,1] $..book[:2]	the first
$..book[?(@.isbn)]	filter all
$..book[?(@.price<10)]	filter all
$..*	all Elem JSON st

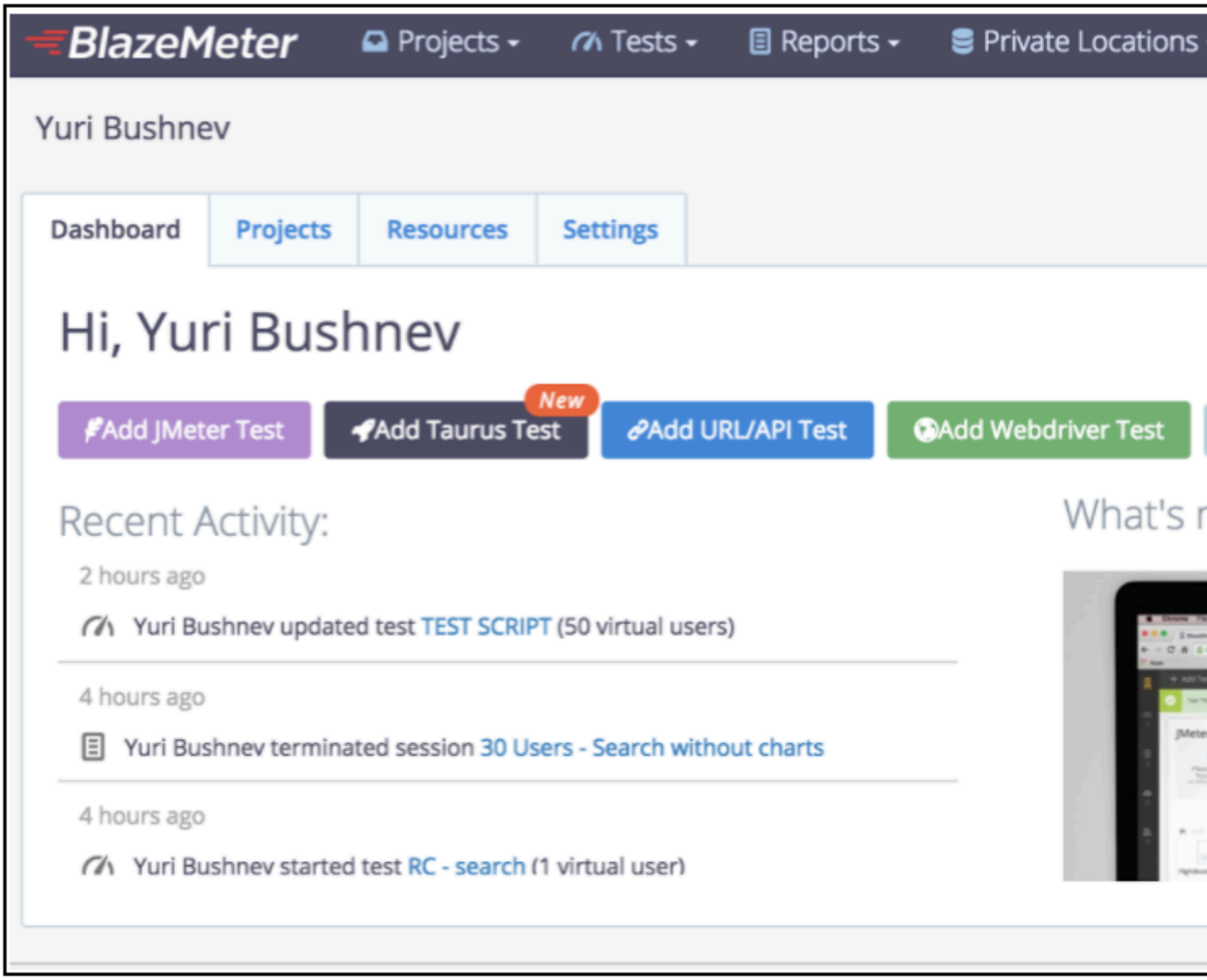
Through this link you can find a more detailed description of the JSON Path format, with related examples.

Automated Correlation by Using BlazeMeter's 'SmartJMX'

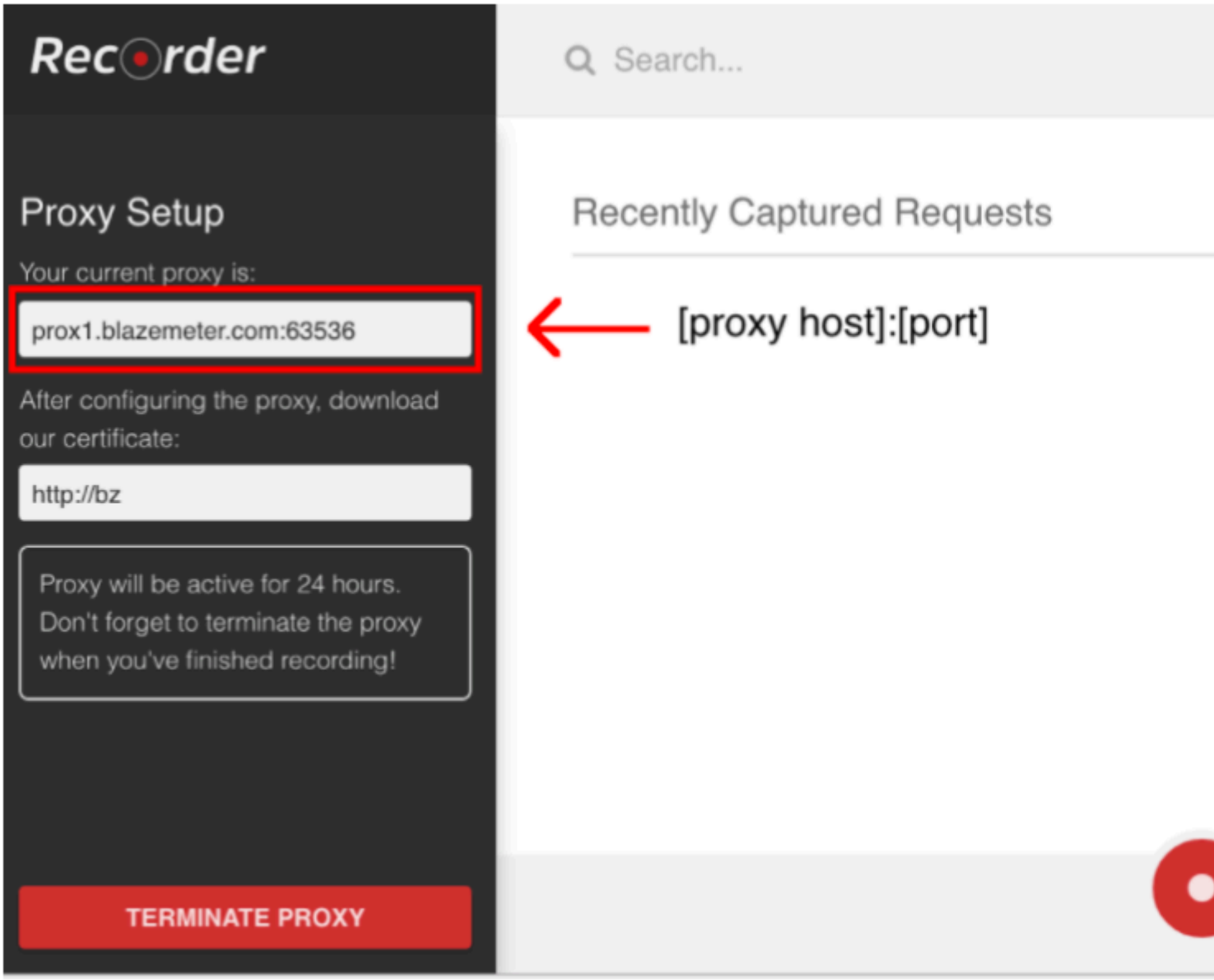
When you manually write your performance scripts, you need to deal with correlation yourself. But there is another option to create your scripts - automation scripts recording. On the one hand, the manual approach helps your write structured scripts and you can add all the required extractors at the same time. On the other hand, this approach is very time consuming.

Automation scripts recording is very easy and lets you to do the same work, only much faster. But if you use common recording ways, the scripts will be very unstructured and usually require adding additional parametrization. The “Smart JMX” feature on the Blazemeter recorder combines the advantages of both ways. It can be found at this link: [https://a.blazemeter.com/app/recorder/index.html][1]

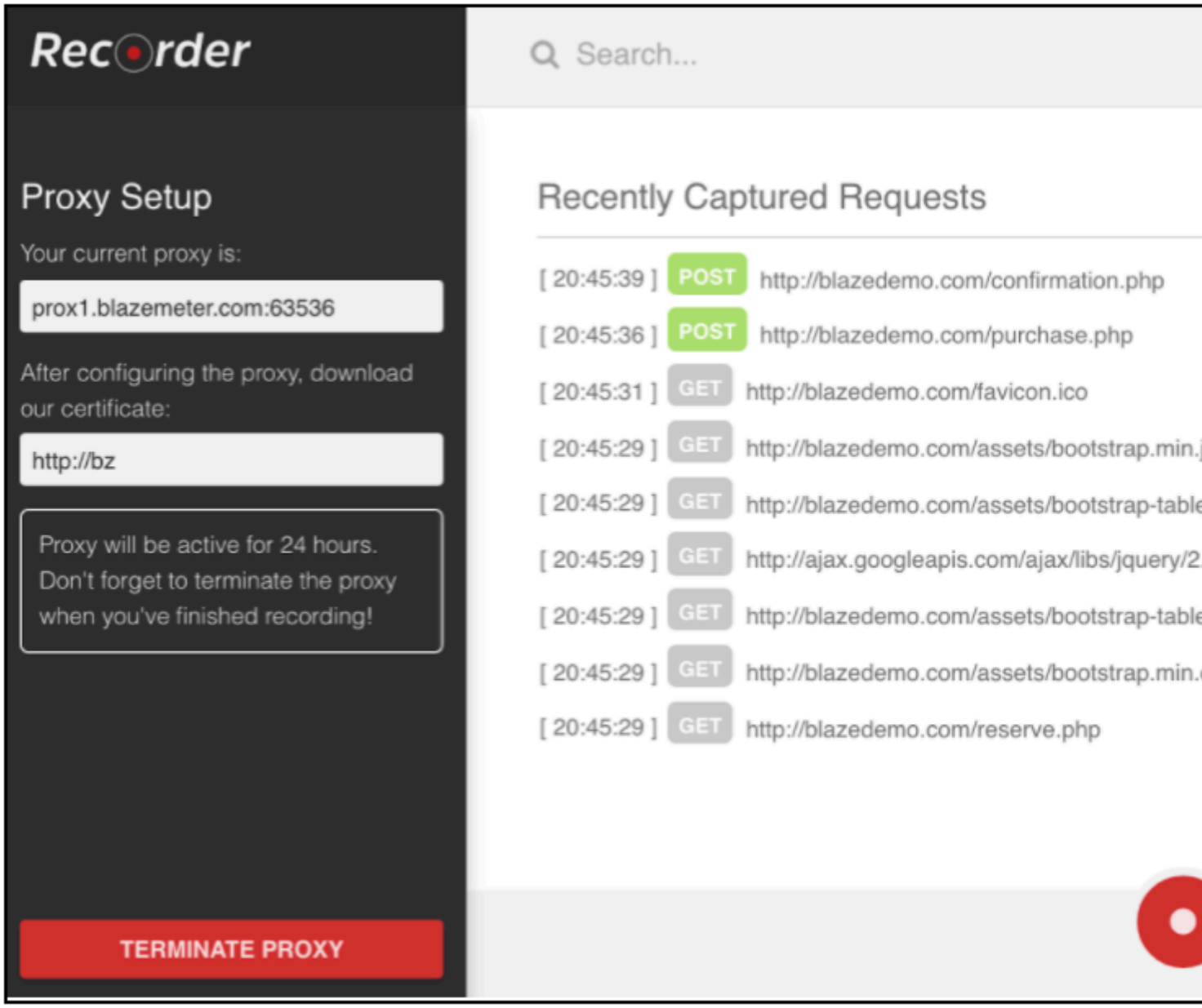
After registration go to “Recorder” section.



To start script recording, first you need to configure your browser’s proxy (covered here), but this time you should get a proxy host and a port provided by the BlazeMeter recorder.



When the browser is configured, you can go ahead with script recording by pressing the red button at the bottom. Now you can go to the application under test and perform user workflows for recording.



After the script is recorded, you can export the results into a “SMART” JMX file. An exported jmx file contains a list of options that allow you to configure your script and parameterize, without additional efforts. One of these improvements is that the “SMART” JMX automatically finds correlation candidates, substitutes it with the appropriate extractor, and provides an easy way for further parametrization.

Read Apache JMeter Correlations online: https://riptutorial.com/jmeter/topic/8978/apache-jmeter-correlations

Chapter 3: Apache JMeter parameterization

Introduction

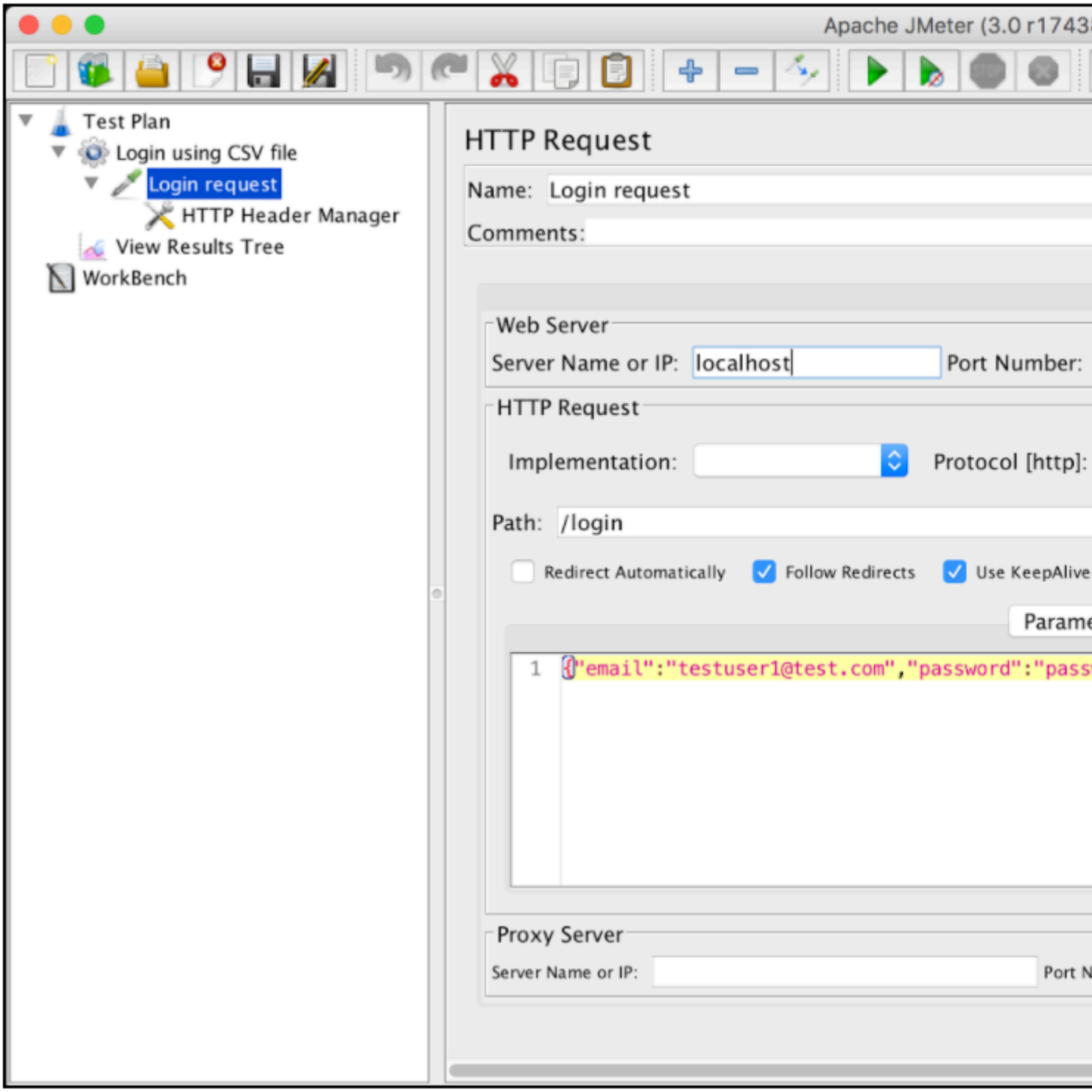
Parameterization is the creation of different data sets for different users in the same test script. For example, running multiple users with different credentials in the same script. This makes it one of the main aspects in performance tests creation.

Examples

Parameterization using external files

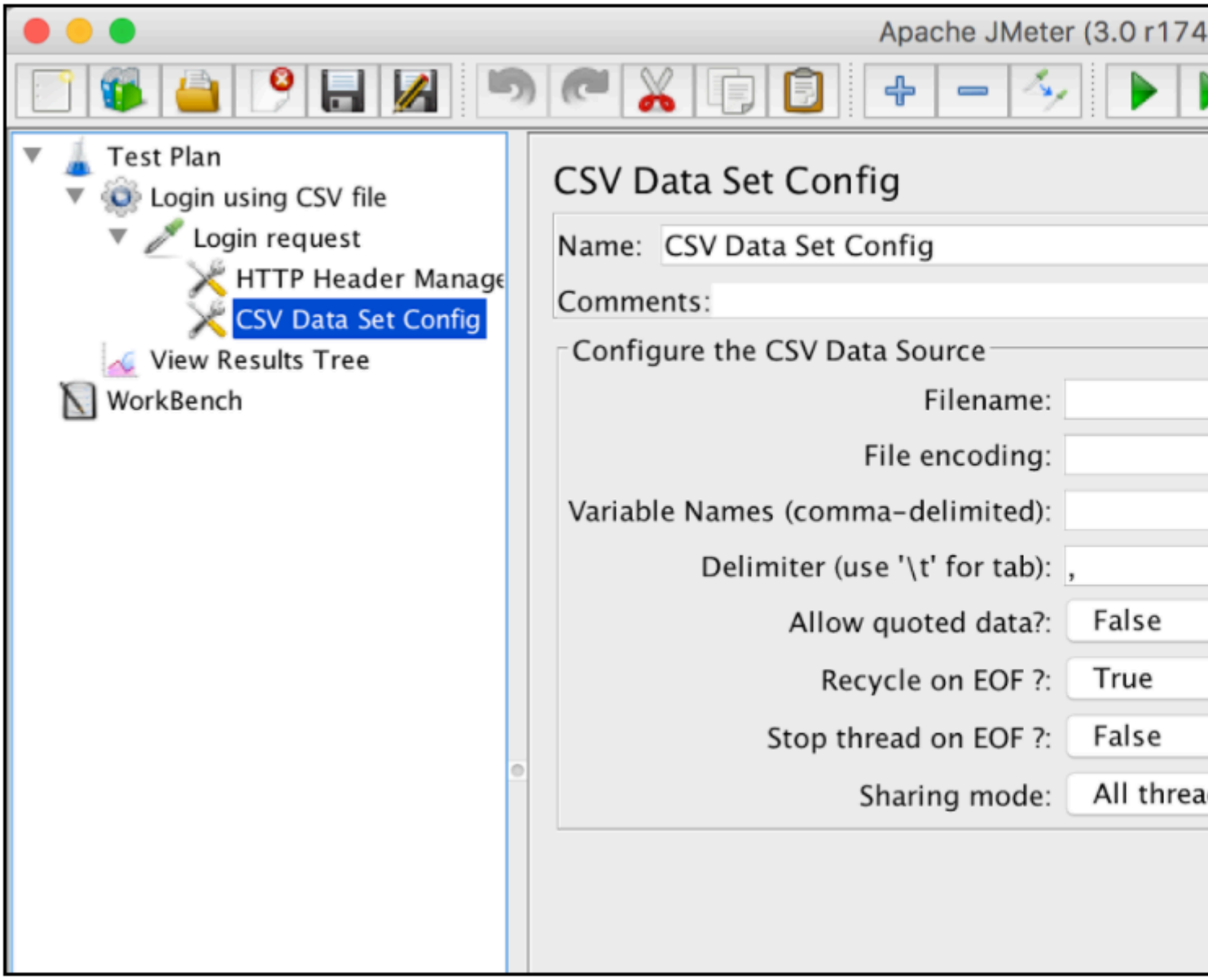
One of the common ways to parametrize your performance scripts is to use a CSV file. The best example of CSV input files usage is a login process. If you want to test your application across different users, you need to provide a list of user credentials.

Let’s assume that we have a login request that works for one specific user:



We can easily parametrize that request by using an external CSV file and running the script across different users. To add CSV parametrization config:

Right click on login request -> Add -> Config Element -> CSV Data Set Config

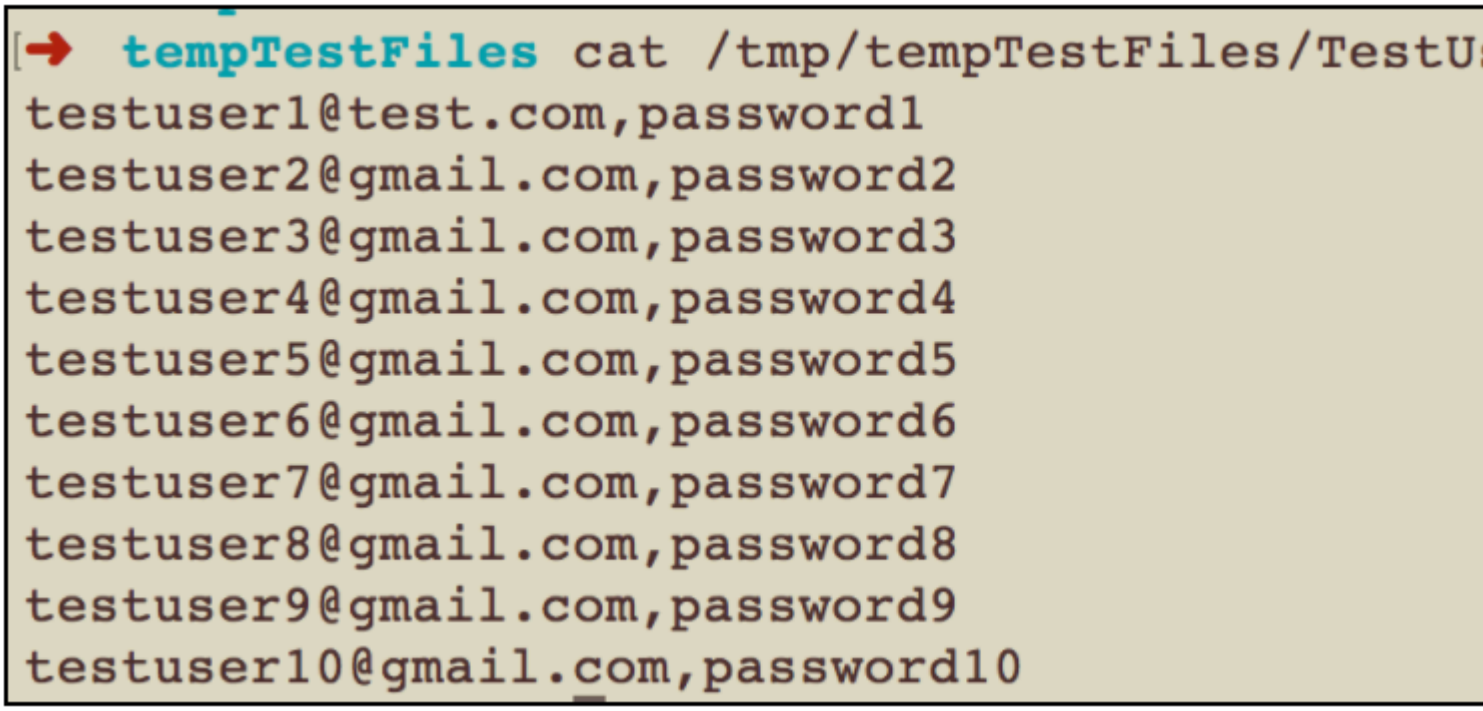


A short explanation of 'CSV Data Set Config' parameters:

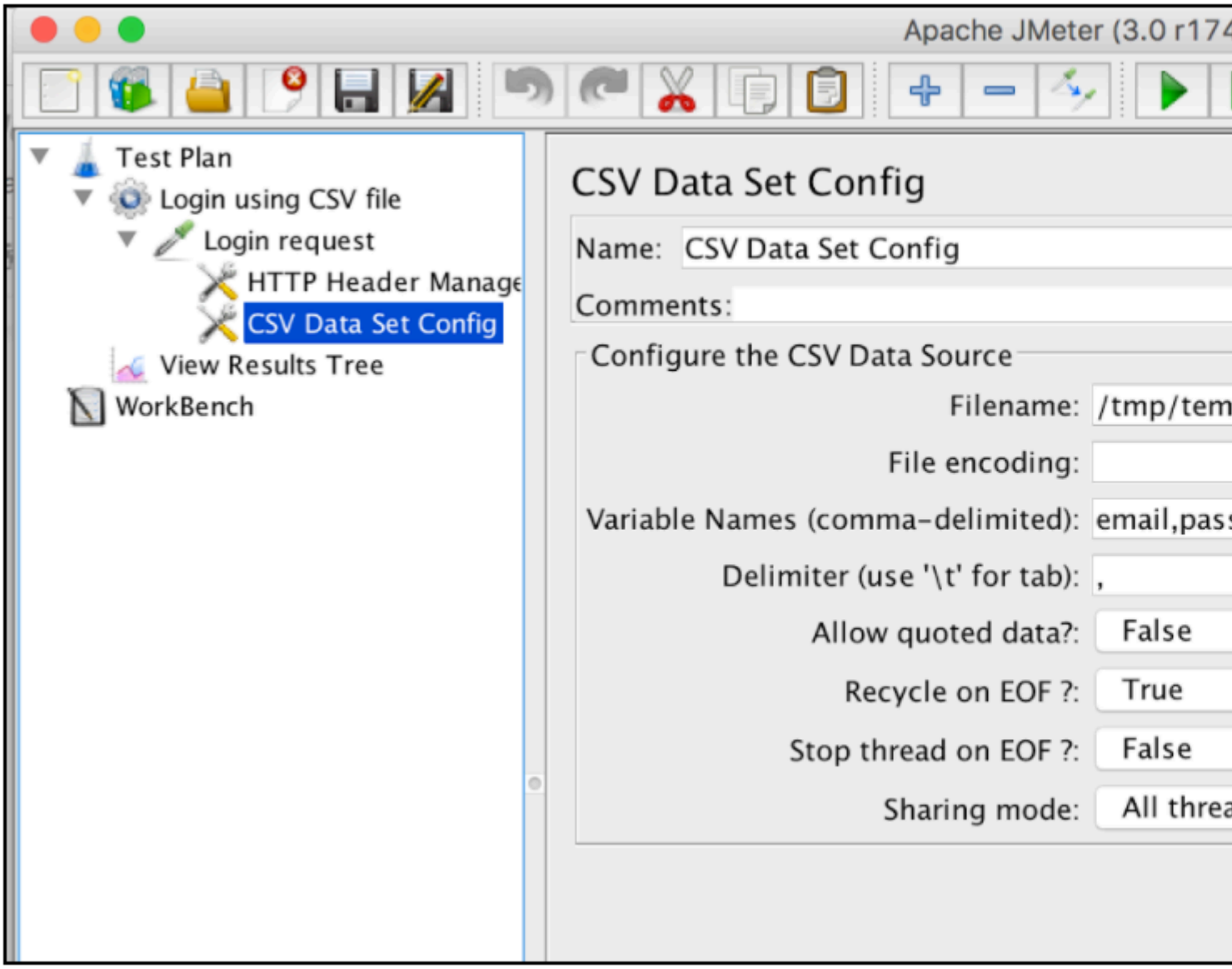
- Name - element name as it will be used in the JMeter tree
- Filename - name of the input file. Relative file names are resolved based on the path of the active test plan. Absolute filenames are also supported
- File Encoding - encoding of input file, if it's not the platform default
- Variable Names - list of separated variable names that will be used as a container for parsed values. If empty, the first line of the file will be interpreted as the list of variable names
- Delimiter - delimiter that will be used to split the parsed values from the input file
- Allow quoted data? - true in case you want to ignore double quotes and allow such elements to contain a delimiter.
- Recycle on EOF? - true in case the file test plan should iterate over the file more than once. It will instruct JMeter to move the cursor at the beginning of the file
- Stop thread on EOF? - false in case of loop iteration over the CDC file and true if you want to stop the thread after reading the whole file
- Sharing mode:
    - All threads - the file is shared between all virtual users (the default)

- Current thread group - the file will be opened once for each thread group
- Current thread - each file will be opened separately for each of threads
- Identifier - all threads sharing the same identifier also share the same file

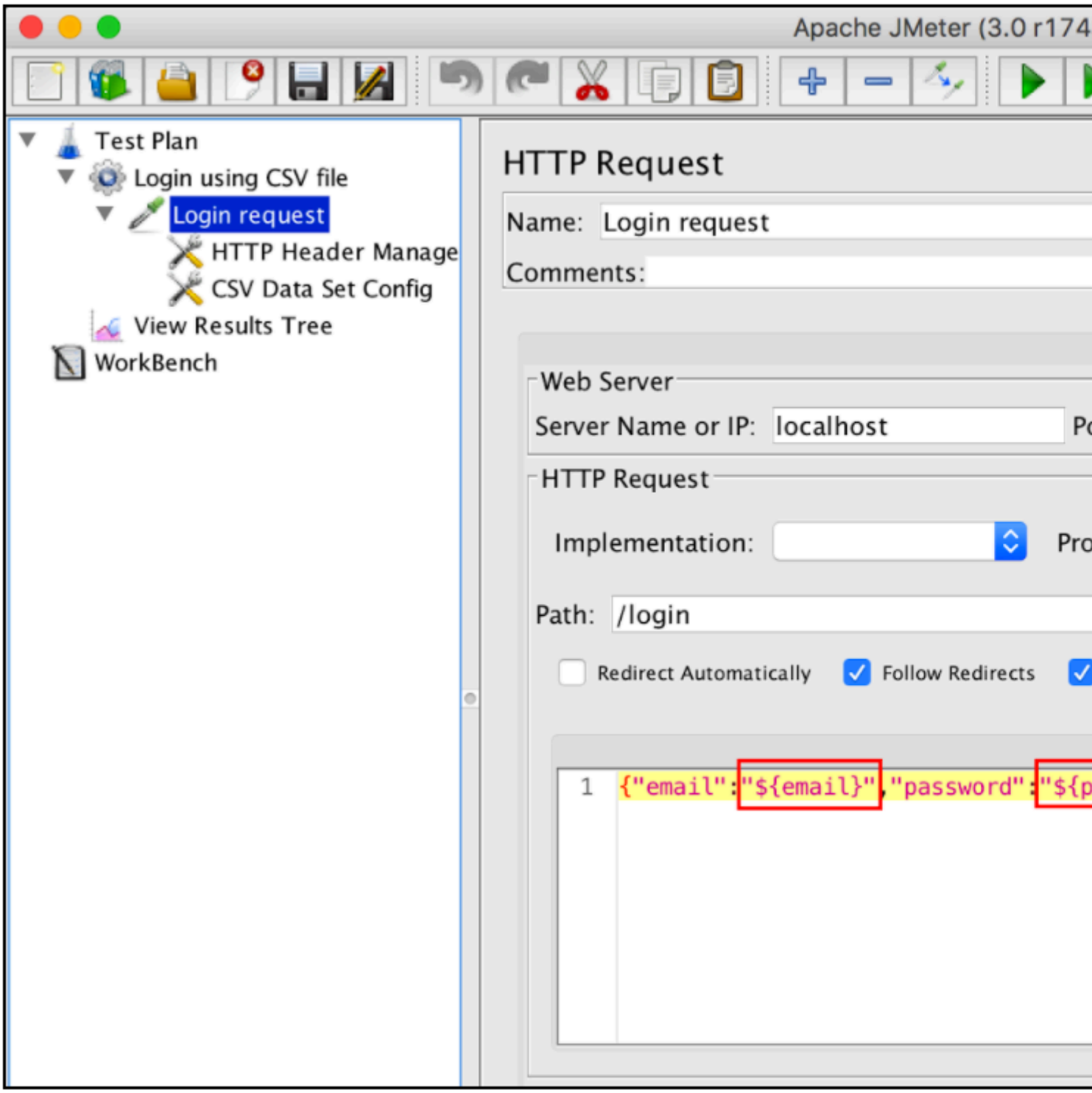
Let's create a csv file containing different users with names and passwords:



We can now use this file with the CSV Data Set config. In our case, it's enough to add the "Filename" and "Variables Names" config values:

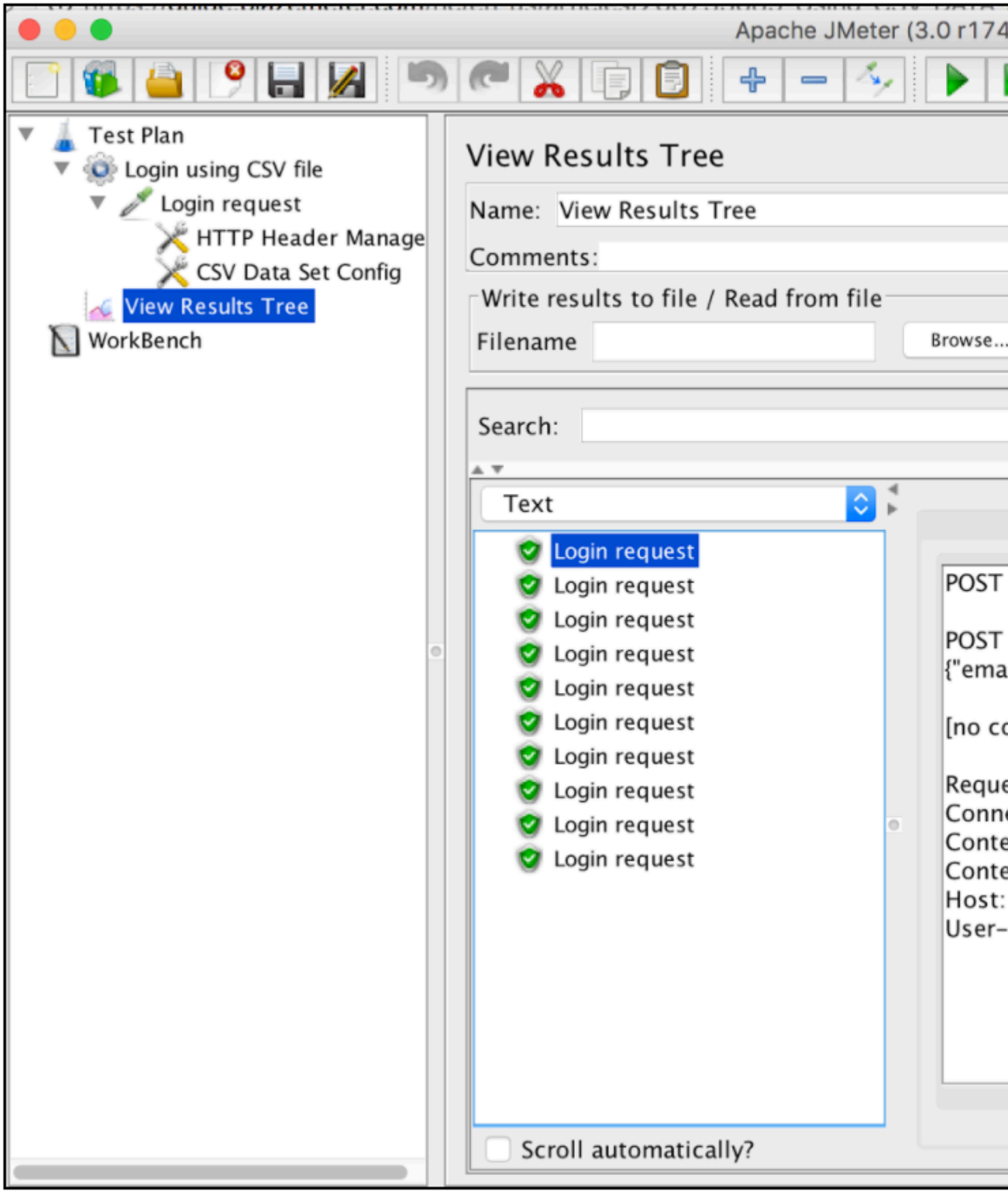


The last step we have to take is to parametrize the login request with CSV variables. This can be done by substituting the initial values with appropriate variables from the “Variable Names” configuration field of the CSV Data Set Config, like this:

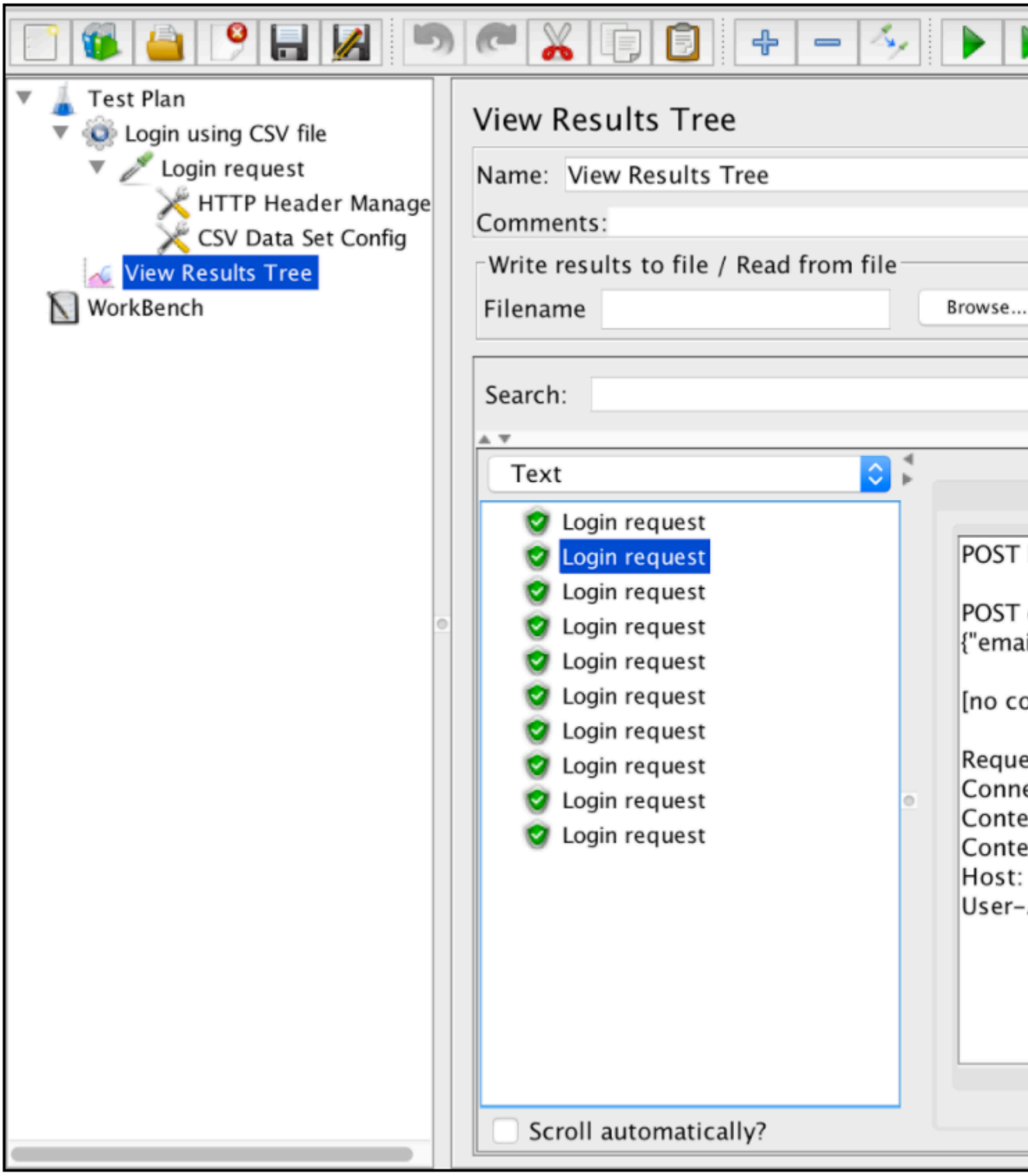


If we run our test script now, JMeter will substitute these variables with values from the 'TestUsers.csv' file. Each JMeter virtual user will receive credentials from the following csv file line.

The login request by the first user:



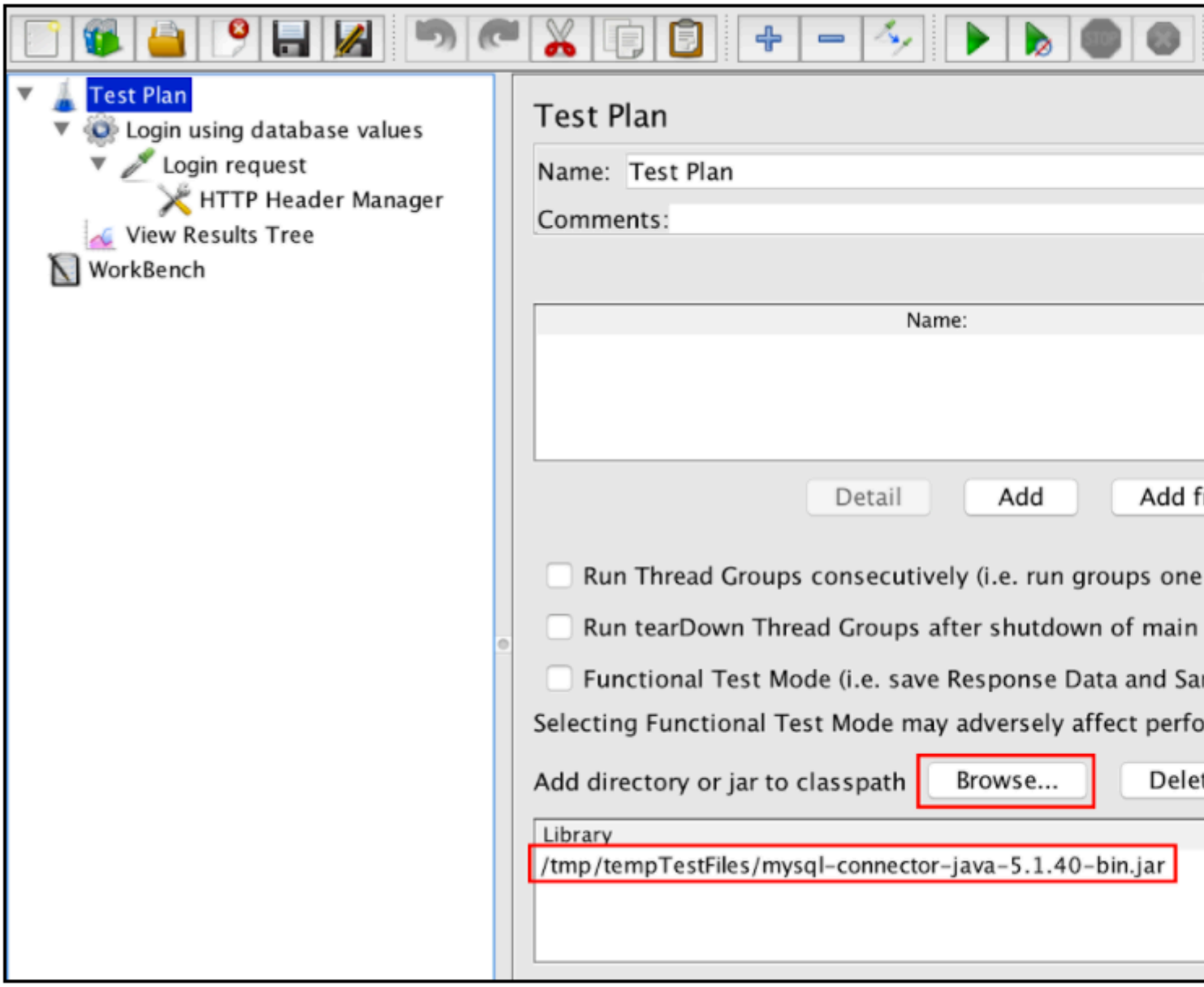
The login request by the second user:



Parameterization using databases

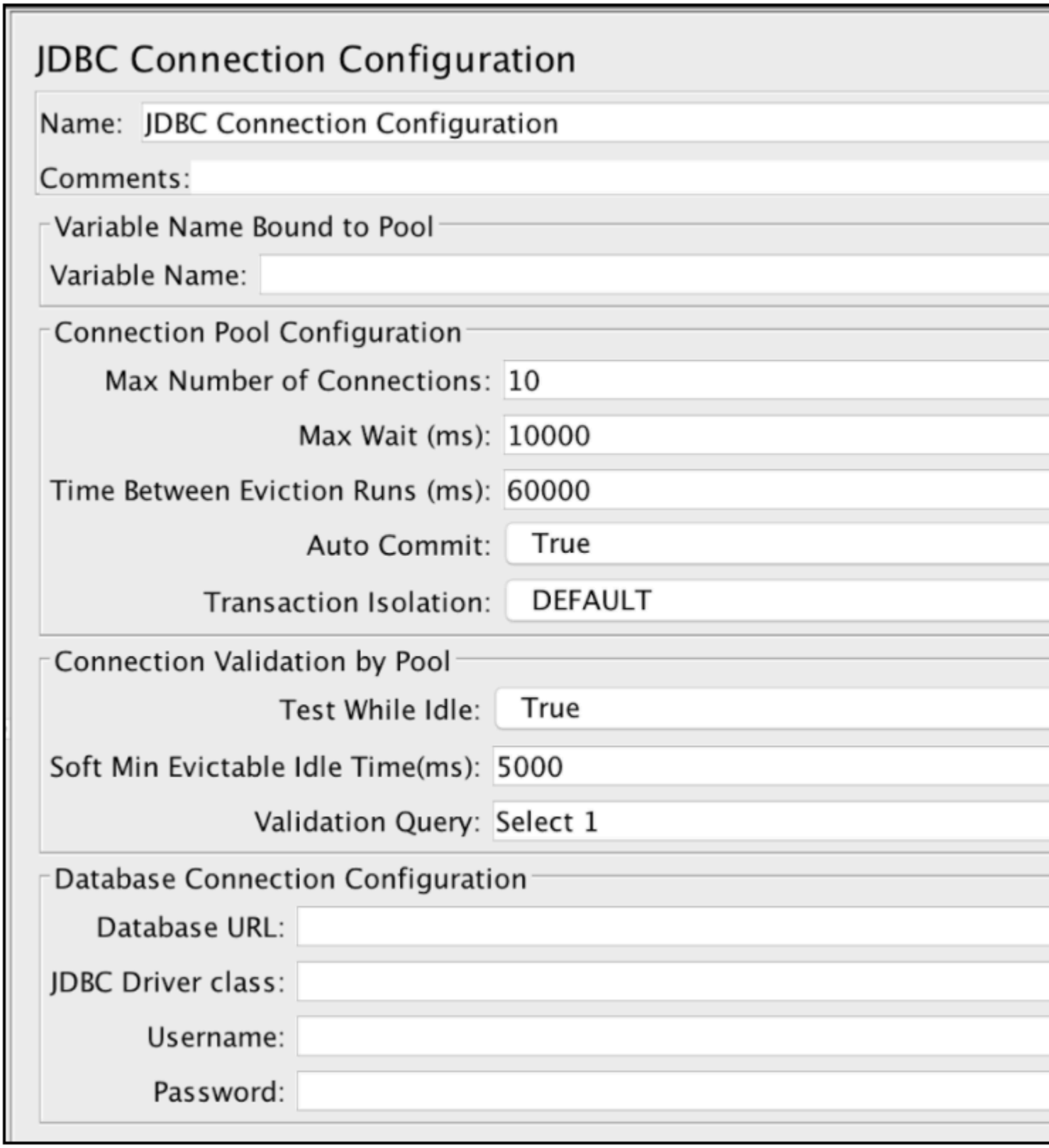
Another way to parametrize your performance scripts is to use database data through JDBC. JDBC is an application programming interface that defines how a client can access a database.

First of all, download the JDBC driver to your database (refer to the database vendor). For example, mysql driver can be found here. Then, you can add it by adding the .jar file to the test-plan by using the form below:



But it is better to add the Jar file in lib folder and restart JMeter.

After that, configure the database connection by using the 'JDBC Connection Configuration' element. Like this: *Right click on Thread Group -> Add -> Config Element -> JDBC Connection Configuration*



'JDBC Connection Configuration' parameters:

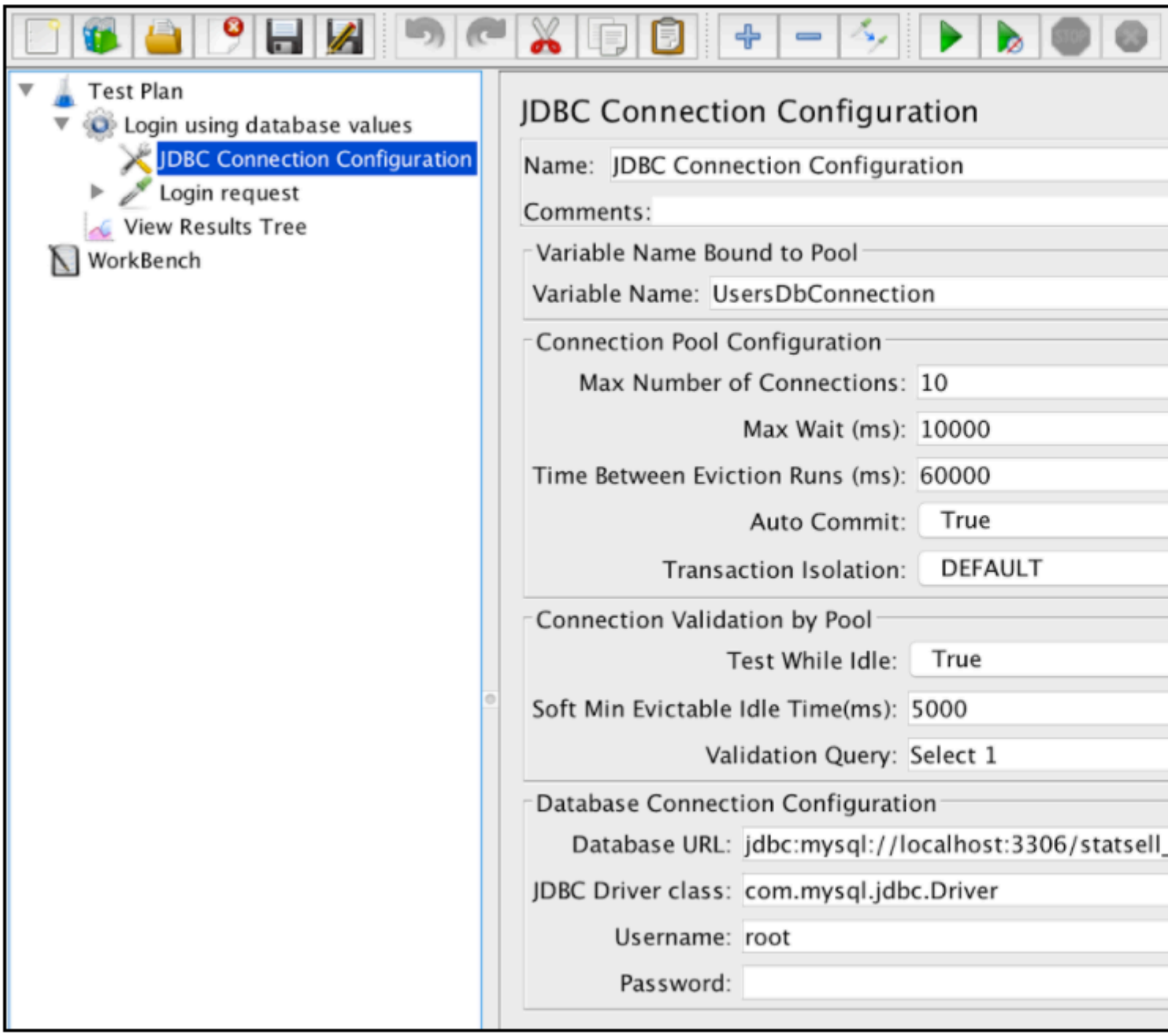
- Name - name of the connection configuration that will be shown in the thread group tree
- Variable Name - name that will be used as unique identifier for the db connection (multiple connections can be used and each one will be tied to a different name)
- Max Number of Connections - maximum number of connections allowed in the connection pool. In case of 0, each thread will get its own pool with a single connection in it

- Max Wait (ms) - pool throws an error if the specified timeout is exceeded during db connection
- Time Between Eviction Runs (ms) - number of milliseconds to pause between runs of the thread that evicts unused connections from the db pool
- Auto Commit - yes to turn auto commit for related db connections
- Test While Idle - check idle connections before an effective request is detected. More details: BasicDataSource.html#getTestWhileIdle
- Soft Min Evictable Idle Time(ms) - period of time during which a specified connection might be idle in the db pool before it can be evicted. More details: BasicDataSource.html#getSoftMinEvictableIdleTimeMillis
- Validation Query - healthcheck query that will be used to verify if the database is still responding
- Database URL - JDBC connection string for the database. See here for examples
- JDBC Driver class - appropriate name of driver class (specific for each db). For example, ‘com.mysql.jdbc.Driver’ for MySql db
- Username - database username
- Password - database password (will be stored unencrypted in the test plan)

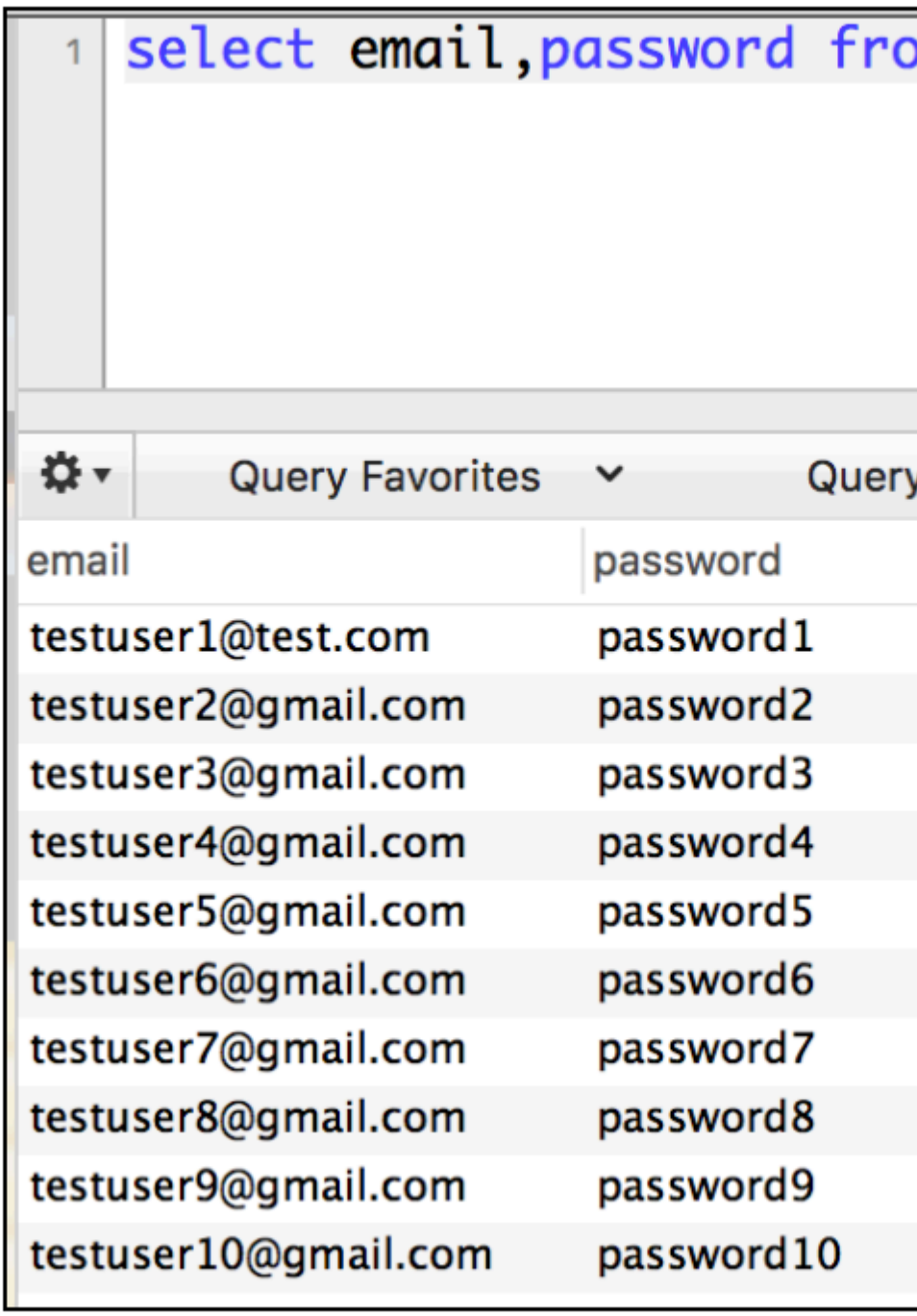
In our case we need to setup the mandatory fields only:

- Variable name Bound to Pool.
- Database URL
- JDBC Driver class
- Username
- Password

The rest of the fields in the screen can be left as defaults:

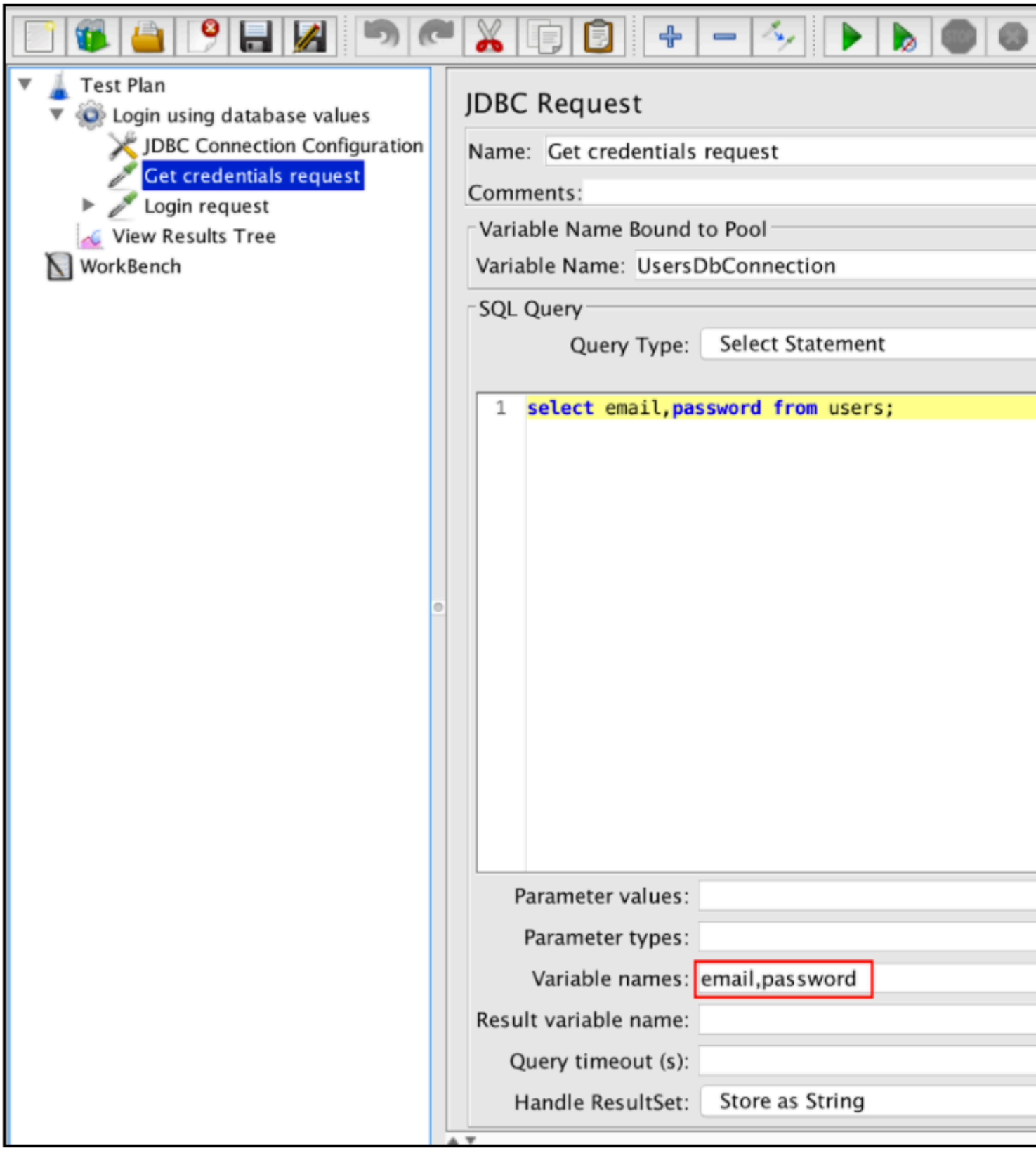


Let’s assume that we store test user credentials in the database:



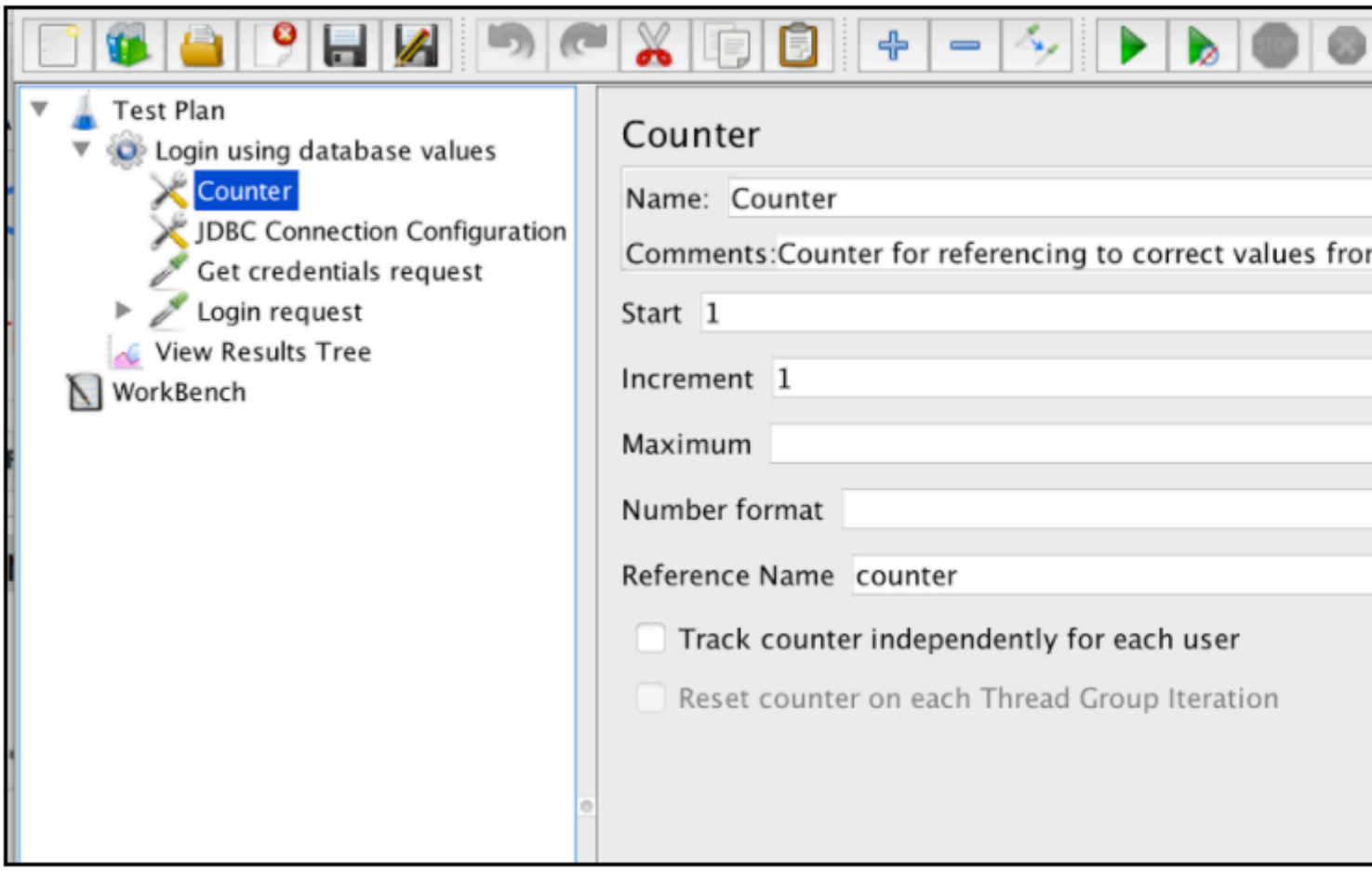
Now when the database connection is configured, we can add the JDBC request itself and use its query to get all the credentials from database: *Right click on Thread Group -> Add -> Sample -> JDBC Request*

By using the ‘Select Statement’ query and ‘Variable Names’ we can parse the response to custom variables.

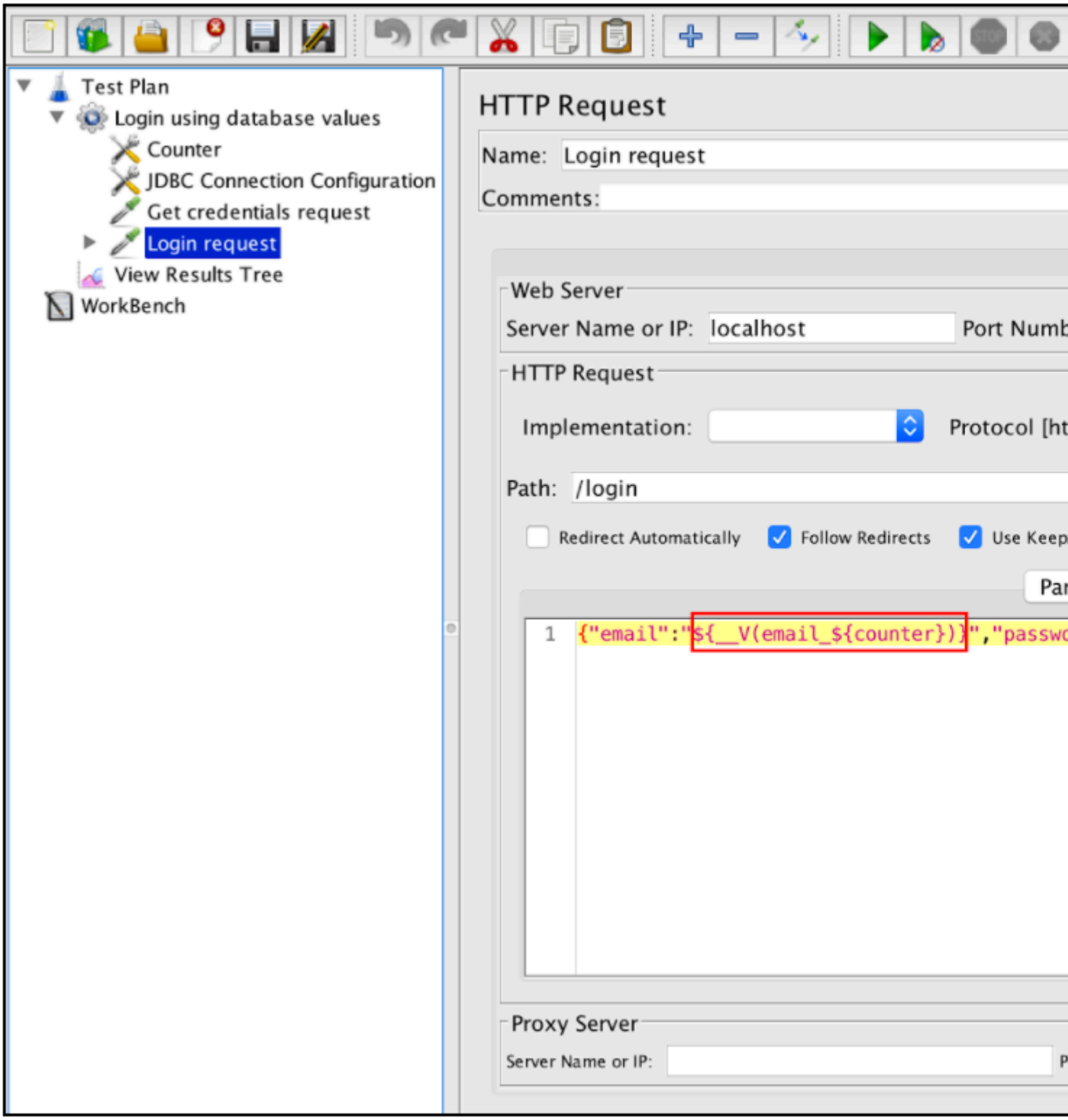


We will now have JMeter variables that can be used further in subsequent requests. Specified variables will be created with incremental suffix (email_1, email_2, email_3…..).

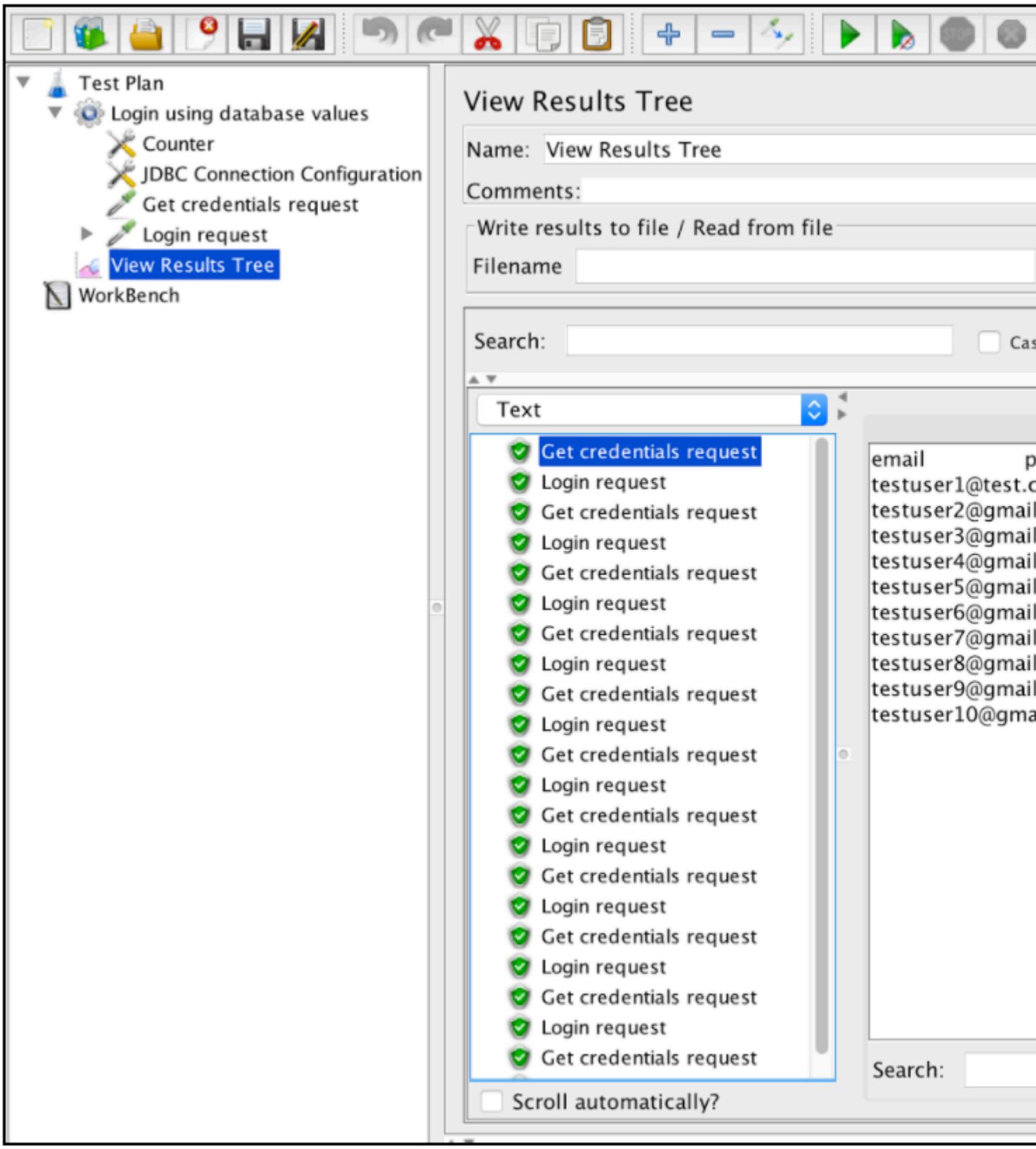
To use these variables in the 'Login request', we need to add a counter that will be used to access the right values from the JDBC query response. To add the 'Counter' element in JMeter: *Right click on Thread Group -> Add -> Config Element -> Counter*

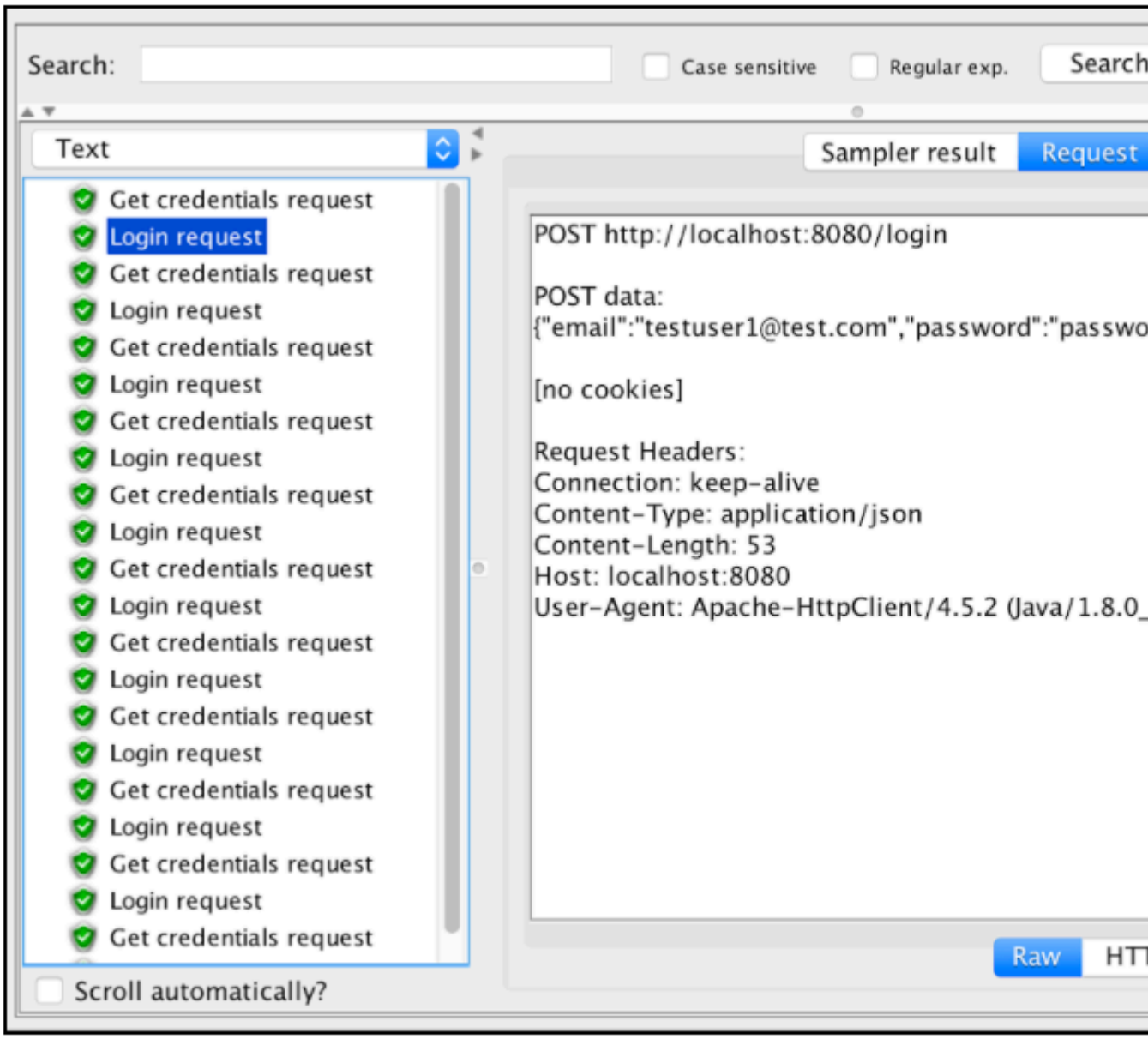


After that, we can update the 'Login request' using the __V function. This returns the result of evaluating a variable name expression and can be used to evaluate nested variable references:



The specified configuration is enough to use database values to run the script across different users:


Test Plan
Login using database values
Counter
JDBC Connection Configuration
Get credentials request
Login request
View Results Tree
WorkBench
View Results Tree
Name: View Results Tree
Comments:
Write results to file / Read from file
Filename
Search:
Text
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Login request
Get credentials request
Scroll automatically?
email
testuser1@test.c
testuser2@gmail
testuser3@gmail
testuser4@gmail
testuser5@gmail
testuser6@gmail
testuser7@gmail
testuser8@gmail
testuser9@gmail
testuser10@gma
Search:

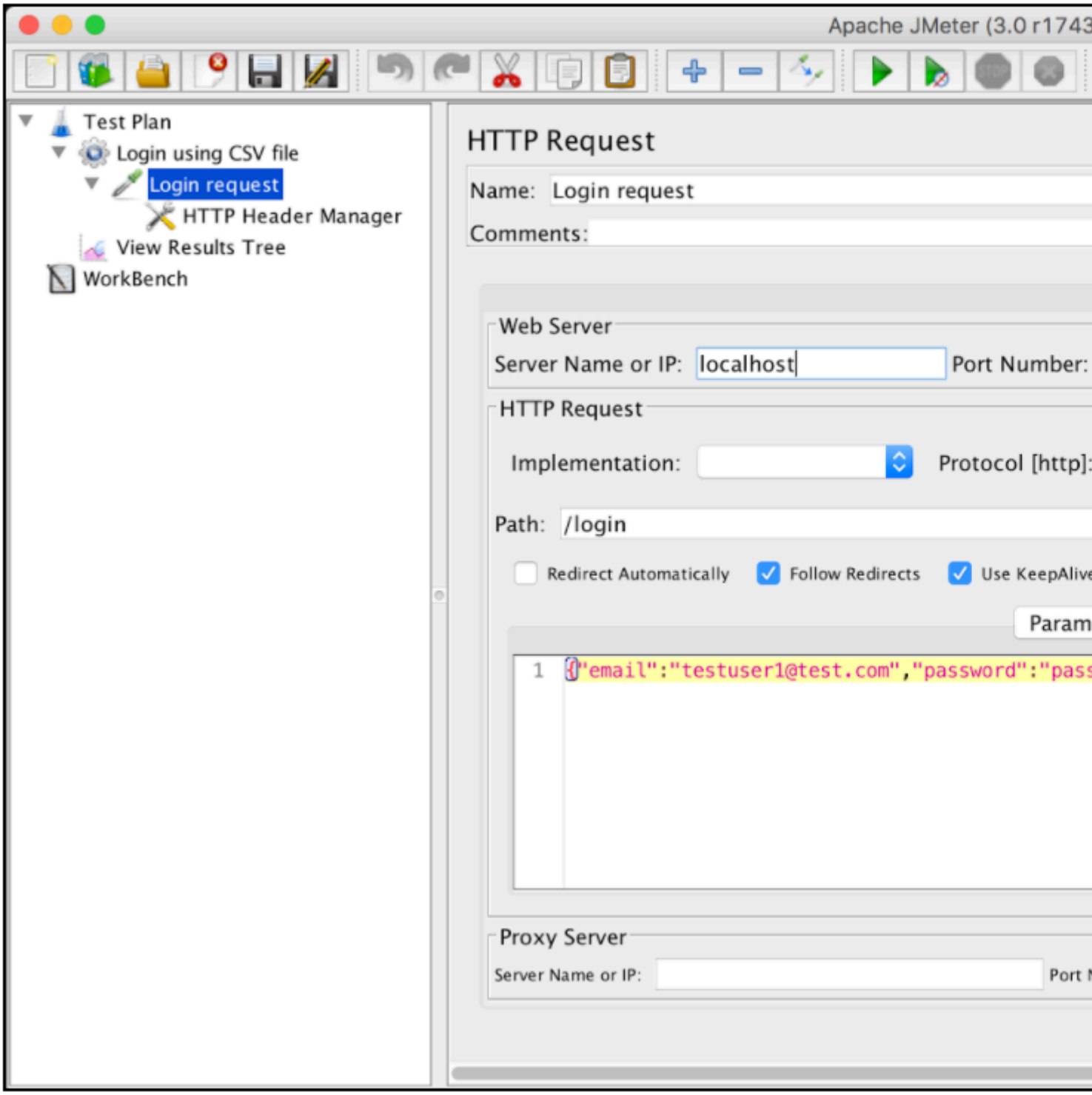


Parameterization using the 'Parameterized Controller' plugin

If you need to execute a repeating sequence of the same action with different parameters, use the 'Parameterized Controller' 3rd party plugin from JMeter-Plugins project.

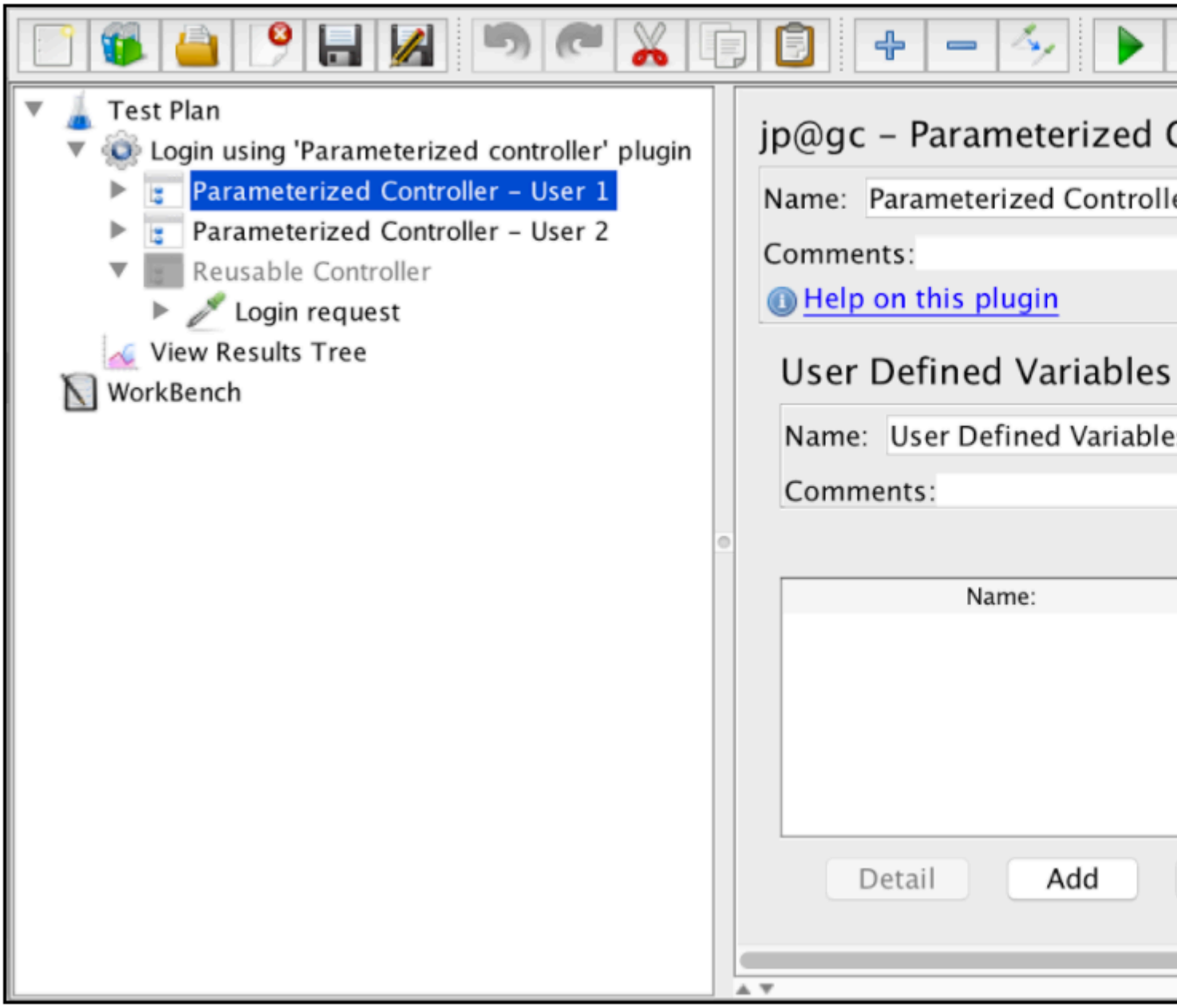
You need to install this plugin first by following installation procedure.

Let's assume that we want to parameterize the login workflow:

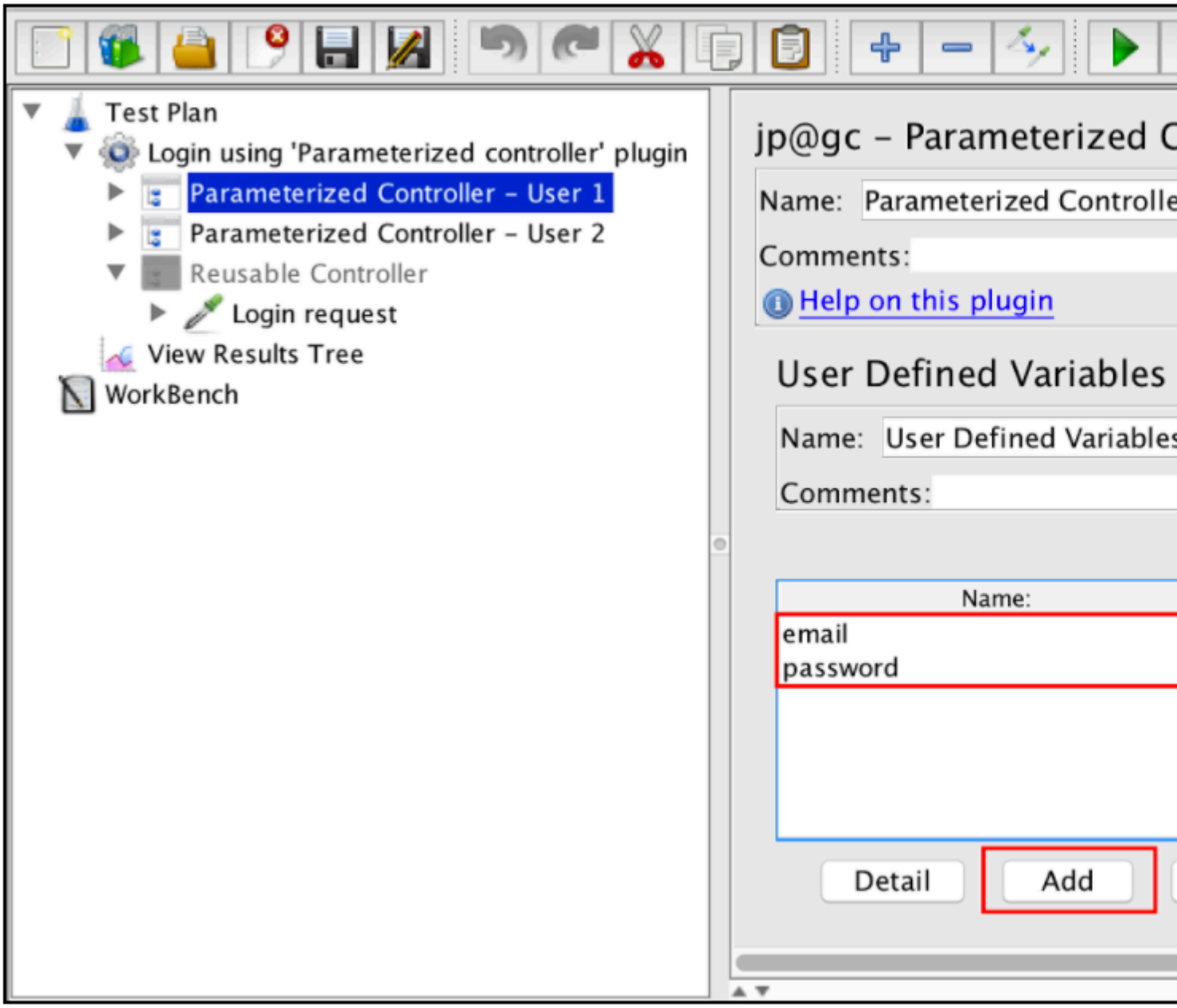


First of all, you need to install the ‘Parameterized Controller’ plugin as it is not included into JMeter core. Installation steps of that process can be found here.

Let’s move the ‘Login Request’ into a separate controller and disable it (right-click it and select "Disable"). This is the most preferable way to have a modules container inside your test plan and avoid using Workbench as such a container. After the installation is over, you can add two ‘Parameterized Controller’ controllers with different user credentials: *Right click on Thread Group -> Add -> Logic Controller -> Parameterized Controller*

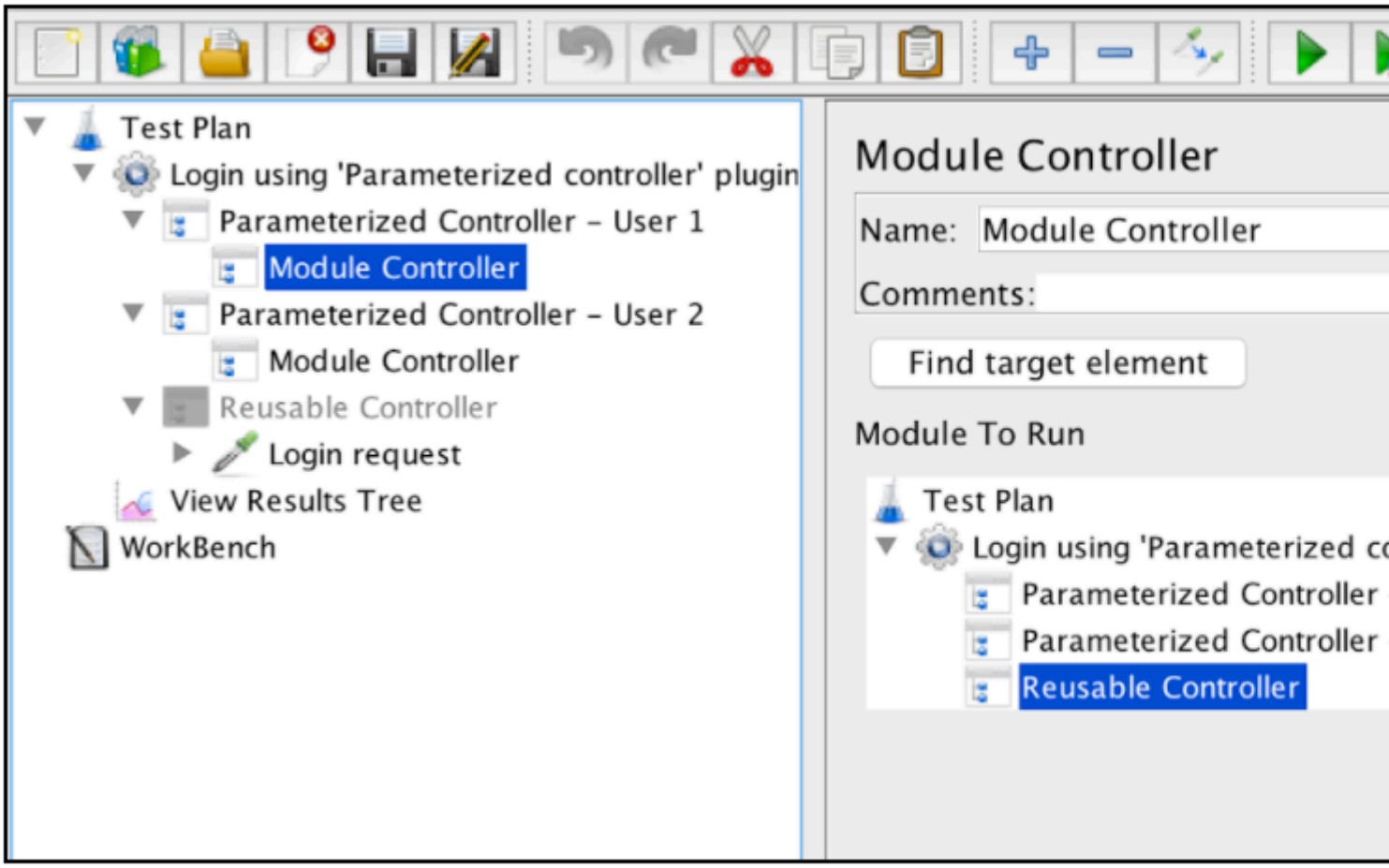


Parameterized Controllers contains the ‘User Defined Variables’ section, where you can specify your parameters. Put the credentials of the first user in the first parameterized controller and the second user credentials in the second parameterized controller.

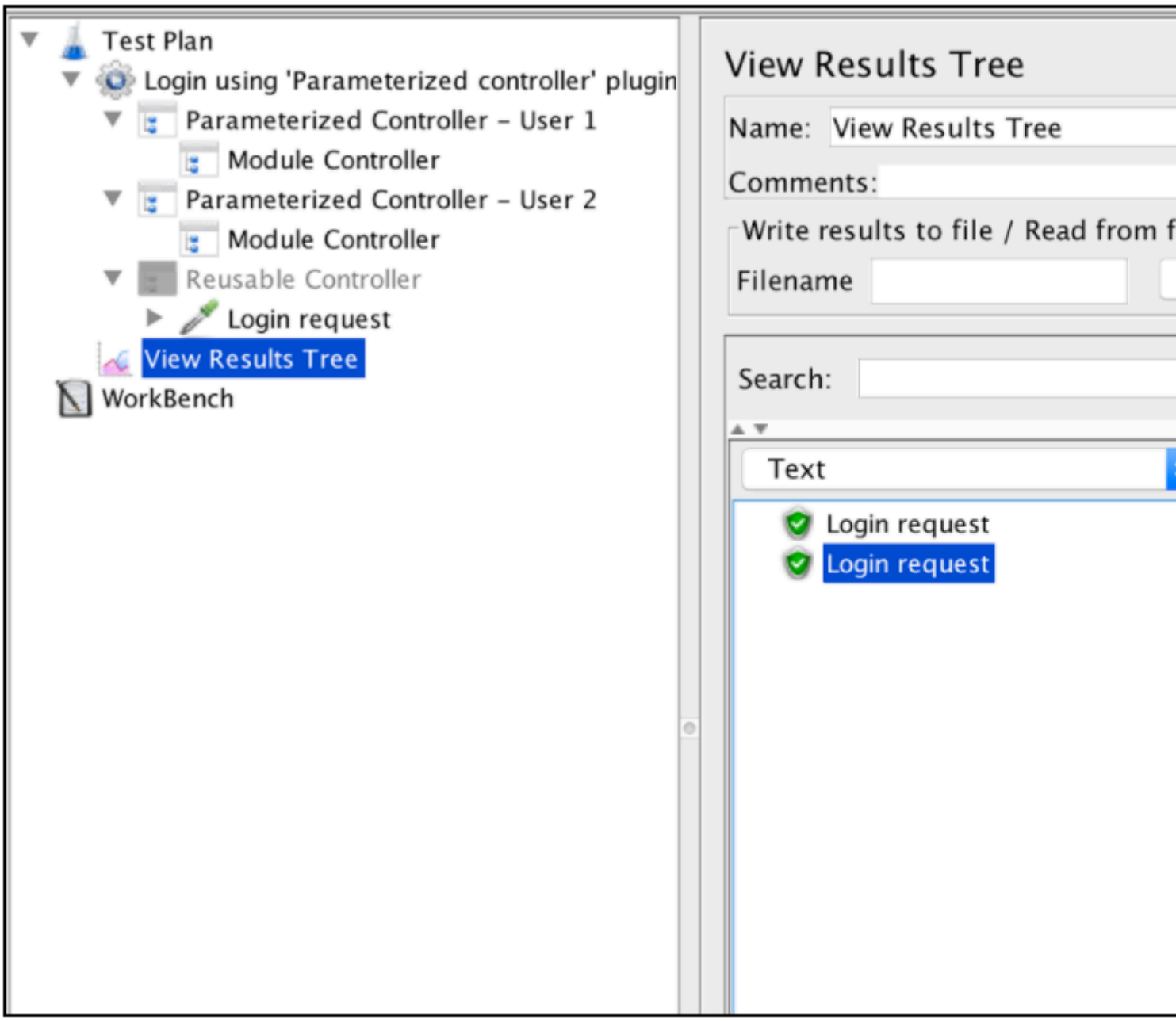


Inside both parameterized controllers, add references to the ‘Reusable Controller’ to call the ‘Login request’ with different parameters. It can be done this way:

Right click on ‘Parameterized Controller’ -> ‘Add’ -> ‘Logic Controller’ -> ‘Module Controller’



When running your script, you will see that the ‘Login request’ triggered each of the parameterized controllers separately. It can be very useful in case you need to run your script across different combinations of input parameters.



Read Apache JMeter parameterization online: https://riptutorial.com/jmeter/topic/9602/apache-jmeter-parameterization

Chapter 4: Apache JMeter: Test scenario recording

Introduction

Recording test scenarios is one of the most convenient ways to create test scripts. This is because test recordings let you mimic realistic user workflows, instead of having to manually create a test script. The recordings capture all browser requests to the web application, and then automatically create a jmx file that can be run in performance tests. By using JMeter's recording/playback functionality or 3rd party tools like BlazeMeter and BadBoy, testers can make their work 3 times faster.

Examples

Script Recording with the JMeter Template Feature

In version 2.10, JMeter introduced a mechanism that saves you time when creating scripts - JMeter Templates. These templates are skeletons that can be reused as the base for your new scripts.

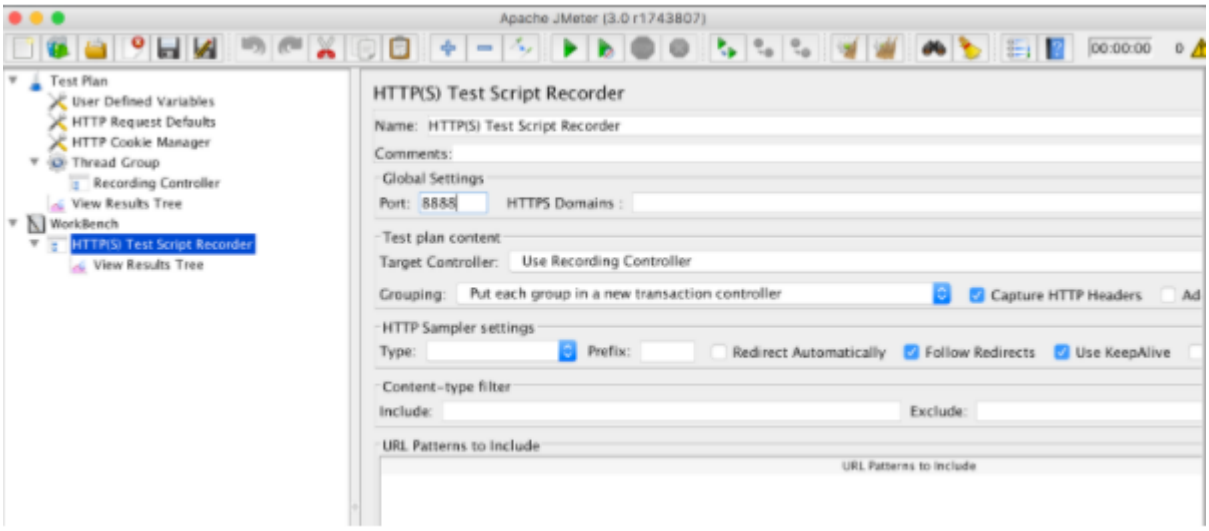
JMeter already has several available templates with detailed descriptions, and you can also add your own. Templates have all the required configuration and elements to record your performance scripts from scratch.

Here's how to use the JMeter Template Feature:

Configure JMeter

1. Open JMeter

2. Select the template for script recording:
   File -> Templates… -> Select Template -> Recording -> Create JMeter will add the relevant elements to the test tree.



Configure your browser proxy

To use the JMeter Recorder, you need to configure your browser to send all the requests via proxy. Any browser can be used for these needs, although there might be differences between the locations of the browsers' configurations, which are browser-specific and might vary according to OS.

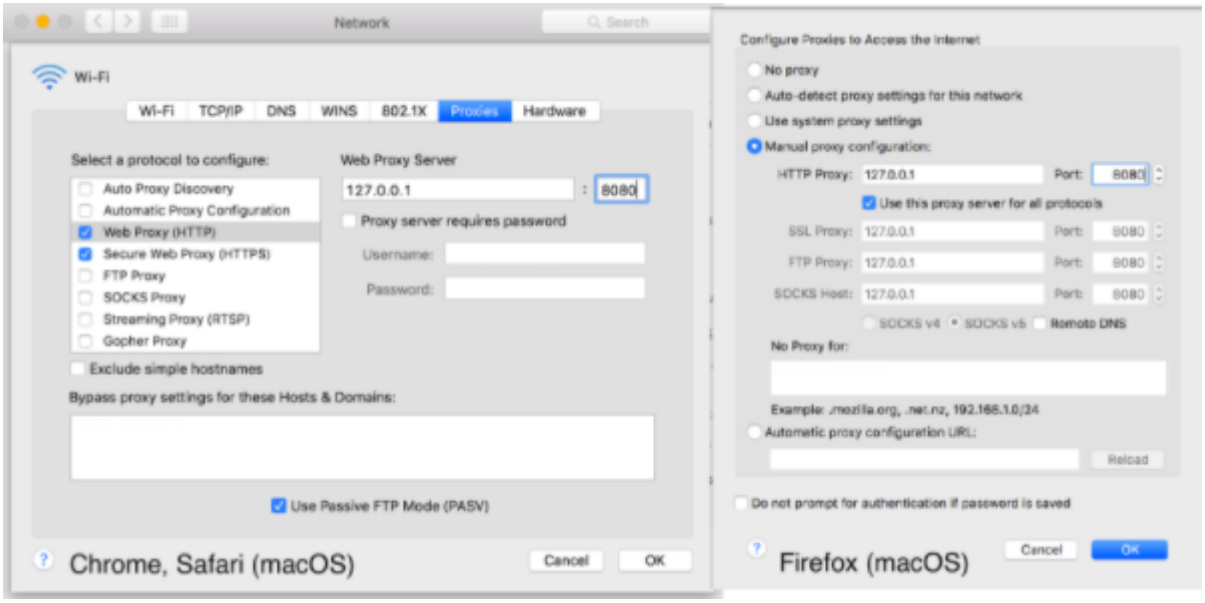
3. To configure your browser:

   Chrome: Menu button -> Settings -> Show advanced settings… -> Network -> Change proxy settings

   Safari: Preferences -> Advanced -> Proxies -> Change Settings...

   Firefox: Menu button -> Preferences -> Advanced -> Network -> Connection -> Settings..

4. For example, you can use the localhost 127.0.0.1. Change the port to the port in the HTTP(S) Script Recorder.



If you have issues while completing this stage, check that you don't have any 3rd party plugins

that can manage your browser's proxy settings. If you do, like Hola VPN for example, the proxy settings will be unavailable in your browser's menu.

5. Click on the 'Start' button, which is at the bottom of the "HTTP(S) Test Script Recorder page, and go through the web application workflow you want to test. When you go back to JMeter, you should see all the captured requests from your browser.

Script Recording with the JMeter Proxy Recorder

JMeter also enables you to manually configure your workspace. It's more complex, but you can make the scripts fit your exact needs.

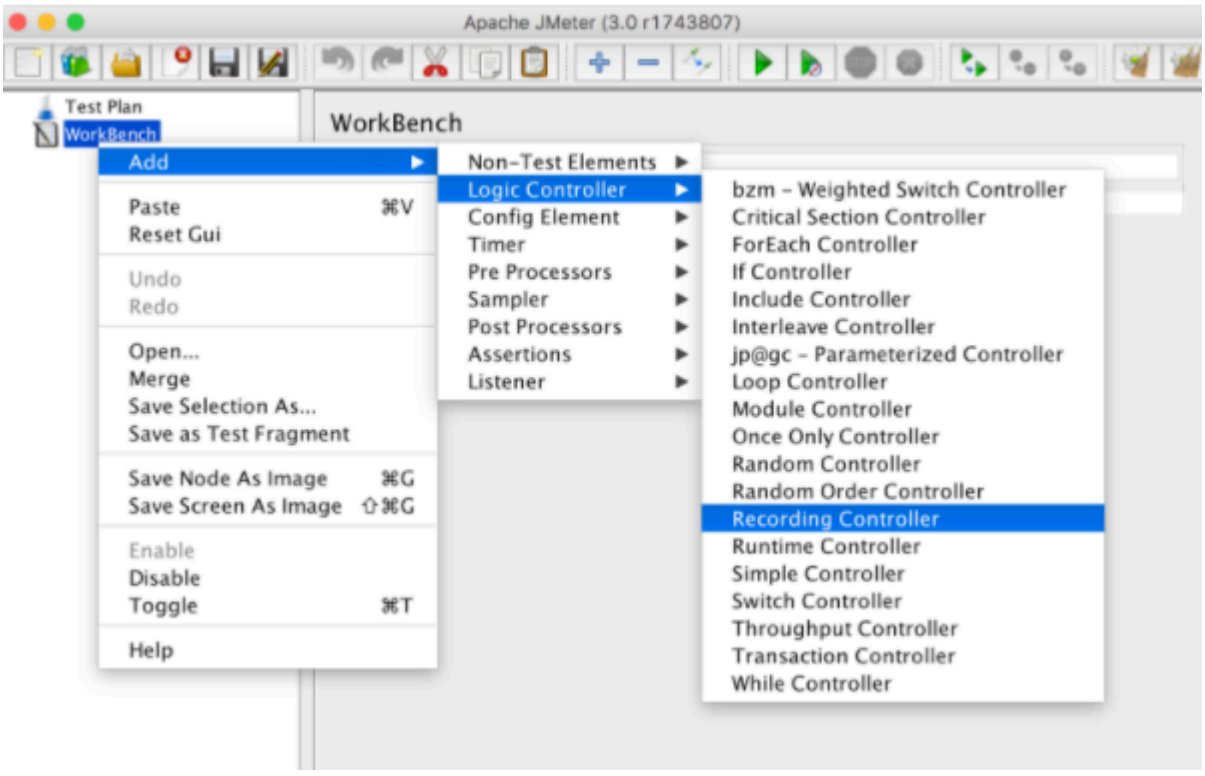
Configure your browser proxy

1. Configure your browser, as described in chapter 1.

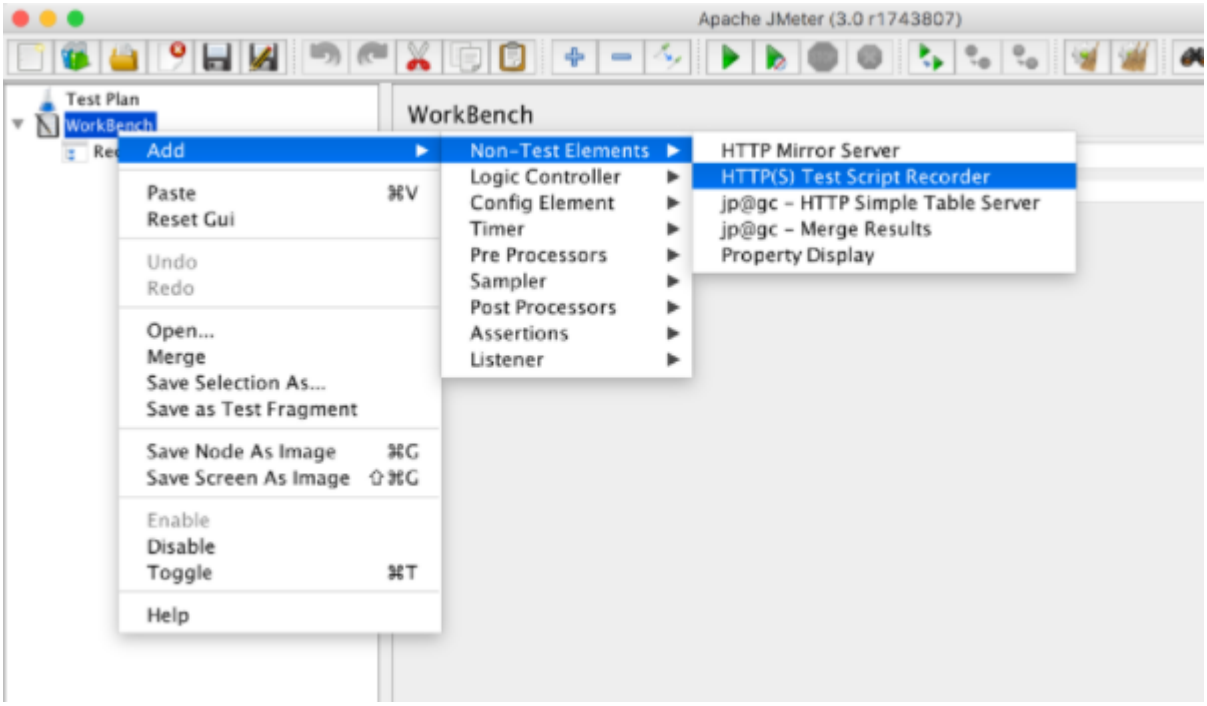
Configure JMeter

The "WorkBench" branch can be used as a temporary workspace for creating scripts. Bear in mind that entries added to this section will not be saved with the test plan. Therefore, if you want to reuse the same recording configuration in the future, you will need to copy and paste it to the "Test Plan" section.

2. Add "Recording Controller" to "WorkBench": Right click on "WorkBench" -> "Add" -> "Logic Controller" -> "Recording Controller"



3. Add "HTTP(S) Test Script Recorder" to the same "WorkBench": Right click on "WorkBench" -> "Add" -> "Non-Test Elements" -> "HTTP(S) Test Script Recorder"

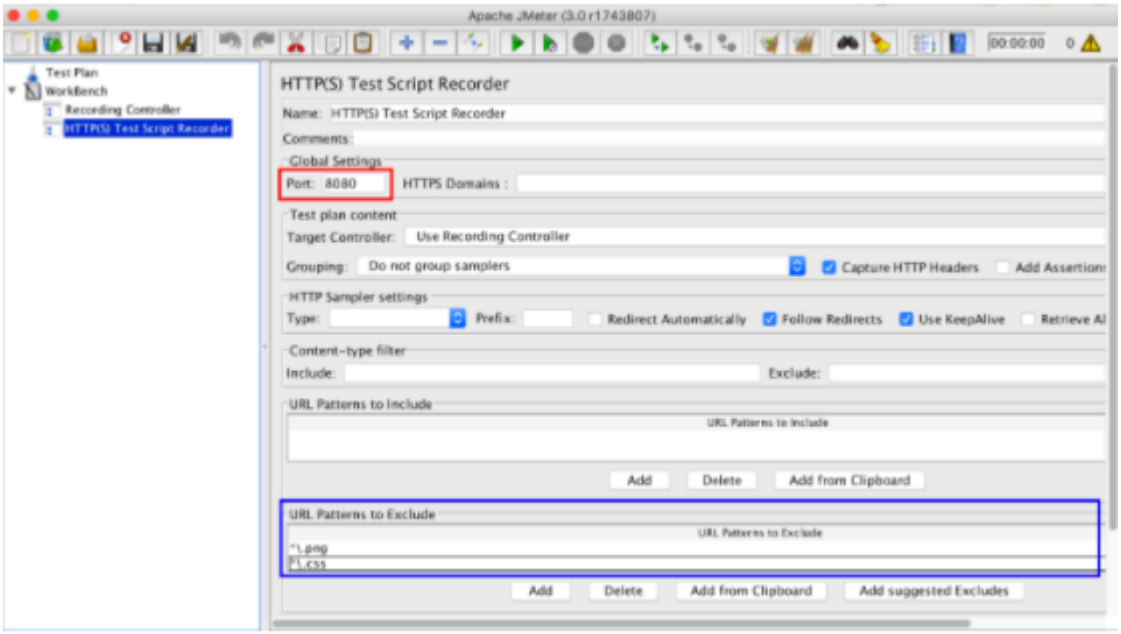


4. On the “HTTP(S) Test Script Recorder” configuration page in “Global Settings: Port”, you need to put the same port that is specified into your browser’s proxy configuration, for example 8080.

5. If you want to exclude requests to specific resources, you can use the “URL Patterns to Exclude” section. This can be useful if you want to include only the types of content you want to request (e.g. *.html, *.php, etc) or to exclude the types of content you do not want to request (e.g. *.jpg, *.png, *.js, etc).

When would we use this? For example, when recording a script that calls a third-party application or when testing a server-side script, you might not want to download assets since they might clutter up your tests and consume bandwidth, or you might want to record certain requests from a certain path.

The most common exclude patterns are: “*..png“, “*..jpg“, “*..gif“, “*..css”, “*..js”. You can also combine different patterns together. This combined pattern should get rid of all redundant requests that might distract you from important ones: “*..(bmp|css|js|gif|ico|jpe?g|png|swf|woff)”

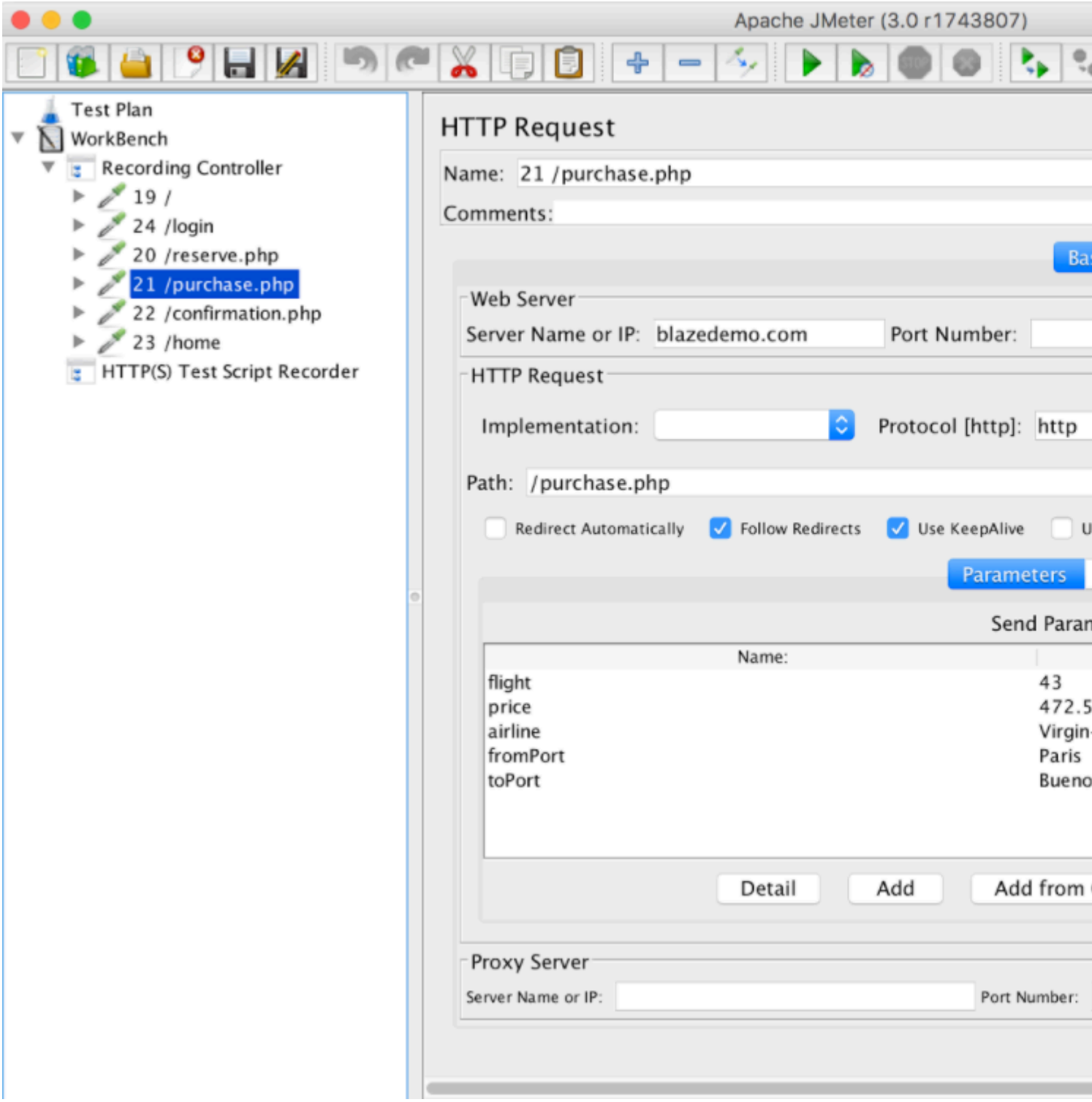


6. On the other hand, you might want to mimic the complete browser behavior and include the loading of all resources. In this case, it's not necessary to exclude URL patterns. Keep in mind through that the browser downloads all embedded resources from the requested page and that it has a caching mechanism, which might affect result performance.

In this case, it is recommended to download all the embedded resources in the script: Right click on "Test Plan" -> "Add" -> "Config Element" -> "HTTP Request Defaults" -> "Advanced" -> select "Retrieve All Embedded Resources" checkbox.

7. To make JMeter behave more like a real browser it is recommended to add the "HTTP Cache Manager", which lets you simulate browser caching functionality in your performance tests. Right click on "Test Plan" -> "Add" -> "Config Element" -> "HTTP Cache Manager".

8. Now click on the 'Start' button, which is at the bottom of the "HTTP(S) Test Script Recorder" page, and go through the web application workflow you want to test. When you go back to JMeter, you should see all the captured requests from your browser under the "Recording Controller".



Recording Performance Scripts for Mobile Devices

JMeter can also can be used for recording mobile performance testing. Mobile scripts recording is very similar to web application scripts recording.

Configure JMeter

1. Configure “JMeter Templates” as specified in chapter 1.

Configure your mobile phone

After the JMeter configuration is prepared, including the JMeter “HTTP(S) Test Script Recording” element started on a specified port, you can configure your mobile phone to send a request to the web application you are testing via the JMeter proxy.

2. **IOS**:

- Setting -> Wi-Fi
- Click on the connected network
- Go to the “HTTP PROXY” configuration section
- Click on the “Manual” tab
- Set the IP of the computer JMeter application is running on under “Server”
- Set the port that is specified on the “HTTP(S) Test Script Recording” under “Port”

Android:

- Setting -> Wi-Fi
- Long click on the connected network and click the ‘Modify Network’ option
- Click on the “Advanced options” checkbox
- Set the “Proxy” option to “Manual”
- Set the “Proxy hostname” as your computer’s IP address and “Proxy Port” as specified on the “HTTP(S) Test Script Recording” configuration under “Port”
- Click “Save”

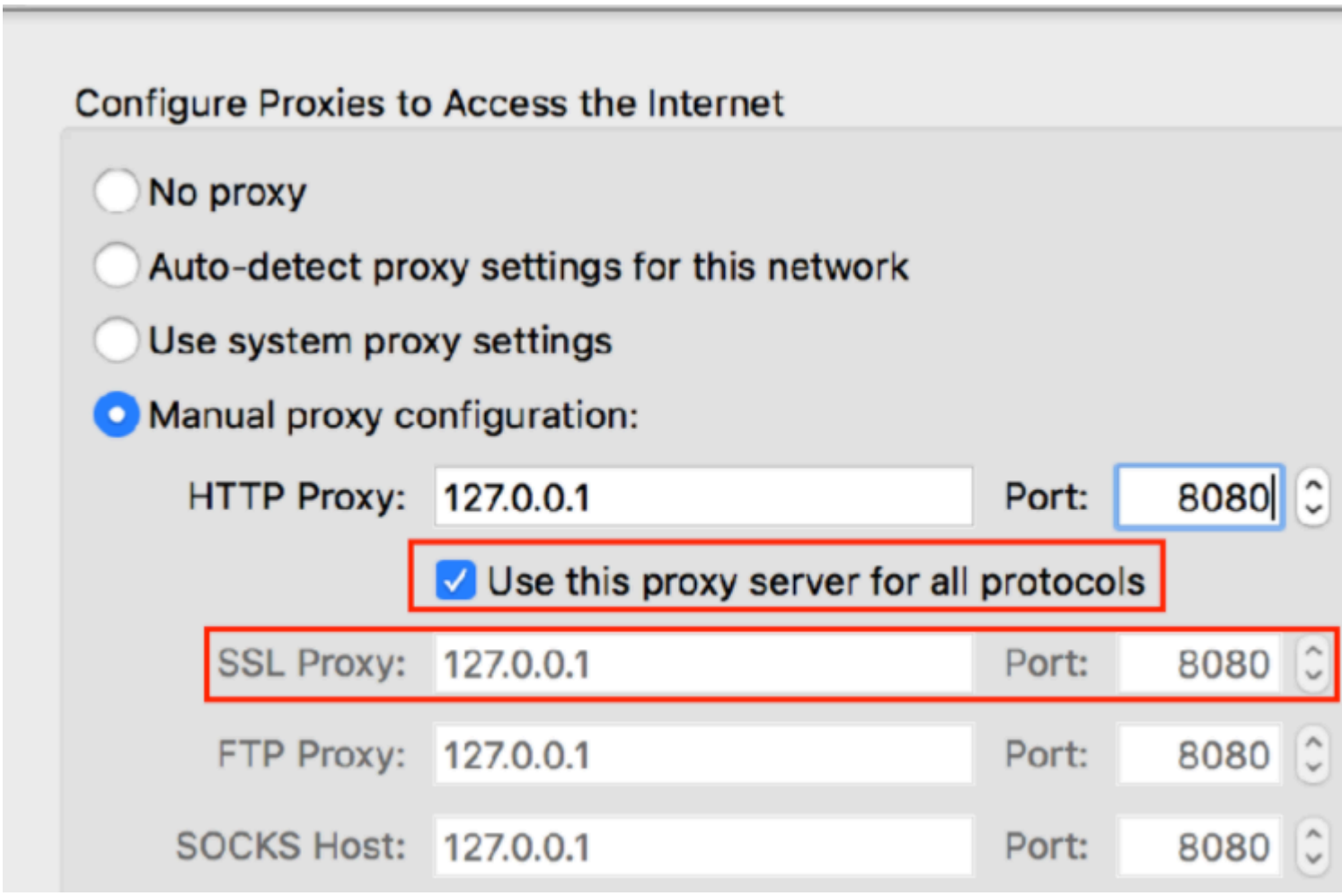
3. You can now start running the application on your mobile device. The requests will be recorded on JMeter.

Recording HTTPS Traffic

If your web application uses the SSL encryption, you need to capture HTTPS traffic instead of HTTP. To record HTTPS Traffic with JMeter, you need to configure the SSL certificates.

Configure your SSL proxy

1. Make sure the SSL proxy is configured the same way the HTTP proxy is configured:

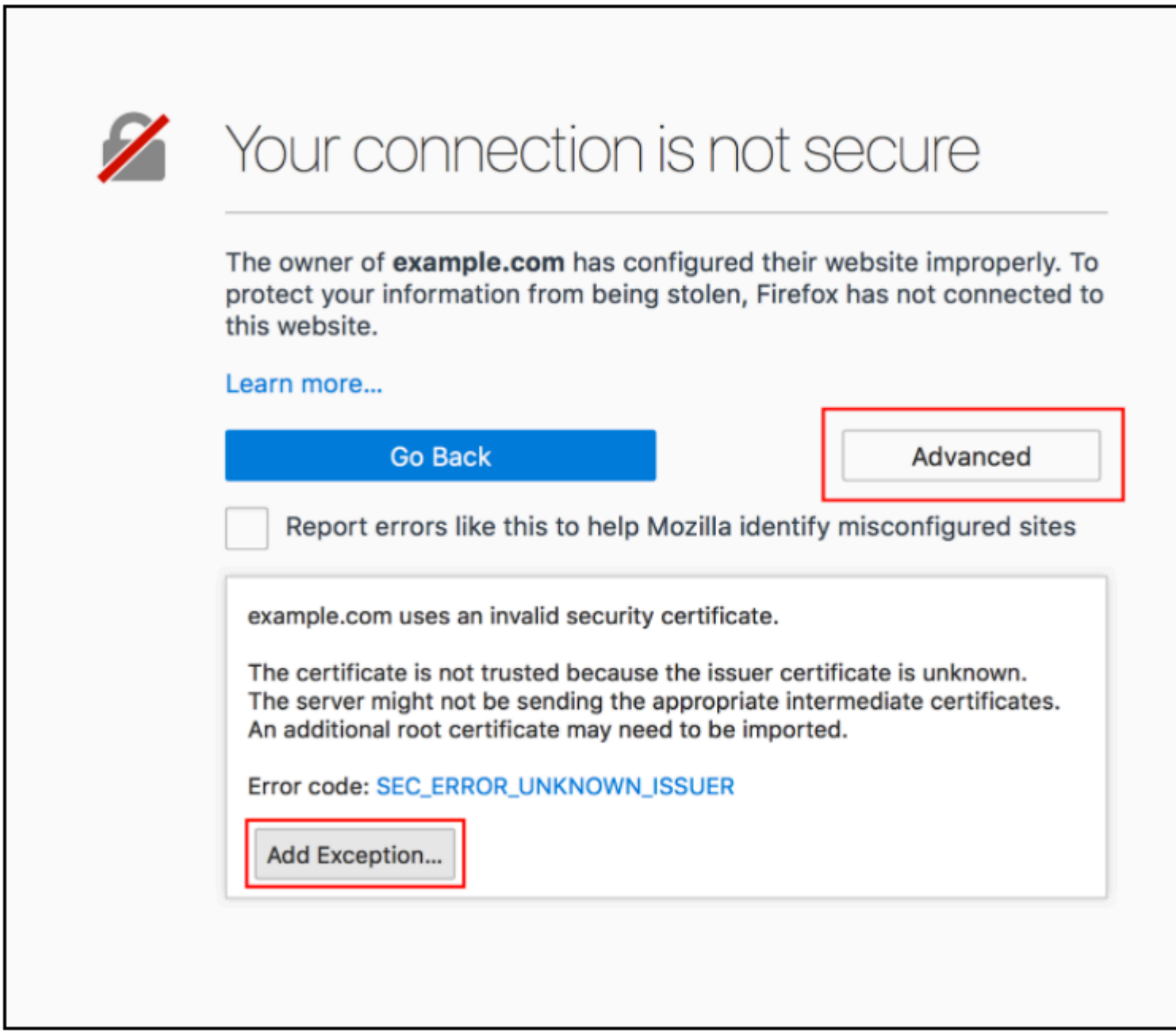


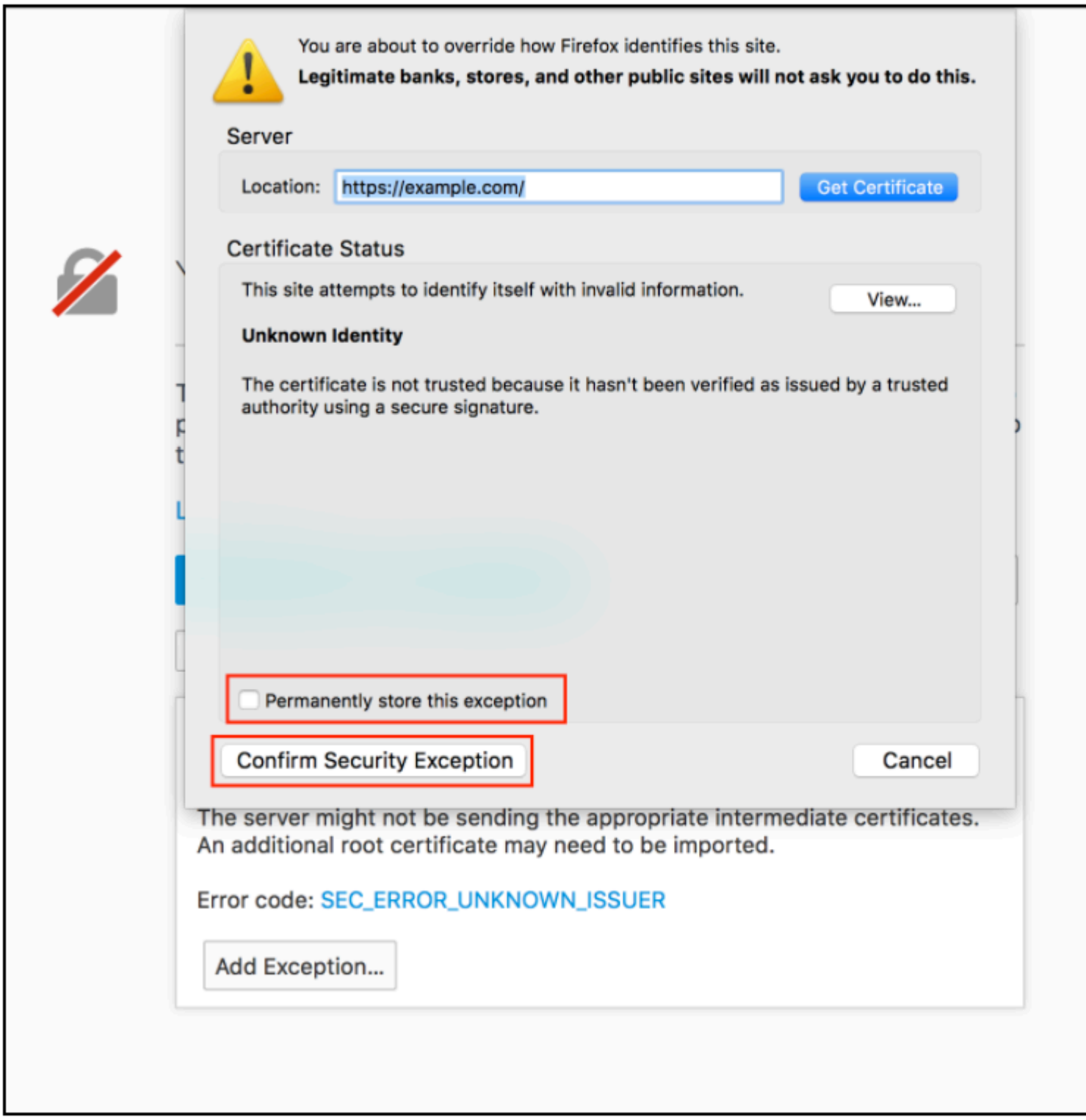
Configure JMeter

2. Start script recording by using the “JMeter Recording Template” feature as explained in example "Script Recording with the JMeter Template Feature".

3. After opening the web application, you will see a message regarding an unsecure connection. To proceed, you just need to accept the JMeter dummy certificate:

- Click on ‘Advanced’
- Click on ‘Add Exception...’
- Uncheck ‘Permanently store this exception’
- Click on ‘Confirm security exception’


Your connection is not secure
The owner of example.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.
Learn more...
Go Back
Advanced
Report errors like this to help Mozilla identify misconfigured sites
example.com uses an invalid security certificate.
The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.
Error code: SEC_ERROR_UNKNOWN_ISSUER
Add Exception...



4. If you see the “This site provides valid, verified identification. There is no need to add an exception.” warning message, you need to clear the browser history for your application, including cookies, cache, offline website data. Then, proceed with the same steps again.

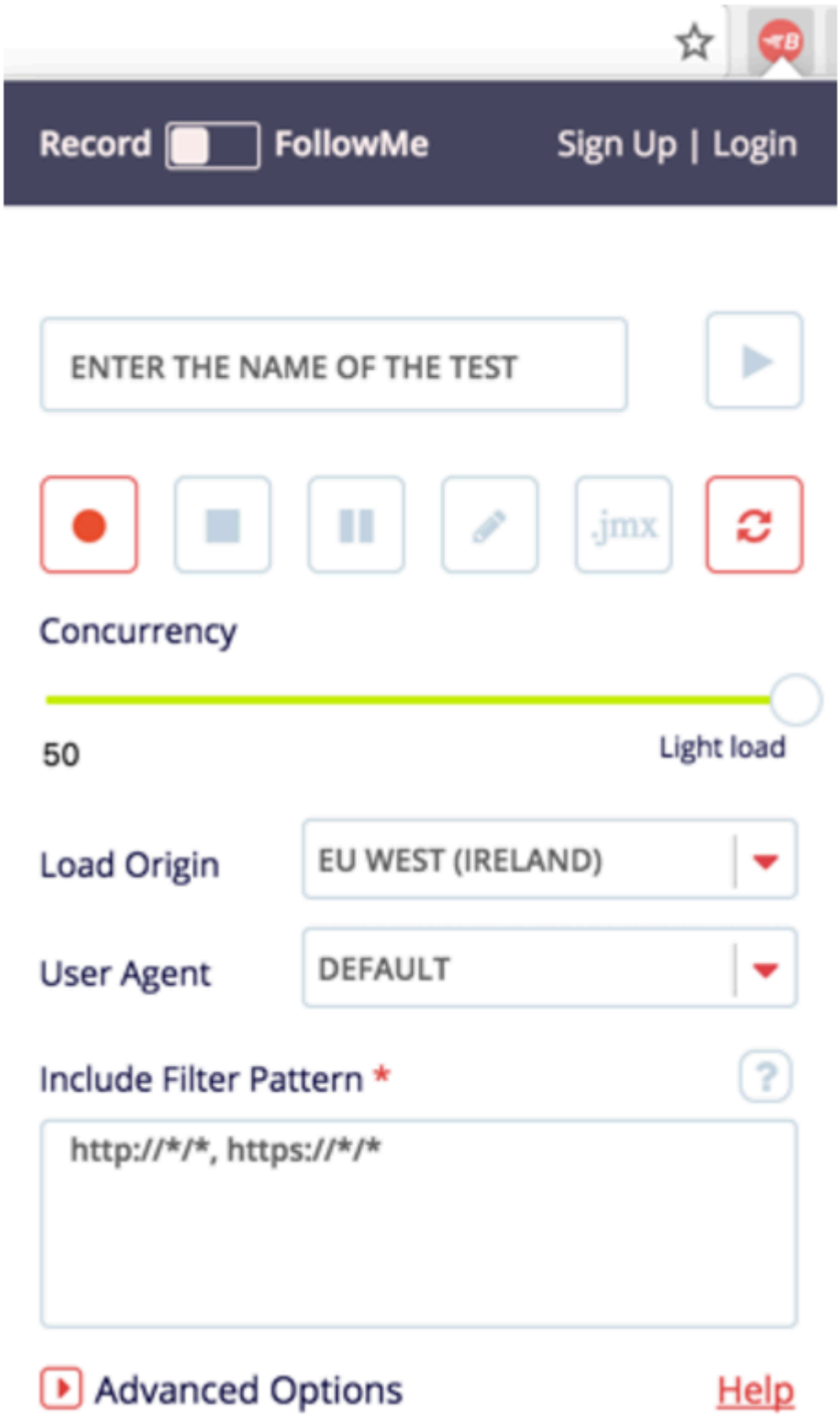
This approach also works for mobile scripts recording, since the JMeter certificate needs to be installed only on the host which is being used to run JMeter.

Script Recording with the BlazeMeter Chrome Extension

So far we’ve covered the basic ways to record test scenarios. But one of the fastest and easiest

ways to record your performance scripts, which is also free, is to use the BlazeMeter Recorder Chrome extension. These recordings can be run in JMeter or in BlazeMeter.

The reason the extension is so useful, is that it lets you record performance scripts from your browser without having to configure your proxy.



To create a new performance script:

1. Open the recorder from your Chrome
2. Enter a test name in the top field
3. Start recording by clicking on the record button, in the shape of a circle, and perform the web actions you want to record. All your requests will be captured. The Blazemeter Chrome Extension also supports recording of HTTPS traffic.
4. After you finish recording, click on the stop button, in the shape of a square. You can also pause your recording and then resume, as well as edit it, in .jmx or JSON format, or in the cloud.
5. Export your recording - to run the test in JMeter, export to .jmx format by clicking on the .jmx button. To run the test in BlazeMeter, click ‘play’.

For more information see here.

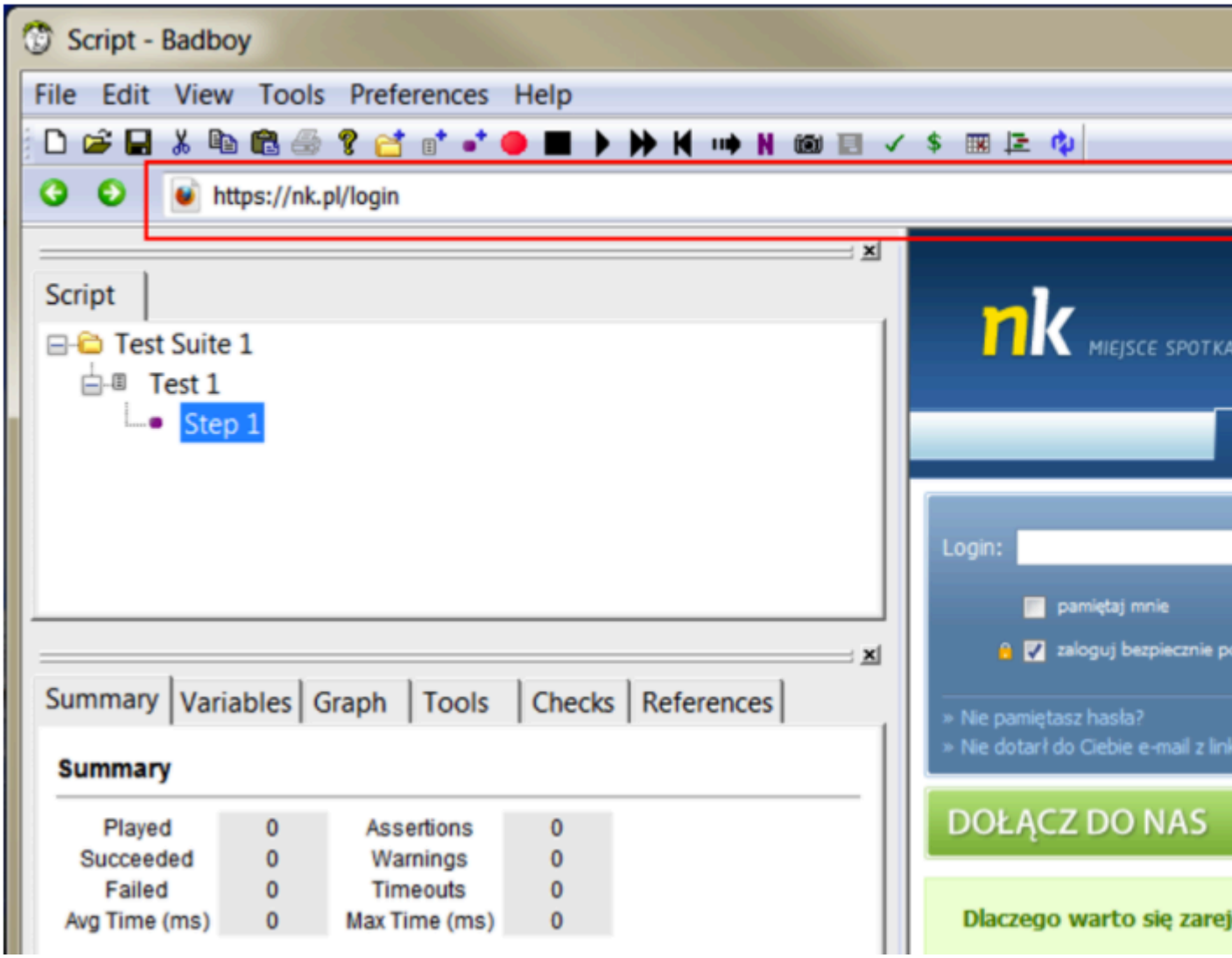
Script Recording with BadBoy

Another useful 3rd party recording tool is BadBoy. However, it works only for Windows OS.

To create a new performance script:

1. Install BadBoy here

2. Enter the URL under test in the address bar



3. Press the record button, shaped like a red circle and perform the actions you want to capture.

4. Export your script to JMeter - File -> Export to JMeter

For more information, see here.

Using a performance scripts recorder is a great way to avoid routine tasks and still get the best test scripts. After recording, configure the test to the number of virtual users you want to test as well as additional test configurations, run your test and analyze the results to identify errors and bottlenecks and characterize trends that show you the health of your system.

Read Apache JMeter: Test scenario recording online:
https://riptutorial.com/jmeter/topic/8798/apache-jmeter--test-scenario-recording

Credits

S. No	Chapters	Contributors
1	Getting started with Apache JMeter	Aliaksandr Belik, Chulbul Pandey, Community, Kiril S., M Navneet Krishna, Milamber, Naveen, NaveenKumar Namachivayam, RowlandB, UBIK LOAD PACK, Venkatesh Achanta
2	Apache JMeter Correlations	UBIK LOAD PACK, Yuri Bushnev
3	Apache JMeter parameterization	UBIK LOAD PACK, Yuri Bushnev
4	Apache JMeter: Test scenario recording	UBIK LOAD PACK, Yuri Bushnev