Jmeter Performance Testing

Winter 2016 – Author: Thao Le

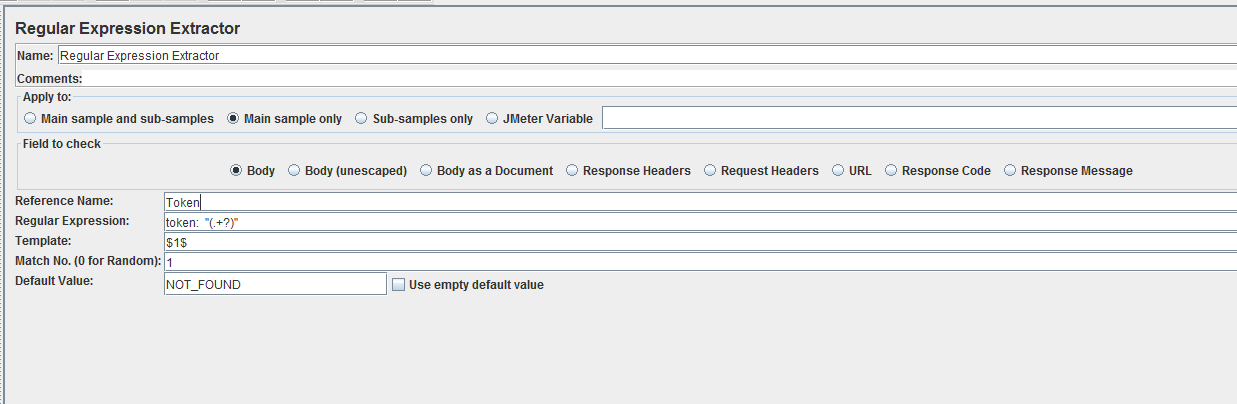
Contents

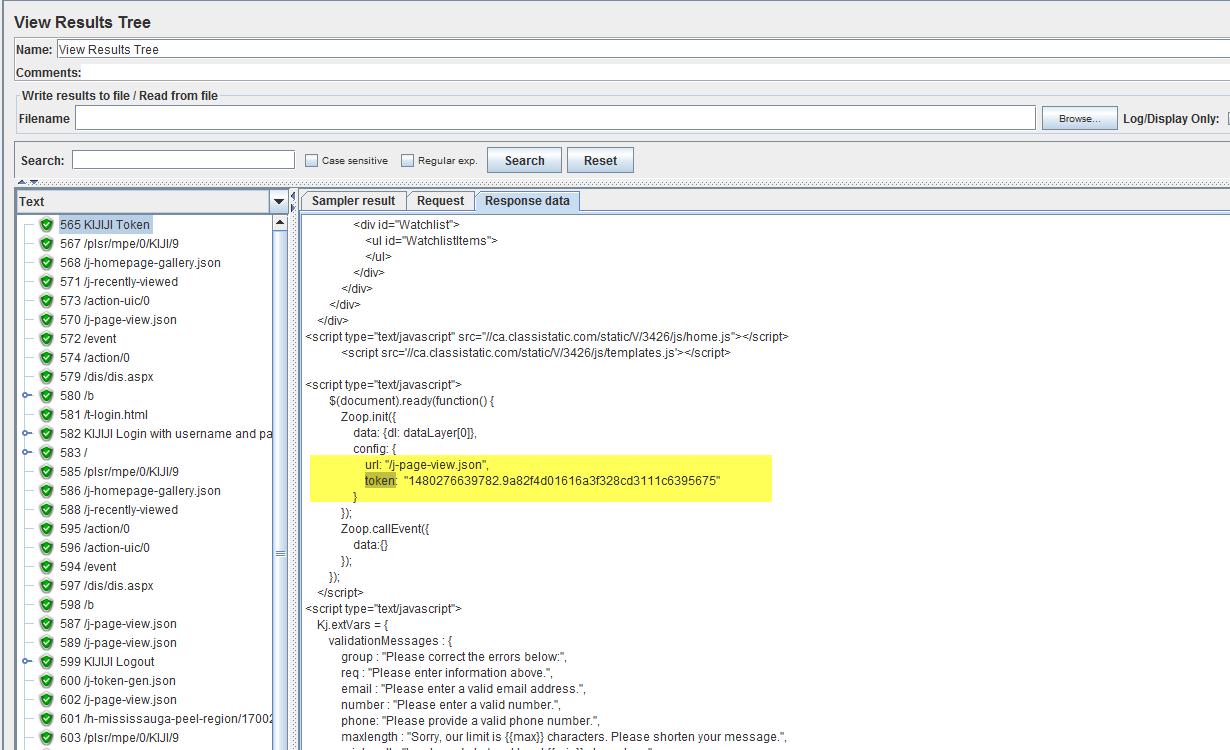
[1. Week 4: Handle Dynamic Token, Assertions, Using CSV Data File 4](#_Toc467782850)

[2. Contact Us 8](#_Toc467782851)

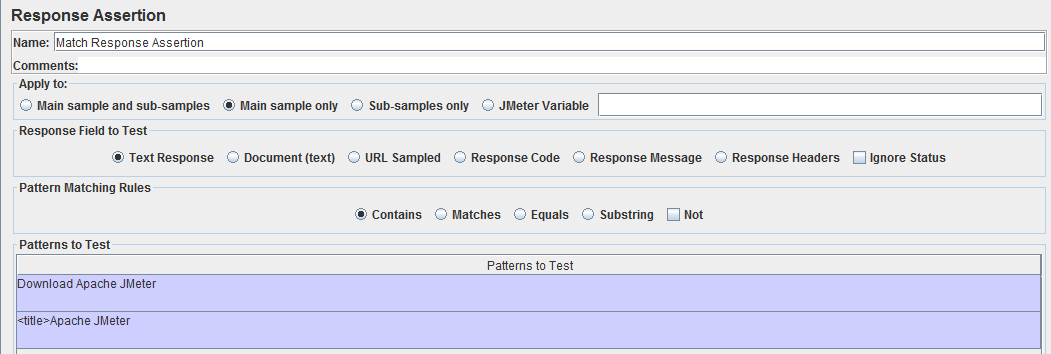
# Week 4: Handle Dynamic Token, Assertions, Using CSV Data File

* What Is Dynamic Token?
  + Dynamic Cross-Site Request token is a token which is generated in the response of the previous request. This is to avoid automation tool to act like a user and attaches the web-site.
  + For example: if you open www.kijiji.com at the first-time, the site will generate a token=123 and when you click to go to the next page, this token will be used to pass to the next request for loading the page. Each time the user open www.kijiji.com, he/she will get a different token and the website will use this to pass to the next page.
  + If we record a test plan for a web-site without changing this dynamic token, at the second time when running test plan, it will be failed.
  + To handle the dynamic token, we need to extract the token from previous response and use it for the next request by using Regular Expression Extractor.
* How To Handle Token With Regular Expression Extractor:
  + The JMeter Regex Extractor saves the values of the dynamic token in a variables.
  + For example, assume:
    - Reference Name: Token
    - Regex: token: "(.+?)"
    - Template: $1$
  + Reference Name: name of variable contains Value extracted.
  + Regex:
    - . match any character
    - + one or more times
    - ? stop when first match succeeds
* Template: is a value group to use in Regular Expression. $1$ is regular expression group 1; $2$ is regular expression group 2.

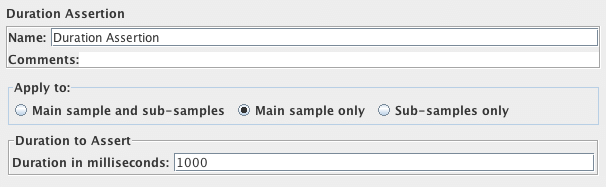




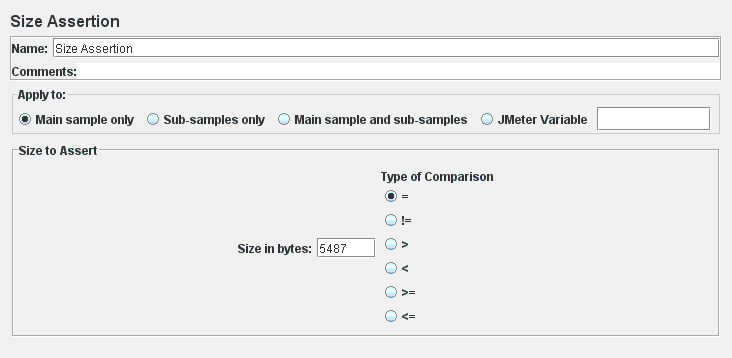
* What Are Assertions?
  + Assertions allow you to assert facts about responses received from the server being tested. Using an assertion, you can essentially test that the response is returning the results you expect it to.
* Assertions:
  + Response Assertion: The response assertion control panel lets you add pattern strings to be compared against various fields of the response.



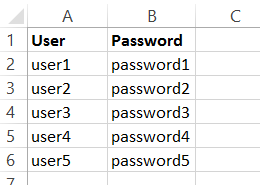
* + Duration Assertion: The Duration Assertion tests that each response was received within a given amount of time. Any response that takes longer than the given number of milliseconds (specified by the user) is marked as a failed response.

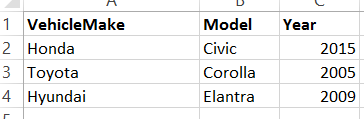
[](http://jmeter.apache.org/images/screenshots/duration_assertion.png)

* + Size Assertion: The Size Assertion tests that each response contains the right number of bytes in it. You can specify that the size be equal to, greater than, less than, or not equal to a given number of bytes.

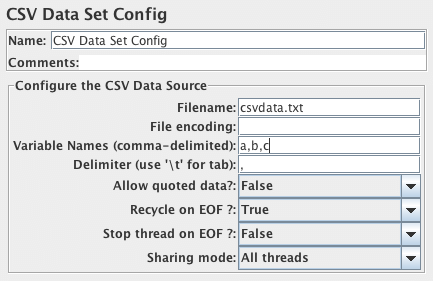


* + XML Assertion: The XML Assertion tests that the response data consists of a formally correct XML document. It does not validate the XML based on a DTD or schema or do any further validation.
  + BeanShell Assertion: The BeanShell Assertion allows the user to perform assertion checking using a BeanShell script. For full details on using BeanShell, please see the BeanShell website (http://www.beanshell.org/).
* Set Up CSV File: for data-driven test, we need to set up a csv file with multiple data to run the test multiple time using different data. For instance, to test how long kijiji.com response with different user searching for different car, we create a User\_PasswordAccount.csv and SearchCars.csv file:





* CSV Data Set Config
  + CSV Data Set Config is used to read lines from a file, and split them into variables. It is well suited to handling large numbers of variables, and is also useful for testing with "random" and unique values.
  + JMeter supports CSV files which have a header line defining the column names. To enable this, leave the "Variable Names" field empty. The correct delimiter must be provided.

[](https://jmeter.apache.org/images/screenshots/csvdatasetconfig.png)

* + File Name: name of the csv file to be read.
  + Variable Names: List of variable names (comma-delimited). JMeter supports CSV header lines: if the variable name field empty, then the first line of the file is read and interpreted as the list of column names.
  + Delimiter: to be used to split the records in the file. If there are fewer values on the line than there are variables the remaining variables are not updated - so they will retain their previous value (if any).
  + Allow quoted data: Should the CSV file allow values to be quoted? If enabled, then values can be enclosed in " - double-quote - allowing values to contain a delimiter.
  + Recycle on EOF: Should the file be re-read from the beginning on reaching EOF? (default is true).
  + Stop thread on EOF: Should the thread be stopped on EOF, if Recycle is false? (default is false).
  + Sharing mode:
    - All threads - (the default) the file is shared between all the threads.
    - Current thread group - each file is opened once for each thread group in which the element appears
    - Current thread - each file is opened separately for each thread
    - Identifier - all threads sharing the same identifier share the same file.