

The configurable JMeter script needs accompanying csv files to run. Let's say the URL of the WebApollo instance is <http://gmod-dev.nal.usda.gov:8080/cercap>. Then in *testdata* directory, we need a subdirectory called *cercap*, and under *cercap* there are two csv files: *usernames.csv* stores usernames and passwords to login the instance, and *operations.csv* stores scaffold names and operations posted to server. The format of *username.csv* is:

```
USERNAME1, PASSWORD1

USERNAME2, PASSWORD2

...
```

All username/password would be used sequentially to login the instance. The format of *operations.csv* is:

```
SCAFFOLD_NAME1  DATA_TO_BE_POSTED1  NUMBER_TO_MATCH1

SCAFFOLD_NAME2  DATA_TO_BE_POSTED2  NUMBER_TO_MATCH2

...
```

The columns are separated by tab. NUMBER_TO_MATCH is the number used for retrieving feature ID from the responded data by regular expression. For instance, the following is responded json data after adding a feature, and to delete the feature next, we need the 4th *uniqueName* (21F1D14EC304A02FE1F74A9889E47427) here. So the NUMBER_TO_MATCH is 4.

regexpal 0.1.4 — a JavaScript regular expression tester

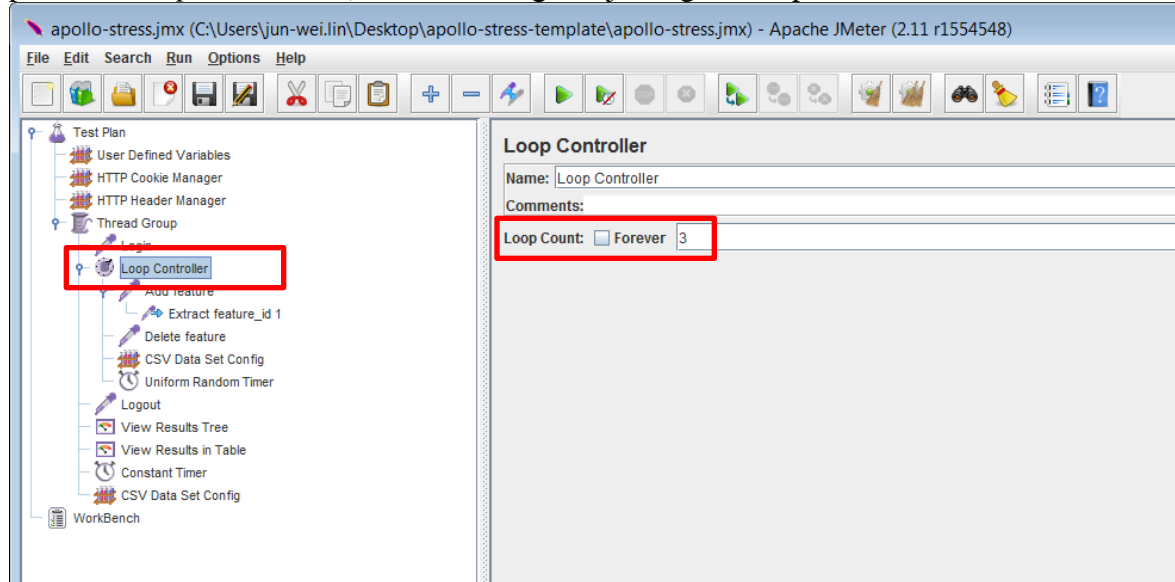
Case insensitive (i) ☐ ☒ \$ match at line breaks (m) ☐ Dot matches all (s; via XRegExp) ☐ Options Quick Reference

RegexBuddy Android Regex Book Version History Blog

uniqueName\":\"(.+?)\"

{\"operation\":\"ADD\",\"sequenceAlterationEvent\":false,\"features\":[{\"location\":{\"fmin\":2960,\"strand\":1,\"fmax\":3244},\"parent_type\":{\"name\":\"gene\",\"cv\":{\"name\":\"sequence\"}},\"name\":\"108609 t\",\"children\":[{\"location\":{\"fmin\":2960,\"strand\":1,\"fmax\":3244},\"parent_type\":{\"name\":\"mRNA\",\"cv\":{\"name\":\"sequence\"}},\"properties\":[{\"value\":\"cercap_user_admin\",\"type\":{\"name\":\"owner\",\"cv\":{\"name\":\"feature_property\"}}}],\"uniqueName\":\"6A68F1C97FBA6216FFD6168E07106E5F\",\"type\":{\"name\":\"CDS\",\"cv\":{\"name\":\"sequence\"}},\"date_last_modified\":1426688460925,\"parent_id\":\"21F1D14EC304A02FE1F74A9889E47427\"},{\"location\":{\"fmin\":3227,\"strand\":1,\"fmax\":3244},\"parent_type\":{\"name\":\"mRNA\",\"cv\":{\"name\":\"sequence\"}},\"properties\":[{\"value\":\"cercap_user_admin\",\"type\":{\"name\":\"owner\",\"cv\":{\"name\":\"feature_property\"}}}],\"uniqueName\":\"CBE9D87E45DAA8DAB2F13B68CD88D87D\",\"type\":{\"name\":\"exon\",\"cv\":{\"name\":\"sequence\"}},\"date_last_modified\":1426688460925,\"parent_id\":\"21F1D14EC304A02FE1F74A9889E47427\"},{\"location\":{\"fmin\":2960,\"strand\":1,\"fmax\":2967},\"parent_type\":{\"name\":\"mRNA\",\"cv\":{\"name\":\"sequence\"}},\"properties\":[{\"value\":\"cercap_user_admin\",\"type\":{\"name\":\"owner\",\"cv\":{\"name\":\"feature_property\"}}}],\"uniqueName\":\"BE7A8204CB7C011FDFA54F9DA647B42B\",\"type\":{\"name\":\"exon\",\"cv\":{\"name\":\"sequence\"}},\"date_last_modified\":1426688460925,\"parent_id\":\"21F1D14EC304A02FE1F74A9889E47427\"},{\"location\":{\"fmin\":2960,\"strand\":1,\"fmax\":2967},\"parent_type\":{\"name\":\"mRNA\",\"cv\":{\"name\":\"sequence\"}},\"properties\":[{\"value\":\"cercap_user_admin\",\"type\":{\"name\":\"owner\",\"cv\":{\"name\":\"feature_property\"}}}],\"uniqueName\":\"21F1D14EC304A02FE1F74A9889E47427\",\"type\":{\"name\":\"mRNA\",\"cv\":{\"name\":\"sequence\"}},\"date_last_modified\":1426688460926,\"parent_id\":\"CBDE71B5B869EB9417D073FF14DA967B\"}]}

By default we have three operations for the jmx script to run, if you want more operations, just put them in *operations.csv*, and don't forget adjusting the loop count here:



To run the jmx script with command in Windows:

```
[JMETER_DIR]/bin/jmeter -t apollo-stress.jmx -Jusers=USER -Jloop=LOOP -  
Jorganism=ORGANISM -Jserver=SERVER
```

For example, the following command triggers 3 users on *cercap* instance performing a test 5 times. Each user in a test performs a cycle of 8 actions (login, add and then delete feature on three scaffolds, logout); each action had a random delay of 0 to 3 seconds.

```
[JMETER_DIR]/bin/jmeter -t apollo-stress.jmx -Jusers=3 -Jloop=5 -  
Jorganism=cercap -Jserver=gmod-dev.nal.udsa.gov
```

We can conduct stress test by spawning processes running on different instances on a server.

p.s. plus *-n* argument can trigger JMeter in non-GUI mode and generate a log file.