ENHANCING LOAD TESTING EFFECTIVENESS: UNLEASHING THE POWER OF THE JMETER SIMPLE TABLE SERVER

JMeter Simple Table Server: An Introduction

JMeter is a popular open-source tool used for performance testing, load testing, and functional testing of web applications. One of the lesser-known features of JMeter is the Simple Table Server (STS), which allows you to store and retrieve test data in a simple tabular format. In this blog post, we'll explore how to implement the STS.

What is the STS?

The STS is a server application that allows you to store and retrieve test data in a tabular format. It provides a simple and easy-to-use interface for creating and managing test data, which can be used in your JMeter test plans. You can store data in tables and use them to simulate real-world scenarios, such as user login credentials, product information, and customer details.

Challenges of using/managing input files locally

- Manually splitting input file record and distributing it on each node.
- If you plan to add a new use case, you will have to repeat the same process again.
- What if you plan to add a new JMeter secondary machine? Now you will have to manage files for each node. For example, 40 files for six nodes are 240 total files.
- What if you want to update part of your input files?

- What if your use case has Maker-Checker flow? Managing FIFO(First In First Out) input list in distributed nodes is quite tedious.

A real-world use case scenario

Consider a scenario where you have 40 thread groups that are using unique user logins. Moreover, you have a distributed JMeter setup with five secondary machines. In this case, you will have to manually split users and copy those files to each secondary machine. so, you end up having 200 input files (40 files for each node equals 200).

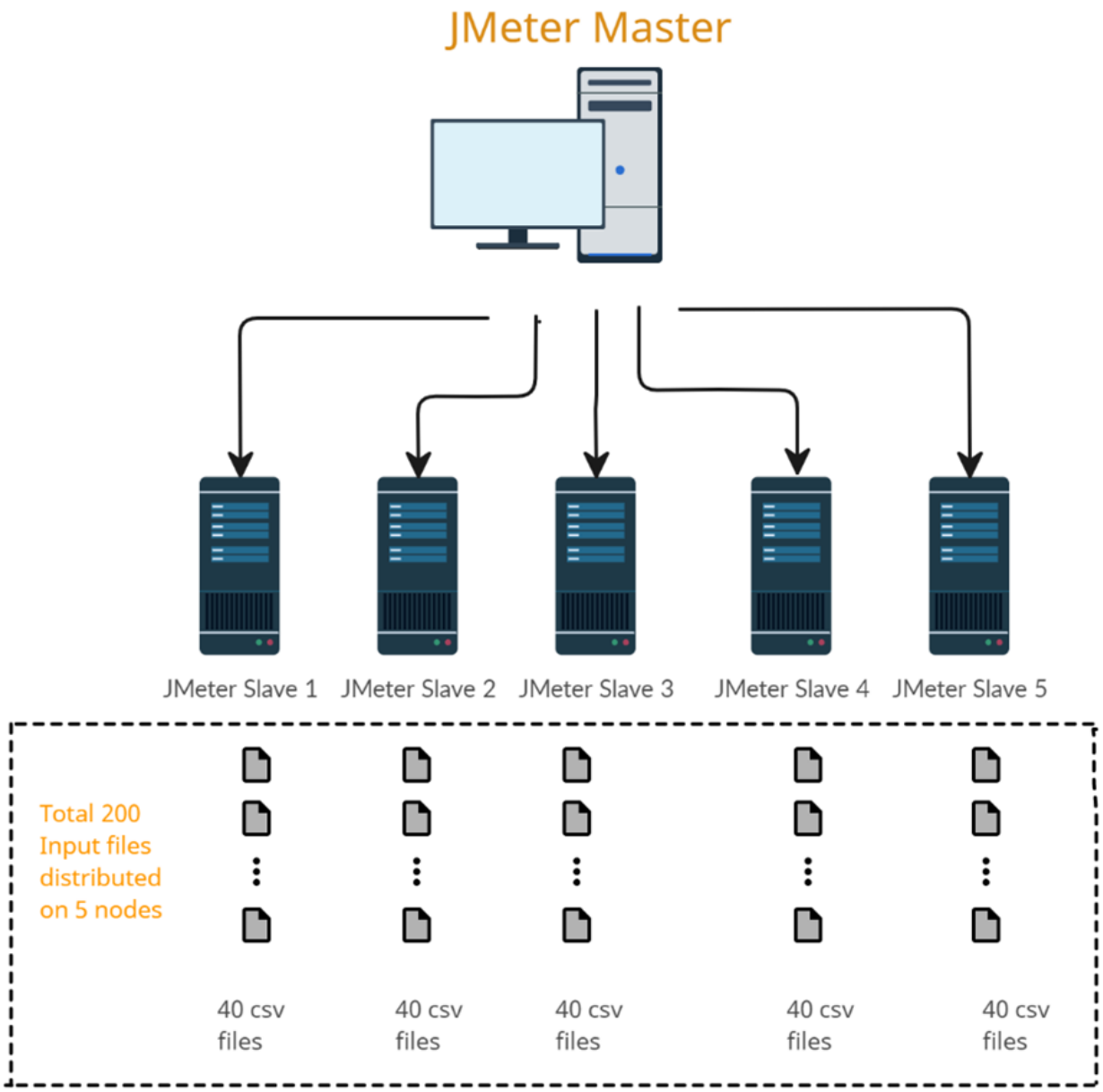


Figure 1.

Benefits of using the STS

- **Improved test performance:** The STS can improve test performance by allowing virtual users to share data. This reduces the need for virtual users to read and write data from external data

files, which can cause delays and increase the test execution time.

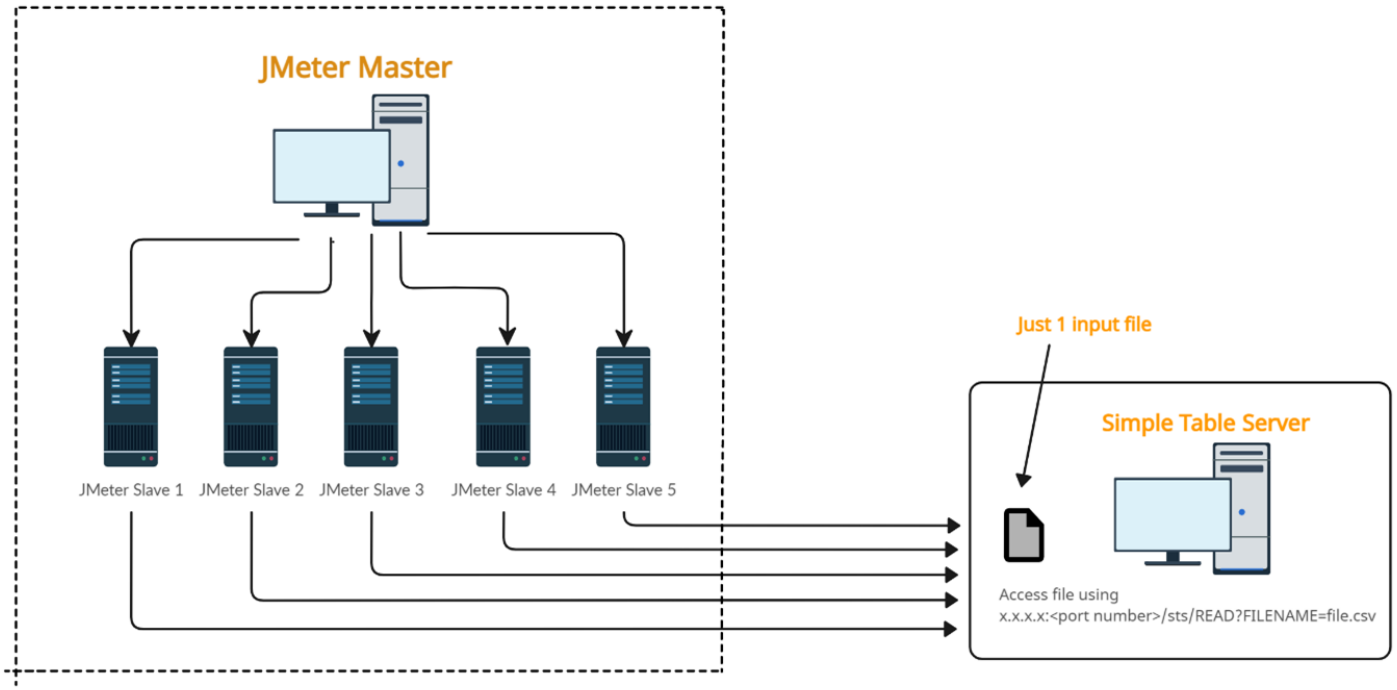


Figure 2.

- **Easy data sharing**: The STS simplifies data sharing between virtual users. Virtual users can easily access and modify data stored in the STS table, making it easier to create complex test scenarios and create, read, update, and delete the table on the fly.
- **Flexibility**: The STS can be used with CSV files, which makes it easy to integrate with different types of applications.
- **Realistic test scenarios**: The STS enables virtual users to share data, which can create more realistic test scenarios. For example, multiple virtual users can access and update the same data simultaneously, simulating a real-world scenario such as Maker-Checker flow.
- **Centralized data management**: The STS provides centralized data management in a single location, which simplifies the management and maintenance of test data.
- **Improved test accuracy**: By allowing virtual users to share, access, and modify the same data, you can reduce data inconsistencies and errors and gain complete control on how you read the file, i.e., unique, sequential, or random.

Configuring the STS

Implementing STS can be done in just a few steps:

Step 1: Download and Install JMeter

To use the STS, you must have JMeter installed on your machine. You can download the latest version from the official website and follow the installation instructions.

Step 2: Download the STS Plugin

Once you have installed JMeter, download the STS plugin and install it by copying the JAR file to the "lib/ext" folder in your JMeter installation directory.

Step 3: Configure the STS

After installing the plugin, you need to configure the STS by adding the following properties in the jmeter.properties file.

In this example, I chose the dataset directory location at C:/jmeter_for_sts/dataset. You can select any location.

By default, the port is 9191, which you can change per your requirements.

```
jmeterPlugin.sts.port=5678
jmeterPlugin.sts.datasetDirectory=C:/jmeter_for_sts/dataset
jmeterPlugin.sts.loadAndRunOnStartup=true
jmeterPlugin.sts.initFileAtStartup=[redacted].csv,[redacted].csv,psrac

jmeterPlugin.sts.charsetEncodingHttpResponse=UTF-8
jmeterPlugin.sts.charsetEncodingReadFile=UTF-8
jmeterPlugin.sts.charsetEncodingWriteFile=UTF-8
```

Figure 3.

Now we will create a CSV file with test data and upload it to the server (at the above "datasetDirectory" location).

The CSV file should contain the test data in subsequent rows as illustrated below:

	A	B	C	D	E	F
1	login1	password1				
2	login2	password2				
3	login3	password3				
4	login4	password4				
5	login5	password5				
6	login6	password6				
7						
8						
9						

Figure 4.

Now you can execute following command in your browser and it will display the number of records in the file: http://hostname:port/sts/INITFILE?FILENAME=env1_login.csv

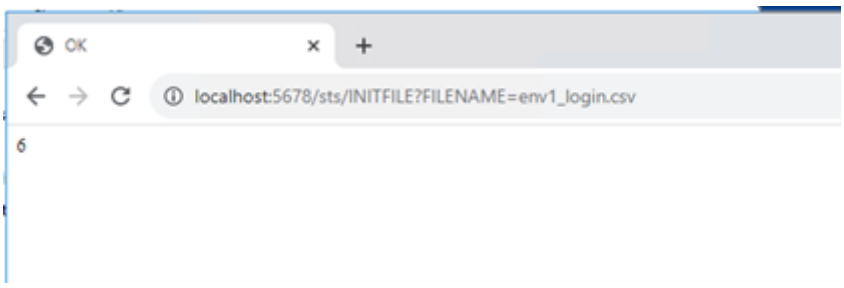


Figure 5. .

Step 4: There are various ways to start the STS. In this example, I am starting by using the simple-table-server.bat file. Go to JMeter's bin directory and click on the "simple-table-server.bat" file.

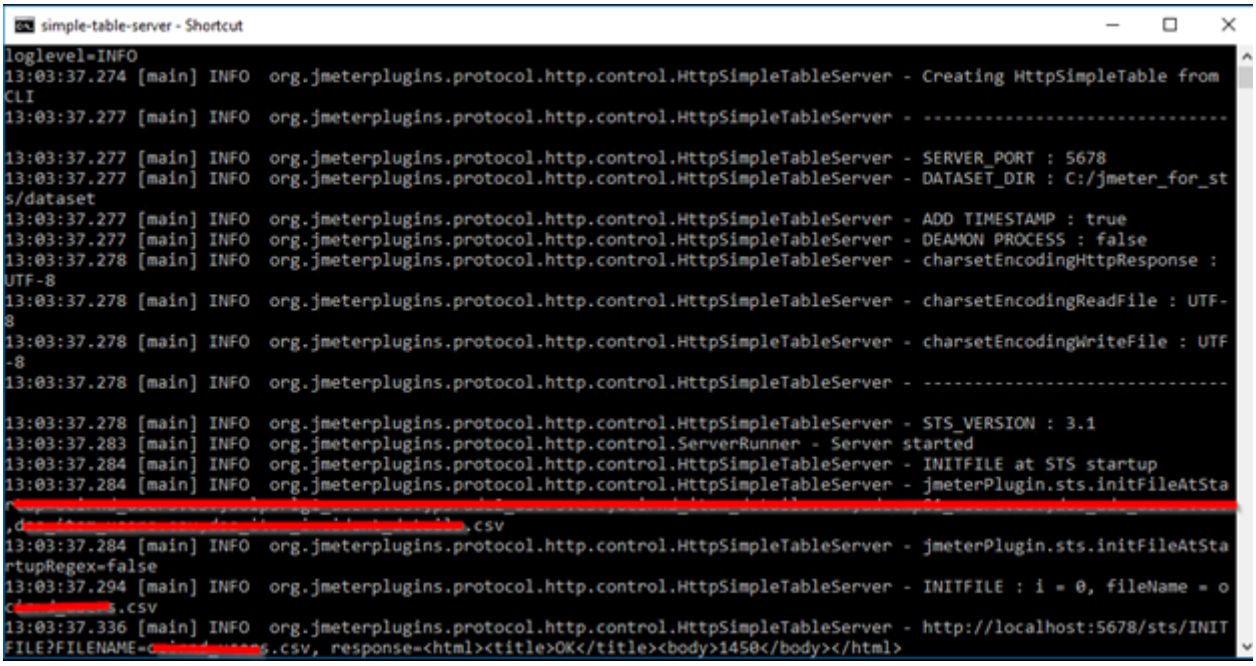


Figure 6.

Upon successful startup, you will see a screen like the one shown above with the server port number and the location of the file.

It will also load the list of files you specified in the initFileAtStartup parameter in jmeter.properties. As a best practice, always add your input files in the "initFileAtStartup parameter." Files that are not mentioned will disappear when the STS restarts.

Now, try to access the URL IP address: <port>/sts.

It should give you the output below with all the commands related to the STS.

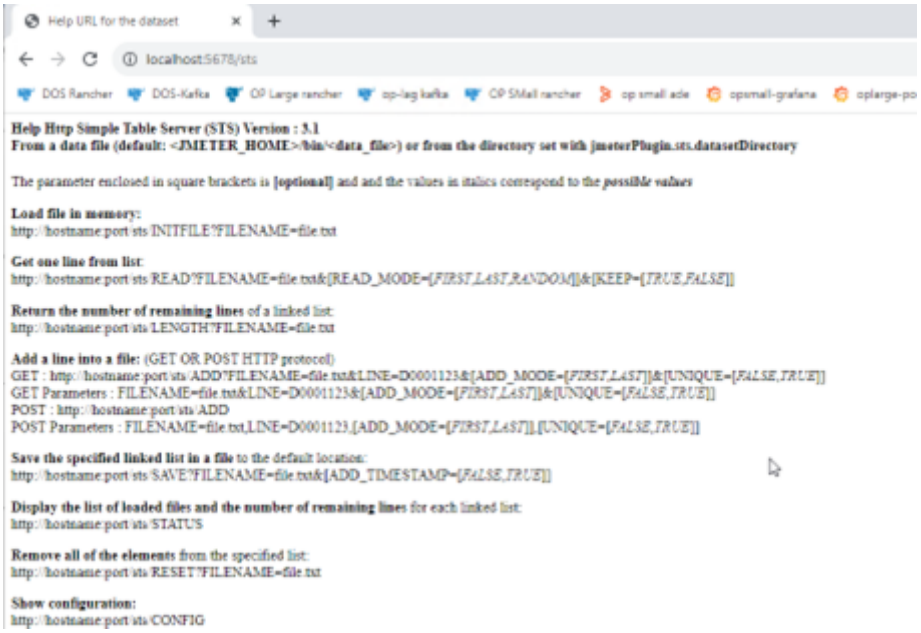


Figure 7.

Once you see this page, you have successfully configured the STS.

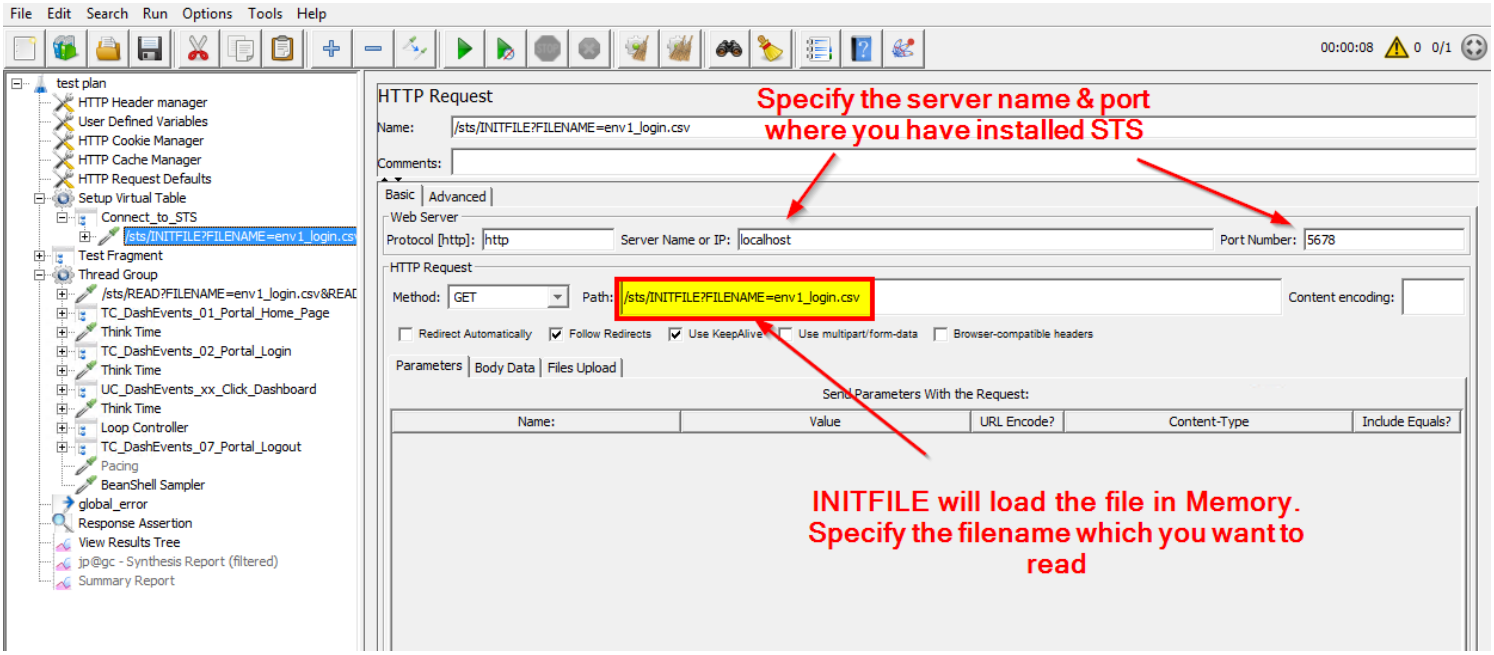

L
Load File
Load file in memory:
http://hostname:port/sts/INITFILE?FILENAME=file.csv
This is command will load the file in STS. You will have to actually copy the file to STS directory
R
Read
Get one line from list:
http://hostname:port/sts/READ?FILENAME=file.txt&[READ_MODE=[FIRST,LAST,RANDOM]] &[KEEP=[TRUE,FALSE]]
There are various ways through which you can read the file. Either First, Last or Random. Keep TRUE is used if you want to keep the record after getting read. False will delete the record once it is read from the table.
S
Save
Save the specified linked list in a file to the default location:
http://hostname:port/sts/SAVE?FILENAME=file.txt&[ADD_TIMESTAMP=[FALSE,TRUE]]
This command will actually save the file. This is mostly used in maker-checker flow.
S
Status
Display the list of loaded files and the number of remaining lines for each linked list:
http://hostname:port/sts/STATUS
This command will display the status of all the files you have loaded in sts. It will also display the number of records in the file
R
Remove
Remove all of the elements from the specified list:
http://hostname:port/sts/RESET?FILENAME=file.txt
It will remove the files which you have loaded in STS.

Figure 8.

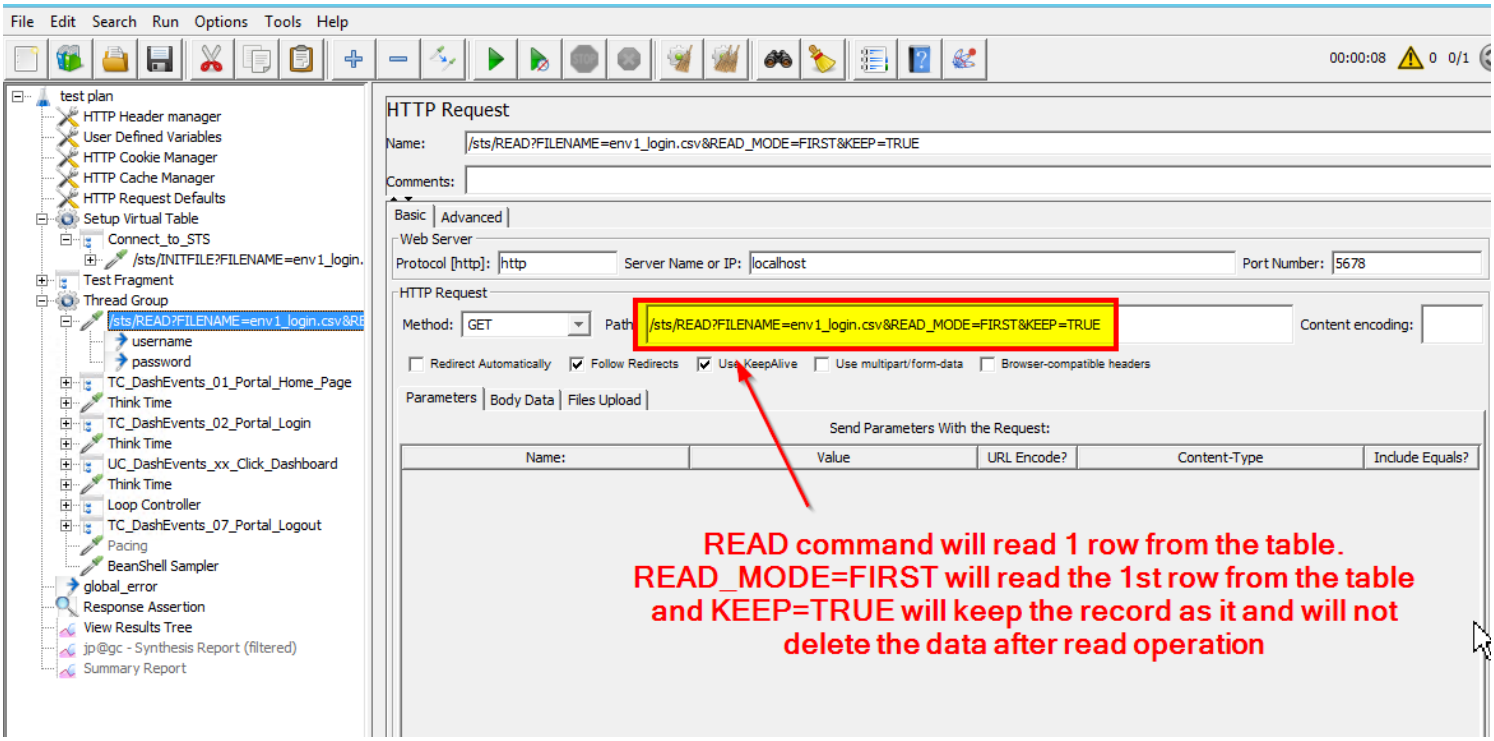
How to integrate the STS into your test plan:

Assuming you followed the above steps and your STS is now fully up and running. Let's try to implement this in your actual test plan. We will practice how one can read the STS data and use it in subsequent requests.

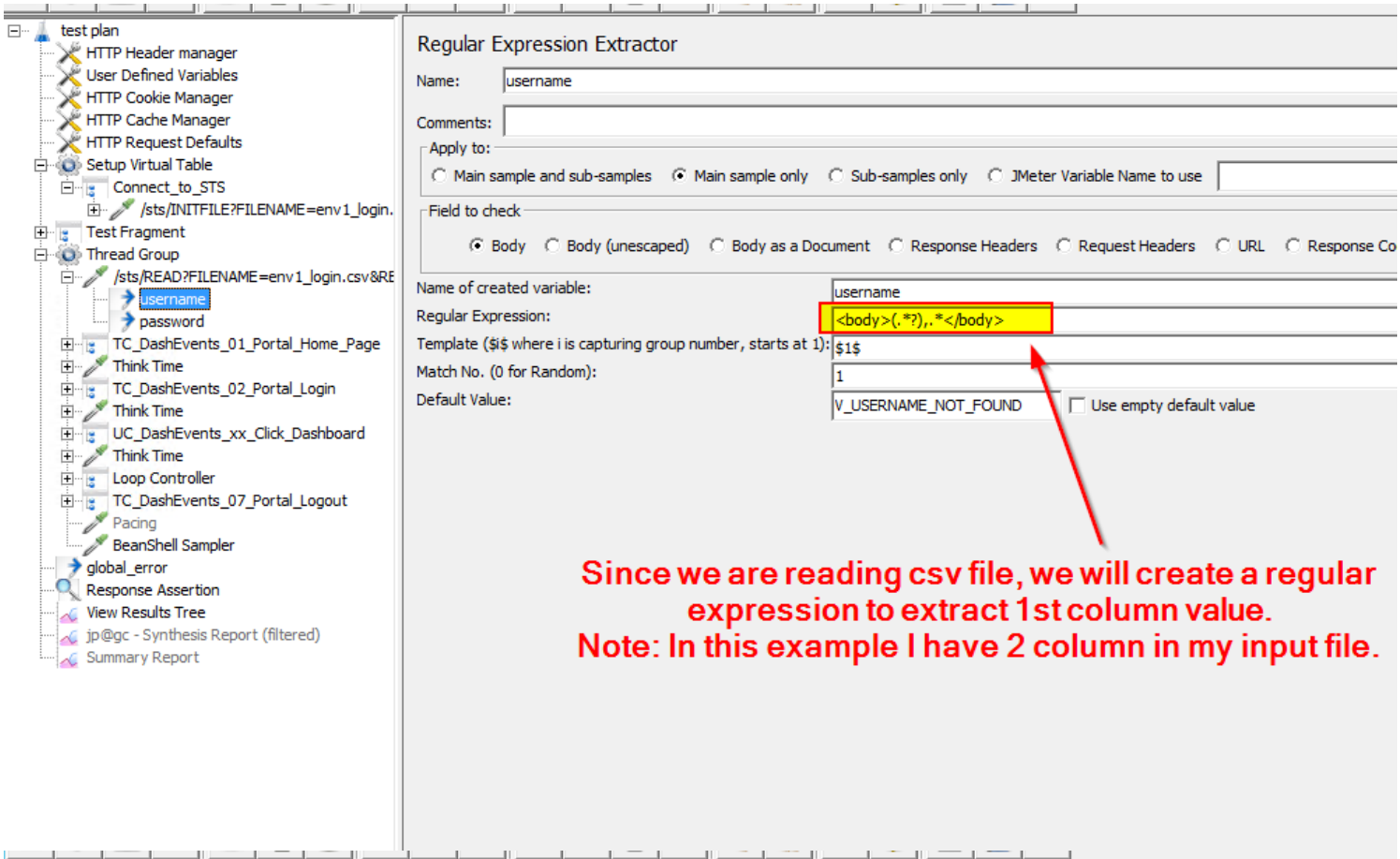
1. Load the file in memory with the INITFILE command.

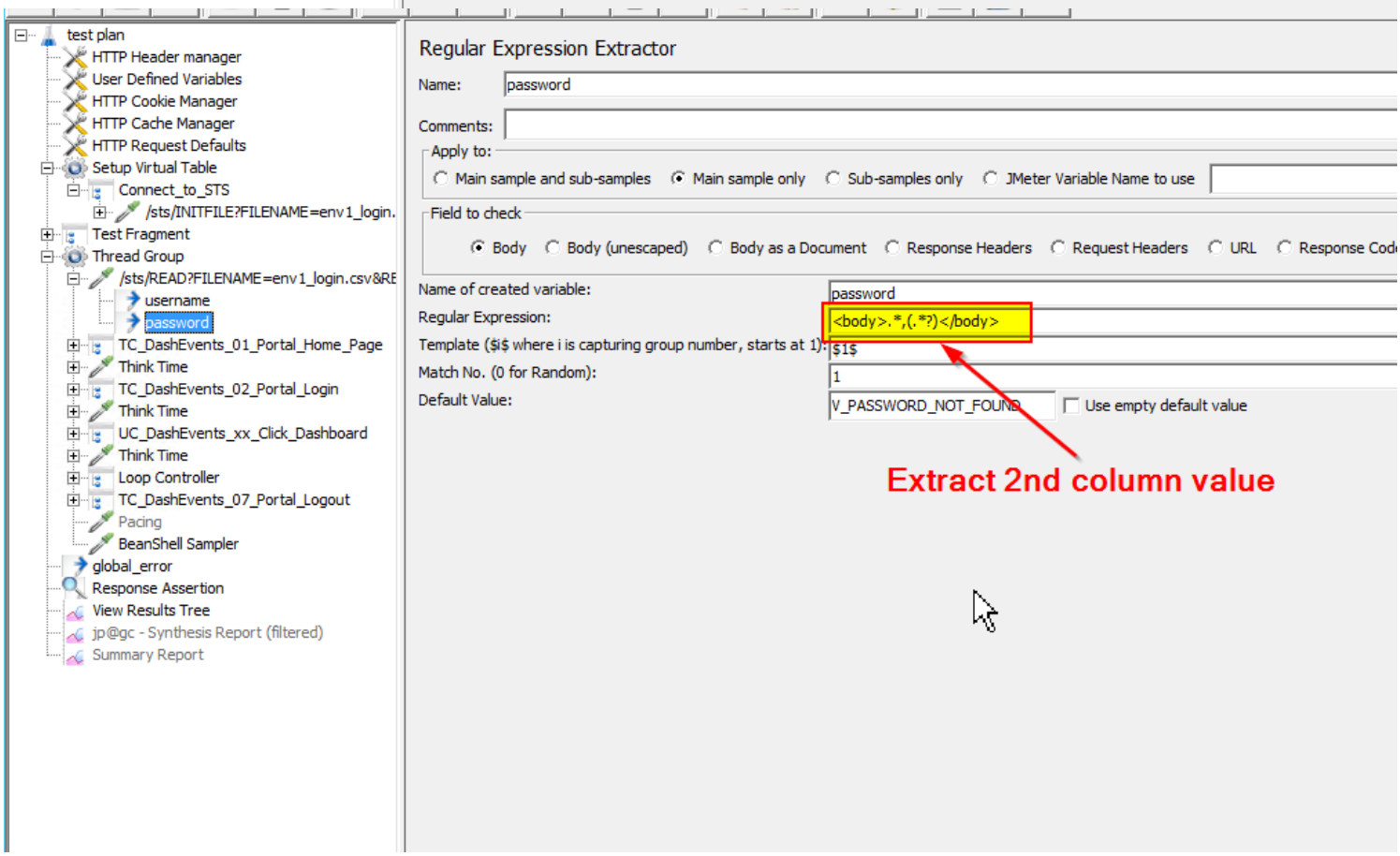


2. Read the file using the READ command and depending upon your choice, you can also read the file through FIRST/LAST/RANDOM commands.



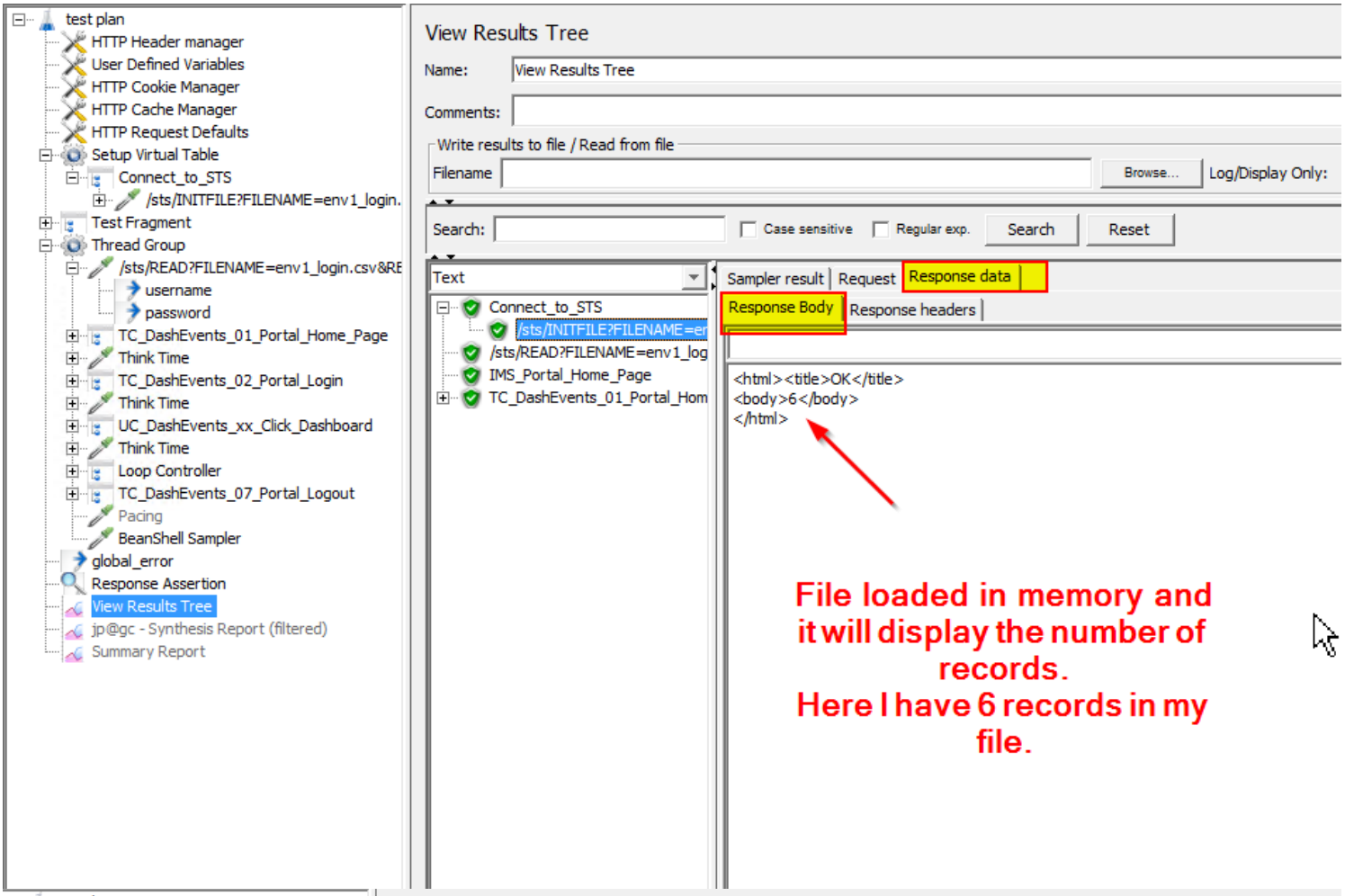
3. Using Regular Expression, extract the username and password from the response data.

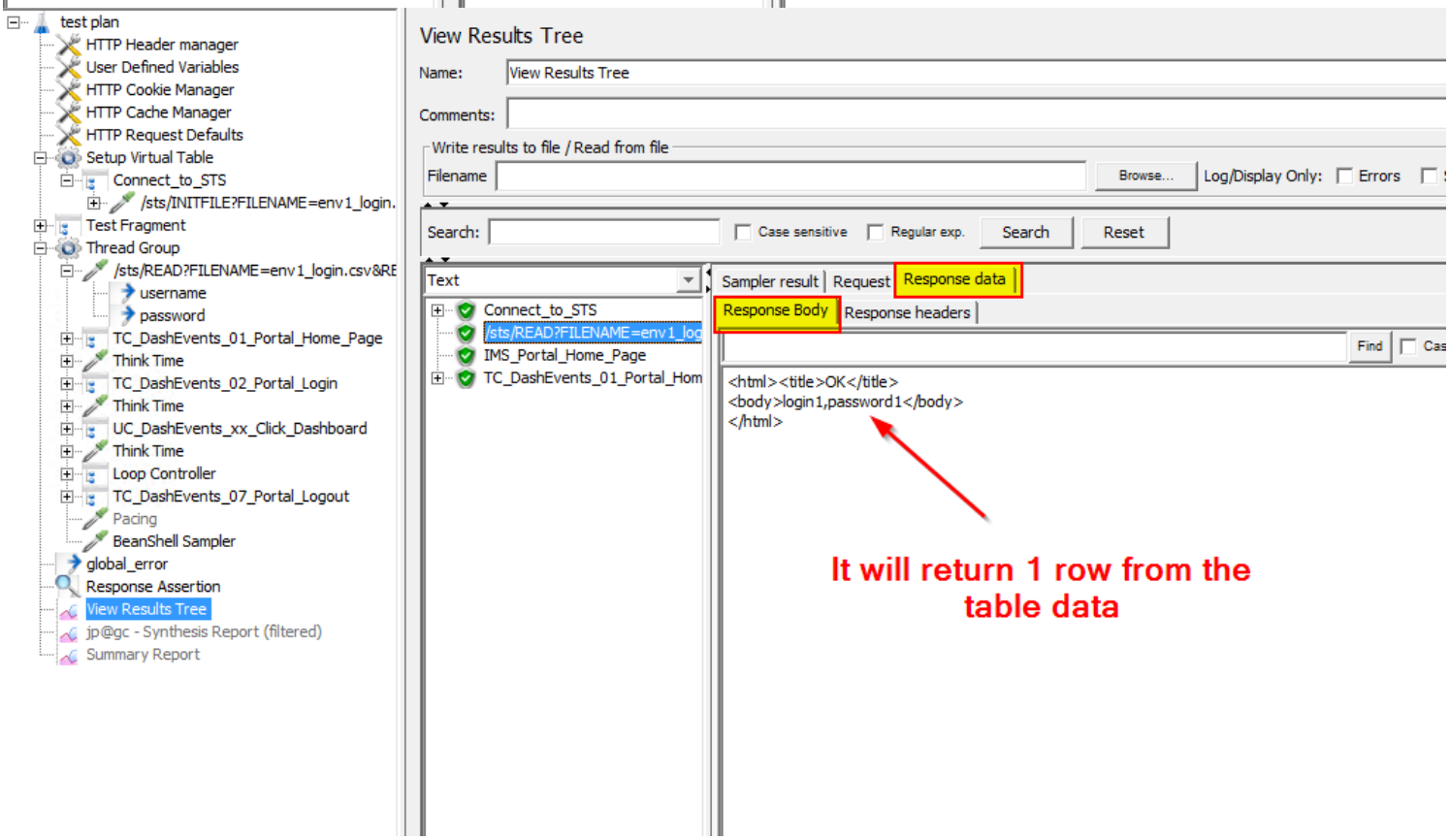




Likewise, you can read other parameters (if you have more column in the files).

4. Let's run the script and check the output.





Now you can use these extracted parameters in subsequent requests, as needed.

Conclusion

The JMeter Simple Table Server is a useful feature that allows you to store and retrieve test data in a tabular format. It provides an easy-to-use interface for managing test data and can be used to simulate real-world scenarios in your JMeter test plans. By following the steps outlined in this blog post, you can easily implement the STS and start using it in your performance testing projects.