09 – CSV Data & Parameterization

🎯 Objective

Learn how to test with **multiple different users** (User1, User2, User3) instead of repeating the same user 100 times. This is called **Parameterization**.

1️⃣ Why Parameterization Matters?

If you run a load test with 1,000 threads but hardcode `username: "testuser1"` :

1. **Server Caching:** The server might cache the response for "testuser1", making your test look faster than it really is.
2. **Logic Errors:** Some systems don't allow the same user to have 1,000 active sessions.
3. **Unrealistic Data:** Real traffic comes from unique IPs, unique accounts, and unique transaction IDs.

👉 **To simulate reality, you need dynamic data.**

2️⃣ The Solution: CSV Data Set Config

JMeter uses a config element called **CSV Data Set Config** to read rows from a file and feed them into variables (e.g., `${username}` , `${password}`).

📂 The CSV File Format

Create a simple text file `users.csv` (save it in the same folder as your `.jmx` script).
Do not use headers in the file unless you specifically configure JMeter to ignore the first line.

Example Content (`users.csv`):

```
user1,pass123
user2,pass123
user3,pass123
user4,pass123
```

3️⃣ Setting Up CSV Data Set Config

Add it to your Test Plan:

`Thread Group` → `Add` → `Config Element` → `CSV Data Set Config`

Configuration Fields:

- **Filename:** `users.csv` (Use relative path if in the same folder, just the filename).
- **File Encoding:** `UTF-8`.
- **Variable Names:** `username,password` (This maps column 1 to `${username}` and column 2 to `${password}`).
- **Ignore First Line:** `False` (unless you have a header row).
- **Delimiter:** `,` (Comma).

4️⃣ Using the Variables

Now, update your **Login API** request body to use the variables instead of static text.

Old Body:

```
{
  "username": "user1",
  "password": "pass123"
}
```

New Parameterized Body:

```
{
  "username": "${username}",
  "password": "${password}"
}
```

👉 **When the test runs:**

- Thread 1 reads Row 1 (`user1`)
- Thread 2 reads Row 2 (`user2`)
- Thread 3 reads Row 3 (`user3`)

5️⃣ Critical Settings: Recycle vs. Stop

What happens if you have **100 Threads** but only **4 rows** in your CSV?

🔹 Option A: Recycle on EOF = True (Default)

- Thread 1-4 use rows 1-4.
- Thread 5 goes back to Row 1 (`user1`).
- **Use Case:** When you don't care if users are reused (e.g., generic read-only users).

🔹 Option B: Stop Thread on EOF = True

- Thread 1-4 use rows 1-4.
- Thread 5 **stops** and does not run.
- **Use Case:** Registration flows where you cannot register the same email twice. Also useful for "consume-once" data like coupon codes.

6️⃣ Sharing Mode (Advanced)

- **All threads:** (Default) Lines are distributed one-by-one to whoever asks next.
- **Current thread group:** Each Thread Group opens its own copy of the file.
- **Current thread:** Each *User* reads the file from the top (Row 1). (Rarely used, as every user would effectively be "User1").

7️⃣ Interview Question

"I have a CSV with 500 records. I want to run a test with 1000 users. What happens?"

Answer:
"It depends on the **'Recycle on EOF'** setting.

- If **True**, JMeter will loop back to the top and reuse the first 500 records.
- If **False**, the first 500 threads will run, and the remaining 500 threads will stop immediately (or fail, depending on config)."

🎯 Mini Exercise

1. Create `data.csv` with 3 lines of mock users.
2. Add **CSV Data Set Config** to your Thread Group.
3. Change your Login Request to use `${username}`.
4. Run with **5 Threads**.
5. Check **View Results Tree** request bodies. You should see different users being sent.