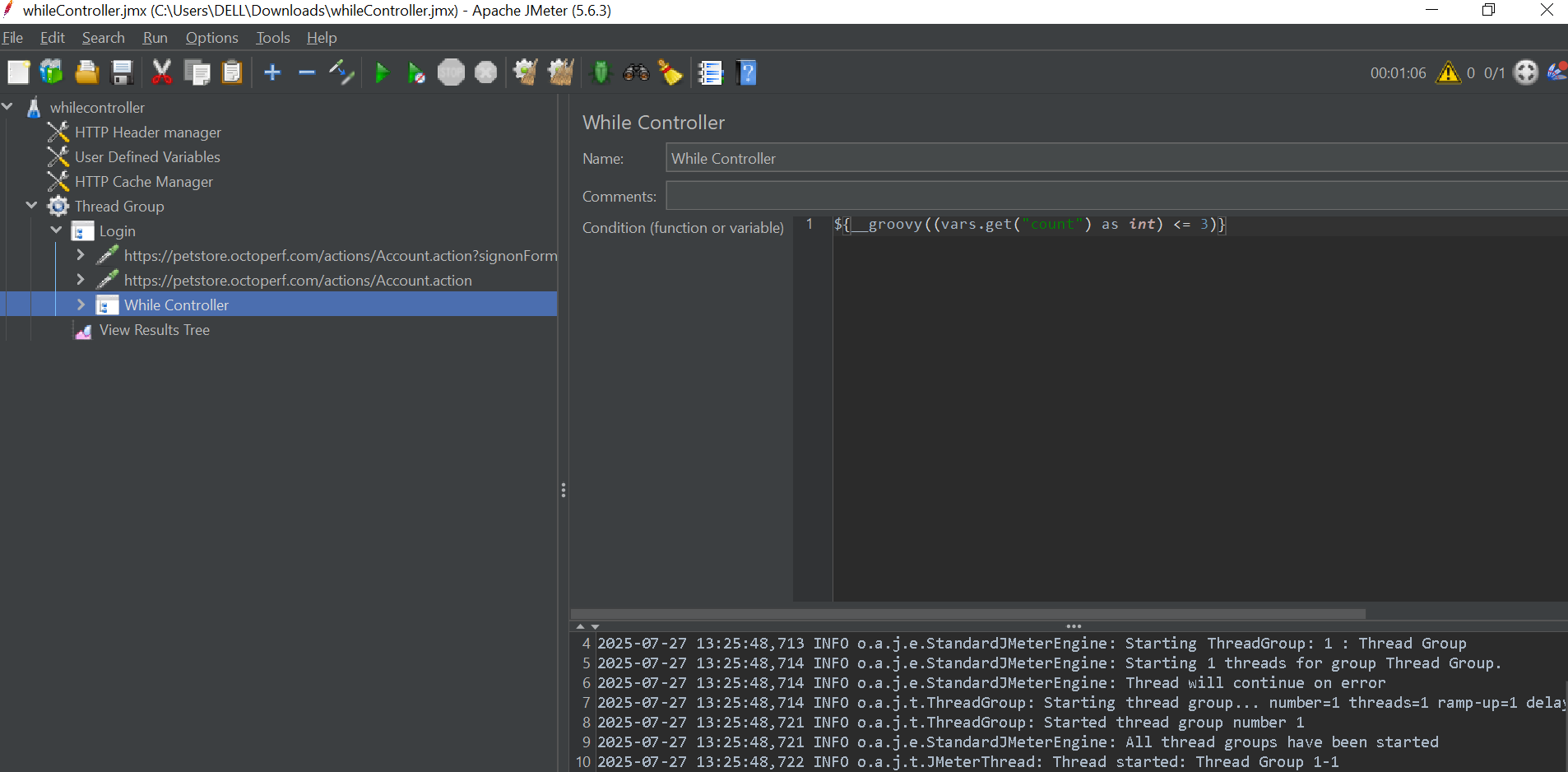
🔁 While Controller in JMeter





**📘 What Is It?**

The **While Controller** is a **Logic Controller** that repeatedly executes its child elements **as long as a specified condition is true**.  
It's similar to the while(condition) loop in programming languages.

**🔧 How to Configure a While Controller**

1. **Add a Thread Group**  
   *Right-click Test Plan → Add → Threads (Users) → Thread Group*
2. **Add a While Controller**  
   *Right-click Thread Group → Add → Logic Controller → While Controller*
3. **Set the Condition (core logic)**  
   In the “Condition” field, you can:
   * Leave it **blank**: Exits when the **last sampler fails**
   * Use **"LAST"**: If the previous sampler fails, skip execution entirely
   * Use **JMeter Functions** to return "true" or "false"

✅ **Recommended: Use JavaScript or JEXL3 Functions**

**✅ Valid Condition Examples**

| **Use Case** | **Condition** |
| --- | --- |
| Loop until a value is reached | ${\_\_javaScript("${myVar}" != "expected",)} |
| Simulate a for loop | ${\_\_javaScript(parseInt(vars.get("counter")) <= 5,)} |
| Poll until status = COMPLETE | ${\_\_jexl3("${status}" == "COMPLETE",)} |
| Continue if response does not contain SUCCESS | ${\_\_javaScript("${response}".indexOf("SUCCESS") == -1,)} |
| Check token not null | ${\_\_groovy(vars.get("token") != "null",)} |

🔒 **Avoid** using ${myVar} == "value" directly — it won’t work properly.

**🧪 Add Child Elements**

Include:

* Samplers (e.g., HTTP Request)
* Post-Processors (like JSON Extractor to update variables)
* Timers (to avoid tight loops)
* Logic Controllers
* Assertions

**🔄 Common Use Cases**

| **Scenario** | **Description** |
| --- | --- |
| **Polling for Status** | Wait for an async operation to finish (e.g., "ORDER → DELIVERED") |
| **Retry on Failure** | Retry a request until success (e.g., payment gateway, file download) |
| **Dynamic Iterations** | Continue until a condition met in response or data |
| **Conditional Loops** | Loop depends on application behavior |
| **For-Loop Alternative** | Combine with Counter to simulate a "for" loop |

**📘 Example: Polling for Order Status**

pgsql

CopyEdit

Test Plan

└── Thread Group (Threads: 1, Loop Count: 1)

├── HTTP Request (Place Order)

│ └── JSON Extractor (Extract orderId)

├── User Defined Variables (orderStatus = "PENDING")

├── While Controller (Condition: ${\_\_javaScript("${orderStatus}" != "DELIVERED",)})

│ ├── HTTP Request (Get Order Status using ${orderId})

│ │ └── JSON Extractor (Update orderStatus)

│ └── Constant Timer (5000 ms)

└── HTTP Request (Finalize / Acknowledge Order)

**Explanation:**

* Place an order → extract orderId
* Set initial orderStatus = PENDING
* Loop until orderStatus == DELIVERED
* In each iteration: get status, extract, wait 5s, repeat
* Exit loop and finalize order

**⚠️ Important Considerations**

| **Tip** | **Description** |
| --- | --- |
| **Clear Exit Condition** | Ensure the loop can eventually evaluate to false |
| **Timers** | Use a Constant or Random Timer to avoid overwhelming the server |
| **Variable Scope** | Make sure variables used in the condition are **updated inside the loop** |
| **Error Handling** | Use If Controllers or custom scripting to manage failures gracefully |
| **Thread Safety** | In complex cases, use JSR223 + Groovy for thread-safe logic |

**🧠 Pro Tip**

For retry mechanisms or polling, combine **While Controller** with:

* **JSR223 Sampler** to increment counters
* **JSON Extractor** to update dynamic values
* **Timers** to pace the loop