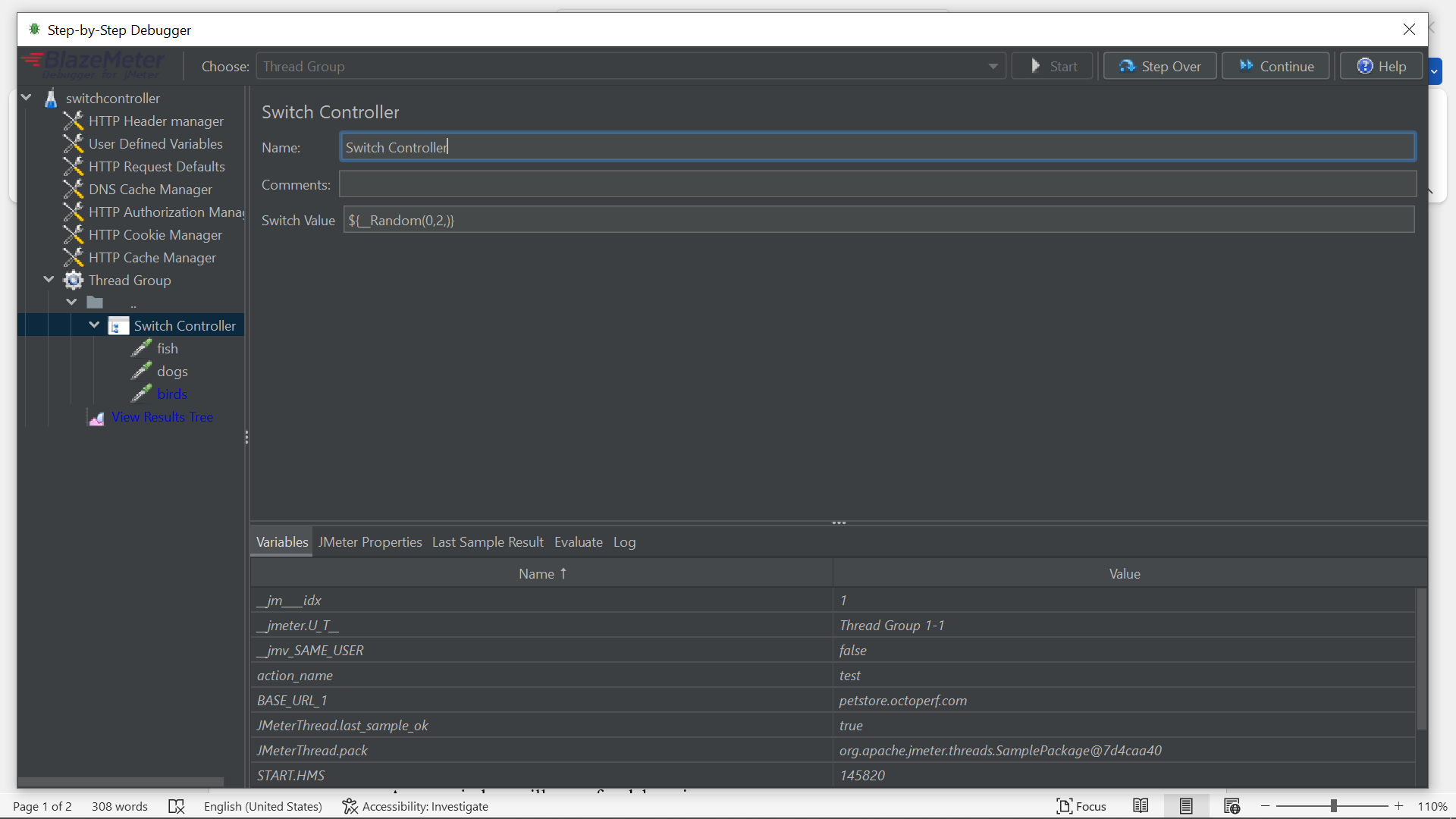
🔍 Step-by-Step Debugger





JMeter does **not have a native debugger**, but you can simulate step-by-step debugging using the **BlazeMeter Step-by-Step Debugger** plugin. Here’s how:

**✅ Step 1: Install JMeter Plugins Manager**

1. **Download the Plugin Manager**
   * Get plugins-manager.jar from the JMeter Plugins site.
2. **Place the JAR file**
   * Copy plugins-manager.jar into JMeter’s lib/ext directory.
3. **Restart JMeter**
   * After restart, go to **Options > Plugins Manager** to confirm it's installed.

**✅ Step 2: Install the BlazeMeter Debugger Plugin**

1. **Open Plugins Manager**
   * Go to **Options > Plugins Manager**.
2. **Install Debugger**
   * In the **Available Plugins** tab, search for:

vbnet

CopyEdit

BlazeMeter Step-by-step Debugger

* + Check it and click **Apply Changes and Restart JMeter**.

**🐞 Step 3: Use the Step-by-Step Debugger**

**🎛️ Launch the Debugger**

* Go to **Run > Step by Step Debugger**.
* A new window will open for debugging.

**👥 Select Thread Group**

* Use the dropdown to choose the thread group to debug (one at a time).

**▶️ Start and Control Execution**

* **Start** – Highlights the first executable element.
* **Step Over** – Executes the current step and moves to the next.

**🔎 Debugger Tabs Overview**

| **Tab Name** | **Purpose** |
| --- | --- |
| **Variables** | View current JMeter variable values (debug extractors, etc.) |
| **Properties** | Inspect JMeter properties (useful for shared or global configurations) |
| **Last Sample** | View request/response of the last sampler like View Results Tree |
| **Evaluate** | Run JMeter functions or expressions dynamically during debug |

**🛑 Set and Use Breakpoints**

* **Right-click** any element > Select **"Breakpoint"**.
* The icon will turn red 🔴.
* In the debugger, click **Continue** to execute until the breakpoint is hit.
* Once paused, inspect or step over.

**⛔ Stop Debugging**

* Use the **Stop** button to end the debug session.

**📌 Benefits of the Debugger**

* Real-time insight into test execution.
* Great for **debugging correlation**, **dynamic variables**, **timing issues**, or **loop logic**.
* Saves time when tuning complex test plans or custom scripts.