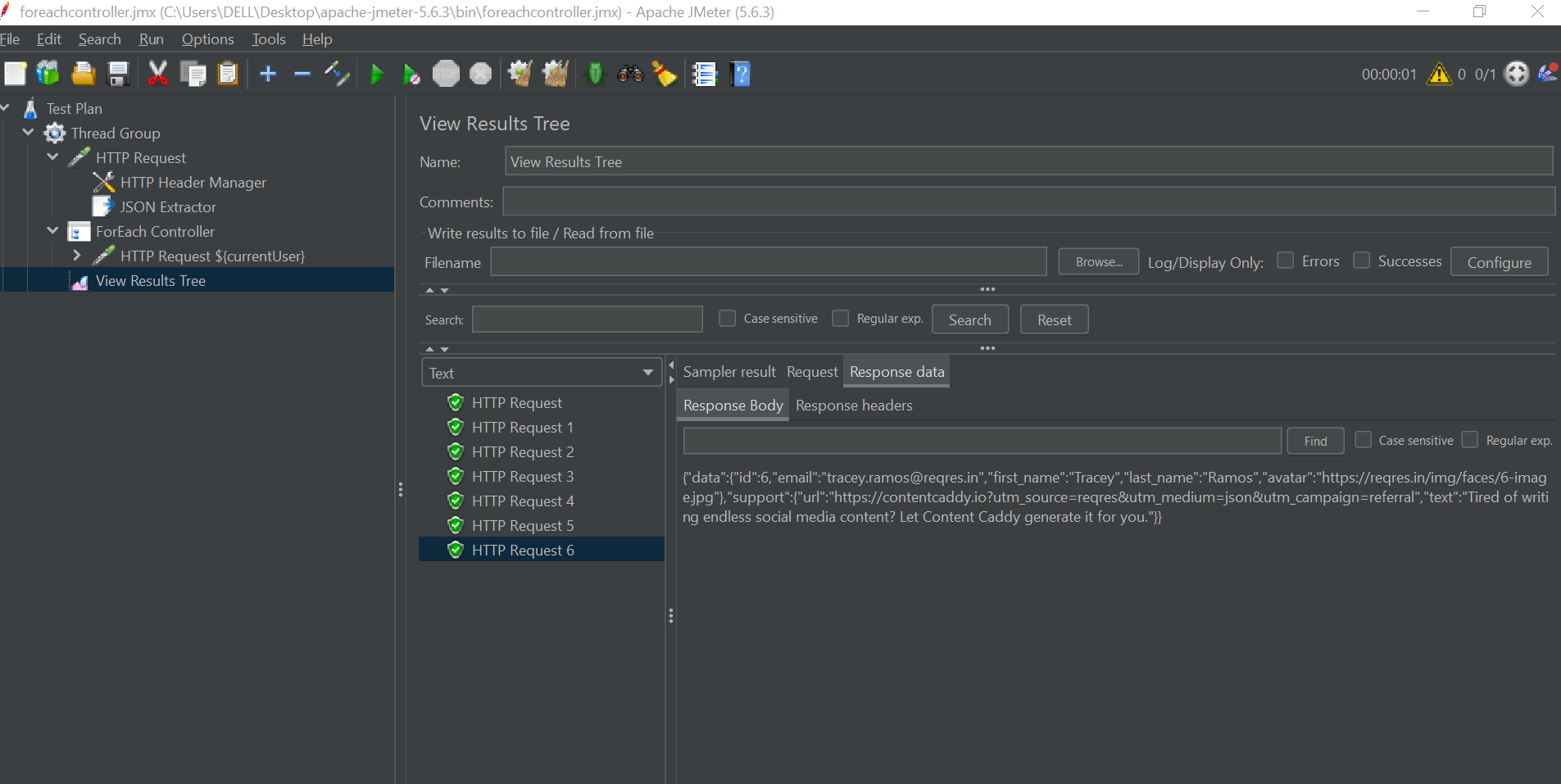
🌀 ForEach Controller





The **ForEach Controller** is a **Logic Controller** in JMeter that enables you to **loop through a list of variables dynamically**. It’s especially useful when working with **data extracted from previous responses**, like items in a JSON array.

**🔧 What Does It Do?**

It iterates over variables that share a **common prefix** and differ by a **numeric suffix** (e.g., item\_1, item\_2, item\_3), executing its child elements for each value. Each value is assigned to a **single output variable** per iteration, making it accessible for samplers or logic.

**📌 When to Use It**

1. **Processing Dynamic Data**  
   → Ideal for looping through items returned by a JSON array or dynamic response (e.g., product IDs, user IDs).
2. **Predefined Data Iteration**  
   → Loop through predefined variables (like usernames or URLs).
3. **Eliminating Redundant Samplers**  
   → Avoid writing repeated samplers for each data point. Just loop with ForEach.

**🧩 Key Configuration Fields**

| **Field** | **Description** |
| --- | --- |
| **Input variable prefix** | Prefix of dynamic variables (e.g., productId) |
| **Start index for loop** | Start index (exclusive). e.g., 0 = start from \_1 |
| **End index for loop** | End index (inclusive). Leave blank to iterate all |
| **Output variable name** | The name used to store current value (e.g., currentProduct) |
| **Add "\_" before number** | Usually checked to match var\_1, var\_2, etc. |

**📖 Example Scenario**

**Scenario:** You call an API that returns a list of users, and you want to fetch each user’s profile.

**API Response:**

json

CopyEdit

{

"users": [

{ "id": "user123", "name": "Alice" },

{ "id": "user456", "name": "Bob" },

{ "id": "user789", "name": "Charlie" }

]

}

**Goal:**

Call /users/{id}/profile for each id.

**🛠️ Step-by-Step Implementation**

1. **Add Thread Group**  
   Test Plan > Add > Threads (Users) > Thread Group
2. **HTTP Request to Get User List**  
   Add GET /users request.
3. **Extract User IDs using JSON Extractor**  
   Add > Post Processors > JSON Extractor
   * **Names of created variables:** userId
   * **JSON Path Expressions:** $.users[\*].id
   * **Match No.:** -1 (extract all)  
     ➜ This will create userId\_1, userId\_2, userId\_3, etc.
4. **Add ForEach Controller**  
   Add > Logic Controller > ForEach Controller
   * **Input variable prefix:** userId
   * **Output variable name:** currentUser
   * Check: **Add "\_" before number**
5. **Inside ForEach Controller: Add HTTP Request**  
   Add > Sampler > HTTP Request
   * **Path:** /users/${currentUser}/profile
   * Method: GET

**▶️ How It Executes**

* Iteration 1: currentUser = userId\_1 → user123 → GET /users/user123/profile
* Iteration 2: currentUser = userId\_2 → user456 → GET /users/user456/profile
* Iteration 3: currentUser = userId\_3 → user789 → GET /users/user789/profile

**🧠 Useful Tips**

* ✅ **Consistency in Variable Naming**  
  Use extractors that auto-generate numbered variables with a clear prefix (e.g., var\_1, var\_2).
* 🔄 **\_\_jm\_\_ForEachController\_\_idx**  
  You can use ${\_\_jm\_\_<ForEachControllerName>\_\_idx} to get the **current index** of the iteration—great for debugging or conditional logic.
* ⚠️ **Scope Awareness**  
  Place your ForEach Controller where the extracted variables are **still in scope**—generally under the same Thread Group.

**📋 Final Notes**

* The **ForEach Controller is essential** when dealing with dynamic lists.
* Combine it with **JSON Extractor** or **Regular Expression Extractor** for powerful, flexible test flows.
* Avoids redundancy and follows the **DRY principle** effectively.