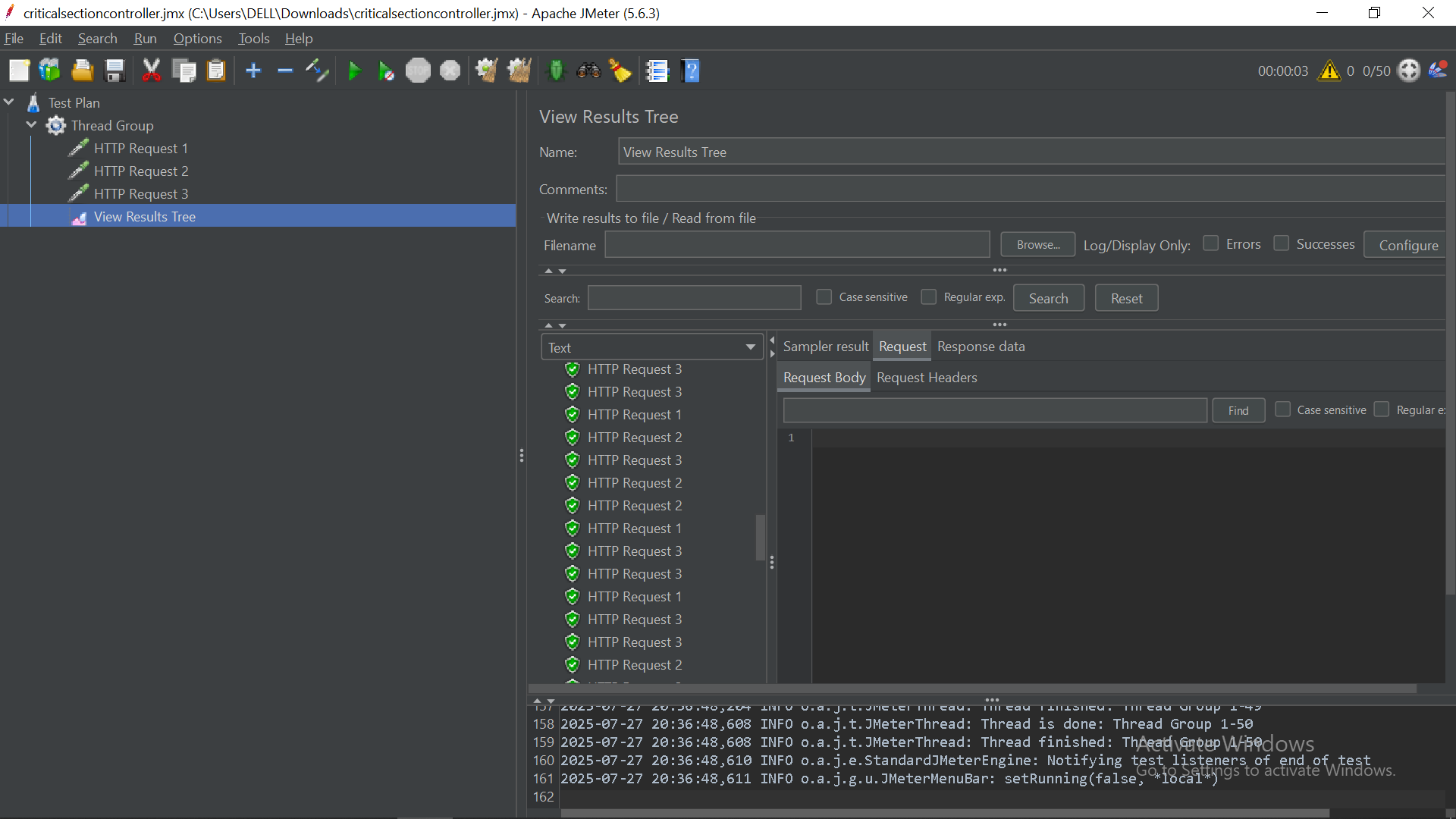
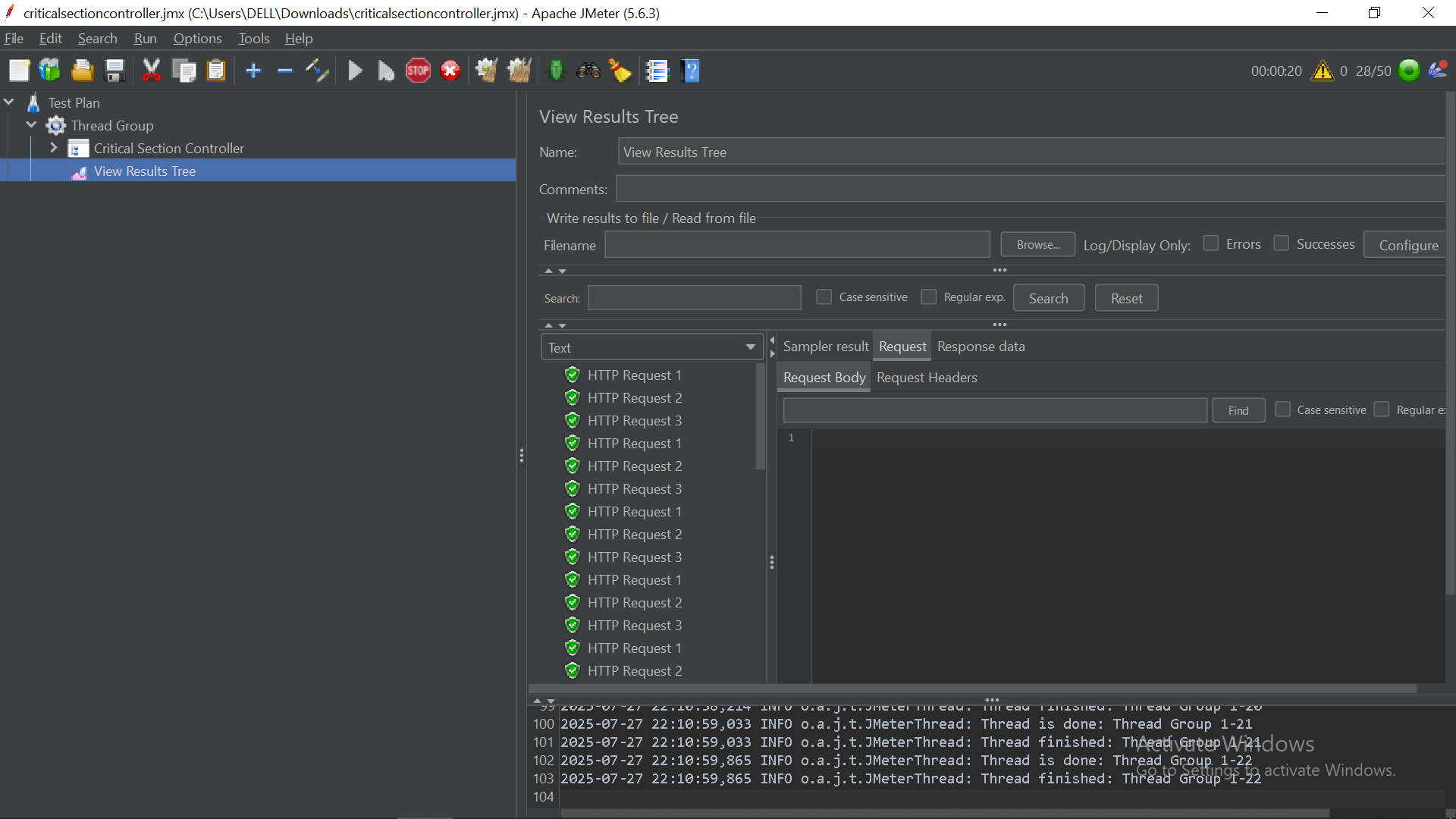
🔐 Critical Section Controller in JMeter





**📘 What Is It?**

The **Critical Section Controller** is a **Logic Controller** that ensures **only one thread at a time** can execute its **child elements** — simulating mutual exclusion, like a "lock" or **synchronized block** in programming.

**⚙️ How It Works**

| **Step** | **Description** |
| --- | --- |
| **1. Named Lock** | You assign a **Lock Name** to identify the critical section. |
| **2. Lock Acquisition** | A thread must acquire the lock before executing child elements. |
| **3. Exclusive Execution** | Only **one thread** can execute inside at any time (even across Thread Groups). |
| **4. Wait and Retry** | Other threads wait until the lock is released. |
| **5. Lock Release** | Once the thread finishes execution of the child elements, the lock is released automatically. |

**🛠 Configuration**

1. **Add a Thread Group**
2. **Add Critical Section Controller**  
   *Right-click Thread Group → Add → Logic Controller → Critical Section Controller*
3. **Set Lock Name**
   * Unique identifier (e.g., MySharedLock)
   * Threads with the same lock name will block each other
4. **Add Child Elements**
   * Place any samplers or logic that must be **executed one thread at a time**

**✅ Common Use Cases**

| **Scenario** | **Why Use It?** |
| --- | --- |
| **Shared Counter / Variable Update** | Prevent race conditions |
| **Shared File Writing** | Avoid corrupting files |
| **Simulate Admin Task** | Allow only one user/thread |
| **Login with Shared Users** | Avoid concurrent login of same account |
| **Database Updates** | Prevent deadlocks or constraint violations |

**📘 Example: Incrementing a Global Counter**

**Test Plan**

less

CopyEdit

Test Plan

└── Thread Group (Threads: 10, Loop Count: 5)

├── Critical Section Controller (Lock Name: MyGlobalCounterLock)

│ └── JSR223 Sampler (Groovy):

**JSR223 Code:**

groovy

CopyEdit

// Get global counter or initialize to 0

long counter = Long.parseLong(props.get("globalCounter") ?: "0");

// Increment

counter++;

// Update property

props.put("globalCounter", counter.toString());

// Log it

log.info("Thread ${Thread.currentThread().getName()} - Global Counter: $counter");

**🔄 Output Behavior:**

* You’ll see sequential logs like:  
  Global Counter: 1, 2, 3, ..., 50
* Ensures no two threads update at the same time.

**⚠️ Important Considerations**

| **Concern** | **Notes** |
| --- | --- |
| **Performance Bottleneck** | Serializes threads → reduces throughput |
| **JVM-Local Lock** | Not effective across **multiple JMeter instances** in distributed testing |
| **Deadlocks Possible** | Avoid multiple inter-dependent Critical Section Controllers |
| **Scope** | Only child elements are serialized; rest of test plan continues normally |

**🧠 Pro Tips**

* Use **only where absolutely necessary** — excessive use defeats the purpose of concurrent load testing.
* For **cross-engine locking**, consider custom scripts or external systems (e.g., DB row lock or Redis).
* Combine with **property variables**, **JSR223 Sampler**, or **BeanShell** to manage shared state safely.