14 – Best Practices & Common Mistakes

🎯 Objective

You now know the tools. This module teaches you the **wisdom** to use them safely. We will cover the most common ways beginners fail at load testing and how to ensure your results are trusted by the organization.

1️⃣ The "Seven Deadly Sins" of JMeter 🛑

❌ 1. Using GUI Mode for Load Testing

- **The Sin:** Running 1,000 users with the JMeter interface open.
- **The Consequence:** Your laptop freezes, JMeter crashes (OOM), and results are garbage because the *client* was the bottleneck, not the server.
- ✅ **Fix:** Always use CLI mode (`jmeter -n -t ...`) for anything above 50 users.

❌ 2. Leaving "View Results Tree" Enabled

- **The Sin:** Forgetting to disable this listener during a load test.
- **The Consequence:** JMeter stores every single response in RAM. Crash imminent.
- ✅ **Fix:** Disable it or use it only for debugging errors (scoped to failed requests).

❌ 3. The Login Loop Trap

- **The Sin:** Putting the "Login API" inside the main loop alongside "Search" and "Add to Cart".
- **The Consequence:** You end up testing the Authentication Server 100x more than the App Server. You essentially DDoS your Identity Provider.
- ✅ **Fix:** Use the **"Login Once, Run Many"** pattern (e.g., using a `Once Only Controller` or a separate `setUp Thread Group`).

❌ 4. Zero Think Time

- **The Sin:** Firing requests as fast as the machine allows (0ms delay).
- **The Consequence:** Unrealistic throughput. You might crush the server with 100 users, whereas in reality, it handles 10,000 real humans fine.

- ✅ **Fix:** Always use **Timers** (Uniform Random Timer) to mimic human pauses.

❌ 5. Hardcoding Data

- **The Sin:** using `ID=12345` for every request.
- **The Consequence:** The Database caches the result for ID 12345. Your test runs lightning fast because it's hitting the cache, not the disk.
- ✅ **Fix:** Use **CSV Data Set Config** to query random, distinct records.

❌ 6. Running Load from the Same Network

- **The Sin:** Testing your Production Cloud Server from your Office WiFi.
- **The Consequence:** You are testing your Office WiFi bandwidth, not the server's capacity.
- ✅ **Fix:** Use **Azure Load Testing** or cloud VMs to generate load *from* the cloud *to* the cloud.

❌ 7. Ignoring Client-Side Health

- **The Sin:** Seeing high response times and immediately blaming the server.
- **The Consequence:** Maybe your JMeter machine reached 100% CPU?
- ✅ **Fix:** Monitor the load generator's health. If JMeter CPU > 80%, your results are invalid.

2️⃣ The "Golden Checklist" Before Execution ✅

Before you press "Start" on a big test, verify this list:

1. **[ ] Functionality:** Did I run a smoke test (1 user) to verify the script works?
2. **[ ] Clean Up:** Are all listeners (View Results Tree) disabled?
3. **[ ] Data:** Do I have enough rows in my CSV file for the number of users?
4. **[ ] Parameterization:** Did I remove all hardcoded tokens/IDs?
5. **[ ] Network:** Am I whitelisted? (Will the WAF/Firewall block me?)
6. **[ ] Monitoring:** Is the monitoring dashboard (Azure Monitor/Dynatrace) open?

3️⃣ Scaling: Distributed Testing

When a single machine isn't enough (e.g., you need 50,000 users), you use **Distributed Testing**.

- **Master (Controller):** Tells the slaves what to do and collects results.
- **Slaves (Injectors):** The machines that actually send the traffic.
- **Azure Load Testing:** Handles this complexity for you automatically.

4️⃣ Final Advice for the Interview 🎓

If they ask: **"What was your biggest challenge in performance testing?"**

Good Answer:
"The biggest challenge was **data management**. Generating unique, valid data for 10,000 users (like unique order IDs or available stock) is difficult. I solved it by creating SQL scripts to pre-seed the database before the test and using CSV files to ensure every virtual user had isolated data."

🏁 Course Completion

Congratulations! 🚀
You have gone from "What is an API?" to "Designing Enterprise-Scale Load Tests."

Your Next Steps:

1. **Build:** Create the Mock API and JMX script we discussed.
2. **Break:** Intentionally break the script (wrong assertions, no headers) to see how it looks.
3. **Run:** Execute it via CLI and generate the HTML report.
4. **Automate:** Try to trigger it via a simple script or Azure DevOps task.

You are now ready for the project.