Apache JMeter Öğrenme (Learning Apache JMeter)

1. Giriş

Bugünkü odağım: **Apache JMeter nedir, nasıl kullanılır ve performans testlerinde nerede durur?** JMeter, performans testlerinde en yaygın kullanılan açık kaynak araçlardan biri. Ücretsiz, esnek ve eklentilerle büyüyebiliyor. Sadece API değil; **web uygulamaları** ve **veritabanları** (JDBC) üzerinde de yük testi çalıştırabiliyorum.

Gerçek kullanım alanları:

- **Load Test** (normal kullanıcı yükü)
- **Stress Test** (kapasite üstü yük)
- **Endurance Test** (uzun süre sabit yük)
- **Peak Test** (ani yük dalgası)
- **Scalability Test** (kaynak artınca performans tepkisi)

> Not: Huawei Cloud'da pratik yapmadım; **CPTS tarafını teorik** inceledim. JMeter'ı **açık/free API'lerle** yerel ortamda test ettim.

---

2. Apache JMeter Nedir? (What is Apache JMeter?)

- **Tanım:** Java tabanlı, açık kaynak **performans & yük testi** aracı.
- **Öne çıkanlar:**
  - **GUI** (kolay kurgu), **Non-GUI/CLI** (büyük testler için)
  - **Plugin ekosistemi** (raporlama, thread modelleri, ekstra sampler'lar)
  - **Çoklu protokol:** HTTP(S), JDBC, FTP, gRPC (plugin), vb.
- **Kullanım alanları:**
  - **API Testing:** REST/SOAP
  - **Web App Testing:** UI/HTTP seviyesinde
  - **Database Testing:** JDBC ile SQL performansı

---

3. Kurulum & İlk Çalıştırma (Installing JMeter)

- **Gereksinim:** Java (JDK).
- **Kurulum:**
  1. Java'yı kur ve doğrula:

```
java -version
```

2. Apache JMeter'i indir → klasöre çıkar.
3. Çalıştır:
    - Linux/Mac: `bin/jmeter`
    - Windows: `bin\jmeter.bat`

- **Arayüz ana parçaları:**
    - **Test Plan:** Tüm senaryonun çatısı
    - **Thread Group:** Sanal kullanıcı, ramp-up, döngü sayısı
    - **Sampler:** Ne isteği atacağım (HTTP, JDBC, vb.)
    - **Listener:** Sonuç/rapor bileşenleri
    - **Assertions:** Yanıt doğrulama

---

4. Bugün Yaptığım Pratikler (Free API'lerle)

Aşağıdaki denemeleri **kendi makinemde**, **açık API'lerle** yaptım. Amacım JMeter'ın iş akışını ve temel bileşenleri "elle hissetmek"ti. **Ağır yük vermeden**, saygılı oranlarla test ettim (etik notlar aşağıda).

4.1 Deneme #1 — JSONPlaceholder ile Basit GET

- **Hedef:** `https://jsonplaceholder.typicode.com/posts`
- **Kurgu (tree):** `Test Plan` → `Thread Group (10 kullanıcı, 10s ramp-up, 1 loop)` → `HTTP Request Defaults (Base URL: https://jsonplaceholder.typicode.com)` → `HTTP Request (GET /posts)` → `View Results Tree` + `Summary Report`
- **Assertion:** Response Code = **200**, JSON'da `userId` alanı görünüyor mu (JSON Assertion).
- **Kazandıklarım:** Listener'larda **Response Time**, **Throughput** ve **Error %** okumayı tekrar ettim.

4.2 Deneme #2 — httpbin ile POST + Header + Body

- **Hedef:** `https://httpbin.org/post`
- **Kurgu:** `HTTP Header Manager (Content-Type: application/json)` `HTTP Request (POST /post, raw JSON body: {"name": "harun"})`
- **Assertion:** Dönen JSON'da `"json.name" == "harun"` (JSON Assertion veya **JSR223 Assertion (Groovy)** ile kontrol).
- **Kazandıklarım:** Header/body parametreleme ve **Request/Response** inceleme akışı.

4.3 Deneme #3 — reqres.in ile Login + Token'ı Çekip Kullanma

- **Flow:**
    1. `POST /api/login` → token dönüyor.
    2. Token'ı **JSON Extractor** ile yakala: `$.token` → `${authToken}`
    3. `HTTP Header Manager` → `Authorization: Bearer ${authToken}`
    4. `GET /api/users/2` isteğinde **Authorization** kullan.
- **Kazandıklarım: Correlation** (dinamik değer yakalayıp sonraki isteklerde kullanma) ve **Header Manager** ile güvenli istek akışı.

4.4 Deneme #4 — CSV ile Parametrik Senaryo + Think Time

- **CSV Data Set Config:** `data/users.csv` (ör. email, password)
- **Timer:** `Constant Timer (500 ms)` → her istek arası düşünme süresi (think time)
- **Kazandıklarım:** Parametrik veri, daha gerçekçi kullanıcı davranışı.

> Not: Bu denemelerde **yükü düşük tuttum** (ör. 10–20 kullanıcı, kısa süre), çünkü bu API'ler herkese açık. Aşağıda "Etik ve Sınırlar" bölümüne not düştüm.

---

5. JMeter Senaryosu Tasarlarken En İyi Pratikler

- **HTTP Request Defaults:** Base URL ve ortak ayarları tek yerde tut.
- **HTTP Header Manager:** `Content-Type`, `Authorization` gibi başlıkları merkezi yönet.
- **Cookie/Cache Manager:** Gerçek tarayıcı davranışına yaklaşmak için ekle.
- **Extractor'lar:**
  - **JSON Extractor** (token, id vb.)
  - **RegExp Extractor** (gerekirse metin yakalama)
- **Assertions:** Metin, JSON alanı, response code.
- **Listeners:** Geliştirme sırasında `View Results Tree` faydalı; **büyük testlerde** kapat (ağırdır).
- **Non-GUI Çalıştır:** Büyük testleri CLI'dan koştur (aşağıda).
- **Timer/Throughput:** `Constant Timer`, `Uniform Random Timer`, `Constant Throughput Timer` ile akışı gerçekçi tut.
- **Temizlik:** Büyük koşularda **ağır listener** ve **response save** kapalı olsun; sadece **Simple Data Writer** ile `.jtl` çıkar.

---

6. JMeter ile Test Türlerini Kurguya Çevirmek

- **Load Test (örn.)**
  - `Concurrency Thread Group` (plugin) veya klasik `Thread Group`
  - **50–100 VU**, **yumuşak ramp-up**, süre bazlı çalışma (`Scheduler`)
  - İsteğe bağlı: `Constant Throughput Timer` ile sabit RPS hedefi
- **Stress Test**
  - Kademeli artır (Stepping Thread Group / Ultimate Thread Group plugin)
  - Üst limitleri yokla, hata eşiğini gözle
- **Endurance Test**
  - **Uzun süre** (1–2 saat) sabit/orta yük
  - **Memory/CPU** gözlem (sistem tarafı metrikleri izlemek şart)

- **Peak Test**
  - **Kısa sürede** yüksek VU dalgası
  - Ani artışa tepki ölç
- **Scalability Test**
  - Kaynak artırımı (yatay/dikey) sonrası aynı senaryoyu yeniden çalıştır; metrik farkını kıyasla

> Bu hafta **yüksek ölçek** koşmadım; hedefim **mekanizmayı öğrenmek** ve **doğru kurgu** yapmaktı.

---

7. Raporlama & CLI (Non-GUI) Çalıştırma

- **CLI koşu (Non-GUI):**

```
# test.jmx planını koş, sonuçları results.jtl'e yaz
jmeter -n -t test.jmx -l results.jtl
```

- **HTML Dashboard üretimi:**

```
jmeter -g results.jtl -e -o ./report
```

`./report` klasöründe **Response Time (avg/p90/p95)**, **Throughput**, **Errors** gibi grafikler oluşur.

- **Temel metrikler:**
  - **Response Time:** Ortalama, p90/p95 gecikmeler
  - **Throughput:** Zaman başına istek sayısı (req/s)
  - **Error Rate:** Hata yüzdesi

---

8. Etik ve Sınırlar (Önemli)

- **Sadece izinli/size ait sistemlerde** yüksek yük uygulayın.
- Açık/free API'lerde **çok düşük concurrency** ile test yapın (nazik olun).
- **Think time** ekleyin; DoS etkisi yaratmayın.
- Servis şartları/limitleri (rate limit) varsa **saygılı davranın**.

---

9. Kapanış

Bugün JMeter'ı **teoriden pratiğe** taşıdım: kurulumu yaptım, arayüzde test planı kurguladım, **JSONPlaceholder**, **httpbin**, **reqres.in** gibi **ücretsiz API'lerle** küçük senaryolar koştum.

- **Header/Body/Assertion** yönetimini tekrar ettim.
- **JSON Extractor** ile **token yakalama ve header'a enjekte etme** (correlation) pratiği yaptım.
- **CSV Data Set** ve **Timer** kullanarak daha gerçekçi kullanıcı davranışını simüle ettim.

- Büyük koşular için **Non-GUI** ile raporlama akışını öğrendim.

> Huawei Cloud / CPTS tarafında **pratik yapmadım**; şu an için **teorik bilgi** seviyesindeyim. JMeter ile temelim oturdu; ileride **CI/CD** içinde otomatik performans testleri ve **CPTS** ile bulut tabanlı senaryolara geçebilirim.
