Rest Assured ile Java/Maven/jUnit4 — Otomatik Servis Testleri (Proje Sunumu)

Bu sunum, Rest Assured kütüphanesi kullanarak Java/Maven/jUnit4 ile hazırlanmış bir servis testi projesini özetler. Hedef: akademik jüriye test yaklaşımı, örnek GET/POST testleri, doğrulama kriterleri ve yapay zeka destekli test mühendisliği uygulamalarını kısa, net ve teknik olarak sunmak.

Özet ve Amaç

Amaç: Otomatik regresyon testi altyapısı sunmak; en az bir GET ve bir POST testi ile status code, response body doğrulamaları ve süre kontrollerini göstermek.

Kapsam: Java + Maven proje yapısı, jUnit4 test runner, Rest Assured test kod örnekleri, CI ile entegrasyon önerisi ve sonuç raporlaması.

Teknik Mimari ve Proje Yapısı

1

Proje Yapısı

standard Maven layout: src/main , src/test/java (test sınıfları), pom.xml ile bağımlılıklar (rest-assured, junit, json-path, hamcrest).

2

Continuous Integration

kodun her update'de otomatik olarak test edilmesi için şu yapıldı:

- **GitHub Actions Entegrasyonu:** Proje kök dizinine eklenen bir `.github/workflows/main.yml` dosyası ile GitHub üzerinde bir "Pipeline" oluşturulur.

3

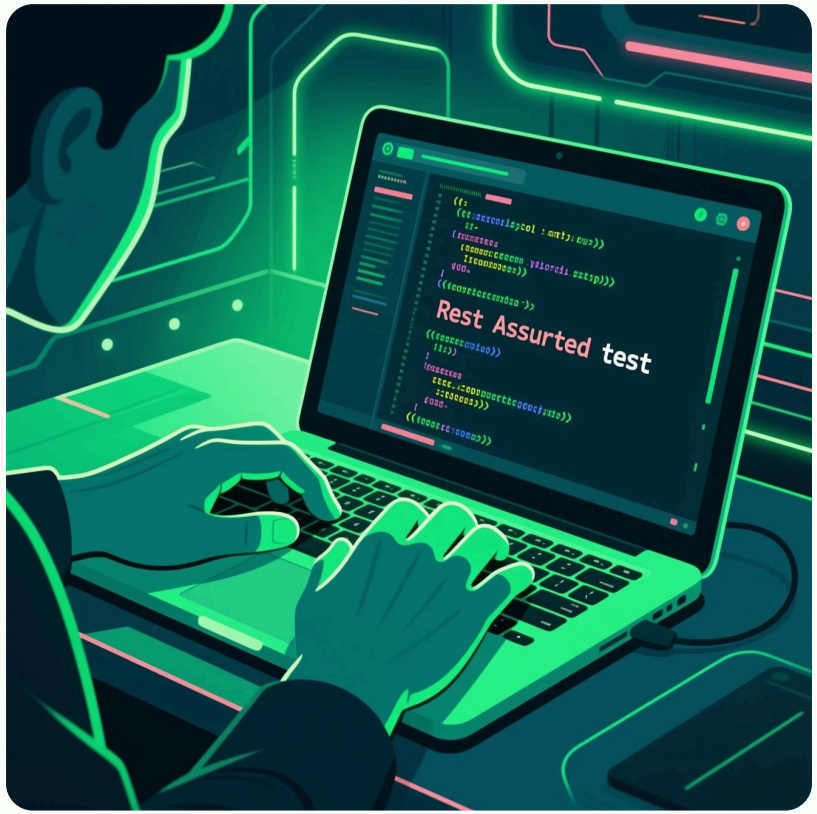
Test Runner

jUnit4 ile Test Metotları: Her bir servis testi (GET ve POST), `@Test` ile isaretlenip birbirinden bağımsız ve otomatik olarak calistirilir hale getirilmistir.

Raporlama ve Loglama: Testlerin sonunda `log().all()` komutu kullanılarak isteğin ve cevabın tüm detayları konsola yazdırılmakta, böylece manuel bir raporlama adımı yerine otomatik regresyon testi çıktıları elde edilmektedir.

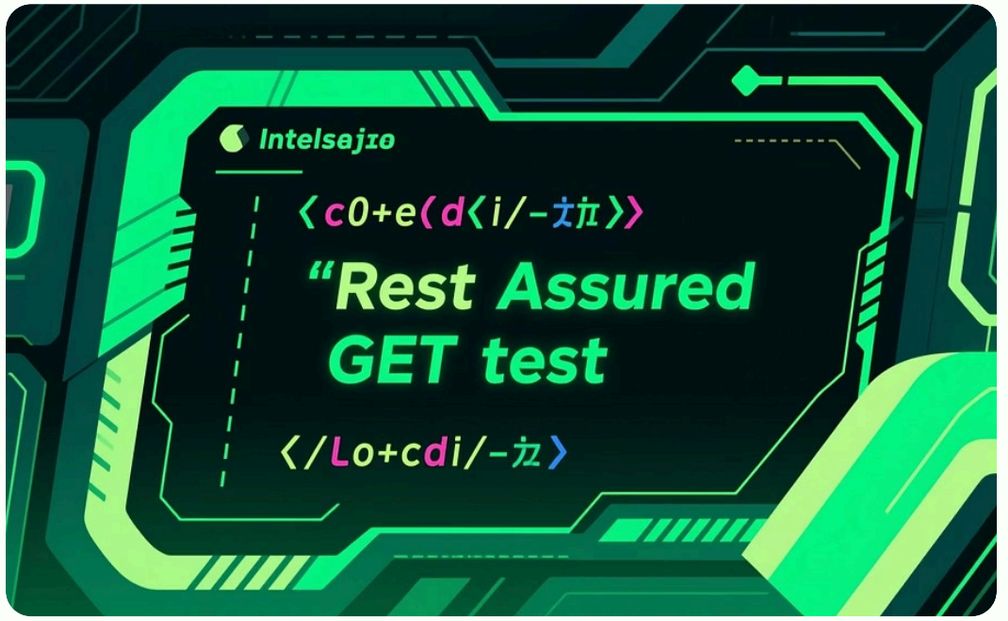
Rest Assured Kullanımı — Temel Prensipler

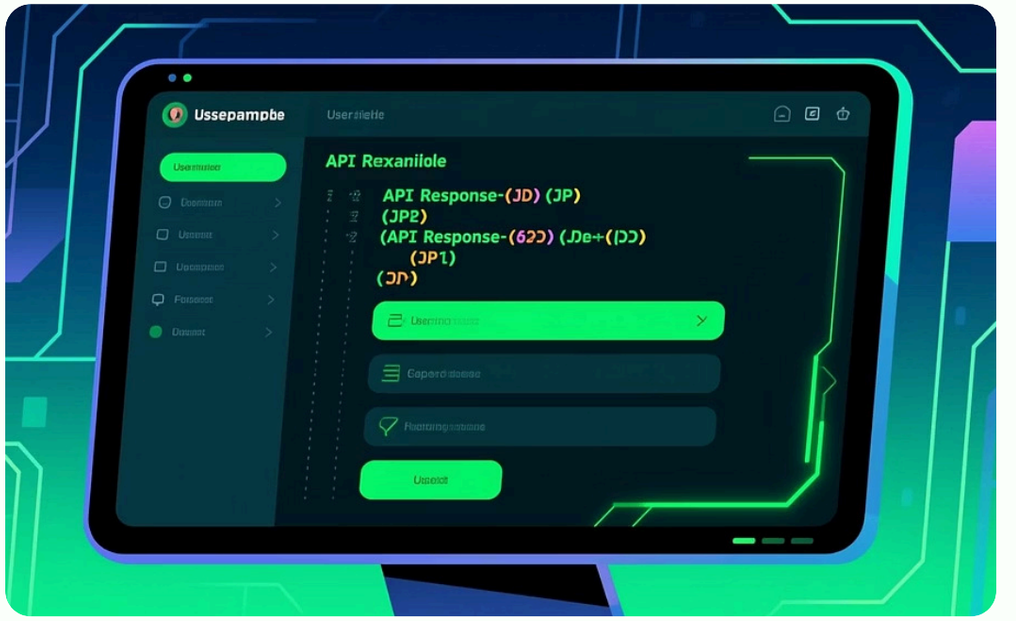
- **Temel Akış (BDD Yapısı):** `given()` (ön koşullar) – `when()` (aksiyon) – `then()` (doğrulama) zinciri kullanılarak test adımları standardize edilmiştir.
- **İstek Yönetimi:** Servis çağrılarında gerekli olan Header yönetimi, Query Parametreleri(enlem/boylam) ve POST istekleri için Request Body gönderimi bu yapı üzerinden yönetilir.
- **Status Code Doğrulama:** Servis yanıtının beklenen başarı kodunu döndüğü `.then().statusCode(200)` (veya POST için 201) komutuyla teyit edilir.
- **Body (İçerik) Doğrulama:** Yanıt içerisindeki verilerin doğruluğu, `.body("list[0].main.aqi", equalTo(deger))` yöntemiyle kontrol edilir.
- **Performans (Süre) Kontrolü:** Regresyon testlerinin bir parçası olarak servisin hızı `.time(lessThan(3000L), TimeUnit.MILLISECONDS)` ifadesiyle denetlenerek performans kriterlerine uyum sağlanır.



Örnek 1 — GET İsteği

Senaryo: Belirli Koordinatlar Için Hava Kirliliği Verisi Sorgulama. Doğrulamalar: 200 OK, body içinde beklenen kullanıcıdan beklenen maddeler var.





GET Test (kısaltılmış)

@Test public void havaKalitesiVeriKontrolu() { given() .queryParam("lat", "41.00").queryParam("lon", "28.97") .queryParam("appid", "API_ANAHTARINIZ") .when() .get("/data/2.5/air_pollution") .then() .statusCode(200) // Status code kontrolü .body("list[0].main.aqi", notNullValue()) // Body değer kontrolü .time(lessThan(2000L)) // 2 saniye altı cevap kontrolü .log().selection(); // Sadece önemli kısımları yazdırır }

Beklenen Alanlar

*Status Code 200

*Hava Kalitesi Indeksi: Null olmayan, sayısal bir değer.

***Yanıt Süresi (Response Time):** 3000ms altı

***Koordinat Dogˇrulaması:** Sorgulanan enlem ve boylam bilgilerinin uyumu.

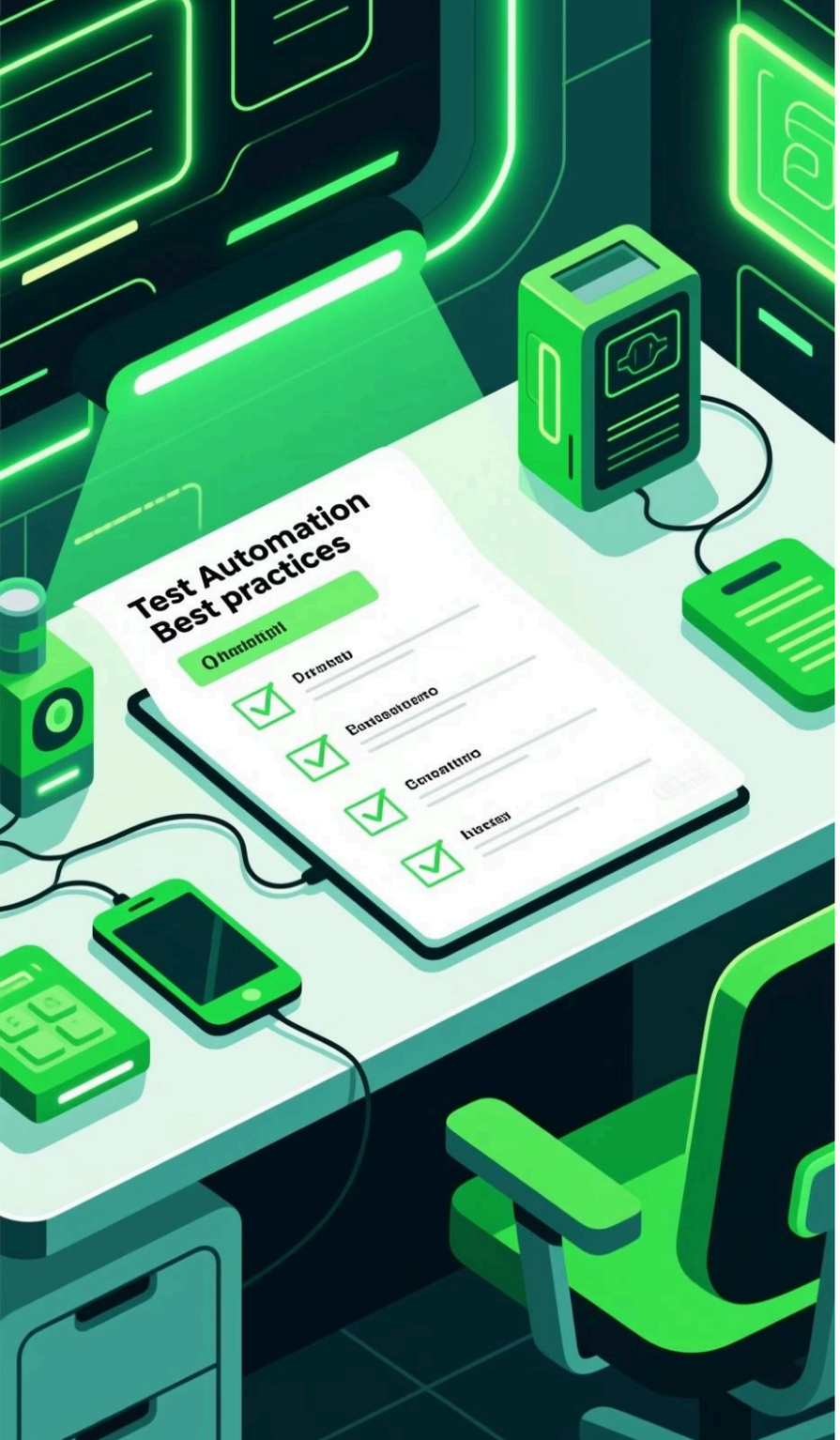
Örnek 2 — POST İsteği (Request Body Dahil)

Senaryo: eni Hava Kalitesi Raporu Oluşturma. Doğrulamalar: 201 Created, response body'de gönderilen verinin karşılığı, cevap süresi 3oooms altında.

POST Test (kısaltılmış)

```
@Test public void postHavaRaporu_Senaryosu() { // 1. Gönderilecek verinin (Request Body) hazırlanması String jsonBody = "{"city": "Istanbul", "aqi_level": 2}";

RestAssured.given()
    .body(jsonBody) // Request Body kullanımı
.when()
.post("https://jsonplaceholder.typicode.com/posts") // POST isteğinin yapıldığı satır
.then()
    .statusCode(201) // Status code kontrolü (201 Created) [cite: 6]
    .body("city", equalTo("Istanbul")) // Body değer kontrolü
    .time(lessThan(3000L))
    .log().all();
}
```



Doğrulama Stratejisi ve Test Pratikleri

1. Kritik Yol Testleri

API'nın temel işlevleri (auth, CRUD) için smoke ve regresyon testleri oluşturun.

2. Veri Fabrikaları ve Fixture'lar

Tekrarlanabilir test verileri için factory pattern veya JSON fixture kullanılmalı.

3. Performans Eşiği

Yanıt süreleri SLA'lara göre test edilip uyarı eşiği belirlenir.

4. Raporlama

JUnit raporları, Allure entegrasyonu veya HTML raporlar ile sonuçlar saklanmalı.

Yapay Zeka Destekli Yazılım Test Mühendisliği

Teorik Çerçeve: AI, test tasarımını hızlandırır, anomalileri tespit eder ve test kapsamı önerileri sunar. ML tabanlı analizlerle flaky testlerin kök neden analizi yapılabilir.

- Test veri üretimi için generative modeller
- Regression test prioritization (önceliklendirme)
- Anormal davranış tespiti için anomaly detection

AI Destekli Test: Sınırlar, Riskler ve Etik Kullanım

1

AI Yöntemlerinin Sınırları

Yapay zeka modelleri, istatistiksel olasılıklara dayalı sonuçlar üretir, bu da onların "mantıksal akıl yürütme" yerine "örüntü eşleştirme" yapmasına neden olur.

- **Bağlamsal Hatalar:** İş mantığını yanlış anlayarak işlevsel olarak hatalı kodlar üretebilir.
- **Halüsinasyon Riski:** Var olmayan API parametrelerini veya kütüphane metodlarını gerçekmiş gibi sunabilir.
- **Veri Kısıtı:** Modeller, sadece eğitildikleri veri setindeki bilgilerle sınırlıdır; güncel bilgiye sahip olmayabilirler.

2

Doğruluk ve Risk Dengesi

Yazılım testinde AI kullanımı, hız (speed) ve kesinlik (precision) arasında bir denge gerektirir.

- **Hız Avantajı:** Saniyeler içinde onlarca test senaryosu ve "request body" taslağı oluşturabilir.
- **Doğruluk Riski:** "False positive" (hatayı fark etmeme) veya "false negative" (hatasız durumu hata sanma) üretme riski vardır.
- **Akademik Yaklaşım:** Tam otomasyon yerine "insan denetimli otomasyon" (Human-in-the-loop) yöntemi tercih edilmelidir.

3

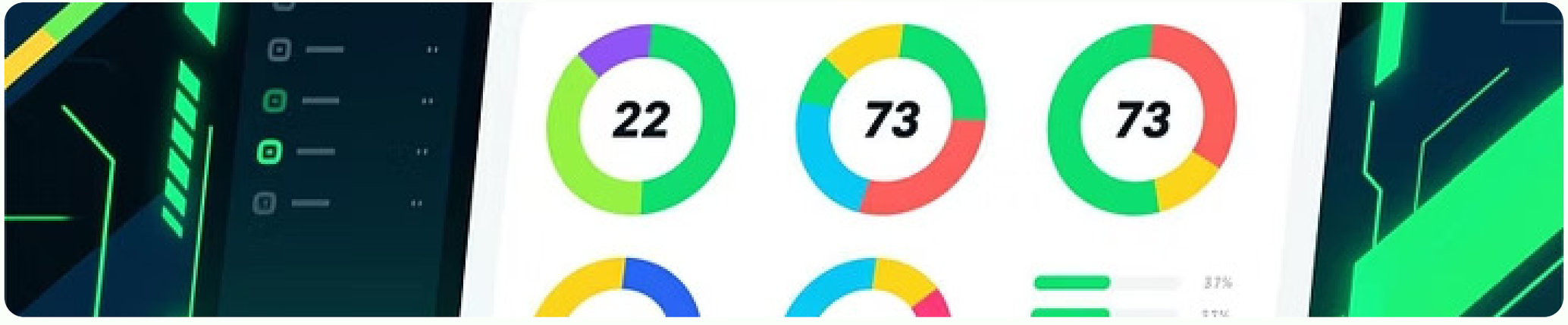
Etik Kullanım ve Akademik Dürüstlük

Sunumdaki "kodun nasıl yazıldığını sunma" ve "güzel hazırlanmış olması" maddeleri akademik dürüstlükle doğrudan ilişkilidir.

- **Şeffaflık:** Kodun veya sunumun hangi bölümlerinde AI'dan destek alındığı açıkça belirtilmelidir.
- **Fikri Mülkiyet:** AI'nın ürettiği kodun açık kaynak lisanslarına uyumu ve veri gizliliği esastır.
- **Sorumluluk:** Sunulan kodun ve test raporunun doğruluğundan AI değil, öğrencinin kendisi sorumludur.

Canlı Sunumda Gösterilecekler — Demo Akışı

1. Proje dizin yapısının gösterimi ve pom.xml bağımlılıkları
2. IDE üzerinde GET testinin çalıştırılması, assertion ve süre doğrulaması
3. POST testinin çalıştırılması (request body gösterimi) ve response doğrulamaları
4. Test raporu ve CI çıktısı (GitHub Actions örneği)
5. AI destekli test önerileri ve kısa demo (örnek: test önceliklendirme önerisi)



Sonuçlar, Teslim ve İleri Adımlar

Repository

Cihangir1/
litesiServisTesti

erilerini sağlayan bir servisin test
ıyla hazırlanmış örnek proje.

0 Issues 0 Stars 0 Forks

GitHub

GitHub – HuseyinCihangir1/Hava...

Hava kalitesi verilerini sağlayan bir servisin test edilmesi amacıyla...