o load -> aynı makinede çok fazla örnek açarsak, makineyi aşırı yükleyebilir. çalışan testlerle sonuçlanacak yavaşlar ve başarısızlık oranını artırır.

o bu, GRID kullanan farklı makinelerde testler yürüterek ele alınabilir.

o projemde bazı test durumları paralel olarak çalışmadı.

Sayfa 89

API

1. API nedir?

- Bağlantı anlamına gelir. Demek istediğim, API istekleri alan ve bir sisteme ne yapmak istediğinizi söyleyen ve sonra yanıtı size geri döndürür.
- API, **Uygulama Programlama Arayüzünün** (yazılım aracısı olan) kısaltmasıdır. birbirleriyle konuşmak için uygulamalar.

2. API ve Web Hizmetleri mi?

- API = *tarayıcı* : Selenium WebDriver, *veritabanı* : JDBC, *MsOffice* : Apache POI
- Web hizmetleri = bir API iletişim için İnternet kullanıyorsa, bu bir web hizmetidir. * Tüm web hizmetleri API'dir.
- UI yok (kullanıcı arayüzü) → UI ile web uygulaması ve Selenium Webdriver kullanıyoruz
- Kullanıyoruz:
  - SABUN → XML
  - REST → JSON, XML, METİN
  - Postacı, Huzurlu Kütüphane

3. SoapUI nedir? ve bunu mevcut projenizde nasıl kullandınız?

- SOAP UI, önde gelen açık kaynaklı platformlar arası API Test aracıdır
- SOAPUI, test uzmanlarının farklı Web API'sinde otomatikleştirilmiş işlevsellik, regresyon, uyumluluk ve yük testleri yürütmesine olanak tanır.
- SOAPUI, her tür API'yi test etmek için tüm standart protokolleri ve teknolojileri destekler.
- SOAPUI arayüzü, hem teknik hem de teknik olmayan kullanıcıların sorunsuz bir şekilde kullanmasını sağlayan basittir.

4. REST tabanlı mimaride yaygın olarak kullanılan bazı HTTP yöntemlerinin adı?

- Oluştur → POST (sunucuya veri gönder)
- Oku → GET (belirli bir URI kullanarak belirli bir sunucudan veri alır)
- Güncelle → PUT (Hedef kaynağın tüm mevcut temsillerini yüklenen içerikle değiştirir)
- Sil → SİL (Bir URI tarafından verilen hedef kaynağın tüm mevcut temsillerini kaldırır.)

5. HTML Durum Kodları?

- 1xx → Bilgilendirici
- 2xx → Başarılı (istek başarıyla kabul edildi) (200 → Tamam, 201 → Oluşturuldu, 202 → Kabul Edildi, 204 → İçerik Yok)
- 3xx → Yönlendirme
- 4xx → İstemci Hatası (400-Hatalı İstek, 401-Yetkisiz, 403-Yasak, 404-Bulunamadı, 405-Yönteme İzin Verilmiyor)
- 5xx → Sunucu Hatası (500-Dahili sunucu Hatası, 502-Hatalı Ağ Geçidi, 501-Uygulanmadı, 503-Hizmet Kullanılamıyor)

6. Yanıt aldığınızda ilk kontrol ettiğiniz şey nedir?

- Durum teklifi (200 her zaman Tamam anlamına gelir)
- Her zaman 404'ün bulunamadı anlamına geldiğini kontrol ederiz
- rest-assured.io ==> ECS makinesini uzak Masaüstü arama türünde bulmaya yönelik otomasyon için

7. HTTP yöntemleri ve istek türleri

- Get vücut gerektirmez
- Yerleştirme gerekli gövde, GÜNCELLEME bilgisi anlamına gelir
- Gönderi gerekli gövde, CREATE bilgi anlamına gelir
- Silme , gövde gerektirmez
- GET -> OKU, YAYINLA -> OLUŞTUR, PUT -> GÜNCELLEME, SİL -> SİL
- POST VS PUT

Sayfa 90

8. Parametreler api

- 2 TİP:
  - YOL PARAMETRESİ (DEĞER URL'NİN BİR PARÇASI OLACAKTIR) SORGU / TALEP
  - PARAMETRELER (ANAHTAR + DEĞER FORMATI)

9. Hamcrest Matcher ne içindir?

- Hamcrest, 'eşleşme' kurallarının bildirimsel olarak tanımlanmasına izin veren eşleştirici nesneler yazmak için bir çerçevedir.

```
import org.junit.jupiter.api.Test ;
statik kuruluşu içe aktar . hamcrest . MatcherAssert . assertThat ;
statik org içe aktar . hamcrest . Eşleştiriciler . *;

public class BiscuitTest {
    @Ölçek
    public void testEquals () {
        Biscuit theBiscuit = new Biscuit ( "Ginger" );
        Biscuit myBiscuit = new Biscuit ( "Ginger" );
        assertThat ( theBiscuit , equTo ( myBiscuit ));
      assertThat ( "çikolata parçaları" , theBiscuit . getChocolateChipCount (), equTo ( 10 ));
      assertThat ( "fındık" , theBiscuit . getHazelnutCount (), equTo (3));
    }
}
```

```
// ilk bağımsız değişkenin ikinciye eşit olup olmadığını doğrulayın
assertThat ( str1 , is ( "Kunkka" )) ;
assertThat (str1 , olduğu (str2)) ;

// ilk bağımsız değişkenin ikinciye eşit OLMADIĞINI doğrulayın
assertThat (str1 , bir ( değil ( "Tidehunter" ))) ;

// büyük / küçük harfleri görmezden gelin
assertThat ( str1 , equToIgnoringCase ( "kunkka" )) ;

// öncesi ve sonrası boşluğu yok saymayı karşılaştırın
assertThat ( str1 , equToIgnoringWhiteSpace ( "Kunkka" )) ;

// sayıları karşılaştır
assertThat ( 10 , daha büyükThan ( 9 )) ;
assertThat ( 10 , lessthan ( 11 )) ;
assertThat ( 10 , lessThanOrEqualTo ( 11 )) ;

// boş olmadığını doğrula
assertThat ( str1 , notNullValue ()) ;

List <String> list = Diziler. asList ( "bir" , "çok" , "ağaç" ) ;
    assertThat (list , hasSize ( 3 )) ;
    assertThat (list , containsInAnyOrder ( "çok" , "ağaç" , "bir" )) ;
    assertThat (liste , hasItems ( "bir" , "çok" ) ) ;

<Tamsayı> sayıları listele = Diziler. asList ( 11 , 12 , 13 ) ;
assertThat (sayılar , her öğe (daha büyükThan ( 9 ))) ;
```


Sayfa 91

10. RestAssured Günlük Kayıt Günlükleri

- **Günlük Kaydı İste**

```
given (). log (). all () // Parametreler, başlıklar ve gövde dahil tüm istek belirtimi ayrıntılarını günlüğe kaydedin
given (). log (). params () // Yalnızca isteğin parametrelerini günlüğe kaydedin
given (). log (). body () // Yalnızca istek gövdesini günlüğe kaydedin
given (). log (). headers () // Yalnızca istek başlıklarını günlüğe kaydedin
given (). log (). cookies () // Yalnızca istek çerezlerini günlüğe kaydedin
given (). log (). method () // Yalnızca istek yöntemini günlüğe kaydedin
given (). log (). yol () // Yalnızca istek yolunu günlüğe kaydedin
```

- **Yanıt Günlüğü**

```
get ("/ x"). sonra (). log (). body ()
get ("/ x"). sonra (). log (). ifError ()
get ("/ x"). sonra (). log (). tümü ()
get ("/ x"). sonra (). log (). statusLine () // Yalnızca durum satırını günlüğe kaydet
get ("/ x"). sonra (). log (). headers () // Yalnızca yanıt başlıklarını günlüğe kaydet
get ("/ x"). sonra (). log (). cookies () // Yalnızca yanıt çerezlerini günlüğe kaydet
get ("/ x"). sonra (). log (). ifStatusCodeIsEqualTo (302)
      // Yalnızca durum kodu 302'ye eşitse günlüğe kaydedin
get ("/ x"). sonra (). log (). ifStatusCodeMatches (eşleştirici)
      // Yalnızca durum kodu sağlanan Hamcrest eşleştiriciyle eşleşirse günlüğe kaydedin
```

11. Serileştirme ve Seriyi Kaldırma

- Serileştirme; bir Java nesnesini API JSON formatına eşleştirdiğimizde (JAVA NESNESİNİ JSON'A DÖNÜŞTÜR);
    - Java nesnesi (POJO (Düz Eski Java Nesnesi), FASULYE) → API JSON / XML ile eşleştirin
    - Bir sınıftan bir nesneye sahip olduğumuzda ve onu RESTful API'mizde bir JSON formatına eşleştirdiğimizde

```
{make: "Toyota",
Model: "Camry"}
Araba arabası = yeni Araba ();
car.setMake ("Toyota");
car.setModel ("Camry");
verilen (). body (car) .when (). post (uri)
```

- Seriyi kaldırma; API JSON / XML → Java Nesnesine (JSON'dan JAVA NESNE) EŞLEME

```
Araba araba2 = yeni Araba ();
car2 = ne zaman (). get (uri) .body.as (car.class);
car.setMake ("Toyota");
car.setModel ("Camry");
```

Sayfa 92

12. RestAssured Kitaplığı ile API / Web Hizmetleri?

```
import static io.restassured.RestAssured. *;
URI uri = yeni URI ("... / yöntemler (alma, yayınlama)")
```

- GET;

```
Yanıt yanıtı = verildi (). Kabul et (ContentType.JSON) .when (). Get (URI);
response.then (). assertThat (). statusCode (200).
      ve (). assertThat (). ContentType (ContentType.JSON);
```

- POST;

```
Yanıt yanıtı = verildi (). ContentType (ContentType.JSON) .with (). Accept (ContentType.JSON)
. ve (). body (JSONbody) .when (). post (URI);
response.then (). assertThat (). statusCode (200);
```

-

statik org.hamcrest.Matchers. *;
sonra (). assertThat (). body ("Id", Matchers.equalTo (123));

•

JsonPath json = JsonPath (JSONbody);
json.getString ("anahtar");
json.getInt ("anahtar");
json.getList ("key1.key2");

13. EndPoint nedir?

• <protokol>: // <hizmet adı> / <Kaynak Türü> / Kaynak Kimliği → URI (Tekdüzen Kaynak Tanımlayıcı)
Temel URI / kaynak? Parametreler
(http://www.google.com/search?source=book…) →? → sorgu parametreleri

14. Yetkilendirme ve Kimlik Doğrulama

• kimlik doğrulama -> sen kimsin
• yetkilendirme -> hangi haklara sahipsiniz
• Kimlik doğrulama, kullanıcı ve paroladır
• Yetkilendirmenin türleri vardır:
  ○ Yetki yok
  ○ Temel Yetkilendirme
  ○ Taşıyıcı Jetonu
  ○ Yetkilendirmeyi ebeveynden devral

15. RESTful Web Hizmeti / API

• REST, Temsili Devlet Transferi anlamına gelir
• RESTful, REST mimari konsepti uygulanarak yazılan web servisleri için yönlendirilir.
  ○ RESTful'da, CRUD işlemlerini gerçekleştirmek için GET, POST, PUT, DELETE gibi web hizmeti http yöntemleri kullanılabilir.
  ○ CRUD = Oluştur → Oku → Güncelle → Sil

Sayfa 93

16. Yanıt bedeninizdeki bir değeri nasıl doğrularsınız?

• Exp için: kimliğin doğru numarayı içerdiğini doğrulayın
  o *Hamster Matcher* iddia kitaplığıdır.

sonra (). assertThat (). body ("Id", Matchers.equalTo (123));

  o JsonPath'e ayrıştırın ve Id değerini okumak için getInt (), getList (), getString () yöntemlerini kullanın.
Ve JUnit Assertion'ı kullanabilirim:

String body = ... thenReturn (). Body (). AsString ();
JsonPath json = new JsonPath (gövde);
assertEquals (123, json.getInt ("Id"));

  o Bir (POJO) nesnesine (veya Nesne Eşleme) serileştirme

POJO myPojo =… ne zaman (). Post (url) .thenReturn (). Body (). As (Pojo.class);
assertEquals (123, myPojo.getId ());

Ve JUnit Assertion'ı kullanabilirim.

17. API Kimlik Doğrulama Türleri

• Temel
  ○ Önleyici
    ■ Bir hizmet önleyici olacak şekilde yapılandırılmışsa, bir istemciden kimlik bilgilerini istemeyecektir.
bunu gerektirir.
    ■ Bir istek, kimlik bilgileri içermiyorsa, **401 Yetkisiz** durum kodunu döndürür .
  ○ Meydan okuma
    ■ İstek API'ye ulaştığında, API kimlik bilgileri gerektirdiğini söyleyecek ve ardından müşteri sağlayacaktır
kimlik bilgileri.
  ○ oauth -> kimlik doğrulamak için üçüncü taraftan anahtarların ve belirteçlerin kullanıldığı kimlik doğrulama türleri. Var
2 tür oauth:
    ■ oauth1 → uygulaması zor
    ■ oauth2 → daha güvenli

- sindirmek
  - Temelden daha şifrelenmiştir. https ...

18. SABUN kullanmanın avantajı nedir?

- REST, çok çeşitli veri formatlarına izin verirken, SOAP yalnızca XML'e izin verir.
- JSON ile birleştirildiğinde (genellikle verilerle daha iyi çalışır ve daha hızlı ayrıştırma sağlar), REST genellikle dikkate alınır çalışmak daha kolay.
- JSON sayesinde, REST tarayıcı istemcileri için daha iyi destek sunar.
- REST, özellikle değiştirilmemiş ve dinamik olmayan bilgileri önbelleğe alarak üstün performans sağlar
- Yahoo, Ebay, Amazon ve hatta Google gibi büyük servisler için en sık kullanılan protokoldür.
- REST genellikle daha hızlıdır ve daha az bant genişliği kullanır. Mevcut web siteleriyle entegre etmek de daha kolaydır. refactor site altyapısı. Bu, geliştiricilerin bir siteyi sıfırdan yeniden yazmak için zaman harcamak yerine daha hızlı çalışmasını sağlar. Bunun yerine, ek işlevler ekleyebilirler.

Sayfa 94

19. SOAP ve RESTful web servisleri arasındaki fark nedir?

- RESTful JSON, XML, TEXT'i destekler, ancak SOAP yalnızca XML'i destekler
- REST, SOAP tabanlı web hizmetlerinden daha hızlıdır

20. URI nedir, amaç ve biçim?

- URI, Tekdüzen Kaynak Tanımlayıcısı anlamına gelir
- URI'nin amacı, web hizmetini barındıran sunucuda bir kaynak bulmaktır.
- Bir URI aşağıdaki biçimdedir:
  - <protokol>: // <hizmet-adı> / <KaynakTürü> / <KaynakKimliği>

21. Projenizde hangi Web Hizmetlerini kullanıyorsunuz?

- Temsili Aktarım Durumu olan Restful kullanıyorum ve XML ve JSON ile iletişim kuruyor, ancak şu anki proje JSON kullanıyor

22. XML nedir?

- Bilgi işlemde, Genişletilebilir Biçimlendirme Dili (XML), kodlama için bir dizi kural tanımlayan bir biçimlendirme dilidir hem insan tarafından okunabilen hem de makine tarafından okunabilen bir biçimde belgeler.

23. JSON nedir?

- JavaScript Nesne Gösterimi (minimum, verileri yapılandırmak için okunabilir bir biçimdir.)
- XML'e alternatif olarak, öncelikle bir sunucu ile web uygulaması arasında veri iletmek için kullanılır.
- Temel olarak, XML'in basit bir sürümü
- Anahtar: Değer biçimi
- Anahtar her zaman çift tırnak içindedir ve eğer dize çift tırnaklarsa ve sayılar tırnak işareti yoksa değer
- Tamamen http protokolüne dayalıdır - bu nedenle tarayıcıdaki bağlantıya ulaşır ve sonuçları görür

24. Swagger'ı biliyor musunuz? Swagger nedir

- Swagger, geliştiricilerin tasarım yapmasına yardımcı olan geniş bir araç ekosistemiyle desteklenen açık kaynaklı bir yazılım çerçevesidir. RESTful Web hizmetlerini oluşturun, belgeleyin ve kullanın.
- Swagger, makinelerin okuyabilmesi için API'lerinizin yapısını tanımlamanıza olanak tanır.
- API'lerin kendi yapılarını tanımlama yeteneği, Swagger'daki tüm mükemmelliklerin köküdür
- xml şemasına benzer ancak Json için

25. json vs gson

- JSON, anahtar ve değerlere sahip bir biçimdir
- GSON, bir dönüştürme sürecidir
  - java'dan json'a (serileştirme),
  - json'dan java'ya (seriyi kaldırma)

26. Nasıl ve nereye talep gönderiyorsunuz?

- Rest kullandığım için uç noktaları var. Geliştiricilerim genel URL'ler oluşturur ve bu URL'ye istekler gönderilir

27. Web hizmetleri olmayan herhangi bir API kullanıyor musunuz?

- - Tarayıcı için Selenium API, veritabanı için JDBC ve API için RestAssured kullanıyorum

28. API'niz için API dokümantasyon web siteniz var mı?

- Evet, API belgelerimiz için swagger kullanıyoruz ve API uç noktalarının açıklamaları ve yönergeleri burada

Sayfa 95

29. API'yi projenizde nasıl test ediyorsunuz?

- Mevcut projemde sadece şirketlerimizin API'sini değil diğer harici API'leri de test ediyoruz.
  - Örneğin, yetkili son kullanıcının bilgilerini veritabanımıza kolayca aktarmak için LinkedIn api kullanıyoruz.
- Test kullanıcısı olarak bir API isteği gönderiyor ve api'nin durum kodunu, yanıt gövdesini ve uç noktalarını kontrol ediyoruz.
  URL beklendiği gibi çalışıyor
  - Örneğin, projemde API 57'nin Pozitif / Negatif testini de yapıyorum
- Olumlu - Geçerli istekler, başlıklar, parametreler ve Json gövdesi gönderiyorum ve yanıtın 200/201 olduğunu doğruluyorum
- Negatif - Durumun 200 olmamasını bekliyorum, geçersiz istekler, başlıklar, parametreler ve gövde gönderiyorum

30. Rest api'sini nasıl test edersiniz?

- Her REST API uç noktasının beklendiği gibi çalışıp çalışmadığını doğrularım.
- Manuel API testi için POSTMAN kullanıyorum ve otomasyon için Java'da RESTASSURED kitaplığı kullanıyorum.
- POST, PUT, GET, DELETE tür istekleri gönderiyorum ve yanıt durum kodunu ve yanıt gövdesini, başlığı doğrularım.
- API'nin pozitif ve negatif testlerini de yapıyorum.
- Pozitif test yaptığımda, geçerli istek parametreleri, geçerli başlıklar, geçerli istek json gövdesi gönderiyorum ve
  yanıt durum kodu 200 başarılı ve Json yanıt gövdesi verileri de beklenenle eşleşiyor.
- Negatif test yaptığımda, geçersiz istek parametreleri veya geçersiz başlıklar veya geçersiz istek json gövdesi gönderiyorum ve
  bu yanıtı doğrula
- durum kodu 200 değil ve Json yanıt gövdesi hata mesajı içeriyor.

31. Tüm API uç noktaları tüm Http protokollerini kullanabilir mi?

- Buna bağlıdır, API geliştiricim bu URL'nin GET, POST, PUT veya DELETE istekleriyle çalışıp çalışmayacağına karar verir

32. API'nizi manuel olarak nasıl test edersiniz?

- Postman kullanıyorum → bu, REST API URL'sini test eden bir REST API istemci aracıdır

33. API testi için hangi araçları kullanıyorsunuz?

- Manuel test için postacı
- Kendinden Emin Kitaplık

34. Rest API'deki İstek türleri nelerdir?

- Al, Gönder, Koy ve Sil istekleri vardır
  - Okunan verileri alın
  - Gönderi veri oluşturur
  - Güncelleme verilerini koy
  - Sil verileri siler

35. REST API'deki başlıklar nelerdir?

- Kabul Et (İçerik Türü.JSON) türünü kullanıyorum - aldığım şeyin JSON veya XML biçiminde olması gerektiğini kontrol eder
- Ve ContentType. (Contenttype.Json) - gönderdiğim şeyin JSON biçiminde olması gerektiğini kontrol eder

36. RestAssured Kütüphane nedir?

- BDD biçiminde olan ve seriyi kaldırma ve serileştirmeyi kullanarak java kodunu entegre etmeye yardımcı olan web dışı bir hizmet api
  verileri saklamak, doğrulamak ve doğrulamak için Json'dan verileri çıkarın ve bir java nesnesine dönüştürün.
  beklenen biri.

37. Projenizde Enum'u nasıl kullanıyorsunuz?

- Yanıt türümün JSON biçimi olduğundan emin olmak için içerik Türü kullanıyorum

Sayfa 96

38. JsonPath nedir?

- Yanıt gövdesini doğrulamanın başka bir yolu
- JsonPath j = response.jsonpath;

39. Yanıt verilerinin boyutunu doğrulamak için hangi yöntemleri kullanıyorsunuz?

- Hamcrest'ten Matchers kullanıyorum
  - hasitems ()
  - equTo ()

40. Yanıt arayüzünü nasıl kullanırım?

✪ Raporlama

- Mvn Verify başarısız olsa bile testleri çalıştırır (hatayı yok sayar)

- Tüm testin bitmesini bekler
- Başarısızlığı yok sayar, bc bu bizim derleme yapılandırmamızda var

<testFailureIgnore> doğru </testFailureIgnore>

- Doğrulama, testten sonra gelen bir Maven yaşam döngüsüdür

• Mvn testi, bir şey başarısız olursa testi çalıştırmayı durdurur
• Aldığımız orijinal html raporu o kadar iyi değil, istatistiksel verilere ihtiyacımız var
• Örn; "html: hedef / salatalık-raporu" → Kaç testin başarılı / başarısız yüzdeleri olduğu gibi
• Daha fazla istatistiksel veri raporlama için *Cucumber Sandwich'i* (bu, pom xml'de bir bağımlılık dosyasıdır) kullanacağız
• cukesrunner'da "json: target / cucumber.json" ekleyin

- Bu, JSON dosyasından alınan bir html raporu → Bu rapor nasıl çalışır? Json dosyası, raporu oluşturmak için kullanılır
- Sürüm 3.15 (vid'dan)
- pom'a yeni bir yapı xml ekleyin (zaten pom dosyanızda, TestProject adında)
  - Yalnızca bu json raporlamasını alacaksınız (grafikler ve istatistikler içeren salatalık raporu. Bu rapor,

    Yalnızca SİZİN görebileceğiniz yerel olun, Jenkins için değil) YALNIZCA MVN Doğrulaması'nı çalıştırırsanız
  - ANCAK ÇALIŞTIKTAN SONRA DAİMA JSON DOSYASI (salatalık raporundan farklı) ALACAKSINIZ

    TEST, BİLE DOĞRULAYACAK
    - Bu JSON dosyası, salatalık raporu eklentisi için Jenkins için çok önemlidir

• TestProject derlemesi:

- <id> yürütme </id>
- <phase> doğrula </phase> - bu nedenle html (json) raporu yalnızca doğrulama kullanılırken oluşturulur
- <hedefler>
- <goal> oluştur </goal> </goals>
- Rapor ayrıca size bir json dosyası verecektir
- Doğrulamayı kullanarak testleri çalıştırmak için, pom dosyasına sağ tıklayın ve maven oluştur seçeneğine tıklayın…
  - Ayrıca parametreler de ekleyebilirsiniz (koşucu değişkeni ve xml dosyası olan değer gibi) - Hedefleri yazın: doğrulayın

• Bunu komut satırında çalıştırmak için

- Pom dosyasının konumuna gidin ve mvn doğrula yazın
- Sözdizimi mvn <yaşam döngüsü / hedef> şeklindedir

• mvn doğrulaması kullanılarak Yürütme Sırası

1. Pom dosyasına karşı koşun
2. Pom dosyası xml dosyasını çalıştırır
3. Xml, cukesrunner dosyasını çalıştırır
4. Cukesrunner, salatalık özellik dosyasını / testini çalıştırır

• json salatalık raporu ekran görüntüsü gösteriyor mu?

Sayfa 97

41. 100 limitli parametreye ve çalışan id = 100 yol parametresine ihtiyacım olan bir yöntemi nasıl yazabilirim?

• Yazardım;

- . Ve (). Params ("limit", 100)
- . Ve (). PathParams ("employee_id", 110)

42. Backend-API nedir?

• Uygulama mantık kodunun bulunduğu yerdir. Koşullarınız vb.
• Nasıl test edilir?

1) El ile → Postacı vb. Araçları kullanma İstekler göndererek ve yanıtları doğrulayarak.
2) Otomasyon → Java + RestAssured Kitaplığı

43. HTTP İsteği ve HTTP Yanıtı nedir?

HTTP istek yöntemi dört bileşenden oluşur:

- **İstek Yöntemi** → Al, Gönder, Koy, Sil (bunlar yaygın olanlardır)
- **URI talep et** → kaynağın URL'si
- **Üstbilgi İste** → Dil Kabul Et, Kodlama Kabul Et, Kullanıcı Aracısı, Ana Bilgisayar
- **Talep Gövdesi** → bu kaynağa gönderilecek veridir

HTTP yanıt yöntemi üç bileşenden oluşur:

- **Yanıt Durum Kodu** → 200, 301, 404, 500 (bunlar en yaygın olanlardır)
- **Yanıt Başlığı Alanları** → Tarih, Sunucu, Son Değiştirilen, İçerik Türü
- **Yanıt Gövdesi** → bu, sunucudan istemciye geri gelen verilerdir