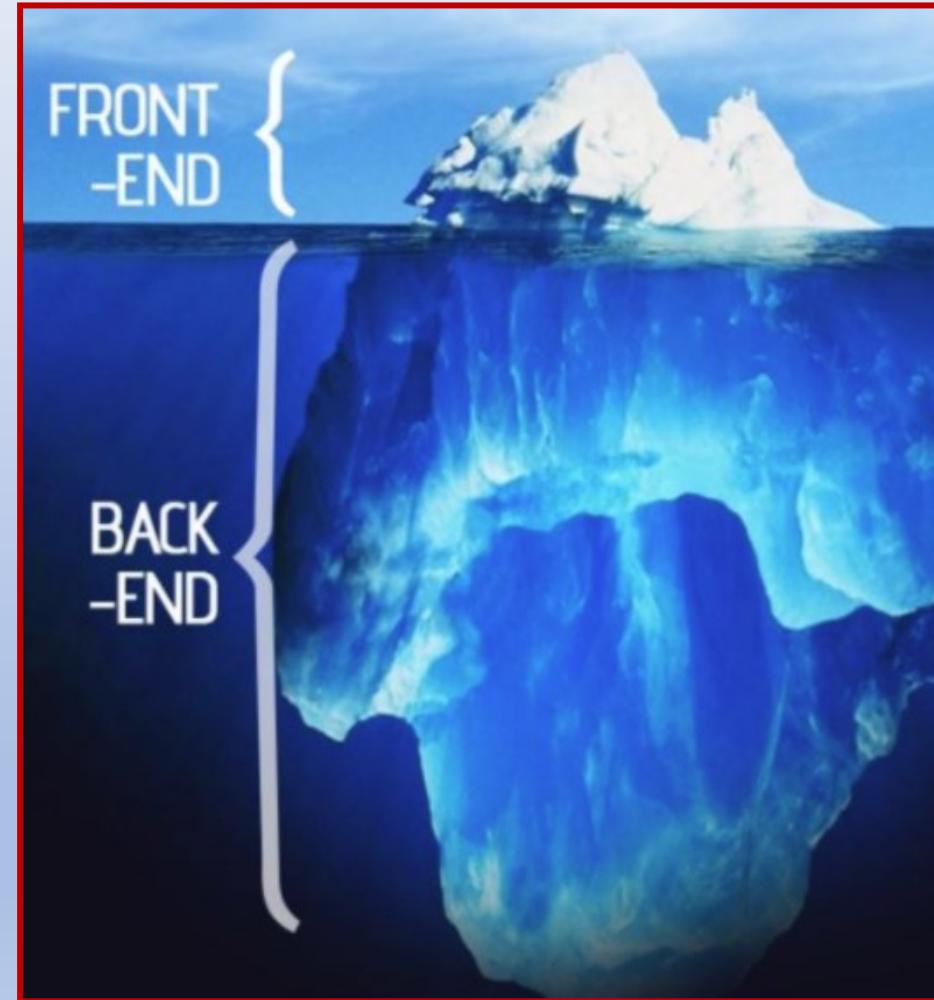


API Nedir ?

Application Programming Interface

Uygulama Programlama Arayüzü

- BOLUM 1 – Temel API kavramlari
- BOLUM 2 – Postman ile manuel API testing
- BOLUM 3 – API test otomasyonu
- BOLUM 4 – Framework'u geliştirmeye
- BOLUM 5- Farklı Tekniklerle API Test Otomasyonu
(De-serialization, Pojo)



API Nedir ?

Application Programming Interface

Uygulama Programlama Arayüzü

BOLUM 1 – Temel API

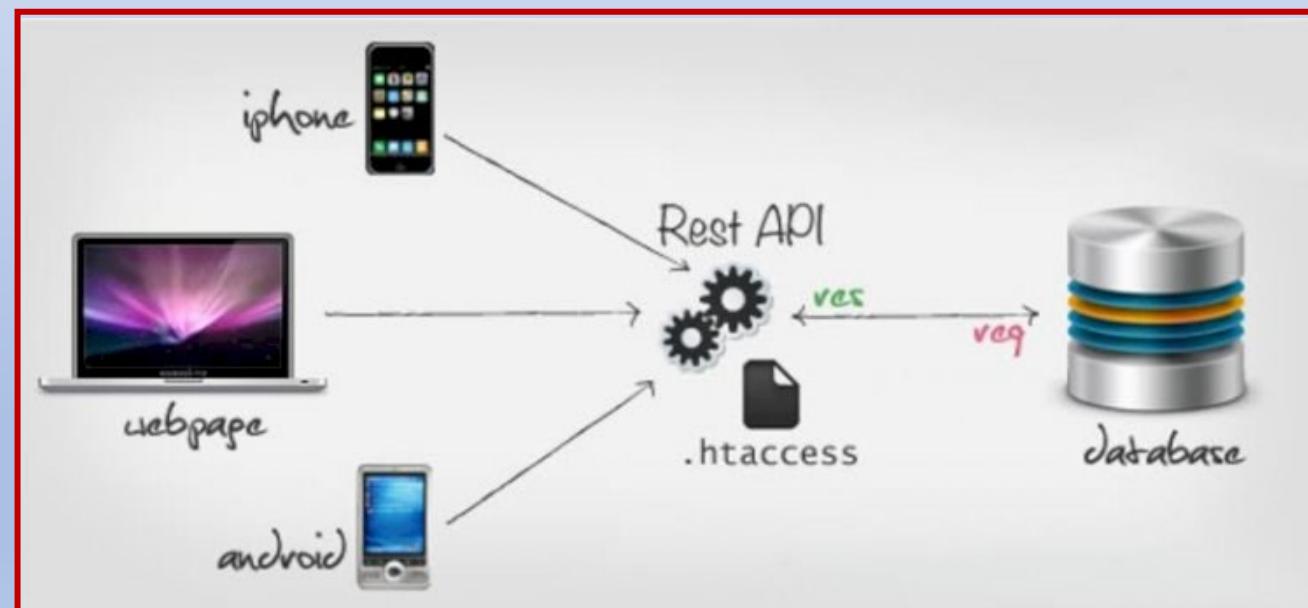
1. API nedir ?
2. API nasıl çalışır ?
3. API nasıl kullanılır ?
4. API ve Web Service aynı midir ?
5. HTTP nedir ?
6. HTTP request ve Response
7. HTTP durum kodları
8. API protokolleri SOAP ve Rest



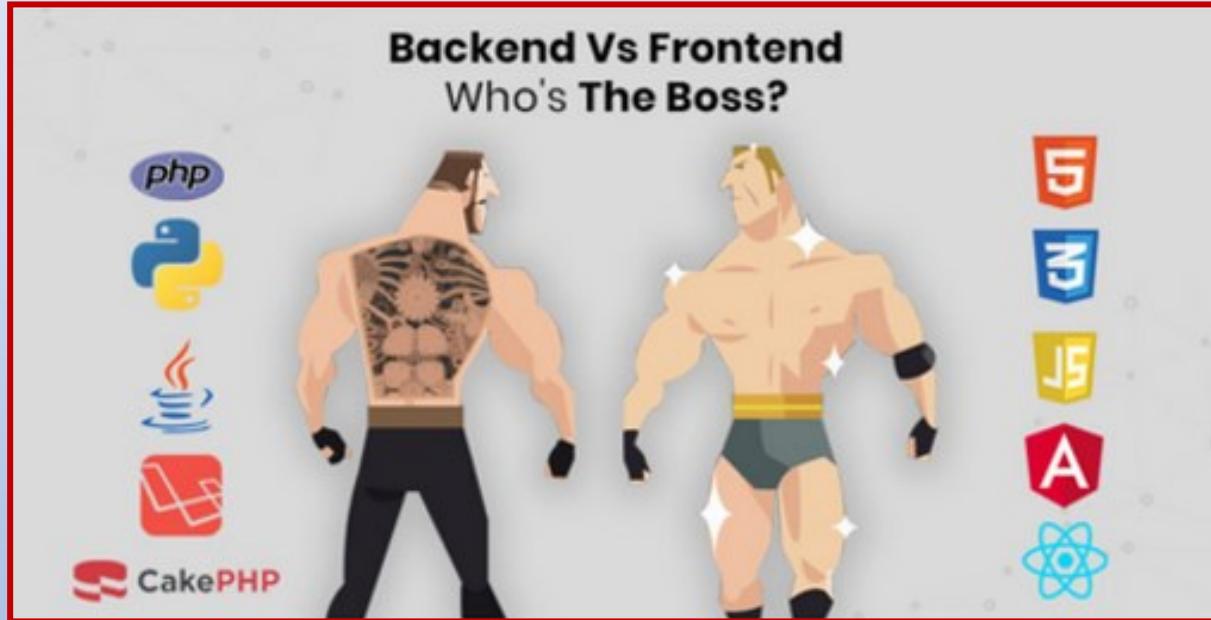
- 9- HTTP metotları
 - GET
 - PUT
 - POST
 - PATCH
 - DELETE
- 10- Endpoints
 - URI
 - URL
- 11- Swagger dokumani
 - Nasıl kullanılır
 - Neler Yapılabilir
 - Token kullanımı

API Nedir ?

- API, bir uygulamanın işlevlerine dışarıdan veya uzaktan erişilip bu işlevlerin kullanılmasını sağlayan arayüzdür.
- API, bir sunucunun üzerindeki uygulamaya farklı platformlardan ulaşılmasına, ve gönderilen request(İstek)'lere response(cevap) dönmesine olanak sağlar.



API Nedir ?



- API uygulamalar arasi bir iletisim oldugu icin UI(User Interface-Kullanici Arayuzu) yoktur.
- API ile normal hayatimizda UI ile yaptigimiz islemleri kodlarla yapabiliriz.
- Uygulamalar arasi tum baglantilari koordine eden ve otomasyon ile onumuze getiren UI degil Backend'dir

API Nasıl Çalışır ?



Corba, Pizza, Kola



API
Garson



1- Request to API
Corba, Pizza, Kola

2- Request to Server
Müşteri Corba, Pizza, Kola istiyor

4- Response to client
Siparisiniz...

3- Response from Server
Siparis hazır

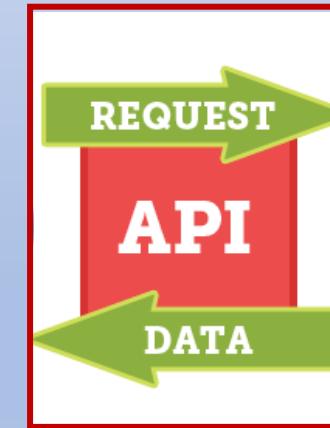
API Nasil Calisir ?



Ayse
1 6

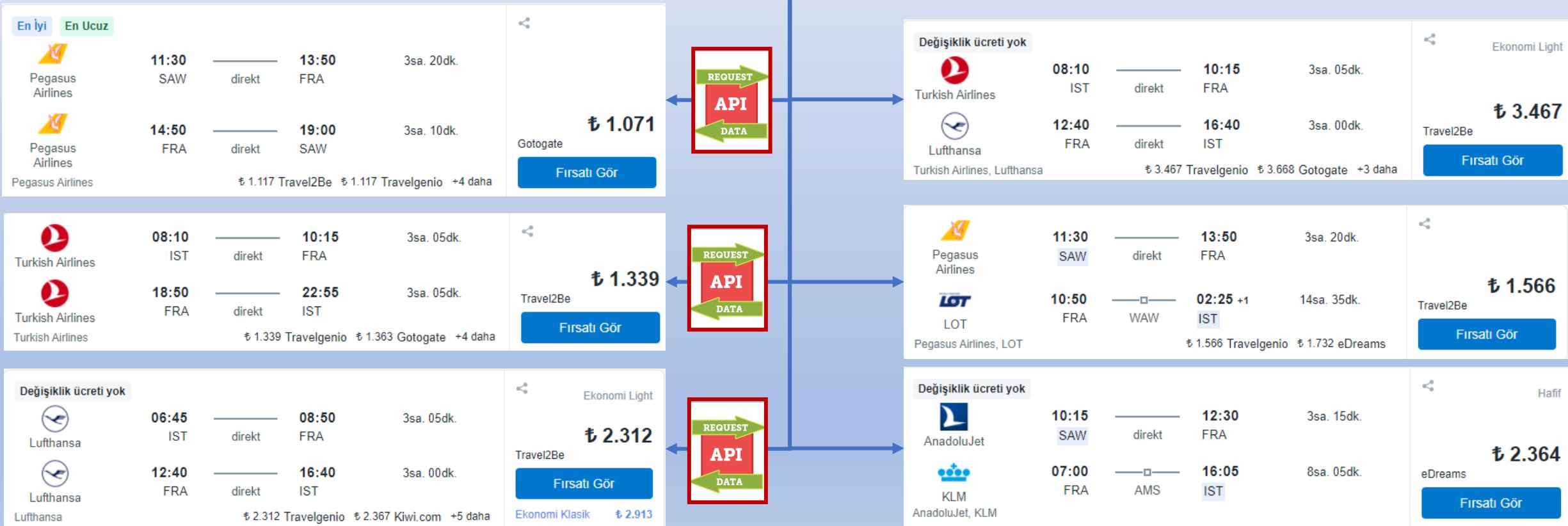


3 4

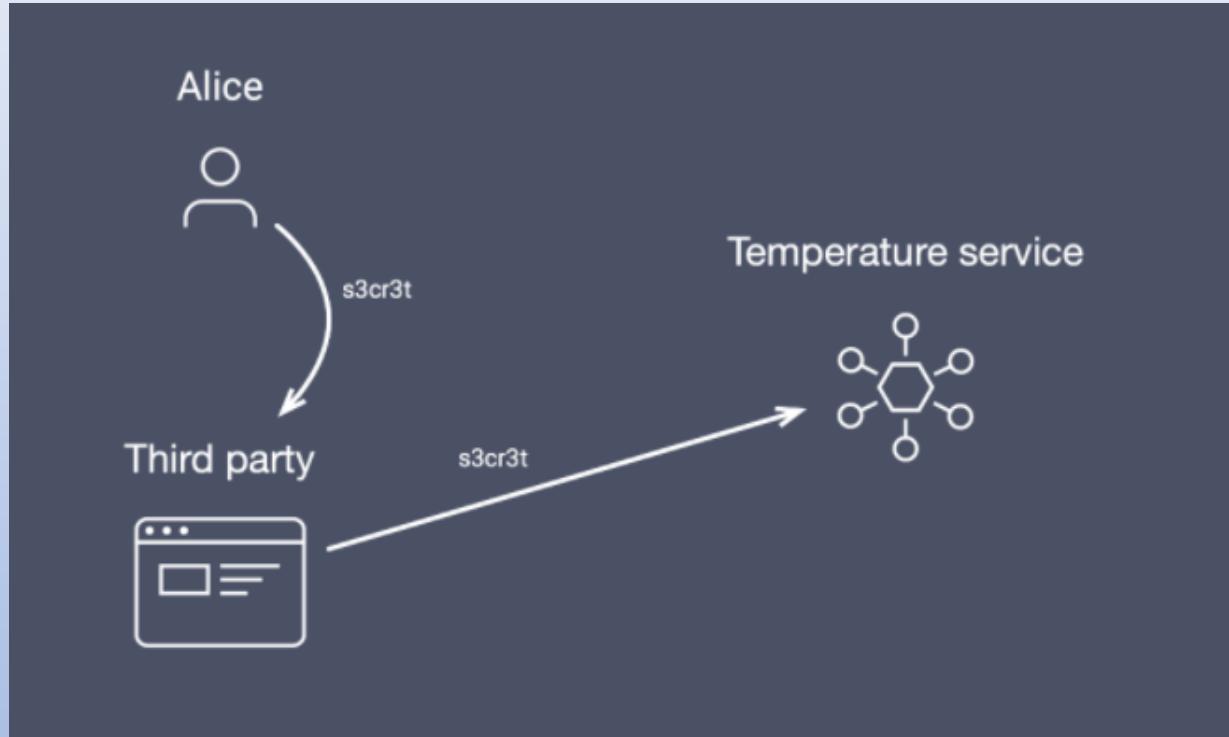


Kart Provider API

API Nasıl Çalışır ?



API Nasıl Çalışır ?



Alice'in evinin mevcut iç ortam sıcaklığını bildirebileceği bir hizmet hesabı var.

Alice ayrıca, sıcaklık verilerini okumak, sıcaklıklarını bir grafik üzerinde çizmek ve diğer hizmetlerden gelen verilerle çapraz referans yapmak için üçüncü taraf bir uygulamaya erişim izni vermek istiyor.

Sıcaklık hizmeti, sıcaklık verileriyle bir API ortaya çıkarır, bu nedenle üçüncü taraf uygulamasının verilere oldukça kolay bir şekilde erişebilmesi gereklidir.

Ayrıca uygulamada yalnızca Alice'in verilerinin kullanılabilmesi için datanın kisiselleştirilmesi gereklidir.

API'da uretebileceğimiz bir key ile verilerimizi kisisellestirebilir ve istersek güvenli hale getirebiliriz

API Nasıl Kullanılır ?

<https://collectapi.com/tr/>

Kategoriler

Tümü

Ücretsiz API'ler

AI Yapayzeka

Akaryakıt

Corona

E-Ticaret

Eczane

Ekonomi

Haberler

Hal Fiyatları

Harita

Hava Durumu

IP Adres

Instagram

Kitap

Mesaj

Oyun

Resim Boyutlandırma

Seyahat

Sinema

Spor

Sözlük

Website

Whatsapp

Yaşam

Çeviri



Whatsapp Business API

Official WhatsApp Business API. WhatsApp'ın resmi c



Whatsapp Business Sandbox

WhatsApp Business API'yi test edebilmeniz için sand



COVID-19 Koronavirus İstatistik API

Dünya genelindeki COVID-19 virüsünün etkilerini Dö



Instagram DM API

Unofficial Instagram DM API kullanarak HTTP istekle



Akaryakıt Fiyatları API

Şehirlere göre farklı akaryakıt istasyonlarındaki benz



Nöbetçi Eczane API

Türkiye il ve ilçelerdeki günün nöbetçi eczanelerini ge



Altın, Döviz ve Borsa API

Anlık döviz kurları, bitcoin API'leri ve Türkiye



Haberler API

7 farklı ülkeden, ülkelerin anadillerine göre ge



Hava Durumu API

Dünyanın her yerindeki hava durumunu günde



Faiz Oranları API

Bankaların genel faiz oranları ve farklı kredi ti



Şans Oyunları API

Son sayısal ve süper lotoda kazanan numaraları



IP Adresi API

İp adresini girerek lokasyon bilgisi, şehir bilgi



Namaz Vakitleri API

İstediğiniz şehrin namaz vakitlerini getiren, so



IMDb API

IMDb sitesinde isimle yada id ile arama yaparak film



Araç Tanıma API

Resimlerdeki araçların sayısını, marka, model, renk b



Plaka Tanıma API

Resimdeki araçların plakalarını yakalayın.



Nesne Tanıma API

Resimde neler olduğunu, hangi nesnelerin bulunduğu



Duygu Analiz API

Resmin ya da yazının içерdiği duyguları dereceleriley



E-Ticaret Ürün Öneri API

Kullanıcının bulunduğu e-ticaret sitesinde bakmış ol



Çiplaklıklık Algılama API

Resimlerin çiplaklıklık içerip içermediğini öğrenin.

API Nasıl Kullanılır ?

2

END POINTS

1

GET /dutyPharmacy

Şehirlerdeki gün nöbetçi olan eczaneleri getiren servis.

PARAMETRELER

3

Alan	Açıklama	Tip	Başlık	Zorunlu
il	Eczaneleri görmek istediğiniz şehri giriniz.	text		✓
ilce	İlçelere göre bakmak istiyorsanız, girdiğiniz şehrin ilçelerini yazabilirsiniz. O şehirdeki tüm eczanelere ulaşmak istiyorsanız burayı boş bırakabilirsiniz.	text		

ÖRNEK

4

```
curl --request GET \
--url 'https://api.collectapi.com/health/dutyPharmacy?ilce=%C3%87ankaya&il=Ankara' \
--header 'authorization: apapikey 700IIqcDndqPfwSEQBrke7:63xbVxUvHLL0CeD3nubi40' \
--header 'content-type: application/json'
```

Nöbetçi Eczane API



5

Response

```
{
  "success": true,
  "result": [
    {
      "name": "Sevinç Eczanesi",
      "dist": "Çankaya",
      "address": "Maltepe Mah. Güzaltan Sk. No:1/C Maltepe",
      "phone": "0312 231 71 75",
      "loc": "39.92887565500589,32.84444332122803"
    },
    {
      "name": "Alper Yüce Eczanesi",
      "dist": "Çankaya",
      "address": "Angora Cad. No:154/Beysukent",
      "phone": "0312 236 36 96",
      "loc": "39.88566860058295,32.71559000015259"
    },
    ...
  ]
}
```

1- Kullanicagim End Point

2- Kullanilacak HTTP metodu ve parametreler

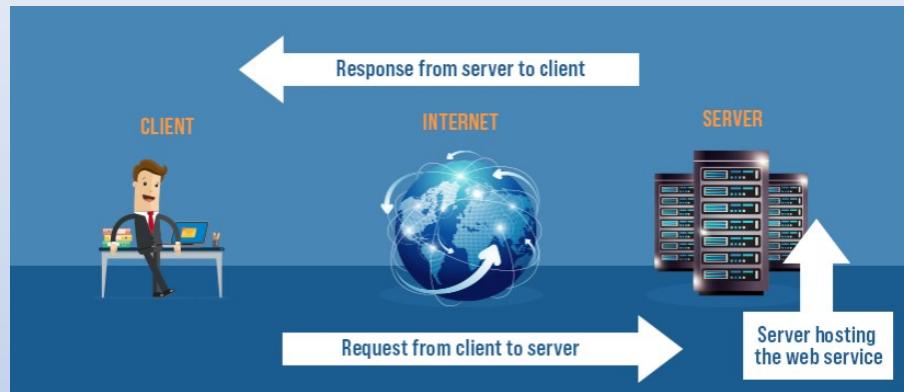
3- Zorunlu alanlar

4- Ornek request ve varsa authorization turu ve apikay

5- Gonderecegimiz request'e karsilik gelecek Response(cevap)

API vs Web Services

Ikisi de uygulamalar arasında iletisim saglar. Aralarındaki tek fark :



Web Service bu iletisimi internet
kullanarak gerceklestirir

API ise internet olmadan da iletisim saglayabilir.

Ornegin; Expedia, KLM Airlines DataBase'ine ulasmak icin internet kullanir (Web Service),
bilgisayarimizdaki Microsoft Word gibi uygulamalar ise farkli uygulamalarla iletisim kurmak icin kendi
API'larini kullanirlar .

NOT: Tum Web Service'ler API'dir ama tum API'lar Web Service degildir.

API

API Testing

DERS 2

BOLUM 1 – Temel API

1. API nedir ?
2. API nasil calisir ?
3. API nasil kullanilir ?
4. API ve Web Service ayni midir ?
5. HTTP nedir ?
6. HTTP request ve Response
7. HTTP durum kodlari
8. API protokolleri SOAP ve Rest



Http Nedir ?

HTTP : Hyper Text Transfer Protocol

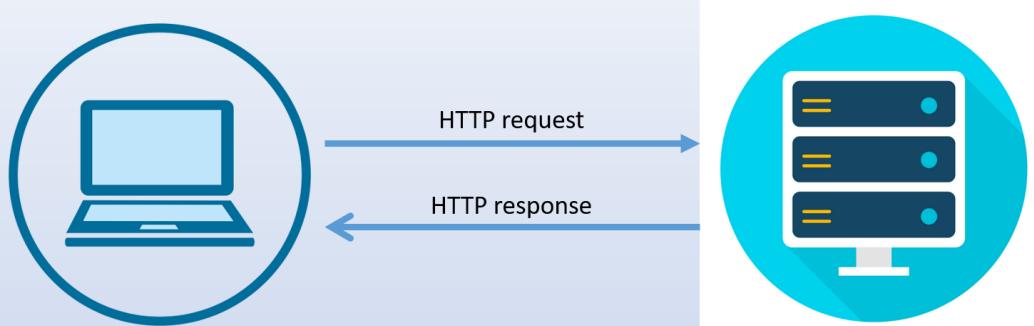
HTTP protokolü istemci (PC) ile sunucu (server) arasındaki alışveriş kurallarını belirler.



İstemci sunucuya bir istek(request) gönderir. Bu istek Internet Explorer, Google Chrome veya Mozilla Firefox gibi web browser'lar aracılığıyla iletilir. Sunucu bu isteği alır ve Apache veya IIS gibi web sunucu programları aracılığıyla cevap(response) verir.

Client ve Server arasındaki tüm iletişim **request** ve **response**'lar ile olur

Request & Response



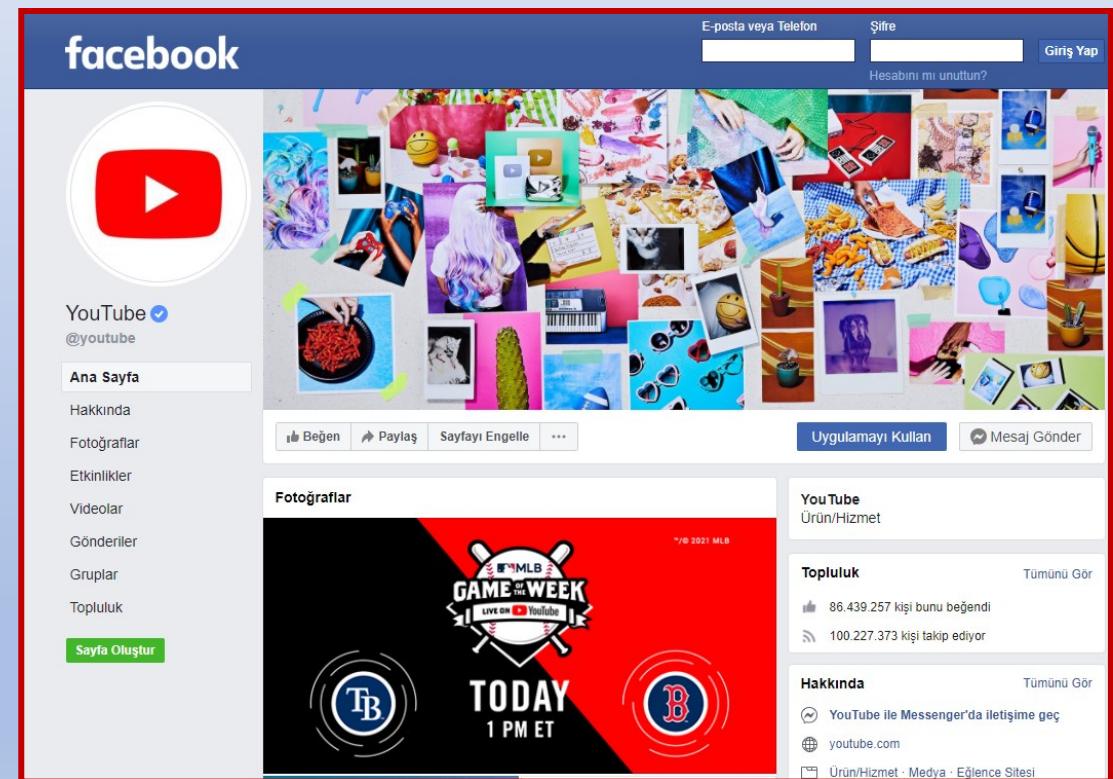
Client

Server

<http://www.facebook.com/youtube>

Request

UI



Response

Request & Response API

Request

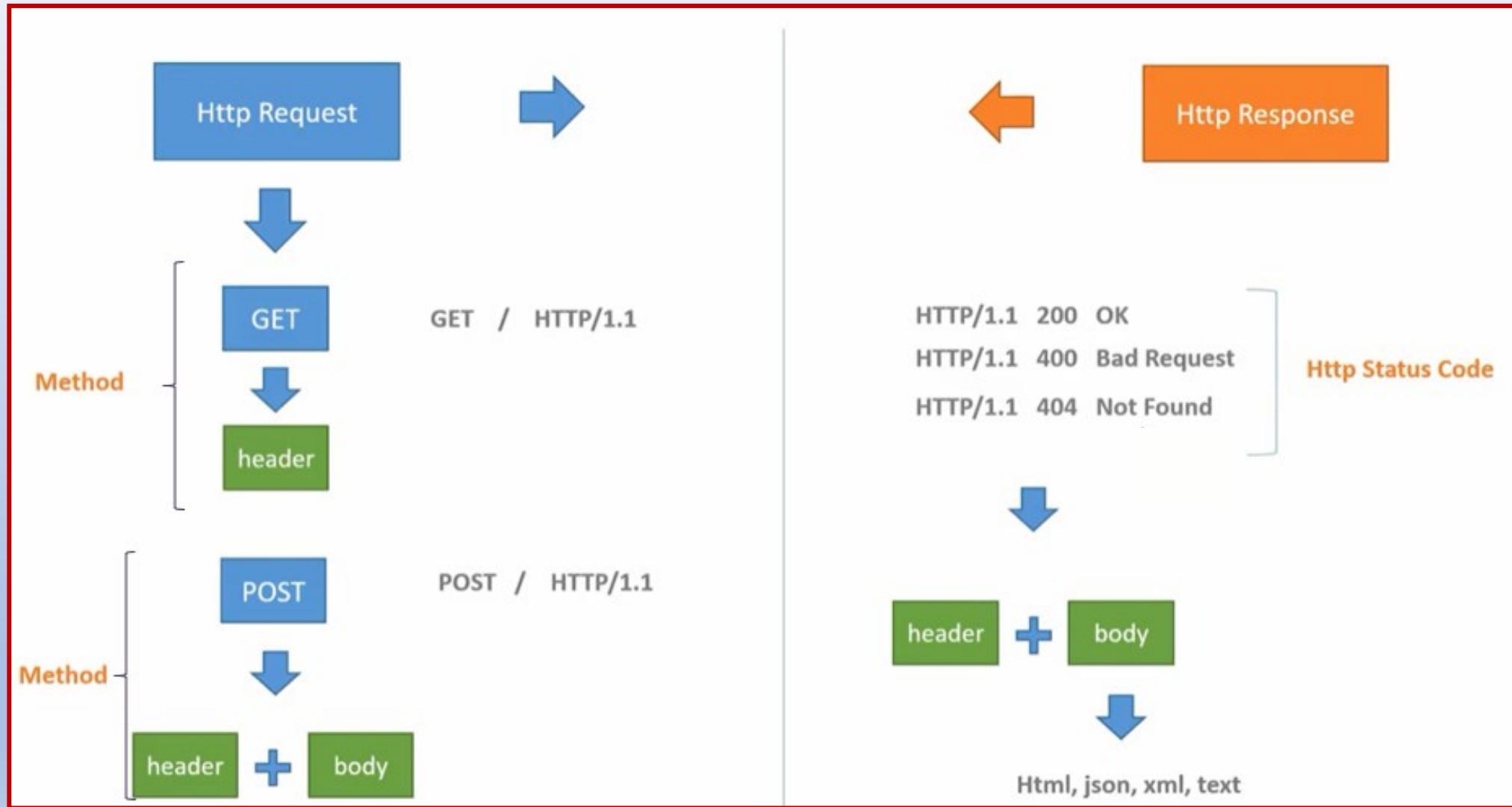
http://graph.facebook.com/youtube

API

The diagram illustrates the process of making a request to a social media API. On the left, a blue bracket labeled "Request" encloses the URL "http://graph.facebook.com/youtube". A red arrow points from this request to the right, where the text "API" is written above a JSON response. The JSON response shows various details about the YouTube page, such as its ID, name, likes count, and cover photo information.

```
{  
  "id": "7270241753",  
  "about": "Discover new channels, watch and share your favor  
  "can_post": false,  
  "category": "Product/service",  
  "checkins": 29,  
  "company_overview": "YouTube provides a forum for people to  
  creators and advertisers large and small. ",  
  "cover": {  
    "cover_id": "10152104891506754",  
    "offset_x": 0,  
    "offset_y": 0,  
    "source": "https://fbcdn-sphotos-f-a.akamaihd.net/hphoto  
oh=0f71bb2d5759df58c96719e5c3f7073c&oe=543B1B52&__gda__=141341  
},  
  "founded": "2005",  
  "has_added_app": false,  
  "is_community_page": false,  
  "is_published": true,  
  "likes": 81768558,  
  "link": "https://www.facebook.com/youtube",  
  "name": "YouTube",  
  "parking": {  
    "lot": 0,  
    "street": 0,  
    "valet": 0  
  },  
  "talking_about_count": 263847,
```

Request & Response API



Request & Response API

http://graph.facebook.com/youtube

API

```
{  
  "error": {  
    "message": "An access token is required to request this resource.",  
    "type": "OAuthException",  
    "code": 104,  
    "fbtrace_id": "ALgvuni-m0EnC1BFUMf1wvp"  
  }  
}
```

NOT:

UI daki kullanici adi ve sifre uygulamasi gibi API'da da bir guvenlik kontrolu vardir.

Ozel kullanici adi ve sifre istenen alanlara girmek icin TOKEN almaniz gerekir

Request & Response API

Sonuc olarak

API'da işlem yapmak için uygulamalar arasındaki iletişimini sağlayacak kurallar net olarak belirlenmiş olmalıdır

Request

Bir uygulamadan herhangi bir bilgi istenecekse bu bilgilere ulaşabilmek için istek sahiplerinin göndermesi gereken request net olarak belirlenmelidir

- 1- End Point
- 2- Parametreler
- 3- HTTP metodu
- 4- Body
- 5- Authorization ihtiyacı

Response

Yapılan request'e verilen cevap da net ve anlaşılır olmalıdır

- 1- Status Code
- 2- Body

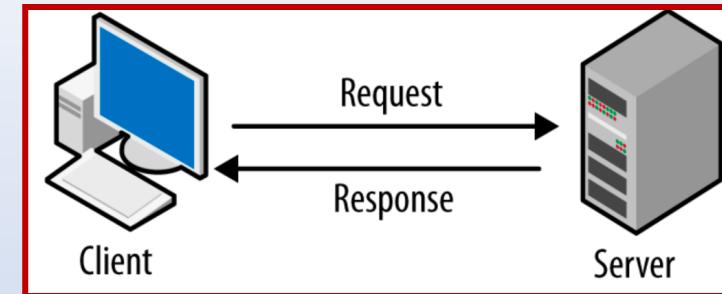


The screenshot shows a browser window with the URL `maps.googleapis.com/maps/api/geocode/json?address=chicago&sensor=false`. The page displays a JSON object representing the geocoding results for the address "Chicago". The JSON structure includes an array of results, each containing address components (long_name, short_name, types), a formatted address ("Chicago, IL, USA"), geometry information (bounds, northeast, southwest), and a location point (lat, lng).

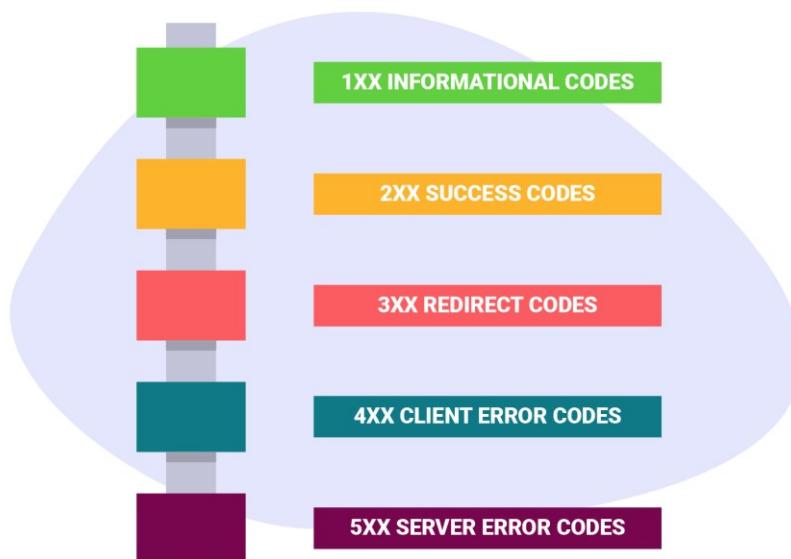
```
{
  "results": [
    {
      "address_components": [
        {
          "long_name": "Chicago",
          "short_name": "Chicago",
          "types": [ "locality", "political" ]
        },
        {
          "long_name": "Cook County",
          "short_name": "Cook County",
          "types": [ "administrative_area_level_2", "political" ]
        },
        {
          "long_name": "Illinois",
          "short_name": "IL",
          "types": [ "administrative_area_level_1", "political" ]
        },
        {
          "long_name": "United States",
          "short_name": "US",
          "types": [ "country", "political" ]
        }
      ],
      "formatted_address": "Chicago, IL, USA",
      "geometry": {
        "bounds": {
          "northeast": {
            "lat": 42.023131,
            "lng": -87.52404399999999
          },
          "southwest": {
            "lat": 41.6443349,
            "lng": -87.9402669
          }
        },
        "location": {
          "lat": 41.8781136,
          "lng": -87.6297982
        }
      }
    }
  ]
}
```

HTTP Status (Durum) Kodlari

İstemci (Request gönderen uygulama) bir sunucuya HTTP kullanarak request gönderdiğinde, gönderdiği request'e sunucunun nasıl bir sonuc donduğunu bilmek ister.



HTTP Status Codes



Request'in ulastigi server (sunucu) gelen Request'e verdiği yanitta, yanıtın durumunu belirten bir sayısal kod da gönderir.

Bu koda HTTP durum kodu (HTTP Status Code) denir ve bazı durumlarda bu kod istemcinin tarayıcısında da gösterilebilir

200, 301, 302, 404 ve 500 kodları en yaygın olanlardır.

Durum kodlarında 1'den 5'e kadar gruplandırılmıştır.

- 1xx Bilgi
- 2xx Başarı
- 3xx Yönlendirme
- 4xx Tarayıcı Hatası
- 5xx Sunucu Hatası

HTTP Status (Durum) Kodlari

- 1) 200 (OK) ==> This is the standard response for successful HTTP requests.
- 2) 201 (CREATED) ==> This is the standard response for an HTTP request that resulted in an item being successfully created.
- 3) 204 (NO CONTENT) ==> This is the standard response for successful HTTP requests, where nothing is being returned in the response body.
- 4) 400 (BAD REQUEST) ==> The request cannot be processed because of bad request syntax, excessive size, or another client error.
- 5) 403 (FORBIDDEN) ==> The client does not have permission to access this resource.
- 6) 404 (NOT FOUND) ==> The resource could not be found at this time. It is possible it was deleted, or does not exist yet
- 7) 500 (INTERNAL SERVER ERROR) ==> The generic answer for an unexpected failure if there is no more specific information available.

HTTP Status (Durum) Kodları

Code	Mesaj	Anlamı
1xx	Bilgi	
100	Continue	Devam
101	Switching Protocols	Anahtarlama Protokolü
102	Processing	İşlem
2xx	Başarı	
200	OK	Tamam
201	Created	Yaratıldı
202	Accepted	Onaylandı
203	Non-Authoritative Information	Yetersiz Bilgi
204	No Content	İçerik Yok
205	Reset Content	İçeriği Baştan al
206	Partial Content	Kısmi İçerik
207	Multi-Status	Çok-Statü
210	Content Different	Farklı İçerik

HTTP Status (Durum) Kodları

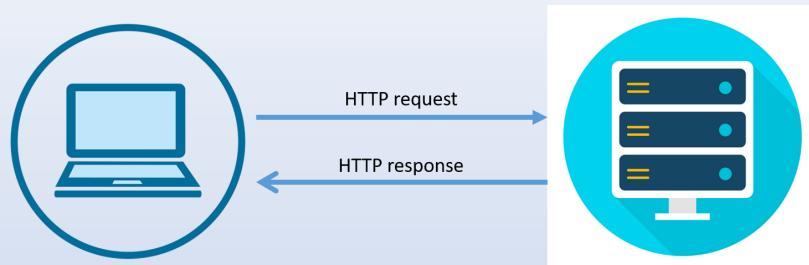
3xx	Yönlendirme	
300	Multiple Choices	Çok Seçenek
301	Moved Permanently	Sürekli Taşındı
302	Moved Temporarily	Geçici Taşındı
303	See Other	Diğerlerine Bak
304	Not Modified	Nitelenemedi
305	Use Proxy	Proxy Kullan
307	Temporary Redirect	Geçici olarak yeniden gönder

5xx	Sunucu Hatası	
500	Internal Server Error	
501	Uygulanmamış	
502	Geçersiz Ağ Geçidi	
503	Hizmet Yok	
504	Gateway Timeout	
505	HTTP Version not supported	

HTTP Status (Durum) Kodları

4xx	Tarayıcı Hatası	
400	Bad Request	Kötü İstek
401	Unauthorized	Yetkisiz
402	Payment Required	Ödeme Gerekli
403	Forbidden	Yasaklandı
404	Not Found	Sayfa Bulunamadı
405	İzin verilmeyen Metod	
406	Not Acceptable	Kabul Edilemez
407	Proxy Sunucuda login olmak gereklidir	
408	İstek zaman aşamına ugradı	
409	Conflict	(Hatalar) Çakıştı,Çakışma
410	Gone	Bak
411	Length Required	
412	Precondition Failed	
413	Request Entity Too Large	
414	Request-URI Too Long	
415	Unsupported Media Type	
416	Requested range unsatisfiable	
417	Expectation failed	
422	Unprocessable entity	
423	Locked	
424	Method failure	

Request / Response Body



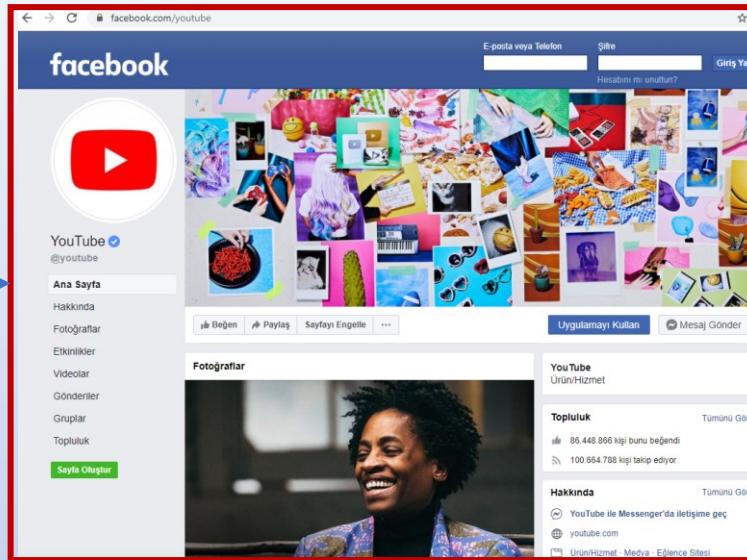
Client

Server

facebook.com/youtube

Request

UI



Response
(Kullanıcının
Gordüğü)

Html

```
<!DOCTYPE html>
<html lang="tr" id="facebook" class="tinyViewport tinyWidth">
  <head>...</head>
  <body class=" _4-u5 _2yq UIPage_LoggedOut _-kb _605a b_c3pyn-ahh chrome webkit win x1 Locale_tr_TR cores-gte4 hasAXNavMenubar _19_u" dir="ltr" style="margin-bottom: 180px;">
    <script nonce>requireLazy(["bootstrapWebSession"],function(j){j(1620996330)})</script>
    <div class=" _1i" id="u_0_6_eF">...</div>
    <div style="display:none">...</div>
    <script>...</script>
    <script>...</script>
    <script>...</script>
    <script>
      onloadRegister_DEPRECATED(function () {try { $("email").focus(); } catch (_ignore) {}});
      onloadRegister_DEPRECATED(function () {try { $("email").focus(); } catch (_ignore) {}});
      onafterloadRegister_DEPRECATED(function ()
        CavalryLogger.getInstance("6962126229364696643-0").collectBrowserTiming(window));
      onafterloadRegister_DEPRECATED(function ()
        {window.CavalryLogger&&CavalryLogger.getInstance().setTimeStamp("t_paint"));
      onafterloadRegister_DEPRECATED(function () {if (window.ExitTime)
        CavalryLogger.getInstance("6962126229364696643-0").setValue("t_exit",
        window.ExitTime);});});
    </script>
    <div class="AdBox Ad advert post-ads"></div>
  </body>
***</html> == $0
```

Response
(Bilgisayara
Gelen)

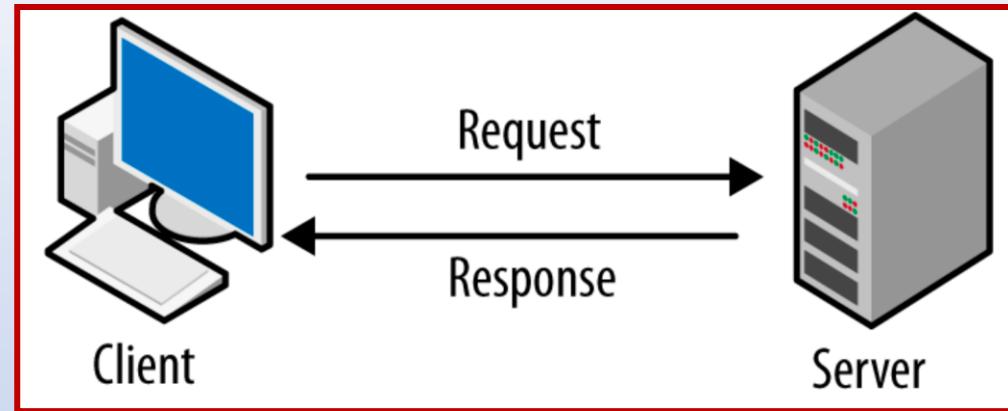
Request / Response Body

Bir sunucuya request gönderildiginde istegimizi sunucuya doğru şekilde anlatabilmemiz ve donen Response'u doğru şekilde anlamlandırmamız icin kullanılan 4 temel data formatı vardır.

- 1- Html
- 2- Text
- 3- XML

```
<customer>
    <customer_id> 1001 </customer_id>
    <customer_name> Mark Star </customer_name>
</customer>
```

- 4- Json {
 "firstname": "Eric",
 "lastname": "Wilson",
 "totalprice": 712,
 "depositpaid": false,
}



JSON (Java Script Object Notation) Nedir ?



Gunumuzde bir web uygulamasindan bahsediyorsak, JavaScript kullanilmamis olmasi neredeyse imkansizdir.

JSON formati JS ile olusturulan web uygulamalarinda data saklamak ve uygulamalar arasinda data alisverisi yapmak (Request/Response) icin en çok tercih edilen formattir.

JSON formatindaki bir data icin uc temel bolum vardir.

1- **Suslu parantezler** : JSON formatindaki bir datanin nerede baslayip, nerede bittigini gosterir. İhtiyac oldugunda NESTED (ic ice) JSON datalari olusturulabilir

2- **Keys** : JSON datalari icinde bulunan variable isimleridir.

3- **Values** : JSON datalari icinde bulunan variable'lara atanan degerlerdir.

Keys ve **Values** arasında : kullanilir.

API Protokollerı

Web servis mimarisinin temeli HTTP üzerine kurulmuştur. Yani genel olarak web servise bir istek gelir ve web servis bu isteği yapıp bir sonuç döndürür. Web servisin bu işlemi yapabilmesi için tanımlanmış farklı yöntemler bulunmaktadır. Bu yapılardan en çok kullanılan ikisi **SOAP** ve **REST**'dir.

SOAP (Simple Object Access Protocol) uygulamalar ile web servislerin bilgi aktarımını sağlayan **XML** tabanlı bir protokoldür.

Yani web servise giden bilgi XML olarak gönderilir, web servis bu bilgiyi yorumlar ve sonucunu XML olarak geri döndürür. XML, makine ve insan tarafından okunabilir şekilde tasarlanmıştır.

SOAP tabanlı bir web servisin, gönderilen XML verisini nasıl yorumlayacağının tanımlanması gereklidir. Bu web servis tanımlaması için standartlar belirlenmiştir.

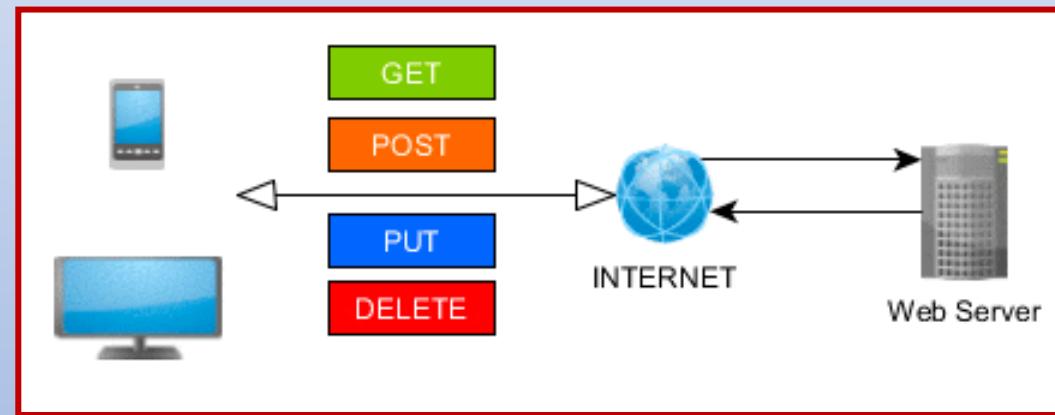
```
<customer>
    <customer_id> 1001 </customer_id>
    <customer_name> Mark Star </customer_name>
</customer>
```

Güvenlik ve dokumantasyona erişim açısından SOAP daha avantajlı olsa da esnekliği ve çok daha düşük data büyüklüğü sayesinde REST daha fazla kullanılmaktadır.

API Protokollerı

REST (Representational State Transfer), REST mimarisinde ise işlemler resource kavramıyla yapılır. Resource URI (Uniform Resource Identifiers) ile tanımlanır.

REST'te SOAP'ta olduğu gibi XML yardımıyla metodlar çağrılmaz bunun yerine o metodu çağıracak URL'ler ile web servise HTTP protokolüyle istek yapılır. Böylece işlemler **tamamen HTTP metodları** üzerinden yapılır.



RESTin döndürdüğü veri tipinin de XML olması zorunlu değildir JSON (**Java Script Object Notation**) , XML, TXT, HTML gibi istenen veri türünde değer döndürülebilir.

Rest Vs SOAP

- SOAP XML veri tipini desteklerken REST istenen veri türüyle işlem yapabilir. JSON veri tipi ile XML'den çok daha düşük boyutlarla veri tutulabildiği için REST ile daha hızlı işlem yapılabilir.
- SOAP için WSDL ile tanımlama yapmak gereklidir REST için böyle bir zorunluluk yoktur. (WADL REST için kullanılan WSDL'e benzer bir yapıdır fakat kullanma zorunluluğu yoktur.) Bir dile ihtiyaç duymadan HTTP metodlarıyla tasarlanabildiği için REST'i kullanması ve tasarlaması daha kolaydır.
- SOAP için birçok geliştirme aracı mevcuttur, REST için geliştirme araçlarına ihtiyaç duyulmaz, tasarlaması kolaydır.
- SOAP; XML-Scheme kullanırken REST; URI-scheme kullanır yani metodlar için URI'ler tanımlanır.
- Her ikisi de HTTP protokolünü kullanırlar. Fakat REST için HTTP zorunluluğu varken SOAP; TCP, SMTP gibi başka protokollerle de çalışabilir.
- Test ve hata ayıklama aşaması REST için daha kolaydır. Çünkü HTTP hatalarını döndürür ve bunlar bir toola ihtiyaç duymadan görülebilir. SOAP için hata ayıklama araçları gerekebilir.
- REST basit HTTP GET metodunu kullandığı için cache'leme işlemi daha kolaydır. SOAP ile cache'leme yapabilmek için karmaşık XML requestleri yapılmalıdır.
- İkisi de HTTPS destekler, SOAP için WS-SECURITY adlı bir ekleni mevcuttur.
- Güvenlik açısından SOAP daha gelişmiştir çünkü hazır yapılar bulunmaktadır.
- Dokümantasyon bakımından SOAP daha gelişmiştir ve daha fazla kaynak bulunmaktadır.

API

API Testing

DERS 3

BOLUM 1 – Temel API

- 1. API nedir ?
- 2. API nasil calisir ?
- 3. API nasil kullanilir ?
- 4. API ve Web Service ayni midir ?
- 5. HTTP nedir ?
- 6. HTTP request ve Response
- 7. HTTP durum kodlari
- 8. API protokoller SOA ve Rest

9- HTTP metotlari

- GET
- PUT
- POST
- PATCH
- DELETE

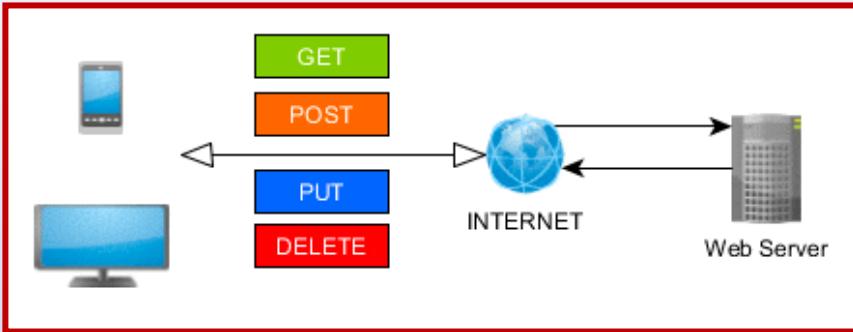
10- Endpoints

- URI
- URL

11- Swagger dokumani

- Nasil kullanilir
- Neler Yapilabilir

HTTP METOTLARI



HTTP metodlarının REST ile kullanımı;

REST tabanlı web servislerde HTTP metodlarına özel anlamlar yüklenir.

Istemcinin Request'i gönderdiği METOT'a göre SUNUCU'nun Response'u da farklılaşacaktır.

Kısaca, API sorgularında ne yapmak istediğimizi bilmemiz, istegimize uygun bir **metot** ve ihtiyaca göre **body** seçmemiz çok önemlidir.

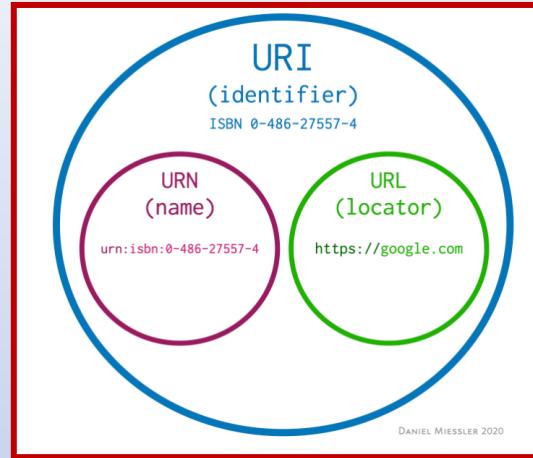
GET : Adresi verilen nesneyi döndürmek için kullanılır. Bu metot kullanıldığındaki kayıtlarda bir değişiklik yapılmaz. Sadece istediğimiz bilgiyi net olarak Request'e yazmamız yeterlidir.

PUT : Var olan bir nesneyi değiştirmek için (veya eğer yoksa yeni bir tane oluşturmak için) kullanılır. Bu metot kullanıldığındaki değiştirmek istediğimiz datayı ve yapmak istediğimiz değişiklikleri request'e yazmaliyiz.

POST : Yeni bir nesne oluşturmak için kullanılır. Her seferinde yeni bir nesne oluşturur. Yeni bir nesne oluşturacağımız için nesne için gerekli tüm bilgileri Request'e yazmamız gereklidir.

DELETE : Adresi verilen nesneyi silmek için kullanılır. Hangi nesneyi sileceğimizi Request'te net olarak belirtmemiz yeterlidir.

ENDPOINTS



Endpoint kaynaga nasıl erişebileceğimizi gösteren URI (Uniform Resource Identifiers)'lara denir. Bir API oluşturduğunuzda kullanıcıların ulaşabilmesi için bu endpoint'i ve kullanabilecek Http method'larını kullanıcılarla bildirmemiz gereklidir.

URI günlük hayatımda kullandığımız URL (Uniform Resource Locator)'a benzer.

URL kullanıcının görebileceği ve etkileşimde bulunacağı bir web sayfasını ifade ederken URI o sayfanın içeriği bilgiyi ifade eder.

ENDPOINTS

<https://restful-booker.herokuapp.com/>

The screenshot shows a web browser window with the URL restful-booker.herokuapp.com in the address bar. The page title is "Welcome to Restful-Booker". Below the title, a sub-header reads: "An API playground created by [Mark Winteringham](#) for those wanting to learn more about API testing and tools". A navigation bar contains links: "@2bittester | [Website](#) | [Code](#) | [API Docs](#)". Below the navigation is a button labeled "Buy me a tea" with a coffee cup icon, and a small box showing a heart icon with the number "6". The main content area contains a paragraph about the API's purpose and features, mentioning it's a Create Read Update Delete Web API with authentication, bugs for exploration, and pre-loaded records. It also notes that the API resets every 10 minutes. At the bottom, a message encourages users to let the developer know via Twitter if they find any bugs.

Welcome to Restful-Booker an API that you can use to learn more about API Testing or try out API testing tools against. Restful-booker is a **Create Read Update Delete** Web API that comes with authentication features and loaded with a bunch of bugs for you to explore. The API comes pre-loaded with 10 records for you to work with **and resets itself every 10 minutes back to that default state**. Restful-booker also comes with [detailed API documentation](#) to help get you started with your API testing straight away.

Whilst the bugs you find are intentional, the API shouldn't go down so if you come across any requests that cause the API to go down, let me know via [Twitter](#).

ENDPOINTS

<https://restful-booker.herokuapp.com/apidoc/index.html>

The screenshot shows a detailed view of a POST endpoint for authentication. The endpoint URL is <https://restful-booker.herokuapp.com/auth>. The request body is defined with two fields: 'username' (String) and 'password' (String). The response body contains a single field 'token' (String), which is described as a token to use in future requests.

1. POST
2. https://restful-booker.herokuapp.com/auth
3. curl -X POST \
 https://restful-booker.herokuapp.com/auth \
 -H 'Content-Type: application/json' \
 -d '{
 "username" : "admin",
 "password" : "password123"
 }'
4. Header
5. Success 200

Alan	Tip	Açıklama
Content-Type	string	Sets the format of payload you are sending Varsayılan değer: application/json

Alan	Tip	Açıklama
username	String	Username for authentication Varsayılan değer: admin
password	String	Password for authentication Varsayılan değer: password123

Alan	Tip	Açıklama
token	String	Token to use in future requests

1 Yapılacak işlem ve kullanılacak HTTP metot

2 Kullanılacak Endpoint

3 Gonderecegimiz datanın formatı (Json)

4 Gonderecegimiz body'de olması gereken variable'lar ve bu variable'ların değerleri

5 Gonderecegimiz request çalışırsa almamız gereken HTTP status code

ENDPOINTS

Booking

Booking - GetBookingIds

1.0.0 ▾

Returns the ids of all the bookings that exist within the API. Can take optional query strings to search and return a subset of booking ids.

GET

`https://restful-booker.herokuapp.com/booking`

Example 1 (All IDs):

Example 2 (Filter by name):

Example 3 (Filter by checkin/checkout date):

`curl -i https://restful-booker.herokuapp.com/booking`



`restful-booker.herokuapp.com/booking`

```
[{"bookingid":2}, {"bookingid":1}, {"bookingid":4}, {"bookingid":7}, {"bookingid":5}, {"bookingid":3}, {"bookingid":10}, {"bookingid":9}, {"bookingid":8}, {"bookingid":6}]
```

ENDPOINTS

Booking - GetBooking

Returns a specific booking based upon the booking id provided

GET

`https://restful-booker.herokuapp.com/booking/:id`

Example 1 (Get booking):

```
curl -i https://restful-booker.herokuapp.com/booking/1
```



restful-booker.herokuapp.com/booking/1

I'm a teapot

HTTP/1.1 200 OK

{

```
"firstname": "Sally",
"lastname": "Brown",
"totalprice": 111,
"depositpaid": true,
"bookingdates": {
    "checkin": "2013-02-23",
    "checkout": "2014-10-23"
},
"additionalneeds": "Breakfast"
```

ENDPOINTS

Booking - CreateBooking

Creates a new booking in the API

POST

<https://restful-booker.herokuapp.com/booking>

JSON example usage:

XML example usage:

URLEncoded example usage:

```
curl -X POST \  
  https://restful-booker.herokuapp.com/booking \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "firstname" : "Jim",  
    "lastname" : "Brown",  
    "totalprice" : 111,  
    "depositpaid" : true,  
    "bookingdates" : {  
      "checkin" : "2018-01-01",  
      "checkout" : "2019-01-01"  
    },  
    "additionalneeds" : "Breakfast"  
  }'
```

Booking - UpdateBooking

Updates a current booking

PUT

<https://restful-booker.herokuapp.com/booking/:id>

JSON example usage:

XML example usage:

URLEncoded example usage:

```
curl -X PUT \  
  https://restful-booker.herokuapp.com/booking/1 \  
  -H 'Content-Type: application/json' \  
  -H 'Accept: application/json' \  
  -H 'Cookie: token=abc123' \  
  -d '{  
    "firstname" : "James",  
    "lastname" : "Brown",  
    "totalprice" : 111,  
    "depositpaid" : true,  
    "bookingdates" : {  
      "checkin" : "2018-01-01",  
      "checkout" : "2019-01-01"  
    },  
    "additionalneeds" : "Breakfast"  
  }'
```

ENDPOINTS

Booking - PartialUpdateBooking

Updates a current booking with a partial payload

PATCH

<https://restful-booker.herokuapp.com/booking/:id>

JSON example usage:

XML example usage:

URLEncoded example usage:

```
curl -X PUT \
  https://restful-booker.herokuapp.com/booking/1 \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json' \
  -H 'Cookie: token=abc123' \
  -d '{
    "firstname" : "James",
    "lastname" : "Brown"
  }'
```

Booking - DeleteBooking

Returns the ids of all the bookings that exist within the API. Can take optional parameters.

DELETE

<https://restful-booker.herokuapp.com/booking/1>

Example 1 (Cookie):

Example 2 (Basic auth):

```
curl -X DELETE \
  https://restful-booker.herokuapp.com/booking/1 \
  -H 'Content-Type: application/json' \
  -H 'Cookie: token=abc123'
```

ENDPOINTS / Swagger DOCUMENTS

Bir API sorgusu yapabilmek için endpoint, kullanılacak HTTP metodu, yazacağımız body'de olması gereken key / value değerleri gibi bilgilere ihtiyacımız vardır.

Sorgu yapmamız istendiginde bu bilgilerin ve ihtiyac varsa TOKEN'in bize verilmesi gerekmektedir.

Swagger Nedir?

Web API geliştirmede en önemli ihtiyaçlardan biri dokümantasyon ihtiyacıdır. API methodlarının ne işe yaradığı ve nasıl kullanıldığından dokümantasyon içerisinde anlaşılır olması gereklidir.

Kendi uygulamanızın API 'ini baskalarının kullanımına acmak istediğinizde kullanıcıları olan tüm dokümantasyonu sunmak ve gerektiğinde güncellemleri yapmak zorundayız ve bunu yapmanın en iyi ve en kolay yolu swagger dokumani hazırlamaktır.

Swagger'ın önemli bir amacı da RestApi'ler için bir arayüz sağlamaktır. Bu API'yi kullancak insanların kaynak koda erişmeden RestApi'lerin özelliklerini görmesine, incelemesine ve örnek sorgularla API'yi anlamasına olanak sağlar.

Swagger DOCUMENTS

<https://petstore.swagger.io/>



Base URL <http://petstore.swagger.io/>

The screenshot shows the Swagger Petstore API documentation for the "pet" resource. The title is "pet Everything about your Pets". Below it are four API operations:

- POST /pet/{petId}/uploadImage** uploads an image
- POST /pet** Add a new pet to the store
- PUT /pet** Update an existing pet
- GET /pet/findByStatus** Finds Pets by status

<http://petstore.swagger.io/pet/findByStatus>

Swagger DOCUMENTS

<https://petstore.swagger.io/>

The screenshot shows the Swagger UI for the 'pet' resource. It lists various HTTP methods and their corresponding URLs and descriptions. A red box highlights the first two methods: 'POST /pet/{petId}/uploadImage' and 'POST /pet'. Another red box highlights the 'GET /pet/{petId}' method, which is described as 'Find pet by ID'. The other methods listed are 'PUT /pet' (Update an existing pet), 'GET /pet/findByStatus' (Finds Pets by status), 'GET /pet/findByTags' (Finds Pets by tags), 'POST /pet/{petId}' (Updates a pet in the store with form data), and 'DELETE /pet/{petId}' (Deletes a pet).

Method	URL	Description
POST	/pet/{petId}/uploadImage	uploads an image
POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet

SWAGGER var olan bir API'yi nasıl kullanacağımızı gösteren bir dokumandır.

Swagger, yeni gittigimiz bir sehrin gezilecek yerlerini ve nerede ne bulabileceğimizi gösteren bir harita gibidir.

1- Base URL <http://petstore.swagger.io/>

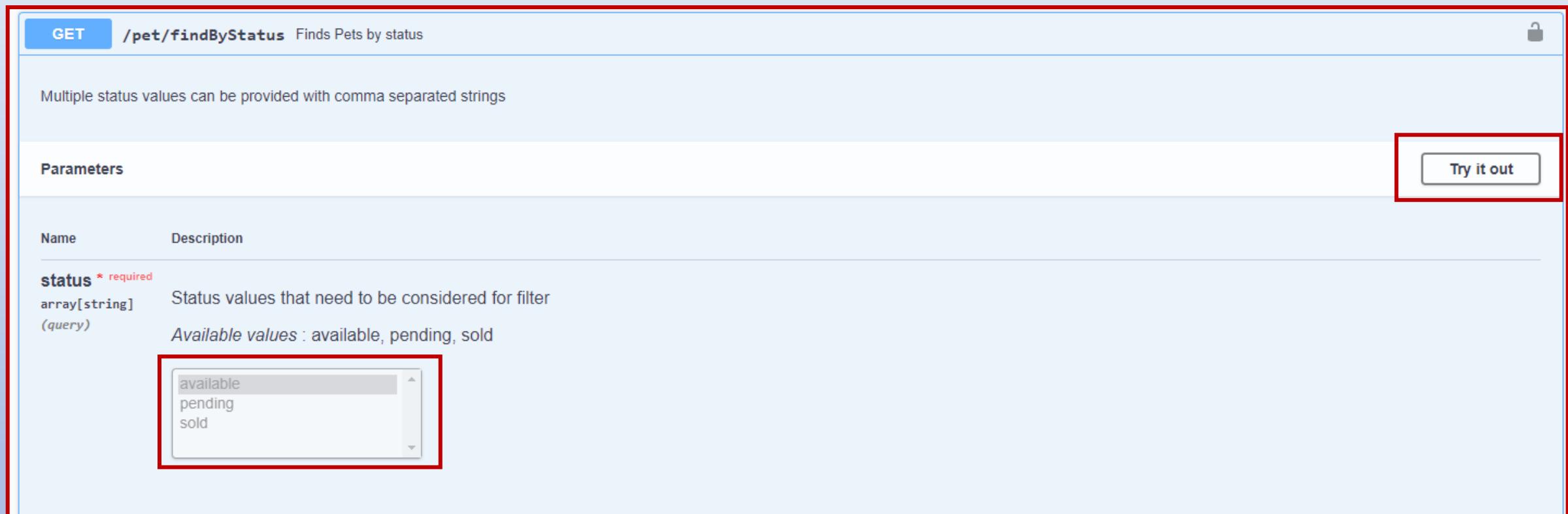
2- Kullanabilecek Http Method'lari

3- Her bir method icin yazilmasi gereklili olan parametreler ve method'un islev aciklamasi
<http://petstore.swagger.io/pet/12345>

Swagger DOCUMENTS

<https://petstore.swagger.io/>

Bir API' in Swagger sayfasindan API üzerinde kullanilacak method'lari, kullanabilecegimiz parametreleri ogrenebilir ve istersek API sorgusu gerceklestirebiliriz.



GET /pet/findByStatus Finds Pets by status

Multiple status values can be provided with comma separated strings

Parameters

Name Description

status * required
array[string]
(query)
Status values that need to be considered for filter
Available values : available, pending, sold

Try it out

available
pending
sold

1- Kullanilabilecek parametreler

2- Swagger dokumanindan ornek API sorgusu yapmak icin

Swagger DOCUMENTS

<https://petstore.swagger.io/>

1- Durum kodu (Status code)

2- Response body

GET /pet/findByStatus Finds Pets by status

Multiple status values can be provided with comma separated strings

Parameters

Name Description

status required
array(string)
(query)
Status values that need to be considered for filter
available
pending
sold

Cancel

Execute Clear

Responses

Response content type application/json

Curl

```
curl -X 'GET' \
  'https://petstore.swagger.io/v2/pet/findByStatus?status=available' \
  -H 'accept: application/json'
```

Request URL

```
https://petstore.swagger.io/v2/pet/findByStatus?status=available
```

Server response

Code Details

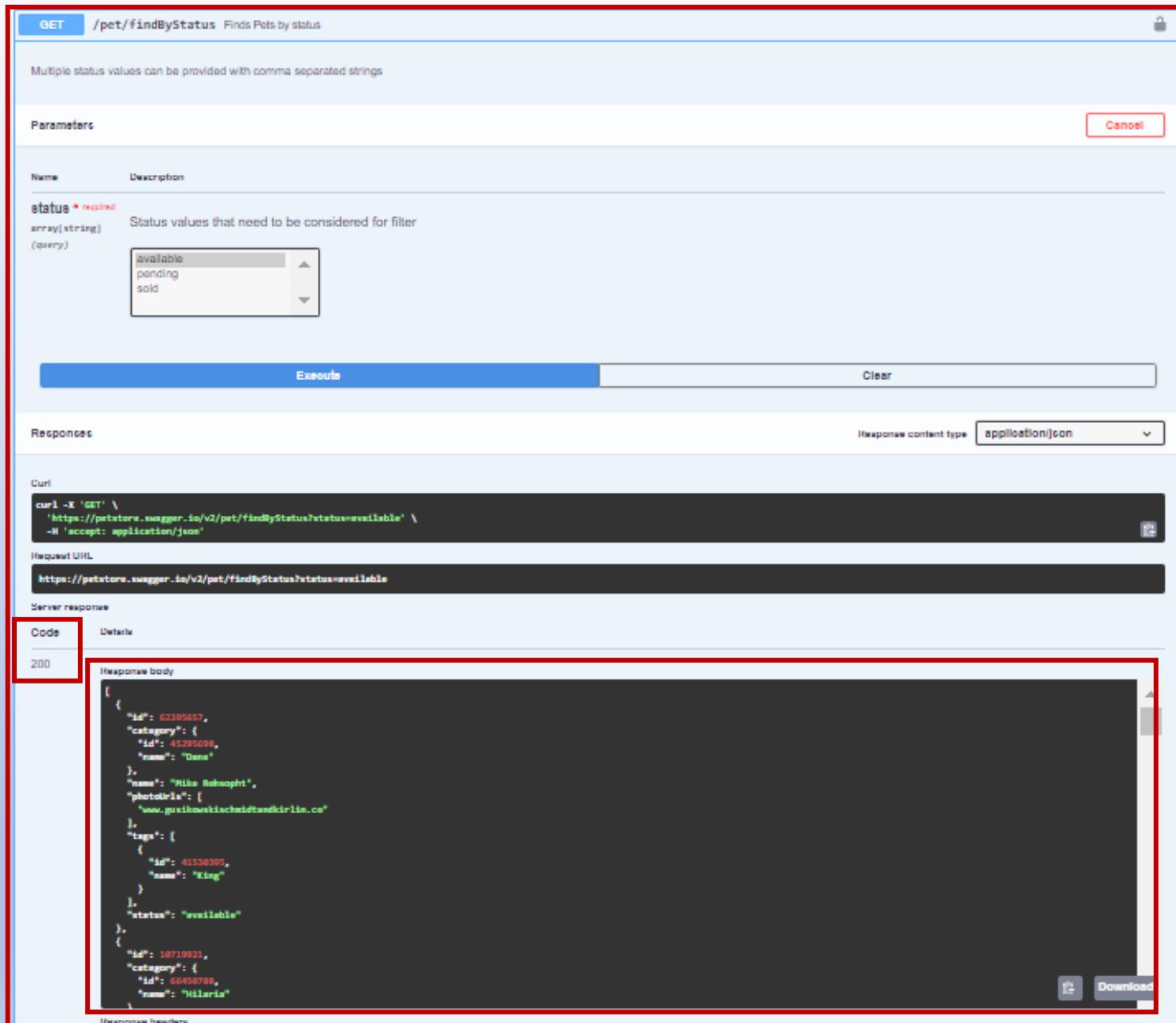
200

Response body

```
{
  "id": 63395657,
  "category": {
    "id": 45295699,
    "name": "Dane"
  },
  "name": "Mike Rohmopt",
  "photoUrl": "www.gutkowskischmidtundkirlin.co",
  "tags": [
    {
      "id": 41530395,
      "name": "King"
    }
  ],
  "status": "available"
},
{
  "id": 10719921,
  "category": {
    "id": 66450798,
    "name": "Hilaris"
  }
}
```

Download

Response headers



Swagger DOCUMENTS

DELETE /pet/{petId} Deletes a pet

Parameters

Name	Description
api_key	string (header) api_key
petId	* required integer(\$int64) (path) Pet id to delete 14859388

Cancel

Execute Clear

Responses

Response content type application/json

Curl

```
curl -X 'DELETE' \
'https://petstore.swagger.io/v2/pet/14859388' \
-H 'accept: application/json'
```

Request URL

```
https://petstore.swagger.io/v2/pet/14859388
```

Server response

Code Details

404

Error:

Response headers

```
access-control-allow-headers: Content-Type,api_key,Authorization
access-control-allow-methods: GET,POST,DELETE,PUT
access-control-allow-origin: *
date: Tue,13 Apr 2021 11:20:25 GMT
server: Jetty(9.2.9.v20150224)
```

Responses

<https://petstore.swagger.io/>

1- Token : Bir kaydi silmek istedigimizde yetkimiz olup olmadigini kontrol ediyor

2- Status code : 404
(Not Found)

3- Response body

Farkli ornek icin :
<https://demoqa.com/swagger/>



BOLUM 2

- 1- POSTMAN GENEL TANITIMI, VARIABLE OLUSTURMA,
GET VE POST REQUEST OLUSTURMA
 - 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA, FARKLI ENDPOINT'LER KULLANMA
 - 3- TRELLO KURULUMU VE API KULLANARAK TRELLO
UZERINDE ISLEMLER YAPMA
-



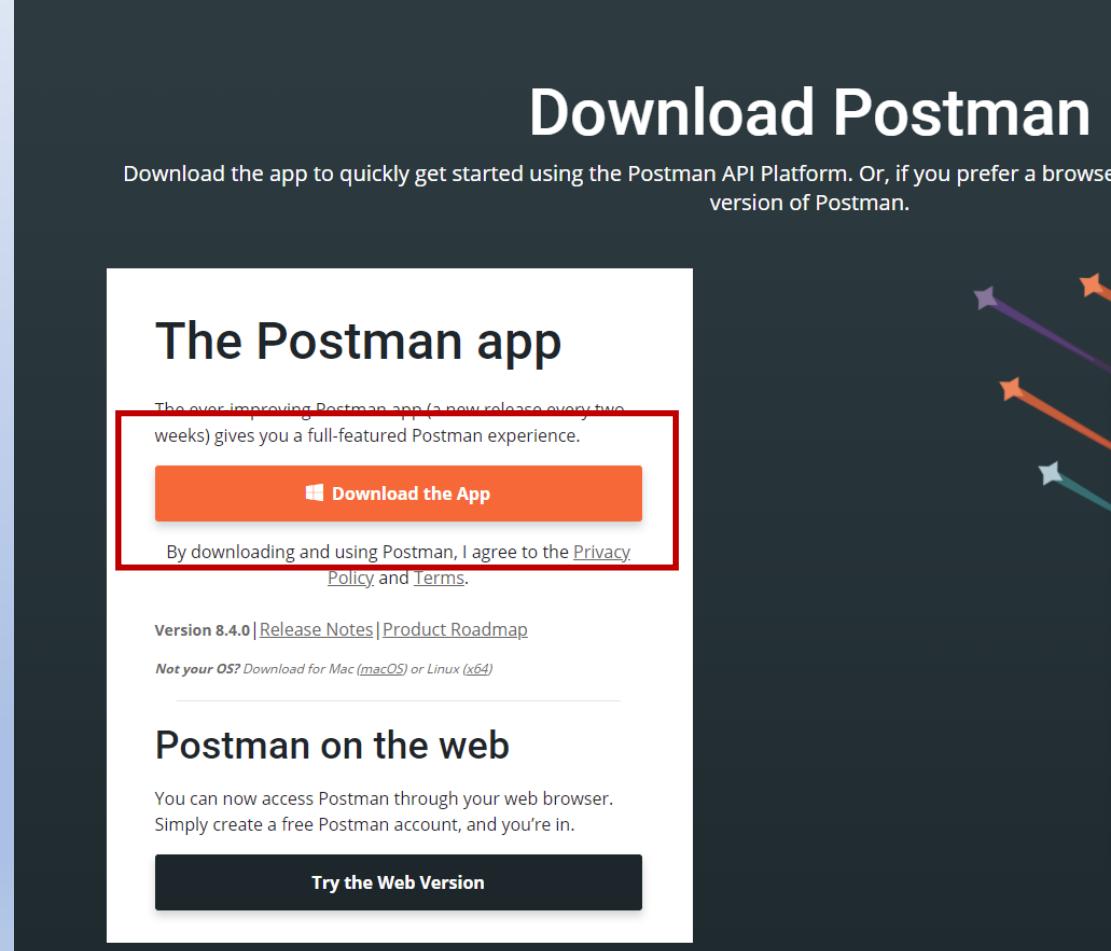
<https://www.postman.com/downloads/>

Postman nedir ?

Postman API sorgusu yapabilecegimiz ve istenirse manuel API testi yapabilecegimiz oldukça kullanışlı bir rest client'tır.

Eğer elinizde var olan bir Endpoint ile neler yapabileceginizi gormek veya API'yi hızlıca test etmek, sonuçlarını incelemek isterseniz Postman kullanıcı dostu arayüzü ile işlerinizi oldukça kolaylaşdıracaktır.

Postman ile yaptiginiz sorguları oluşturacaginiız collection'lar altına organize edebilir, hazırladığınız request'leri export edip arkadaşlarınızla da paylaşabilirsiniz.



The screenshot shows the official Postman download page. At the top, it says "Download Postman". Below that, a sub-headline reads: "Download the app to quickly get started using the Postman API Platform. Or, if you prefer a browser version of Postman." The main section is titled "The Postman app" and contains the text: "The ever-improving Postman app (a new release every two weeks) gives you a full-featured Postman experience." It features a prominent orange "Download the App" button. Below the button, a note states: "By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#)." At the bottom of this section, links are provided for "Version 8.4.0 | Release Notes | Product Roadmap" and "Not your OS? Download for Mac ([macOS](#)) or Linux ([x64](#))". Another section below is titled "Postman on the web" with the text: "You can now access Postman through your web browser. Simply create a free Postman account, and you're in." It includes a "Try the Web Version" button. The background of the page features abstract orange and purple star-like shapes.



The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Collections (highlighted with red box 1), APIs, Environments, Mock Servers, and Monitors. Below the sidebar, a message says "You don't have any collections". There are buttons for "Create Collection" and "History" (highlighted with red box 2). The main area has a search bar at the top right. Below it, a "New" button is highlighted with red box 3. A "GET Untitled Request" card is shown, with its method (GET) highlighted with red box 4. To the right of the card are "Save" (highlighted with red box 5) and "Send" (highlighted with red box 6) buttons. The "Send" button has a dropdown arrow. Below the card, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. Under the Headers tab, there's a table for "Query Params" with columns for KEY, VALUE, and DESCRIPTION. A "Bulk Edit" button is at the bottom of this table. Red box 7 highlights the "Save" button. Red box 8 highlights the top navigation bar. Red box 9 highlights the "Send" button's dropdown arrow. At the bottom, there's a "Find and Replace" and "Console" button, and a "Hit Send to get a response" message with a rocket ship icon.

- 1- Collection bolumu**
- 2- History**
- 3- HTTP metodu**
- 4- Endpoint satiri**
- 5- Request'i calistirmak icin**
- 6- Request ismi**
- 7- Request'i kaydetmek icin**
- 8- Request sekmleri, yeni bir request icin + kullanilabilir**
- 9- Request icin detay ekleyebilecegimiz sekmler**



Untitled Request

GET Enter request URL

1 Params 2 Authorization 3 Headers (6) 4 Body 5 Pre-request Script 6 Tests Settings

Query Params

KEY	VALUE
Key	Value

- 1- Params:** gerekli parametreleri key, value olarak gönderebilir.
- 2- Authorization :** API'lere erişmek için yetkilendirme işlemleri için kullanılır, bir kullanıcı adı / şifre ya da bearer token vs bu kısımdan set edilir.
- 3- Headers :** İhtiyacınıza göre header parametreleri buradan gönderilir, content type JSON gibi.
- 4- Body :** Bir istekteki ayrıntıların özelleştirebileceği kısımdır.
- 5- Pre-request Script :** Bir request gönderilmeden önce çalışan kısımdır, genellikle request'in doğru çalışması için gereken ortam değişkenlerinin set edildiği yerdir.
- 6- Tests :** Bu kısım API'yi test edeceğimiz test scriptlerinin yazıldığı kısımdır.



COLLECTION OLUSTURMA

The screenshot shows the Postman interface with the following elements highlighted:

- 1** The "Collections" icon in the sidebar.
- 2** The "New" button in the top right corner of the main workspace.
- 3** A small number "2" indicating the count of collections.
- 4** A character holding a folder icon.
- 5** The text "You don't have any collections".
- 6** The text "Collections let you group related requests, making them easier to access and run."
- 7** The "Create Collection" button at the bottom.

1- Collection sekmesini secelim

2- + isaretine basarak veya NEW butonuna basip
COLLECTION secerek collection olusturalim

The screenshot shows the "New Collection" dialog box with the following elements highlighted:

- 1** The "API DErsleri" workspace name.
- 2** The "New" and "Import" buttons.
- 3** The "New Collection" input field containing "New Collection".
- 4** The "..." button next to the input field.

3- Mouse ile **New Collection** Yazisinin
uzerine geldiginizde gorunur olan ... yi
secip Rename ile collection'a
istedigimiz ismi verelim

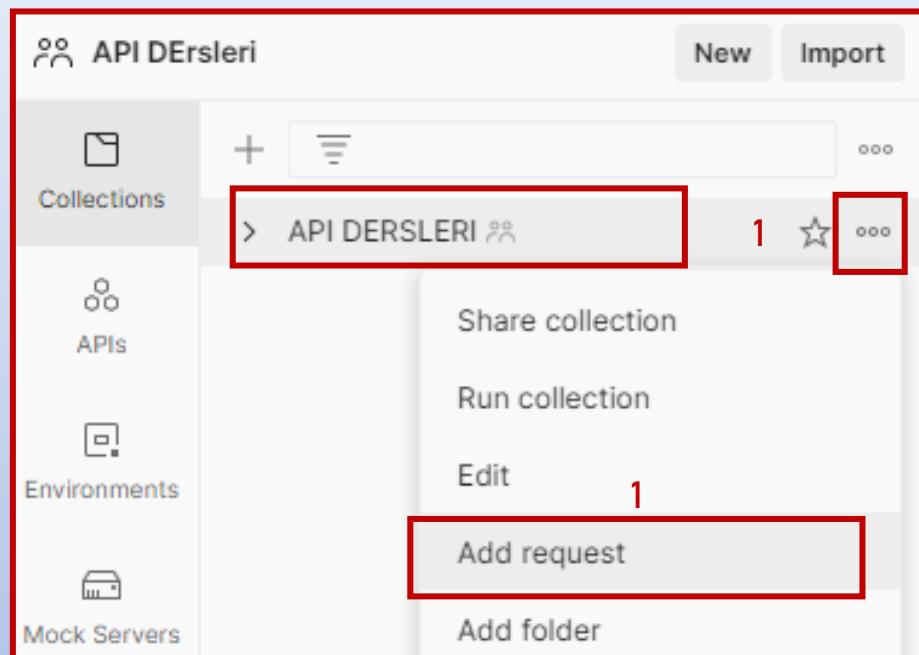
The screenshot shows the "New Collection" dialog box with the following elements highlighted:

- 1** The "API DErsleri" workspace name.
- 2** The "New" and "Import" buttons.
- 3** The "Overview" tab.
- 4** The "New Collection" input field containing "New Collection".
- 5** The "Authorization", "Pre-request Script", "Tests", and "Variables" tabs.
- 6** The "Watch" button.
- 7** The "0" count for tests.
- 8** The "Variables" button.

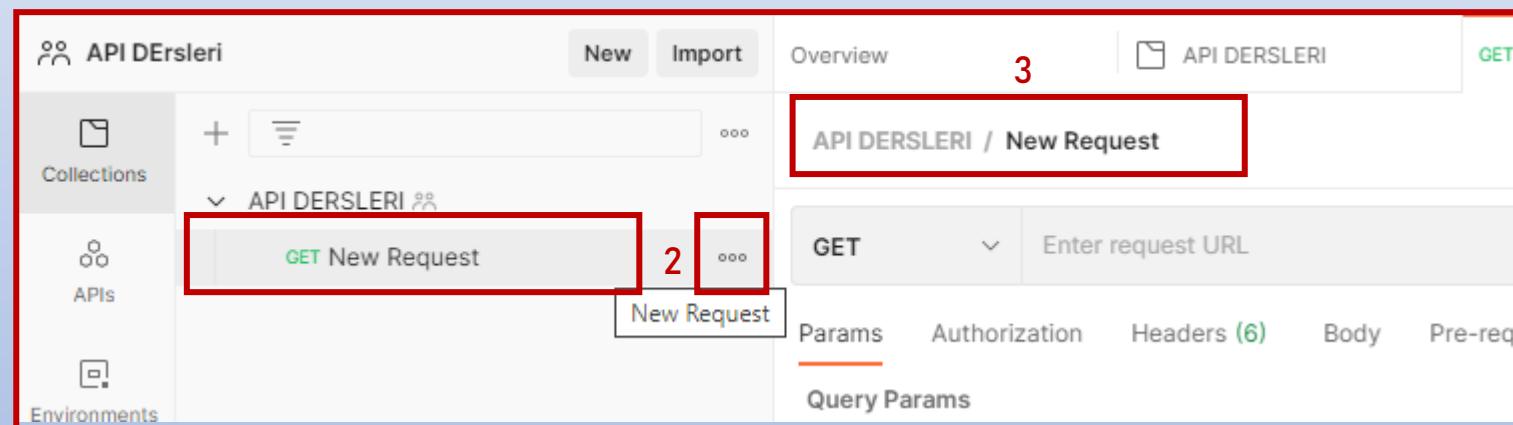
4- Veya mouse ile Request'lerin oldugu bolumdeki **New Collection**
yazisinin ustune geldigimizde ortaya cikan **kalem** isaretine tiklayip
collection'a istedigimiz ismi verelim



REQUEST OLUSTURMA



1- Mouse ile **API DERSLERI** olarak isimlendirdigimiz collection'in uzerine gelelim, cikan ... 'ya basalim ve acilan DROPODOWN menuden **Add request** 'i secelim



2- Mouse ile **New Request** Yazisinin uzerine geldiginizde gorunur olan ... yi secip Rename ile request'e istedigimiz ismi verelim

3- Veya mouse ile Request'lerin oldugu bolumdeki **New Request** yazisinin ustune geldigimizde ortaya cikan **kalem** isaretine tiklayip request'e istedigimiz ismi verelim



<https://restful-booker.herokuapp.com/>

Base URL		Method	Endpoints	Tanim
https://restful-booker.herokuapp.com	JSON	GET	/booking	tum rezervasyonlari listele
		GET	/booking/1	id ile rezervasyon goruntule
		POST	/booking?firstname=Ali&lastname=Can&totalprice=123&depositpaid=true&additionalneeds=Wifi	yenii rezervasyon olustur
		PATCH	/booking/12	Kismi guncelleme
		PUT	/booking/11	guncelleme
		DELETE	/booking/11	silme

Tabloya dikkat edersek her sorgunun basinda <https://restful-booker.herokuapp.com> yazmak zorundayiz. Tum sorgularda kullanilmasi gereken bu kisma **BASE URL** denir

Her sorguda base Url'l yeniden yasmak yerine istersek bunu bir **VARIABLE** olarak tanimlayabilir ve ihtiyac duyduğumuzda kolayca kullanabiliriz.

VARIABLE OLUSTURMA



The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Home, Workspaces, Reports, and Explore. Below these are sections for API Dersleri (Collections), APIs, Environments, Mock Servers, Monitors, and History. A context menu is open over a collection named "API DERSLERI". The menu items are numbered 1 through 4: 1. "Edit" (highlighted with a red box), 2. "Variables" (highlighted with a red box), 3. "herokuapBaseUrl" (variable name), and 4. "https://restful-booker.herokuapp.com" (initial value). The main workspace shows an "API DERSLERI" collection with tabs for Authorization, Pre-request Script, Tests, and Variables (which is active and highlighted with a green dot).

- 1- Collection'in uzerine gelelim, cikan ... 'ya basalim ve acilan DROPODOWN menuden **Edit** 'i secelim
- 2- Variable sekmesini secelim
- 3- Variable ismini yazalim
- 4- Variable'in degerini yazalim



OLUSTURULAN VARIABLE'I KULLANMA

A screenshot of the Postman application interface. On the left, there's a sidebar with icons for Collections, APIs, Environments, Mock Servers, and Monitors. The main area shows a collection named "API DERSLERİ" containing a single API endpoint named "GET Tum id'leri listeleme". This endpoint is set to a GET method and has a URL template starting with "{he". Below the URL, there are tabs for "Params", "Authorization", and "Query Params". A dropdown menu is open over the "Params" tab, listing two variables: "\$randomHexColor" (highlighted in blue) and "herokuapBaseUrl" (highlighted in orange). To the right of the dropdown, status information is shown: INITIAL https://restful-booker.herokuapp.com and CURRENT https://restful-booker.herokuapp.com. A "Settings" button is also visible. The entire screenshot is enclosed in a red border.

- 1- Variable'i kullanmak istediginiz alanda { yazip, variable isminin ilk birkac harfini yazalim
- 2- Sunulan alternatiflerden kullanmak istedigimiz variable'i secelim



GET REQUEST OLUSTURMA

The screenshot shows the Postman interface with the following steps highlighted:

1. Collections list: A collection named "Herokuapp" is selected.
2. Request list: A request titled "Herokuapp / Tum Rezervasyonlari listeleme" is selected.
3. Params tab: The "Params" tab is active.
4. Request URL: The URL is set to "GET {{HerokuappBaseUrl}}/booking ...".
5. Save button: The "Save" button is highlighted.
6. Send button: The "Send" button is highlighted.
7. Response body: The response body is displayed in JSON format, showing a list of bookings with IDs 6, 3, 9, 8, and 7.
8. Status bar: The status bar at the bottom shows "Status: 200 OK Time: 744 ms Size: 407 B".
9. Response format: The "JSON" dropdown in the response panel is highlighted.

8- Status kodunu kontrol edelim 200'lu bir sayı olduğundan emin olalım

9- Response'un formatını kontrol edelim

1- Collection satırında ...'a basıp Add request ile yeni sorgu oluşturalım

2- Request ismini değiştirelim

3- HTTP metodunu seçelim

4- EndPoint'i yazalım

5- Save butonu ile yaptığımız request'i kaydedelim.

6- Send butonu ile request'i çağıralım.

7- Response body'sinin geldiğini kontrol edelim



POST REQUEST OLUSTURMA

The screenshot shows the Postman interface with the following numbered steps:

1. Collections
2. Herokuapp / Rezervasyon Olusturma
3. POST
4. {{HerokuappBaseUrl}}/booking
5. Body
6. raw
7. Request body content:

```
1 {
2   "firstname": "Ahmet",
3   "lastname": "Bulut",
4   "totalprice": 500,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2021-06-10",
8     "checkout": "2021-06-15"
9   },
10  "additionalneeds": "wi-fi"
11 }
```
8. Save
9. Send
10. Response body content:

```
1 {
2   "bookingid": 11,
3   "booking": {
4     "firstname": "Ahmet",
5     "lastname": "Bulut",
6     "totalprice": 500,
7     "depositpaid": true,
8     "bookingdates": {
9       "checkin": "2021-06-10"
10    }
11  }
12 }
```
11. Status: 200 OK

9- Send butonu ile request'i cagiralim.

10- Response body'sinin geldigini kontrol edelim

11- Status kodunu kontrol edelim 200'lu bir sayi oldugundan emin olalim

1- Collection satirinda ...'a basip Add request ile yeni sorgu olusturalim

2- Request ismini degistirelim

3- HTTP metodunu POST yapalim

4- EndPoint'i yazalim

5- Body sekmesini secelim

6- Raw sekmesini secelim.

7- Request body'sini yazalim

8- Save butonu ile yaptigimiz request'i kaydedelim.



BOLUM 2

- 1- POSTMAN GENEL TANITIMI, VARIABLE OLUSTURMA,
GET VE POST REQUEST OLUSTURMA**

- 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA, FARKLI ENDPOINT'LER KULLANARAK
API SORGULARI YAPMA**





PUT REQUEST OLUSTURMA

The screenshot shows the Postman interface with the following numbered steps:

- 1- Click on the 'Collections' icon in the sidebar.
- 2- Select the 'Herokuapp' collection.
- 3- Click on the 'PUT Rezervasyon guncelleme' request.
- 4- Verify the endpoint: `PUT {{(HerokuappBaseUrl)}/booking/1}`.
- 5- Click on the 'Body' tab.
- 6- Click on the 'raw' radio button.
- 7- Paste the following JSON body:

```
1 {  
2   "firstname": "Mehmet",  
3   "lastname": "Hava",  
4   "totalprice": 400,  
5   "depositpaid": true,  
6   "bookingdates": {  
7     "checkin": "2021-06-10",  
8     "checkout": "2021-06-15"  
9   },  
10  "additionalneeds": "breakfast"  
11 }
```

- 8- Click the 'Save' button.
- 9- Click the 'Send' button.
- 10- Check the response body: '1 Forbidden'.
- 11- Check the status bar: 'Status: 403 Forbidden'.

- 8- Save butonu ile yaptigimiz request'i kaydedelim.
- 9- Send butonu ile request'i cagiralim.
- 10- Response body'sinin geldigini kontrol edelim
- 11- Status kodunu kontrol edelim 200'lu bir sayi oldugundan emin olalim

1- Collection satirinda
...'a basip Add request
ile yeni soru
olusturalim

2- Request ismini
degistirelim

3- HTTP metodunu PUT
yapalim

4- EndPoint'i yazalim

5- Body sekmesini
secelim

6- Raw sekmesini
secelim.

7- Request body'sini
yazalim



AUTHORIZATION

The screenshot shows the Postman interface with a red border around the main content area. At the top, there's a navigation bar with tabs like 'New', 'Import', and 'No Environment'. Below the bar, a collection named 'Herokuapp' is selected. A PUT request is being prepared with the URL `1 {{HerokuappBaseUrl}}/booking/1`. The 'Authorization' tab is highlighted with a red box. In the 'Type' dropdown, 'Basic Auth' is selected. A tooltip message is visible: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.' Below the dropdown, there are fields for 'Username' (admin) and 'Password' (password123), both enclosed in a red box. At the bottom of the request panel, there are tabs for 'Body', 'Cookies', 'Headers (8)', and 'Test Results'. The status bar at the bottom right shows 'Status: 403 Forbidden'.

- 1- Authorization sekmesine gelelim
- 2- Authorization type drop-down menusunden Basic Auth. secelim
- 3- Username : admin , password : password123 yazalim



PUT REQUEST OLUSTURMA

The screenshot shows the Postman application interface. On the left, the sidebar includes sections for Collections, APIs, Environments, Mock Servers, Monitors, and History. The main workspace displays a collection named "Herokuapp" with several API endpoints listed under it. A specific PUT request titled "Rezervasyon guncelleme" is selected. The request details show the method as PUT, the URL as `{{HerokuappBaseUrl}}/booking/1`, and the Body tab selected. The JSON body is defined as follows:

```
1 {
2   "firstname": "Mehmet",
3   "lastname": "Hava",
4   "totalprice": 400,
5   "depositpaid": true,
6   "bookingdates": {
7     "checkin": "2021-06-10",
8     "checkout": "2021-06-15"
9   },
10  "additionalneeds": "breakfast"
11 }
```

Below the request details, the response section is visible, showing a status of "Status: 200 OK". The response body is identical to the request body, indicating a successful update. The response body is highlighted with a red box, and the status code "Status: 200 OK" is also highlighted with a red box.

Response body'sinin geldigini ve Status kodunu kontrol edelim 200'lu bir sayi oldugunu kontrol edelim



PATCH REQUEST OLUSTURMA

The screenshot shows the Postman interface with the following numbered steps:

- 1- Collection satirinda ...'a basip Add request ile yeni sorgu olusturalim
- 2- Request ismini degistirelim
- 3- HTTP metodunu PATCH yapalim
- 4- EndPoint'i yazalim
- 5- Body sekmesini secelim
- 6- Raw sekmesini secelim.
- 7- Request body'sini yazalim ve Authorization'i yapalim
- 8- Save butonu ile yaptigimiz request'i kaydedelim.
- 9- Send butonu ile request'i cagiralim.
- 10- Response body'sinin geldigini kontrol edelim
- 11- Status kodunu kontrol edelim 200'lu bir sayi oldugundan emin olalim

Details from the screenshot:

- Collection: Herokuapp
- Request Name: Herokuapp / Kismi guncelleme
- HTTP Method: PATCH
- Endpoint: {{HerokuappBaseUrl}}/booking/10
- Body Type: raw (selected)
- Request Body (Raw):

```
1
2   ...
3   "firstname": "Hasan",
4   "lastname": "Kova"
5
```
- Response Body (Pretty):

```
1
2   {
3     "firstname": "Hasan",
4     "lastname": "Kova",
5     "totalprice": 529,
6     "depositpaid": false,
7     "bookingdates": {
8       "checkin": "2021-04-11",
9       "checkout": "2021-05-09"
10    }
11 }
```
- Status: 200 OK

8- Save butonu ile yaptigimiz request'i kaydedelim.

9- Send butonu ile request'i cagiralim.

10- Response body'sinin geldigini kontrol edelim

11- Status kodunu kontrol edelim 200'lu bir sayi oldugundan emin olalim

1- Collection satirinda ...'a basip Add request ile yeni sorgu olusturalim

2- Request ismini degistirelim

3- HTTP metodunu PATCH yapalim

4- EndPoint'i yazalim

5- Body sekmesini secelim

6- Raw sekmesini secelim.

7- Request body'sini yazalim ve Authorization'i yapalim



DELETE REQUEST OLUSTURMA

The screenshot shows the Postman interface with the following steps highlighted:

1. Collections list: A collection named "Herokuapp" is selected.
2. Request list: A request titled "Herokuapp / silme" is selected.
3. Params tab: The "Authorization" param is selected.
4. Request URL: The URL is set to `({{HerokuappBaseUrl}})/booking/11`.
5. Save button: The "Save" button is highlighted.
6. Send button: The "Send" button is highlighted.
7. Response body: The response body shows the word "Created".
8. Status code: The status code is shown as "status: 201 Created".
9. Response format: The response format is set to "Text".

8- Response body'sinin kontrol edelim

9- Status kodunu kontrol edelim 200'lu bir sayı olduğundan emin olalım

10- Response'un formatını kontrol edelim

1- Collection satırında ...'a basıp Add request ile yeni soru oluşturalım

2- Request ismini degistirelim

3- HTTP metodunu DELETE yapalım

4- EndPoint'i yazalım

5- Authorization'i yapalım

6- Save butonu ile yaptığımız request'i kaydedelim.

7- Send butonu ile request'i çağıralım.



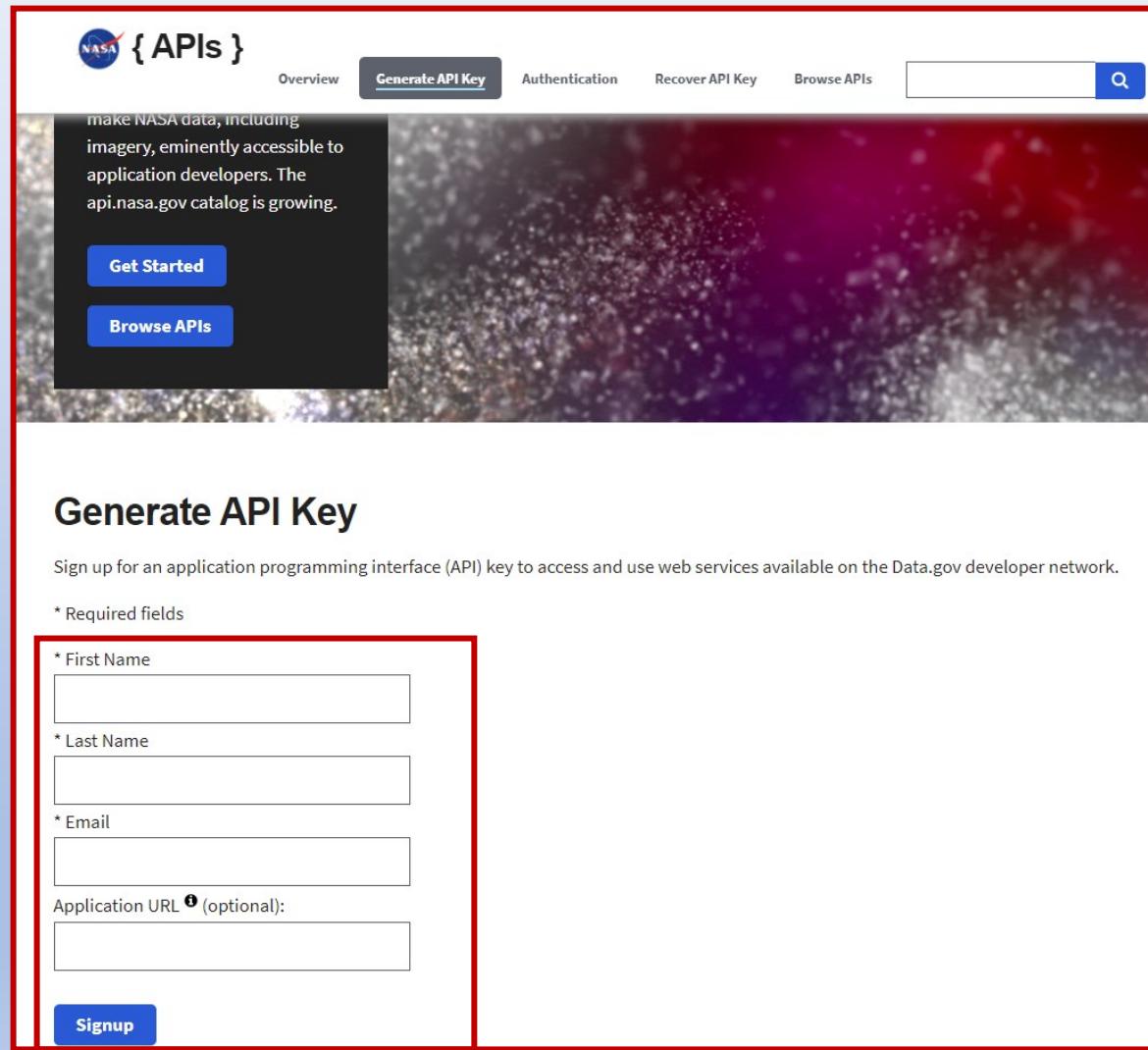
<https://jsonplaceholder.typicode.com>

A screenshot of the JSONPlaceholder homepage. The URL in the address bar is "https://jsonplaceholder.typicode.com". The page title is "JSONPlaceholder". The main heading is "JSON Placeholder". Below it is a sub-heading "Free fake API for testing and prototyping." and a note "Powered by [JSON Server](#) + [LowDB](#)". At the bottom, there is a small note "As of Dec 2020, serving ~1.8 billion requests each month." A red box highlights the "Guide" link in the top navigation bar.

- 1- Getting a resource
Id ile kayit sorgulama
- 2- Listing all resources
Tum kayitlari listeleme
- 3- Creating a resource
Yeni kayit olusturma
- 4- Updating a resource
Kayit guncelleme
- 5-Patching a resource
Kismi guncelleme
- 6-Deleting a resource
Kayit silme

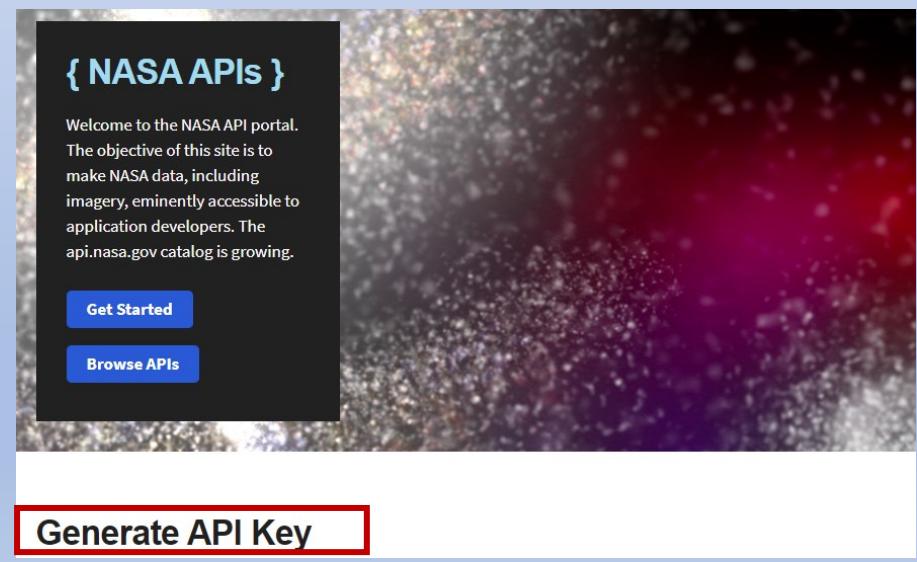
Important: resource will not be really updated on the server but it will be faked as if.

Bu API olusturma,silme,degistirme requestlerini data base'e islemeden fake olarak yapmaktadır



The screenshot shows the NASA APIs portal's "Generate API Key" page. At the top, there are navigation links: Overview, **Generate API Key** (which is highlighted in blue), Authentication, Recover API Key, and Browse APIs. A search bar with a magnifying glass icon is also present. The main content area has a dark background with a starry space image. On the left, there is descriptive text about the NASA API catalog and two blue buttons: "Get Started" and "Browse APIs". The right side contains the "Generate API Key" form. It includes fields for First Name, Last Name, Email, and Application URL (optional), all marked with an asterisk to indicate they are required. Below the form is a "Signup" button. The entire "Generate API Key" section is highlighted with a red border.

- 1- <https://api.nasa.gov> adresine gidelim
- 2- İsim, soyisim ve mail bilgilerini girip Signup butonuna basalim ve register olalım
- 3- Acilan sayfada Generate API Key bolumunde yazılı olan api_key'ini kaydedelim



The screenshot shows a confirmation message from the NASA APIs portal. It says "Welcome to the NASA API portal. The objective of this site is to make NASA data, including imagery, eminently accessible to application developers. The api.nasa.gov catalog is growing." It features the same "Get Started" and "Browse APIs" buttons as the previous screen. Below this, a large red box contains the text "Generate API Key".



1- Üst bölümde bulunan Browse APIs sekmesini seçelim

The screenshot shows the Postman interface for the NASA API. At the top, there's a navigation bar with the NASA logo, a search bar containing '{ APIs }', and several buttons: Overview, Generate API Key, Authentication, Recover API Key, and a red-bordered 'Browse APIs' button. Below the navigation is a large section titled 'Browse APIs' with a red border. This section contains a search bar with a 'Search' button and a list of NASA APIs, each with a '+' sign to its right. The listed APIs include:

- APOD: Astronomy Picture of the Day
- Asteroids NeoWs: Near Earth Object Web Service
- DONKI: Space Weather Database Of Notifications, Knowledge, Information
- Earth: Unlock the significant public investment in earth observation data
- EONET: The Earth Observatory Natural Event Tracker
- EPIC: Earth Polychromatic Imaging Camera
- Exoplanet: Programmatic access to NASA's Exoplanet Archive database
- GeneLab: Programmatic interface for GeneLab's public data repository website
- Insight: Mars Weather Service API
- Mars Rover Photos: Image data gathered by NASA's Curiosity, Opportunity, and Spirit rovers on Mars
- NASA Image and Video Library: API to access the NASA Image and Video Library site at images.nasa.gov
- TechTransfer: Patents, Software, and Tech Transfer Reports
- Satellite Situation Center: System to cast geocentric spacecraft location information into a framework of (empirical) geophysical regions

2- Nasa API'ndan kullanmak istediğiniz endpoint seçelim ve adımları takip ederek api request yazalım



POSTMAN

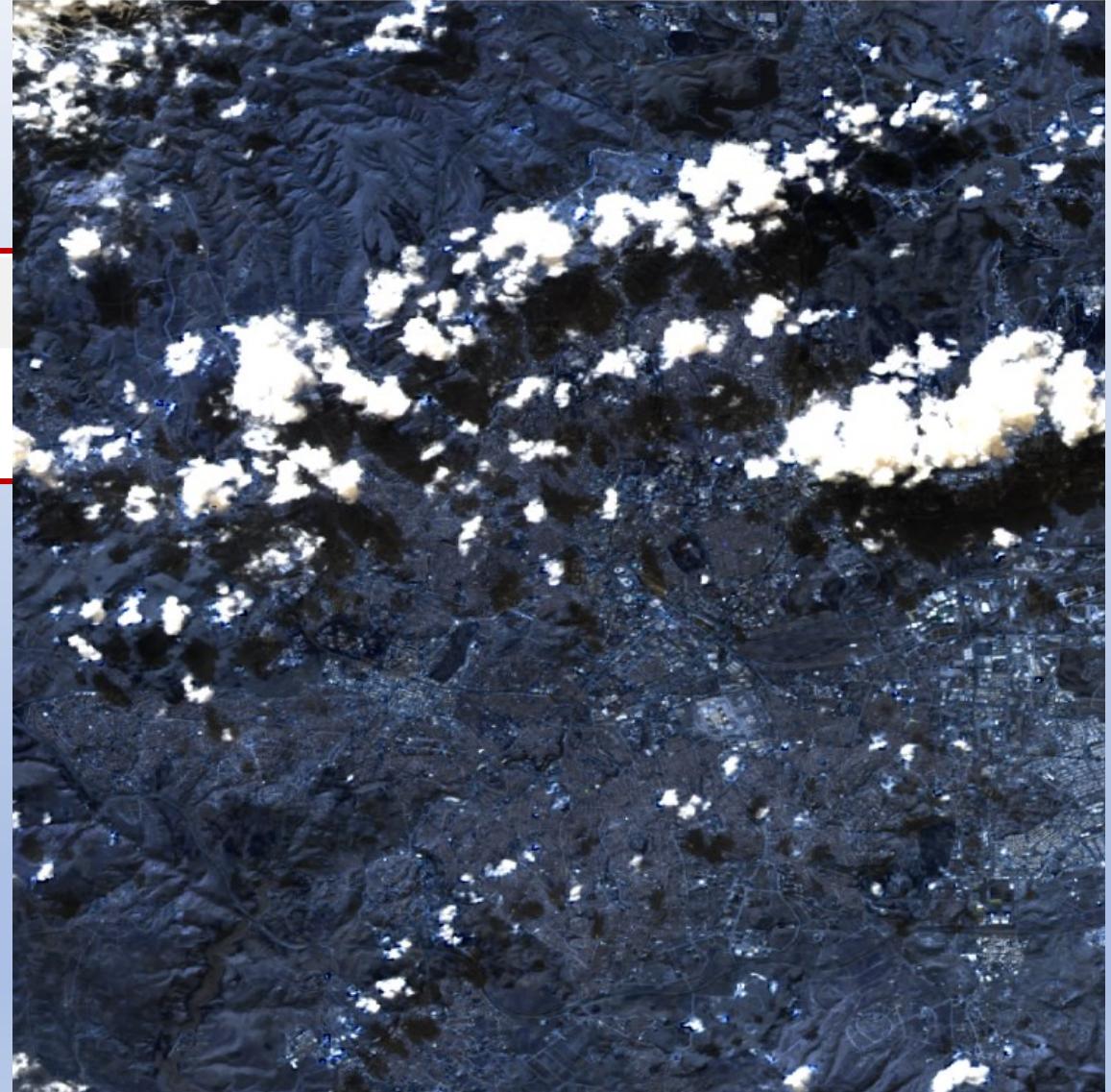
Ornegin Earth'u secebılır istediginiz sehrin
değerlerini girip görüntüsünü alabilirsiniz

Earth: Unlock the significant public investment in earth observation data

Earth

Ankara için örnek soru :

https://api.nasa.gov/planetary/earth/imagery?lon=32.866287&lat=39.925533&date=2021-02-01&api_key=zb1JohVklBe4ctl4GfgQqkzhQXsZ1f0bVgAidcAJ&dim=0.25





Veya APOD'u secebılır istediginiz tarihi girerek gunun astronomy fotografini alabilir ve uygulamalarinizda kullanabilirsiniz

APOD: Astronomy Picture of the Day

APOD

ornek soru :

https://api.nasa.gov/planetary/apod?api_key=zb1JohVklBe4ctI4GfgQqkzhQXsZ1f0bVgAidcAJ



<https://collectapi.com/tr/>

Kategoriler
Tümü
Ücretsiz API'ler
AI Yapayzeka
Akaryakıt
Corona
E-Ticaret
Eczane
Ekonomi
Haberler
Hal Fiyatları
Harita
Hava Durumu
IP Adres
Instagram
Kitap



POSTMAN

Whatsapp Business API Official WhatsApp Business API. WhatsApp'ın resmi c	Haberler API 7 farklı ülkeden, ülkelerin anadillerine göre ge	IMDb API IMDb sitesinde isimle yada id ile arama yaparak film
Whatsapp Business Sandbox WhatsApp Business API'yi test edebilmeniz için sand	Hava Durumu API Dünyanın her yerindeki hava durumunu günü	Araç Tanıma API Resimlerdeki araçların plakalarını yakalayı
COVID-19 Koronavirus İstatistik API Dünya genelindeki COVID-19 virüsünün etkilerini Dü	Faiz Oranları API Bankaların genel faiz oranları ve farklı kredi ti	Plaka Tanıma API Resimdeki araçların plakalarını yakalayı
Instagram DM API Unofficial Instagram DM API kullanarak HTTP istekle	Şans Oyunları API Son sayısal ve süper lotoda kazanan numaral	Nesne Tanıma API Resimde neler olduğunu, hangi nesnelerin bulunduğu
Akaryakıt Fiyatları API Şehirlere göre farklı akaryakıt istasyonlarındaki benz	IP Adresi API ip adresini girerek lokasyon bilgisi, şehir bilgi	Duygu Analiz API Resmin ya da yazının içeriğindeki duyguları dereceleriley
Nöbetçi Eczane API Türkiye il ve ilçelerdeki günün nöbetçi eczanelerini ge	Namaz Vakitleri API istedığınız şehrin namaz vakitlerini getiren, se	Çiplaklılık Algılama API Resimlerin çiplaklı içeriği içermediğini öğrenin.

<https://collectapi.com/tr/>



1- Ekrанин sağ üst tarafındaki **Giriş Yap** sekmesine gidelim

A screenshot of a web browser showing a navigation bar with links for "Dokümanlar", "+ API Yayınlama", "Giriş Yap", and "Türkçe". Below the navigation bar is a large red rectangular box highlighting the "Giriş Yap" button, which is white with black text and has a small arrow icon.

2- Dilediginiz yontemle sisteme giriş yapalim

A screenshot of a login form titled "Giriş Ekranı". It features a logo at the top left and two main buttons: "Email ile Giriş Yap" (highlighted with a red box) and "Google ile Giriş Yap". Below the buttons is a note about accepting terms and conditions. A red box highlights the "Email ile Giriş Yap" button.

3- Ekraniн sağ üst tarafındaki **Hesap** sekmesinden **Profil'i** secelim

A screenshot of a web browser showing a navigation bar with links for "Dokümanlar", "+ API Yayınlama", "Hesap" (highlighted with a red box), and "Türkçe". Below the navigation bar is a large red rectangular box highlighting the "Profil" link under the "Hesap" menu.

4- API Token'i gorunur hale getirelim ve kopyalayalim

A screenshot of a user profile page titled "Profil". It shows a user icon, the name "Mehmet Bulutluoz", and the date "9 Mayıs 2021 yılında katıldı". Below this is a table with three columns: "Bilgiler", "API Token" (highlighted with a red box), and "Bağlı Hesaplar". The "API Token" row contains a key icon, a redacted token value, and two buttons: "Görüntüle" and "Kopyala". A red box highlights the "API Token" column header.



POSTMAN



Akaryakıt Fiyatları API

Şehirlere göre farklı akaryakıt istasyonlarındaki benzin ve dizel fiyatlarına ulaşabilirsiniz.

Görüntüle >

Sorguyu yaparken Headers menusunde “**authorization**” isminde yeni bir header olusturalim ve deger olarak siteden aldigimiz “**apikey token**” i yazalim

CollectAPI / Akaryakit fiyat

GET https://api.collectapi.com/gasPrice/turkeyGasoline?district=kadikoy&city=istanbul&content-type=application/json

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
district	kadikoy	
city	istanbul	
content-type	application/json	
authorization	apikey 7o0llqcDndqPfWsEQBrke7:63xbVxUvHLL0CeD3nubi...	

Key Value Description

Body Cookies (1) Headers (21) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   "result": [
3     {
4       "marka": "Aytemiz",
5       "benzin": 7.7,
6       "katkili": "-"
7     },
8     {
9       "marka": "Alpet",
10      "benzin": 7.71,
11      "katkili": 7.71
12    },
13    {
14      "marka": "M Oil",
15      "benzin": 7.72,
16      "katkili": "-"
17    }
]
```

Status: 200 OK Time: 5.01 s Size: 1.45 KB Save Response



Nöbetçi Eczane API
Türkiye İl ve ilçelerdeki günün nöbetçi eczanelerini getiren API. API sonucunda eczane adres, telefon, konum bilgilerini alabilirsiniz.

Görüntüle >

Sorguyu yaparken Headers menusunde “**authorization**” isminde yeni bir header olusturalim ve deger olarak siteden aldigimiz “**apikey token**” i yazalim

CollectAPI / Nobetçi Eczane

GET https://api.collectapi.com/health/dutyPharmacy?ilce=%C3%87ankaya&il=Ankara

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Postman-Token <calculated when request is sent>
Host <calculated when request is sent>
User-Agent PostmanRuntime/7.28.0
Accept */*
Accept-Encoding gzip, deflate, br
Connection keep-alive
authorization apapikey 7o0llqcDndqPfWsEQBrke7:63xbVxUvHLL0CeD3nubi4O

Status: 200 OK Time: 370 ms Size: 2.45 KB Save Response

Body Cookies (1) Headers (21) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "success": true,  
3   "result": [  
4     {  
5       "name": "BİZİM ZİYA",  
6       "dist": "ÇANKAYA",  
7       "address": "SOKULLU MEHMET PASA CAD. NO.125/B DIKMEN",  
8       "phone": "3124811191",  
9       "loc": "39.880052,32.834802"  
10    },  
11    {  
12      "name": "VENİ ÇINAR",  
13      "dist": "ÇANKAYA",  
14      "address": "TURAN GÜNES BULVARI NO:11/B YILDIZ",  
15      "phone": "3124395553",  
16      "loc": "39.876438,32.864116"  
17    },  
18    {  
19      "name": "YEŞİLBAĞ",  
20    }  
21  ]  
22}  
23
```



BOLUM 2

- 1- POSTMAN GENEL TANITIMI, VARIABLE OLUSTURMA,
GET VE POST REQUEST OLUSTURMA
 - 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA, FARKLI ENDPOINT'LER KULLANARAK
API SORGULARI YAPMA
 - 3- TRELLO KURULUMU VE API KULLANARAK TRELLO
UZERINDE ISLEMLER YAPMA
-



<https://trello.com/>

Trello, takımların ilerleme kaydetmesine yardımcı olur.

İş birliği yapın, projeleri yönetin ve üretkenlikte yeni zirvelere ulaşın. Büyük genel merkezlerden ev ofislere kadar her yerde farklı yöntemlerle takım çalışması yapılabilir. Hepsini Trello ile başarın.

E-posta

Kaydolun -
Ücretsizdir!

- 1- TRELLO sayfasından Kaydolun butonuna basalim
- 2- Acilan giris sayfasından uygun secenegi secelim
- 3- Mailinize gonderilen onay kodunu girip hesabi aktiflestirelim

A screenshot of the Trello sign-up page. It features a large blue header with the Trello logo. Below it is a form with a red border. The form has a text input field labeled "E-posta girin" and a note below it stating "Kayıt olduğunuzda, [Hizmet Kullanım Şartlarını](#) ve [Gizlilik Politikasını](#) okuyup kabul ettiğinizi onaylamış olursunuz." There are four "Devam" buttons with different social logins: Google (with a G logo), Microsoft (with a blue square logo), Apple (with a black logo), and Slack (with a multi-colored logo). At the bottom of the form is a link "Zaten bir hesabınız var mı? Oturum Açın".



Hoş geldiniz, mehmet bulutluoz

Çalışma Alanınızı adlandırın *

Çalışma Alanı türünüze seçin *

Takım üyelerinizi ekleyin *İstedığınız kadar fazla e-posta adresi ekleyin*

Başka bir e-posta adresi girin...

İpucu! Herhangi bir zamanda takım üyeleri ekleyebilirsiniz

Devam

- 1- Calisma alaniniza bir isim verelim
- 2- Faaliyet alaninizi secin
- 3- Devam tusuna basin

Trello

Business Class'ı 30 gün boyunca deneyin

Trello Business Class iş bitirici takımlar için tasarlanmıştır. 30 gün boyunca takımınız tüm Trello Business Class özelliklerini ücretsiz olarak kullanabilir.

Ücretsiz Çalışma Alanı

- 10 Adede Kadar Takım Panosu
- Pano başına 1 Power-Up
- Sınırlı Otomasyonlar

Business Class Çalışma Alanı

- Sınırsız takım Panosu
- Sınırsız Power-Up
- Gelişmiş otomasyonlar
- Yönetici rolleri ata
- 1 gün içinde destek

* Gelişmiş yapılacaklar listeleri, takım şablonları ve daha fazlası gibi ek özellikler!

30 günlük ücretsiz denemeyi başlat

ŞURADAKİ TAKIMLARCA GÜVENİLİR: Google FORTNITE PELOTON

Deneme süresi sona erdikten sonra, yıllık faturalandırma kaydıyla Business Class'ı kullanıcı başına aylık yalnızca \$9.99 karşılığında devam ettirin. [Her zaman Trello'nun ücretsiz sürümüne dönme seçeneklerini olacak.](#)

Business Class olmadan başla

- 4- “Business Class olmadan basla” yi secelim

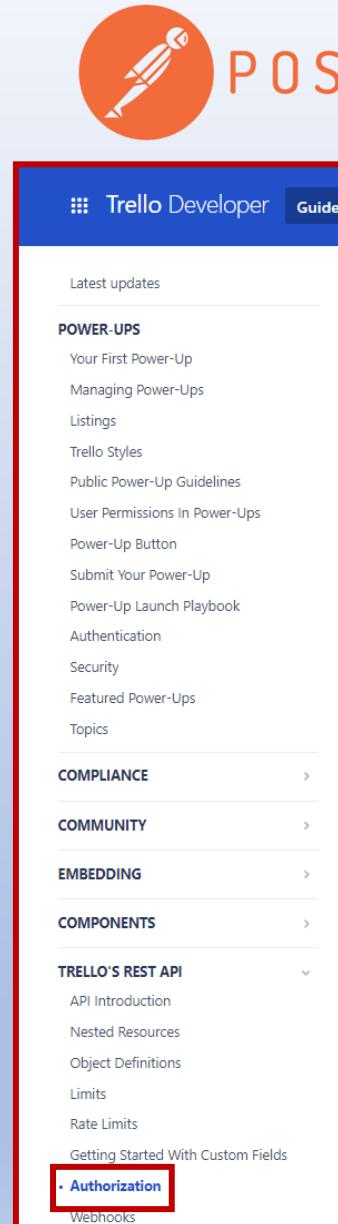
API KEY ALMA



The screenshot shows the Trello Developer documentation homepage. At the top left is the Trello logo. Below it, the text "Build on top of Trello by creating a Power-Up." is displayed. Underneath this, there are two main navigation links: "REST APIs" and "Getting started". The "Getting started" link is highlighted with a red border. To the right of these links, there are two sub-links: "Intro to Power-Ups" and "Trello".

1- <https://developer.atlassian.com/>
Sayfasının alt kısmında bulunan Trello bolumunden **Getting started'i** seçelim

2- Sol taraftaki menuden **TRELLO'S REST API** altında [Authorization](#)'i seçelim



This screenshot shows the "Trello Developer" documentation page with a specific focus on the "Authorization" section. The "Trello Developer" logo is at the top left. The main content area has a sidebar on the left with categories like "POWER-UPS", "COMPLIANCE", "COMMUNITY", "EMBEDDING", "COMPONENTS", and "TRELLO'S REST API". Under "TRELLO'S REST API", the "Authorization" link is highlighted with a red border. The main content area contains sections for "Introduction" and "Authorizing A Client".

Authorizing With Trello's REST API

Introduction

Trello's API uses token-based authentication to grant third-party applications access to the Trello API. Once a Trello user has granted an application access to their Trello account and data, the application is given a token that can be used to make requests to the Trello API on behalf of the user.

There are two ways to authorize a client and receive a User Token. The first is via our `/authorize` route, the second is via basic OAuth1.0. We'll cover the former now. If you'd rather use OAuth, you can skip ahead to [Using Basic OAuth](#).

Authorizing A Client

To begin the authentication process, you need an API key. Every Trello user is given an API key. You can retrieve your API key by logging into Trello and visiting <https://trello.com/app-key>.

Because the API key is tied to the user, it is often a good idea to create a Trello user specifically for building a single application or integration. This ensures that a third-party's integration is disassociated from a third-party integration's developer's Trello account.

3- Ortadaki bolumden [https://trello.com/app-key/](https://trello.com/app-key) linkini kullanarak api-key'inize ulasip kaydedelim

TOKEN ALMA

Geliştirici API Anahtarları

Anahtar:

df524c4a1d9e349a462db5b5f081bfe9

Andaç:

Birçok geliştiricinin her kullanıcından uygulamanızı [doğrulamasını](#) istemesi gerekecektir. Kendiniz için bir uygulama geliştirmeyi düşünüyorsanız veya yerel test yapıyorsanız, manuel olarak [Belirteç](#) oluşturabilirsiniz.

1- Api key gösterilen sayfada alt bolumunden **Belirtec'i** secelim

2- En alt bolumde **Izin Ver** butonuna basalim

3- Verilen belirtec numarası ile kendi trello hesabiniza ulasabilir veya baskalarinin kullanımına acabilirsiniz



Server Token siz devre dışı bırakana kadar hesabınızı kullanabilecek.



Server Token Trello'ya bağlı değildir ve içeriklerinize erişime, izin verdiğinizde tüm ilgili riskleri ve sorumlulukları kabul etmiş olursunuz.

Server Token şunları yapabilecek:

- Adınızı ve kullanıcı adınızı okuyun
- Kendiniz olarak yorum yapın
- E-posta adresini okuma
- Enterprise'larınızı okuyun
- Enterprise'larınızı güncelleyin ve yönetin
- Bütün panolarını ve Çalışma Alanlarını okuma
- Kart, liste, pano ve Çalışma Alanı oluşturma ve güncelleme
- Çalışma Alanınızın Power-Up'larını okuma
- Çalışma Alanınızın Power-Up'larını güncelleme

Server Token şunları **yapamayacak**:

- Trello şifreni görme

Server Token aşağıdaki panolara ve Çalışma Alanlarına erişebilecek:

api



3 Panolar

Server Token ayrıca, ileride erişim elde edeceğiniz tüm panolara ve Çalışma Alanlarına da erişebilecek.

[Trello Gizlilik Politikası](#)

Reddet

Izin Ver



BOARD OLUSTURMA

Trello ile islem yapabilmek icin api-key ve kendi Trello hesabimiza ulasmak icin belirtec almis olduk

The screenshot shows the Postman application interface. On the left, there's a sidebar with sections like 'API DERSLERİ', 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area shows a 'Trello / Board olusturma' collection with a 'POST' request. The request URL is <https://api.trello.com/1/boards/?key=df524c4a1d9e349a462db5b5f081bfe9&token=cef4b32278ac66e68a5426ba92fcc08b58a95b22488f11e5589c6aed9ac...>. The 'Params' tab is selected, showing 'Query Params' with three entries: 'key' (value: df524c4a1d9e349a462db5b5f081bfe9), 'token' (value: cef4b32278ac66e68a5426ba92fcc08b58a95b22488f11e5589c6aed9ac...), and 'name' (value: API ile olusturulan). Below this, the 'Body' tab is selected, showing a JSON response with the following content:

```
1 "id": "60acc76ebf4aa70d6eaa82ca",
2 "name": "API ile olusturulan",
3 "desc": "",
4 "descData": null,
5 "closed": false,
6 "idOrganization": "60abb1f8c74f6f3f28e98a67",
7 "idEnterprise": null,
8 "pinned": false,
9 "url": "https://trello.com/b/9xUPvIvA/api-ile-olusturulan",
10 "shortUrl": "https://trello.com/b/9xUPvIvA",
11 "prefs": {
12     "isPublic": true,
13     "isPrivate": false
14 }
```

1- Dokumantasyondan board olusturmak icin gerekli ornek request'i alalim

2- Request'teki headers bolumunde api-key, token ve name kismina kendi degerlerimizi yazalim

3- Dokumantasyonda belirtildigi uzere POST metodunu secelim

4- Response'da donen board numarasini kaydedelim.

Bundan sonra bu board'la yapacagim her islemde board id'sini kullanabilirim



LIST OLUSTURMA

Trello ile islem yapabilmek icin api-key ve kendi Trello hesabimiza ulasmak icin belirtec ve board'umuzun id'sini almis olduk

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area shows a 'Trello / List Olusturma' collection with several requests listed. A red box highlights the 'Params' tab in the top navigation bar. Below it, a table titled 'Query Params' lists four parameters: 'key', 'token', 'name', and 'idBoard', each with its value filled in. Another red box highlights the 'Body' tab at the bottom, which displays a JSON response with an 'id' key. The status bar at the bottom right indicates a successful '200 OK' response.

1- Dokumantasyondan board olusturmak icin gerekli ornek request'i alalim

2- Request'teki headers bolumunde api-key, token, name ve idBoard kismina kendi degerlerimizi yazalim

3- Dokumantasyonda belirtildigi uzere POST metodunu secelim

4- Response'da donen list numarasini kaydedelim.

Bundan sonra bu list'le yapacagim her islemde board id'sini kullanabilirim



LIST OLUSTURMA

Trello ile islem yapabilmek icin api-key ve kendi Trello hesabimiza ulasmak icin belirtec ve board'umuzun id'sini almis olduk

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main area is titled 'Trello / List Olusturma' and shows a 'POST' request to the URL `https://api.trello.com/1/lists?key=df524c4a1d9e349a462db5b5f081bfe9&token=cef4b32278ac66e68a5426ba92fcc08b58a95b22488f11e5589c6aed9ace0b...`. The 'Params' tab is selected, showing a table with four rows: 'key' (value: df524c4a1d9e349a462db5b5f081bfe9), 'token' (value: cef4b32278ac66e68a5426ba92fcc08b58a95b22488f11e5589c6aed9ace0b...), 'name' (value: API list), and 'idBoard' (value: 60accbb4e8c8d93583983366). Below this, the 'Body' tab is selected, showing a JSON response with the key 'id' highlighted in a red box. The response body is:

```
1
2 "id": "60acc33818a861b93e41f10",
3   "name": "API list",
4   "closed": false,
5   "pos": 8192,
6   "idBoard": "60accbb4e8c8d93583983366",
7   "limits": {}
```

1- Dokumantasyondan list olusturmak icin gerekli ornek request'i alalim

2- Request'teki headers bolumunde api-key, token, name ve idBoard kismina kendi degerlerimizi yazalim

3- Dokumantasyonda belirtildigi uzere POST metodunu secelim

4- Response'da donen list numarasini kaydedelim.

Bundan sonra bu list'le yapacagim her islemde list id'sini kullanabilirim



CARD OLUSTURMA

Trello ile islem yapabilmek icin api-key ve kendi Trello hesabimiza ulasmak icin belirtec ve board'umuzun ve list'imizin id'sini almis olduk

The screenshot shows the Postman interface with a red box highlighting the 'Query Params' section of a POST request to `https://api.trello.com/1/cards`. The 'Query Params' table contains three entries:

KEY	VALUE	DESCRIPTION
key	df524c4a1d9e349a462db5b5f081bfe9	
token	cef4b32278ac66e68a5426ba92fcc08b58a95b22488f11e5589c6aed9ace0...	
idList	60accf2fae45c335edfa8cd9	

The response body is also highlighted with a red box, showing the JSON structure of the created card:

```
1 "id": "60accfb259ad274f055344d4",
2
3
4 "closed": false,
5 "dateLastActivity": "2021-05-25T10:21:38.135Z",
6 "desc": "",
7 "descData": []
8 "emoji": {}
9
10 "dueReminder": null,
11 "idBoard": "60accfed1cb91357fc7fb8958",
12 "idList": "60accf2fae45c335edfa8cd9",
```

Bundan sonra bu card'la yapacagim her islemde card id'sini kullanabilirim

1- Dokumantasyondan card olusturmak icin gerekli ornek request'i alalim

2- Request'teki headers bolumunde api-key, token, name ve idList kismina kendi degerlerimizi yazalim

3- Dokumantasyonda belirtildigi uzere POST metodunu secelim

4- Response'da donen list numarasini kaydedelim.



Simdi olusturdugumuz board, list ve card ile asagidaki islemleri yapalim

1- Kart Goruntuleme

2- Kartin bulundugu listeyi degistirme

3- Kart silme

4- Liste ismi degistirme

5- Listedeki tum kartlari arsive kaldir

6- Board silme

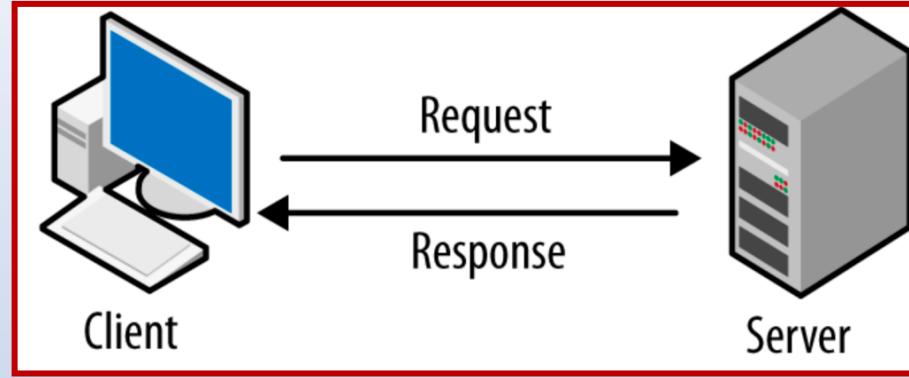




BOLUM 3
API TEST OTOMASYONU

DERS 7
API TESTING NEDIR ?
MANUEL VE OTOMASYON TEST KARSILASTIRMASI
INTELLIJ FRAMEWORK OLUSTURULMASI

API TESTING NEDIR ?



Her API sorgusu Client ile Server arasında bir iletisim demektir.

HTTP protokolu bu iletisimin kurallarını belirler. Hangi işlem için ne tur bir Request göndermemiz gerektiği ve protokole uygun olarak gönderdiğimiz Request karşılığında Server'in ne tur bir cevap vereceği net olarak belli dir.

API'in sağlıklı olarak çalışıp çalışmadığını test etmek için aşağıda belirlenen yöntemle API testleri yapılır.

- 1- Onceden belirlenmiş senaryoya uygun bir Request Server'a gönderilir
- 2- Gonderdigimiz Request'in karşılığında Server'in dönmesi gereken Response belirlenir (Expected Data)
- 3- Gonderdigimiz Request'in karşılığında Server'in döndirdiği Response kaydedilir (Actual Data)
- 4- Expected Data ile Actual Data karşılaştırılır (Assertion) fark varsa rapor edilir.

API TESTING NEDIR ?

The screenshot shows the Postman application interface. A red box highlights the request configuration area. The method is set to 'POST' and the URL is '{{JsonplaceBaseUrl}}/posts'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "title": "Ahmet",
3   "body": "Merhaba",
4   "userId": 1
5 }
```

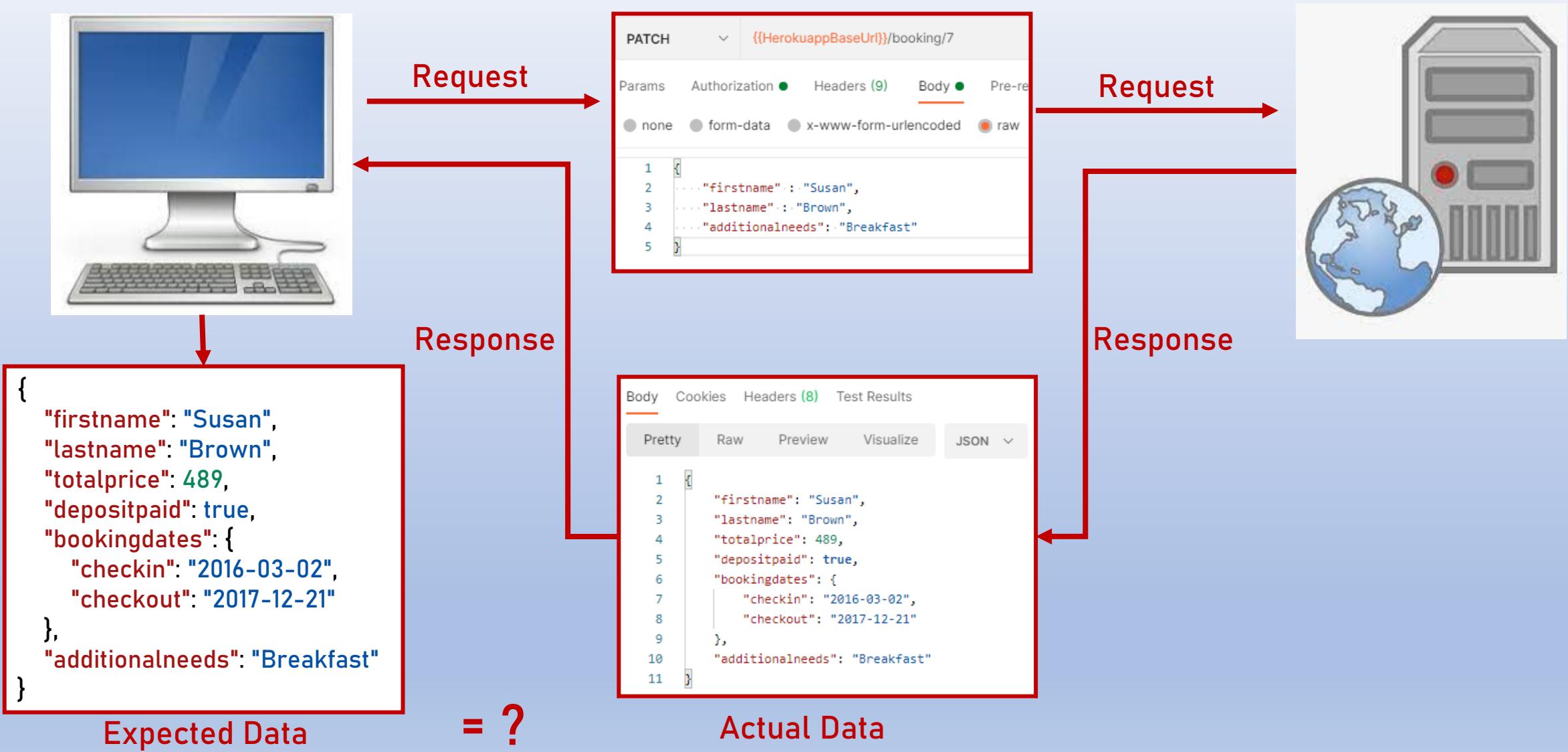
Ornegin; yukaridaki gosterilen Request Server'a gonderildiginde, Server'in yeni bir kayit olusturmasi ve status kodu olarak 200 donmesi beklenir.

Ayrica gonderdigim Request body'sinde yer alan title,body ve userId degerleri ile Server'in bir kayit olusturmasi ve donen Response'daki title,body ve userId degerlerinin Request'teki ile ayni olmasi beklenir

The screenshot shows the Postman interface after a successful request. A red box highlights the response body area. The response is a JSON object with an additional 'id' field:

```
1 {
2   "title": "Ahmet",
3   "body": "Merhaba",
4   "userId": 1,
5   "id": 101
6 }
```

API TESTING NEDİR ?



MANUEL VE OTOMASYON TEST KARSILASTIRMASI

```
{  
    "firstname": "Susan",  
    "lastname": "Brown",  
    "totalprice": 489,  
    "depositpaid": true,  
    "bookingdates": {  
        "checkin": "2016-03-02",  
        "checkout": "2017-12-21"  
    },  
    "additionalneeds": "Breakfast"  
}
```

Expected Data

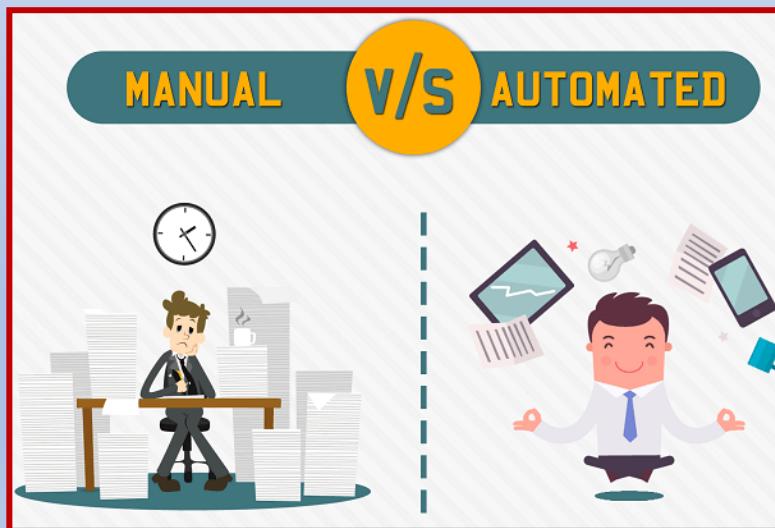
= ?

```
Body Cookies Headers (8) Test Results  
Pretty Raw Preview Visualize JSON ▾  
1 {  
2     "firstname": "Susan",  
3     "lastname": "Brown",  
4     "totalprice": 489,  
5     "depositpaid": true,  
6     "bookingdates": {  
7         "checkin": "2016-03-02",  
8         "checkout": "2017-12-21"  
9     },  
10    "additionalneeds": "Breakfast"  
11 }
```

Actual Data

Request'in manuel olarak Server'a gonderilmesi ve donen Response'in Expected Data ile bire - bir karsilastirilmasidir

Tool olarak Postman kullanilabilir

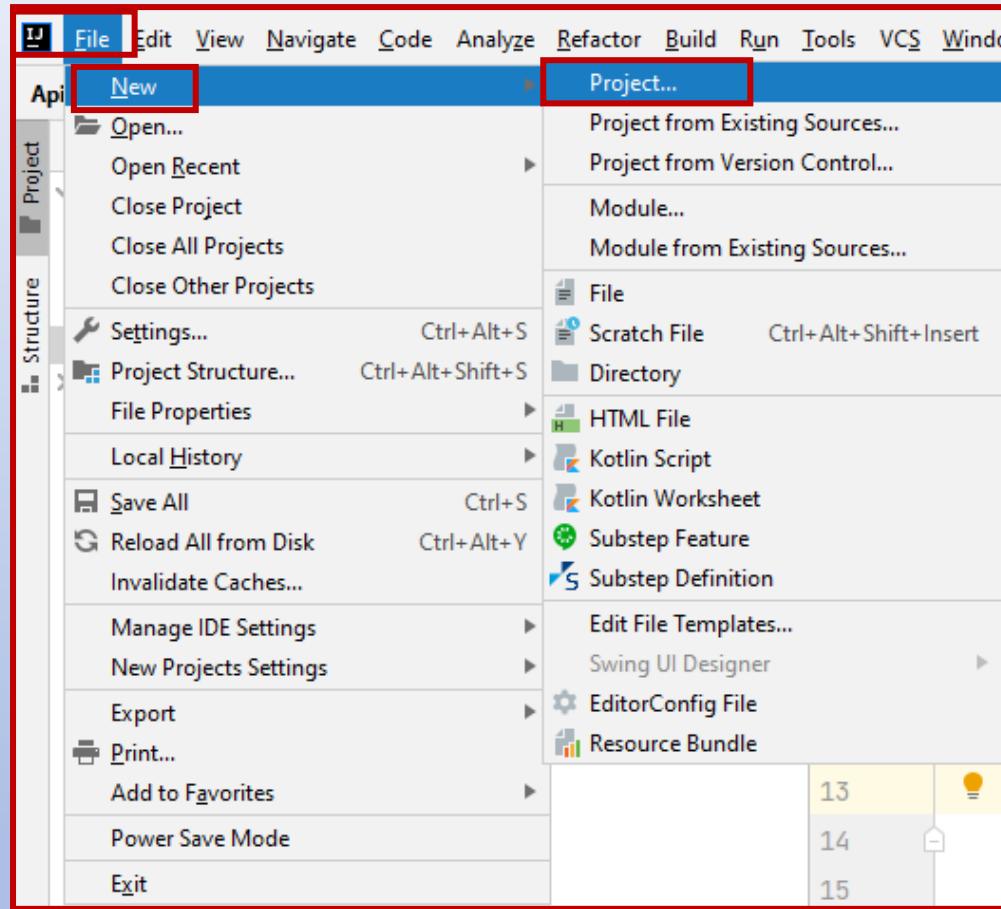


Request'in yazilan otomasyon kodlari ile Server'a gonderilmesi ve donen Response'in Expected Data karsilastirilip raporlanmasinin da yazilan kodlarla yapilmasidir.

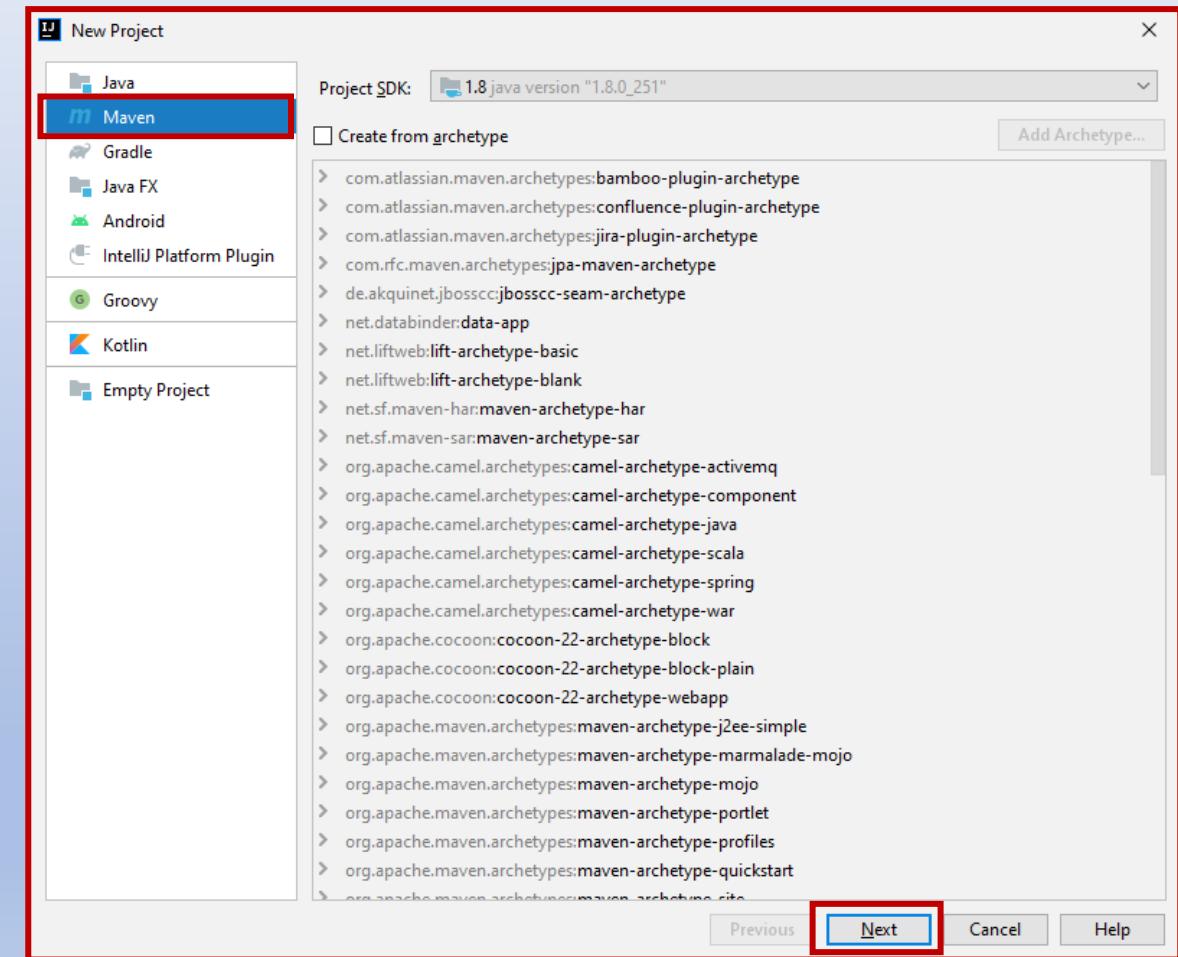
Tool olarak IntelliJ ide ve kutuphane olarak Selenium kullanacagiz

INTELLIJ FRAMEWORK OLUSTURMA

1- File-New-Project secelim

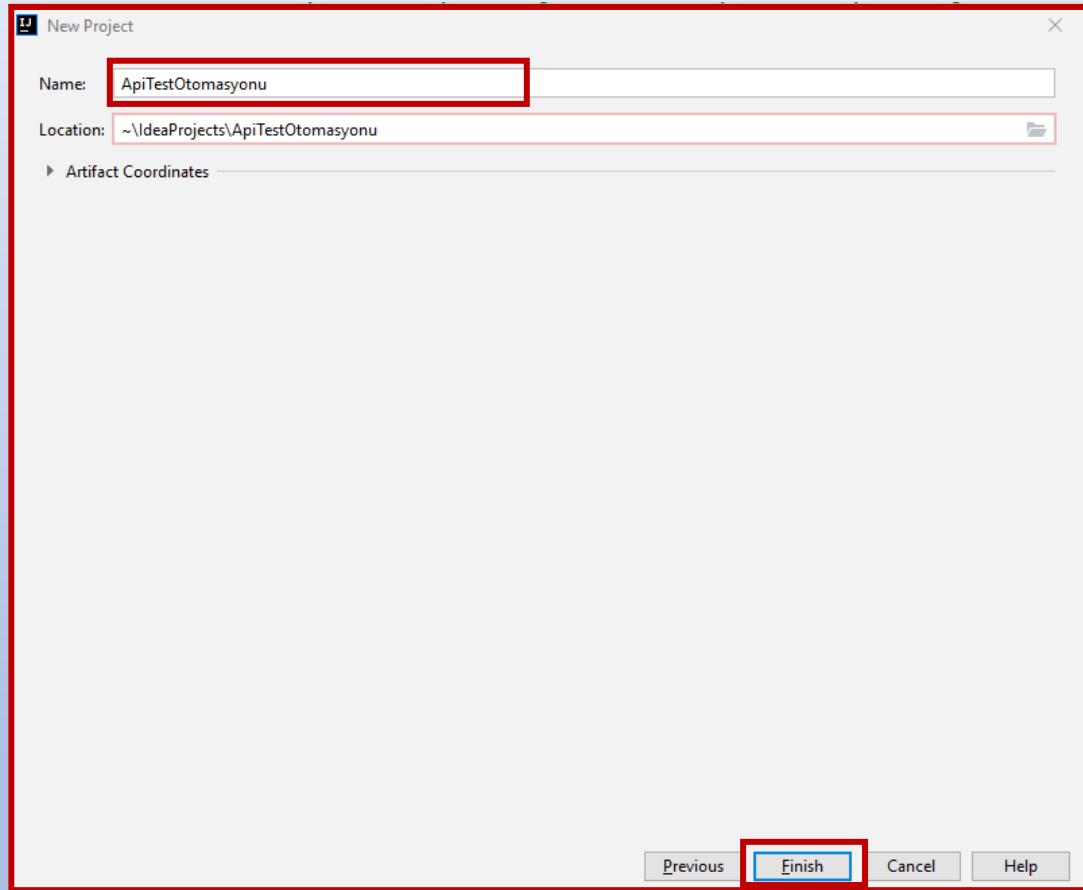


2- Maven'i secip next'e basalim



INTELLIJ FRAMEWORK OLUSTURMA

3- Isim bolumune bir isim yazalim
ve finish'e basalim



4- pom.xml'i secelim

The screenshot shows the IntelliJ IDE with the project 'ApiTestOtomasyonu' open. The 'pom.xml' file is selected in the project tree and is displayed in the code editor. The XML content of the 'pom.xml' file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/
<modelVersion>4.0.0</modelVersion>

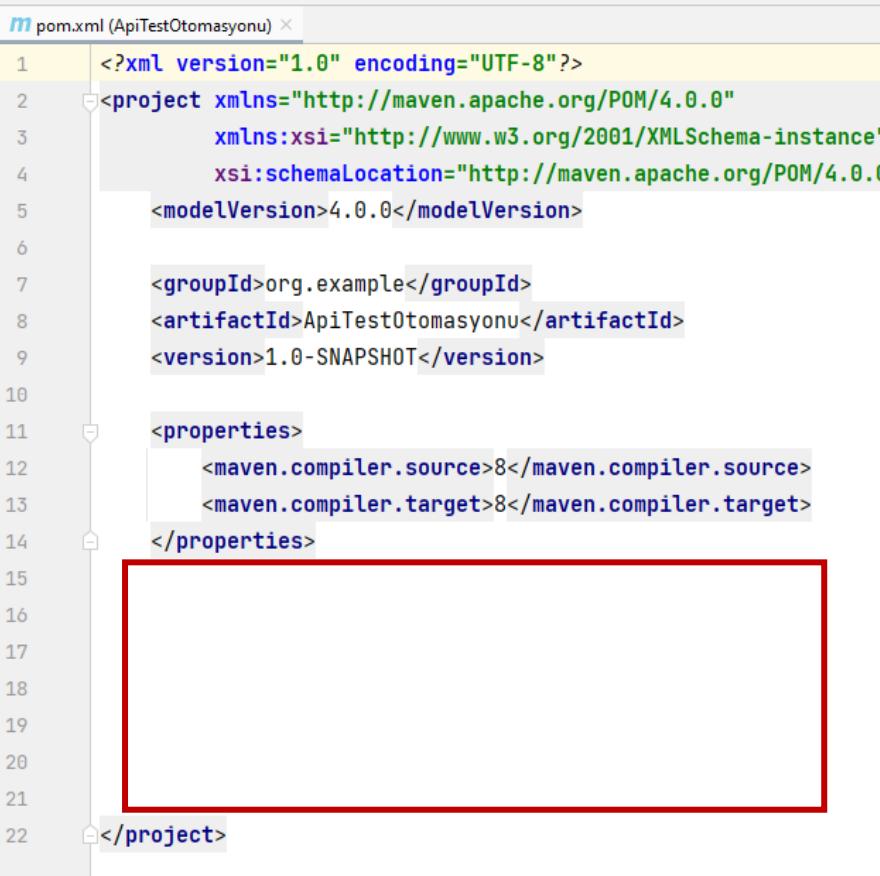
<groupId>org.example</groupId>
<artifactId>ApiTestOtomasyonu</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
</properties>

</project>
```

INTELLIJ FRAMEWORK OLUSTURMA

5- properties'den sonraki isaretli kisma sagdaki dependies'i ekleyelim

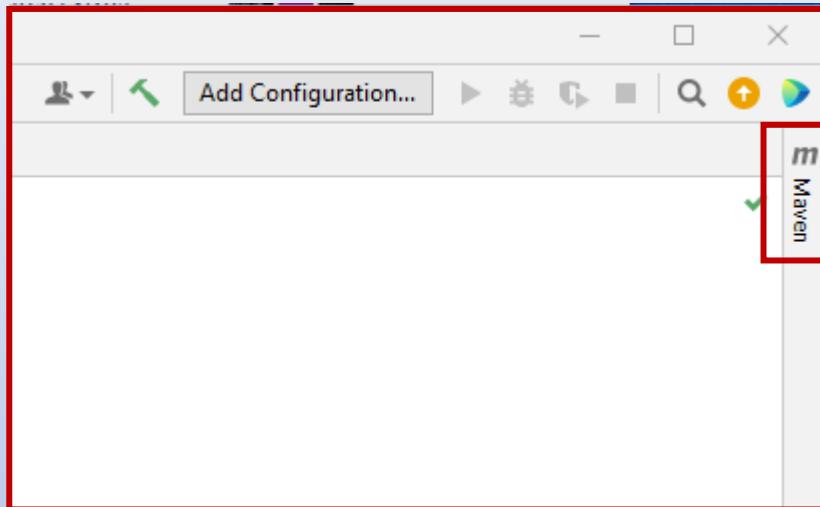


```
pom.xml (ApiTestOtomasyonu) ×  
1  <?xml version="1.0" encoding="UTF-8"?>  
2  <project xmlns="http://maven.apache.org/POM/4.0.0"  
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
5          <modelVersion>4.0.0</modelVersion>  
6  
7      <groupId>org.example</groupId>  
8      <artifactId>ApiTestOtomasyonu</artifactId>  
9      <version>1.0-SNAPSHOT</version>  
10  
11     <properties>  
12         <maven.compiler.source>8</maven.compiler.source>  
13         <maven.compiler.target>8</maven.compiler.target>  
14     </properties>  
15  
16  
17  
18  
19  
20  
21     </project>
```

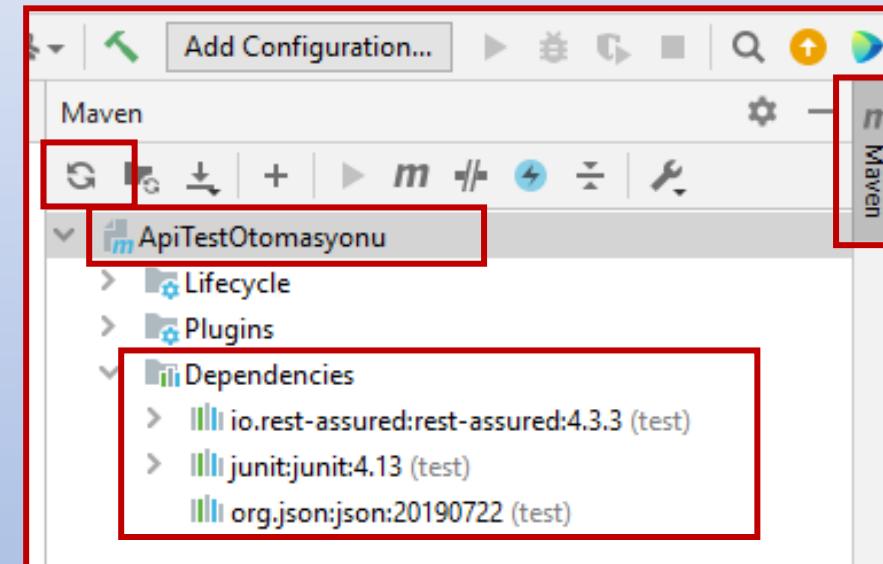
```
<dependencies>  
    <dependency>  
        <groupId>io.rest-assured</groupId>  
        <artifactId>rest-assured</artifactId>  
        <version>4.3.3</version>  
        <scope>test</scope>  
    </dependency>  
    <dependency>  
        <groupId>junit</groupId>  
        <artifactId>junit</artifactId>  
        <version>4.13</version>  
        <scope>test</scope>  
    </dependency>  
    <dependency>  
        <groupId>org.json</groupId>  
        <artifactId>json</artifactId>  
        <version>20190722</version>  
        <scope>test</scope>  
    </dependency>  
</dependencies>
```

INTELLIJ FRAMEWORK OLUSTURMA

6- Ekrani sag ust kosesindeki Maven yazisina tiklayalim



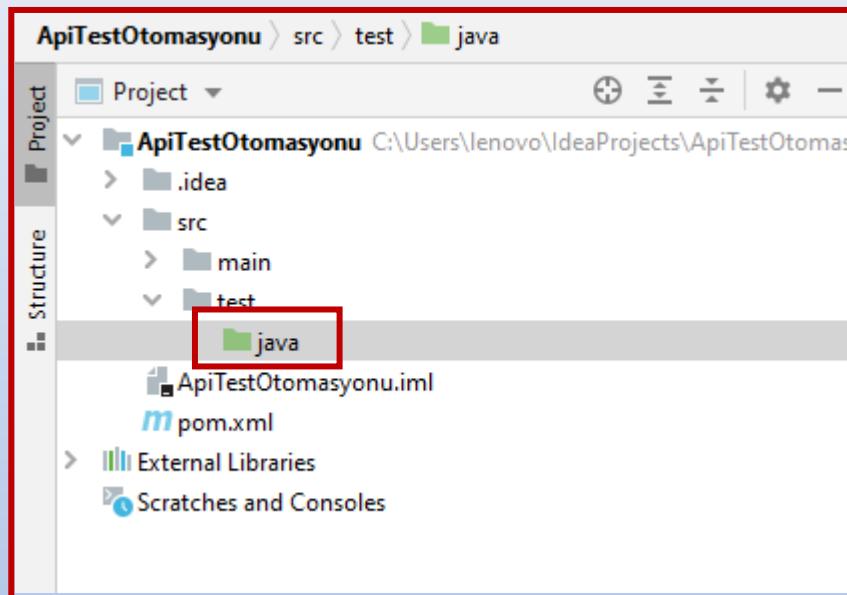
7- Acilan pencerede Proje ismiminin altinda **Dependencies** seceneginin ciktigindan ve Dependencies altında yukledigimiz 3 kutuphanenin oldugundan emin olun



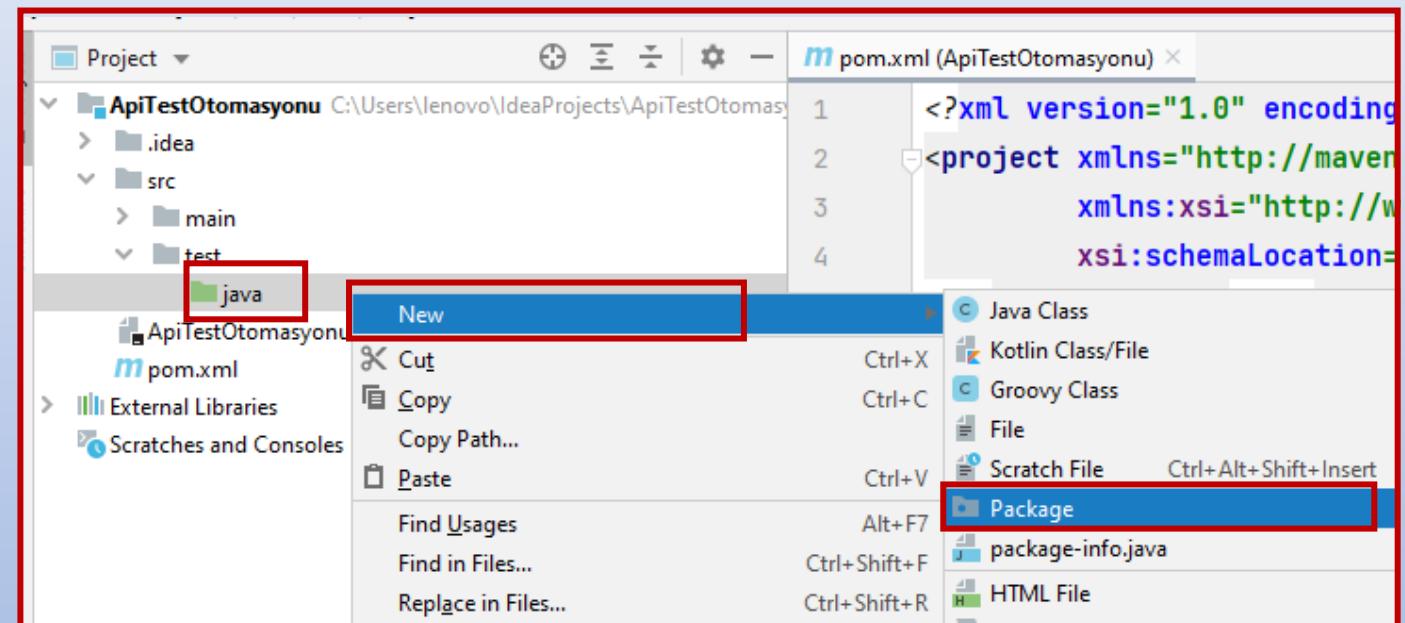
8- Eger kutuphaneler gorunmuyorsa yenile butonuna basin
ve kutuphaneler gorunur oluncaya kadar bekleyin
9- Kutuphaneler gorunur oldugunda Maven yazisina basarak
acilir pencereyi kapatin

INTELLIJ FRAMEWORK OLUSTURMA

10- Projemizin altında src – test – java'yi secelim



11- java klasoru üzerinde sag klik yapip **New – Package** 'yi secelim ve package ismi olarak **test** yazalim



12- Olusturdugumuz test package'l nderinde sag klik yapip **New – Java Class** 'i secelim ve class ismi olarak **C1_Get_ApiSorgulama** yazalim



BOLUM 3
API TEST OTOMASYONU

DERS 8

OTOMASYON ILE GET REQUEST YAPMA

RESPONSE BILGILERININ MANUEL TEST EDILMESI

OTOMASYON ILE GET SORGUSU RESPONSE BILGILERININ TEST EDILMESI

OTOMASYON ILE GET SORGUSU YAPMA

C1_Get_ApiSorgulama

<https://restful-booker.herokuapp.com/booking/10> url'ine bir GET request gonderdigimizde donen Response'un,

status code'unun 200,

ve content type'inin application/json; charset=utf-8,

ve Server isimli Header'in degerinin Cowboy,

ve status Line'in HTTP/1.1 200 OK

ve response suresinin 5 sn'den kisa oldugunu manuel olarak test ediniz.

NOT : IntelliJ'de API sorgulari yapmak icin io.restassured kutuphanesi kullanilir ve Response class'indan bir obje olusturmamiz gereklidir

Response response = given().when().get(url);

Response olustururken kullandigimiz

given : Testimize baslarken bize verilen baslangic degerlerini ifade eder

when : Testimizde gerceklestirdigimiz islemleri ifade eder

then : Response degerlerini degerlendirmek icin yapılan islemleri ifade eder

and : Birbirine bagli islemleri ifade eder

RESPONSE BILGILERININ MANUEL TEST EDILMESI

Olusturdugumuz response objesi ile kullanilabilecegimiz methodlar :

response.prettyPrint	: Response'u yazdirir
response.getStatusCode()	: Response'un status kodunu verir
response.getHeaders()	: Response'un tum basliklarini (headers) verir
response.getHeader("Server")	: Response'un istenen basliginin(header) degerini verir
response.getContentType()	: Response'un Content Type'ini verir
response.getStatusLine()	: Response'un Status Line degerini verir
response.getTime()	: Response'un gerceklesme suresini milisaniye olarak verir

OTOMASYON ILE GET REQUEST RESPONSE BILGILERINI TEST ETMEK

assertThat()

C2_Get_ResponseBilgileriAssertion

<https://restful-booker.herokuapp.com/booking/10> url'ine bir GET request gonderdigimizde donen Response'un,

status code'unun 200,

ve content type'inin application/json; charset=utf-8,

ve Server isimli Header'in degerinin Cowboy,

ve status Line'in HTTP/1.1 200 OK

Olusturdugumuz response objesi ve assertThat() metodu ile yapabilecegimiz assertion'lar

response.

then().

assertThat().

statusCode(200).

contentType("application/json; charset=utf-8").

header("Server","Cowboy").

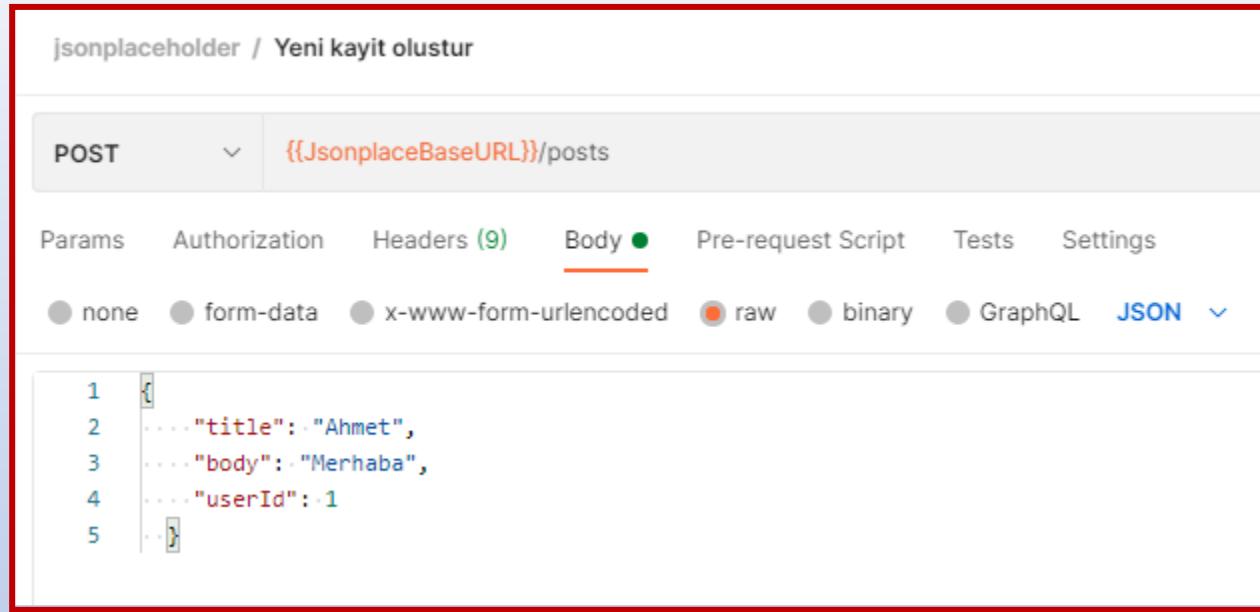
statusLine("HTTP/1.1 200 OK");



BOLUM 3
API TEST OTOMASYONU

DERS 9
JSON OBJECT OLUSTURMA
PUT / POST VE PATCH SORGUSU RESPONSE BILGILERININ TEST EDILMESI

OTOMASYON ILE PUT / POST VE PATCH SORGUSU YAPMA



NOT : PUT, POST ve PATCH request'leri yapılmırken server'a oluşturulması / değiştirilmesini istediğimiz yeni bilgileri göndermek zorundayız.

Bunun için farklı data tipleri ile body oluşturulabilir.

Oluşturma kolaylığı ve data tiplerini kabul esnekliği açısından en çok kullanılan body türü Json olduğundan biz de ilk basta Json objesi oluşturup bunu body olarak request'imize ekleyeceğiz.

İllerleyen zamanlarda Map, Pojo ve Mapper turundan objeler de oluşturup testlerimizde kullanacağız.

JSON (Java Script Object Notation) Nedir ?



JSON formati JS ile olusturulan web uygulamalarinda data saklamak ve uygulamalar arasında data alisverisi yapmak (Request/Response) icin en çok tercih edilen formattir.

JSON formatindaki bir data icin uc temel bolum vardir.

- 1- **Suslu parantezler** : JSON formatindaki bir datanin nerede baslayip, nerede bittiğini gösterir. İhtiyac olduğunda NESTED (ic ice) JSON datalari olusturulabilir
- 2- **Keys** : JSON datalari icinde bulunan variable isimleridir.
- 3- **Values** : JSON datalari icinde bulunan variable'lara atanın degerlerdir.
Keys ve **Values** arasında : kullanılır.

JSON kullandığı key – value yapısı ile Java'dan bildigimiz Map'e çok benzemektedir.

JSON Vs Map

API kullaniminda Key – Value yapisi oldugundan en cok kullanılan data yapıları Map ve JSONObject'tır.

```
public void xx(){  
  
    Map<String, Integer> ornek = new HashMap<>();  
  
    ornek.put("Yas", 25);  
    ornek.put("Isim", "Ali");  
    ornek.put("ogrenciMi", true);  
  
}
```

```
public void xx(){  
  
    JSONObject ornek = new JSONObject();  
  
    ornek.put("Yas", 25);  
    ornek.put("Isim", "Ali");  
    ornek.put("ogrenciMi", true);  
  
}
```

Map olusturulurken kullanacagımız key – value icin data turleri belirtilmek zorundadir, ancak **JSONObject** icin data turlerinin belirtilmesine gerek yoktur.

Map olusturulurken belirttigimiz data turleri disinda data kullandigimizda Compile Time Error CTE olusur, **JSONObject** icin data turlerinin bir onemi yoktur dolayisiyla da kullanım acisından **esnek**tir.

JSON OBJECT OLUSTURMA

C3(JsonObjesiOlusturma)

Aşağıdaki JSON Objesini oluşturup konsolda yazdırın.

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 1  
}
```



```
JSONObject jsonObject=new JSONObject();  
jsonObject.put("title", "Ahmet");  
jsonObject.put("body", "Merhaba");  
jsonObject.put("userId", 1);  
  
System.out.println(jsonObject.toString());
```

```
{"title": "Ahmet", "body": "Merhaba", "userId": 1}  
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 1  
}
```

JSON OBJESİ OLUSTURMA

C3(JsonObjesiOlusturma)

Asagidaki JSON Objesini olusturup konsolda yazdirin.

```
{  
    "firstname": "Jim",  
    "additionalneeds": "Breakfast",  
    "bookingdates": {  
        "checkin": "2018-01-01",  
        "checkout": "2019-01-01"  
    },  
    "totalprice": 111,  
    "depositpaid": true,  
    "lastname": "Brown"  
}
```

```
{  
    "bookingid": 1,  
    "booking": {  
        "firstname": "Jim",  
        "lastname": "Brown",  
        "totalprice": 111,  
        "depositpaid": true,  
        "bookingdates": {  
            "checkin": "2018-01-01",  
            "checkout": "2019-01-01"  
        },  
        "additionalneeds": "Breakfast"  
    }  
}
```

```
{"firstname": "Jim", "additionalneeds": "Breakfast", "bookingdates": {"checkin": "2018-01-01", "checkout": "2019-01-01"}, "totalprice": 111, "depositpaid": true, "lastname": "Brown"}
```

```
JSONObject jsonObjectInner=new JSONObject();  
jsonObjectInner.put("checkin", "2018-01-01");  
jsonObjectInner.put("checkout", "2019-01-01");
```

```
JSONObject jsonObjectBody=new JSONObject();  
jsonObjectBody.put("firstname", "Jim");  
jsonObjectBody.put("lastname", "Brown");  
jsonObjectBody.put("totalprice", 111);  
jsonObjectBody.put("depositpaid", true);  
jsonObjectBody.put("bookingdates", jsonObjectInner);  
jsonObjectBody.put("additionalneeds", "Breakfast");
```

```
{  
    "firstname": "Jim",  
    "additionalneeds": "Breakfast",  
    "bookingdates": {  
        "checkin": "2018-01-01",  
        "checkout": "2019-01-01"  
    },  
    "totalprice": 111,  
    "depositpaid": true,  
    "lastname": "Brown"  
}
```

PUT REQUEST, RESPONSE BILGILERINI ASSERT YAPMA

assertThat()

C4_Put_ResponseBilgileriAssertion

<https://jsonplaceholder.typicode.com/posts/70> url'ine asagidaki Json formatindaki body ile bir PUT request gonderdigimizde

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

donen Response'un,

- status code'unun 200,
- ve content type'inin application/json; charset=utf-8,
- ve Server isimli Header'in degerinin cloudflare,
- ve status Line'in HTTP/1.1 200 OK



BOLUM 3
API TEST OTOMASYONU

DERS 10

API SORGULARINDA RESPONSE BODY'NIN TEST EDILMESI

GET REQUEST BODY BILGILERINI ASSERT YAPMA

C5_Get_ResponseBodyTesti

<https://jsonplaceholder.typicode.com/posts/44> url'ine bir GET request yolladigimizda donen Response'in

status code'unun 200,
ve content type'inin Application.JSON,
ve response body'sinde bulunan **userId**'nin 5,
ve response body'sinde bulunan **title**'in "**optio dolor molestias sit**"
oldugunu test edin.

Response body'sindeki degerleri test etmek icin Matchers class'indan yardim aliriz.

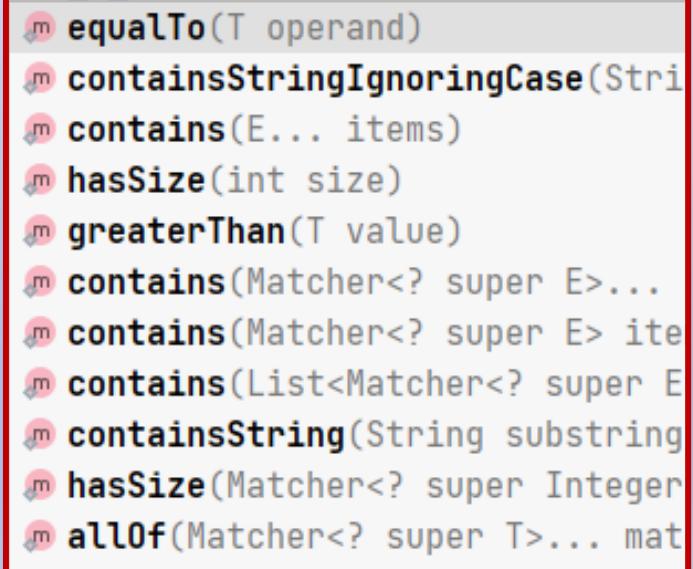
```
response.  
    then().  
    assertThat().  
    body("title", Matchers.equalTo("optio dolor molestias sit")).
```

```
body("userId", Matchers.equalTo(5));
```

key

Metot

value



A screenshot of an IDE's code completion feature. A dropdown menu is open, listing several static methods from the `Matchers` class. The methods listed are: equalTo(T operand), containsStringIgnoringCase(String value), contains(E... items), hasSize(int size), greaterThan(T value), contains(Matcher<? super E>... matchers), contains(Matcher<? super E> item), contains(List<Matcher<? super E> matchers), containsString(String substring), hasSize(Matcher<? super Integer> size), and allOf(Matcher<? super T>... matchers). Each method name is preceded by a small pink circular icon with a white 'm' symbol.

- m equalTo(T operand)
- m containsStringIgnoringCase(String value)
- m contains(E... items)
- m hasSize(int size)
- m greaterThan(T value)
- m contains(Matcher<? super E>... matchers)
- m contains(Matcher<? super E> item)
- m contains(List<Matcher<? super E> matchers)
- m containsString(String substring)
- m hasSize(Matcher<? super Integer> size)
- m allOf(Matcher<? super T>... matchers)

POST REQUEST BODY BILGILERINI ASSERT YAPMA

C6_Post_ResponseBodyTesti

<https://jsonplaceholder.typicode.com/posts> url'ine asagidaki body ile bir POST request gonderdigimizde

```
{  
    "title": "API",  
    "body": "API ogrenmek ne guzel",  
    "userId": 10,  
}
```

donen Response'un,

status code'unun 201,

ve content type'inin application/json

ve Response Body'sindeki,

"title" in "API" oldugunu

"userId" degerinin 100'den kucuk oldugunu

"body" nin "API" kelimesi icerdigini

test edin.

```
body("body", Matchers.containsString("API")).  
body("userId", Matchers.lessThan(100));
```



BOLUM 3
API TEST OTOMASYONU

DERS 11

API SORGULARINDA RESPONSE BODY TEST EDILIRKEN
KOD TEKRARLARINDAN KURTULMA

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

<https://restful-booker.herokuapp.com/booking/10> url'ine bir GET request gonderdigimizde donen Response'un,

status code'unun 200,

ve content type'inin application-json,

ve response body'sindeki

"firstname"in, "Susan",

ve "lastname"in, "Jackson",

ve "totalprice"in, 612,

ve "depositpaid"in, false,

ve "additionalneeds"in, "Breakfast"

oldugunu test edin

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

Su ana kadar yaptigimiz haliyle assertion;

response.

then().

assertThat().

statusCode(**200**).

contentType(ContentType.JSON).

body("firstname",Matchers.equalTo("Susan")).

body("lastname",Matchers.equalTo("Wilson")).

body("totalprice",Matchers.equalTo(**643**)).

body("depositpaid",Matchers.equalTo(false)).

body("additionalneeds",Matchers.equalTo(**null**));

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

body yazilarindan kurtulmak icin
body ile ilgili tum assertion'lar tek body parantezi icinde yapilip virgulle
ayrilabilir

response.

then().

assertThat().

statusCode(200).

contentType(MediaType.JSON).

body("firstname",Matchers.equalTo("Susan"),

 "lastname",Matchers.equalTo("Wilson"),

 "totalprice",Matchers.equalTo(643),

 "depositpaid",Matchers.equalTo(false),

 "additionalneeds",Matchers.equalTo(null));

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

Matchers yazilarindan kurtulmak icin

- 1) Matchers yazilarindan birini silin

```
body("firstname", equalTo("Susan"))
```

- 2) Kirmizi olan equalTo yazisini mouse ile isaretleyip Alt ve Enter tuslarina basin

A screenshot of an IntelliJ IDEA code editor showing a code completion dropdown. The code above the dropdown is: `body("firstname", equalTo("Susan")),
 "lastname", M
 "totalprice"
 "depositpaid"`. The word `equalTo` is highlighted in red. A tooltip above the dropdown says: `Create method 'equalTo' in 'C7_Get_BodyTekrarlardanKurtulma'`. The dropdown menu contains three items: `Import static method...`, `Qualify static call...`, and `Method`.

- 3) Acilan pencereden "import static method"u secin

A screenshot of an IntelliJ IDEA code editor showing a code completion dropdown. The code above the dropdown is: `body("firstname", equalTo("Susan")),
 "lastname", M
 "totalprice",
 "depositpaid"`. The word `equalTo` is highlighted in red. A tooltip above the dropdown says: `Import static method...`. The dropdown menu contains three items: `CoreMatchers.equalTo (org.hamcrest)`, `isEqual.equalTo (org.hamcrest.core)`, and `Matchers.equalTo (org.hamcrest)`.

- 4) Acilan pencereden "import static method"u secin, tum Matchers yazilari ortadan kalkacaktir.

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

Matchers yazilarindan kurtulmak icin

- 5) Matchers.equalTo disinda method da kullanmak istiyorsak, import kismindaki satirdan equalTo'yu silip yerine * yazin

```
package test;

import io.restassured.http.ContentType;
import io.restassured.response.Response;
import org.hamcrest.Matchers;
import org.junit.Test;
import static io.restassured.RestAssured.given;
import static org.hamcrest.Matchers.equalTo;

public class C7_Get_BodyTekrarlardanKurtulma {
```

- 6) Artik Matchers kelimesi yazmadan Matcher class'indaki tum metodlari kullanabiliriz

```
import static org.hamcrest.Matchers.*;
```

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C7_Get_BodyTekrarlardanKurtulma

Matchers yazilarindan kurtulmak icin

response.

then().

assertThat().

statusCode(200).

contentType(MediaType.JSON).

body("firstname",equalTo("Susan"),

 "lastname", equalTo("Wilson"),

 "totalprice", equalTo(643),

 "depositpaid", equalTo(false),

 "additionalneeds", equalTo(null));



BOLUM 3
API TEST OTOMASYONU

DERS 12

KOMPLEKS JSON OBJELERI VE JSONPath KULLANIMI

JSONPath KULLANIMI

C8(JsonPathKullanimi)

```
{  
    "firstName": "John",  
    "lastName" : "doe",  
    "age"      : 26,  
    "address"  : {  
        "streetAddress": "naist street",  
        "city"         : "Nara",  
        "postalCode"   : "630-0192"  
    },  
    "phoneNumbers": [  
        {  
            "type"  : "iPhone",  
            "number": "0123-4567-8888"  
        },  
        {  
            "type"  : "home",  
            "number": "0123-4567-8910"  
        }  
    ]  
}
```

JSONPath JSON verilerini okuma ve update etme fırsatı verir.

Bir JSON objesinin içinde birden fazla data turunde primitive data veya obje bulunabilir.

Ornegin yandaki JSON objesi incelenirse

firstName -- String

lastName -- String

age -- Int

address -- Json obje

phoneNumbers ise içinde iki Json objesi olan bir arraydir

Yandaki Json objesini olusturmak istedigimizde once address ve phoneNumbers objelerini olusturmali sonra bunlari asil Json objemize eklemeliyiz

C8_JsonPathKullanimi

JSONPath KULLANIMI

```
{  
    "firstName": "John",  
    "lastName" : "doe",  
    "age"      : 26,  
    "address"  : {  
        "streetAddress": "naist street",  
        "city"         : "Nara",  
        "postalCode"   : "630-0192"  
    },  
    "phoneNumbers": [  
        {  
            "type"  : "iPhone",  
            "number": "0123-4567-8888"  
        },  
        {  
            "type"  : "home",  
            "number": "0123-4567-8910"  
        }  
    ]  
}
```

```
JSONObject kisiBilgisi=new JSONObject();  
JSONObject adresBilgisi=new JSONObject();  
JSONObject cepTelefonu=new JSONObject();  
JSONObject evTel= new JSONObject();  
JSONArray telBilgileri =new JSONArray();
```

```
cepTelefonu.put("type","Cep Telefonu");  
cepTelefonu.put("number", "555-123-4567");  
evTel.put("type","Ev telefonu");  
evTel.put("number","312-123-4567");  
telBilgileri.put(cepTelefonu);  
telBilgileri.put(evTel);  
  
adresBilgisi.put("streetAddress", "Yenimahalle  
kurtulus cad");  
adresBilgisi.put("city","Ankara");  
adresBilgisi.put("postalCode","06100");
```

```
kisiBilgisi.put("firstName","Ahmet");  
kisiBilgisi.put("lastName","Bulut");  
kisiBilgisi.put("age",49);  
kisiBilgisi.put("address",adresBilgisi);  
kisiBilgisi.put("phoneNumbers",telBilgileri);
```

C8(JsonPathKullanimi

```
{  
  "firstName": "John",  
  "lastName" : "doe",  
  "age"      : 26,  
  "address" : {  
    "streetAddress": "naist street",  
    "city"         : "Nara",  
    "postalCode"   : "630-0192"  
  },  
  "phoneNumbers": [  
    {  
      "type"  : "iPhone",  
      "number": "0123-4567-8888"  
    },  
    {  
      "type"  : "home",  
      "number": "0123-4567-8910"  
    }  
  ]  
}
```

JSONPath KULLANIMI

```
{"firstName":"Ahmet","lastName":"Bulut","adress":{"streetAddress":"Kurtulus  
cad.","city":"Ankara","postalCode":"06100"},"age":49,"phoneNumbers":[{"number":  
"532-555 55 55","type":"cep"}, {"number":"0312-123 4567","type":"ev"}]}
```

```
{  
  "firstName":"Ahmet",  
  "lastName":"Bulut",  
  "address":{  
    "streetAddress":"Kurtulus cad.",  
    "city":"Ankara",  
    "postalCode":"06100"},  
  "age":49,  
  "phoneNumbers": [  
    {  
      "number": "532-555 55 55",  
      "type": "cep" },  
    {  
      "number": "0312-123 4567",  
      "type": "ev" }  
    ]  
  }
```

<https://jsonpath.com/>



BOLUM 3
API TEST OTOMASYONU

DERS 13

API TESTLERINDE, RESPONSE BODY BILGILERI TESTİNDE
JsonPATH KULLANIM

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C09_Post_JsonPathIleBodyTesti

<https://restful-booker.herokuapp.com/booking> url'ine asagidaki body'ye sahip bir POST request gonderdigimizde

```
{  
    "firstname" : "Ahmet",  
    "lastname" : "Bulut",  
    "totalprice" : 500,  
    "depositpaid" : false,  
    "bookingdates" : {  
        "checkin" : "2021-06-01",  
        "checkout" : "2021-06-10"  
    },  
    "additionalneeds" : "wi-fi"  
}
```

donen Response'un,
status code'unun 200,
ve content type'inin application-json,
ve response body'sindeki
"firstname"in,"Ahmet",

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C09_Post_JsonPathIleBodyTesti

```
{  
    "bookingid": 17,  
    "booking": {  
        "firstname": "Ahmet",  
        "lastname": "Bulut",  
        "totalprice": 500,  
        "depositpaid": false,  
        "bookingdates": {  
            "checkin": "2021-06-01",  
            "checkout": "2021-06-10"  
        },  
        "additionalneeds": "wi-fi"  
    }  
}
```

Response body'si iç içe Json objelerinden oluşuyorsa JsonPath yöntemleri kullanılarak assertion yapılabilir.

response.

then().

assertThat().

statusCode(200).

contentType(ContentType.JSON).

body("booking.firstname", equalTo("Ahmet"),

"booking.lastname",equalTo("Bulut"),

"booking.totalprice",equalTo(500),

"booking.depositpaid",equalTo(false),

"booking.bookingdates.checkin",equalTo("2021-06-01"),

"booking.bookingdates.checkout",equalTo("2021-06-10"));

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN LIST KULLANMA

C10_Get_ResponseBodyTestiListKullanimi

http://dummy.restapiexample.com/api/v1/employees url'ine bir GET request yolladigimizda
donen Response'in
status code'unun 200,
ve content type'inin Application.JSON,
ve response body'sindeki
employees sayisinin 24
ve employee'lerden birinin "Ashton Cox"
ve girilen yaslar icinde 61,21 ve 35 degerinin oldugunu test edin
test edin.

```
response.then().  
    assertThat()  
    .body("data.id", hasSize(24))  
    .body("data.employee_name",hasItem("Ashton Cox"))  
    .body("data.employee_age",hasItems(61,21,35));
```



BOLUM 3

API TEST OTOMASYONU

DERS 14

Junit ASSERT
(Hard Assert)

JUnit ASSERT

Response ile ilgili tüm genel bilgileri (statusCode, contentType, headers, statusLine vb.) assertThat() metodu ile yapabiliriz

Aynı şekilde response body'de bulunan value'lari da key'leri kullanarak assertThat() metodundan sonra body() metodu ve Matchers Class'ina ait metodlarla test edebiliriz.

```
response.  
    then().  
        assertThat().  
            statusCode(200).  
            contentType(MediaType.JSON).  
            body("firstname",equalTo("Susan"),  
                "lastname",equalTo("Wilson"),  
                "totalprice",equalTo(643),  
                "depositpaid",equalTo(false),  
                "additionalneeds",equalTo(null));
```

JUnit ASSERT

Assert, bir test senaryosunun PASS veya FAILED durumunu belirlemeye kullanılan yararlı bir yöntemdir. Seçilen metod ve yazılan **boolean koşul'a** göre test sonucu belirlenir.

Assert yöntemleri, Java.lang.Object Class'ına bağlı org.junit.Assert Class'i tarafından sağlanır.

En çok kullandığımız 3 Assert metodu;

1) Assert.assertTrue(**koşul**)

Yazılan koşul'un sonucu True ise test PASS, yoksa test FAILED olur

Assert.assertTrue(**20 > 15**) → Test PASS

True

Assert.assertTrue(**10 > 30**) → Test FAILED

False

JUnit ASSERT

2) Assert.assertFalse(**koşul**)

Yazilan koşul'un sonucu False ise test PASS, yoksa test FAILED olur

Assert.assertFalse(**40 > 50**) → Test PASS

False

Assert. assertFalse(**30 > 20**) → Test FAILED

True

3) Assert.assertEquals(**expected, actual**)

Yazilan expected ile actual esit ise test PASS, yoksa test FAILED olur

Assert.assertEquals(**"Ali" , "Ali"**) → Test PASS

True

Assert. assertEquals(**30 , 20**) → Test FAILED

False

JUnit ASSERT

C11_Get_ExpectedDataOlusturma

<https://jsonplaceholder.typicode.com/posts/22> url'ine bir GET request yolladigimizda donen response body'sinin asagida verilen ile ayni oldugunu test ediniz

Response body :

```
{  
    "userId": 3,  
    "id": 22,  
    "title": "dolor sint quo a velit explicabo quia nam",  
    "body": "eos qui et ipsum ipsam suscipit aut\\nsed omnis non odio\\nexpedita ear  
        um mollitia molestiae aut atque rem suscipit\\nnam impedit esse"  
}
```

```
JsonPath responseJsonPath=response.jsonPath();  
Assert.assertEquals(expectedDataJson.get("userId"),responseJsonPath.getInt("userId"));  
Assert.assertEquals(expectedDataJson.get("id"),responseJsonPath.getInt("id"));  
Assert.assertEquals(expectedDataJson.get("body"),responseJsonPath.getString("body"));  
Assert.assertEquals(expectedDataJson.get("title"),responseJsonPath.getString("title"));
```

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C12_Post_ExpectedDataVeJsonPathIleAssertion

<https://restful-booker.herokuapp.com/booking> url'ine asagidaki body'ye sahip bir POST request gonderdigimizde donen response'un id haric asagidaki gibi oldugunu test edin.

Request body

```
{  
    "firstname": "Ahmet",  
    "lastname": "Bulut",  
    "totalprice": 500,  
    "depositpaid": false,  
    "bookingdates": {  
        "checkin": "2021-06-01",  
        "checkout": "2021-06-10"  
    },  
    "additionalneeds": "wi-fi"  
}
```

Response Body

```
{  
    "bookingid": 24,  
    "booking": {  
        "firstname": "Ahmet",  
        "lastname": "Bulut",  
        "totalprice": 500,  
        "depositpaid": false,  
        "bookingdates": {  
            "checkin": "2021-06-01",  
            "checkout": "2021-06-10"  
        },  
        "additionalneeds": "wi-fi"  
    }  
}
```



BOLUM 3

API TEST OTOMASYONU

DERS 15

TestNG SOFT ASSERT

TestNG ASSERT

1. HARD ASSERT

JUnit'te Öğrendiğimiz AssertionTürü

- i. Assert.assertEquals()
- ii. Assert.assertTrue()
- iii. Assert.assertFalse()

Assert methodları kullanıldığında, bir assertion FAILED olursa execution durur, test method'nun geri kalanı çalışmaz.

Test case'in neden başarısız olduğunu hemen anlamak için hard assertion'u tercih edebiliriz.

Birden fazla assert yapıyorsak assertion başarısız olursa, execution durur, onu düzeltip yeniden calistirdiginizda başka bir assertion FAILED olabilir, her seferinde bir hatayı gormus ve düzeltmis olursunuz. Testteki hataları düzeltmek icin bu yontem cok uygun olmayabilir.

2. SOFT ASSERT (VERIFICATION)

Eğer softAssert başarısız olursa test method'unun geri kalanini durdurmaz ve yürütmeye devam eder. If else statement'da olduğu gibi.

SOFT ASSERT

- SoftAssert doğrulama (verification) olarak da bilinir.
- SoftAssert kullanabilmemiz için object create etmemiz gereklidir.
 - 1.Adım: SoftAssert objesi olusturalım

```
SoftAssert softAssert = new SoftAssert();
```

- 2.Adım: İstedigimiz verification'lari yapalim
 - softAssert.assertTrue();
- 3.Adım: SoftAssert'in durumu raporlamasini isteyelim
 - softAssert.assertAll();



SOFT ASSERT ILE EXPECTED DATA TESTI

C13_Get_SoftAssertIleExpectedDataTesti

http://dummy.restapiexample.com/api/v1/employee/3 url'ine bir GET request gönderdigimizde donen response'un asagidaki gibi oldugunu test edin.

Response Body

```
{  
    "status": "success",  
    "data": {  
        "id": 3,  
        "employee_name": "Ashton Cox",  
        "employee_salary": 86000,  
        "employee_age": 66,  
        "profile_image": ""  
    },  
    "message": "Successfully! Record has been fetched."  
}
```

SOFT ASSERT ILE EXPECTED DATA TESTI

C14_Put_SoftAssertIleExpectedDataTesti

http://dummy.restapiexample.com/api/v1/update/21 url'ine asagidaki body'ye sahip bir PUT request gonderdigimizde donen response'un asagidaki gibi oldugunu test edin.

Request Body

```
{  
    "status": "success",  
    "data": {  
        "name": "Ahmet",  
        "salary": "1230",  
        "age": "44",  
        "id": 40  
    }  
}
```

Response Body

```
{  
    "status": "success",  
    "data": {  
        "status": "success",  
        "data": {  
            "name": "Ahmet",  
            "salary": "1230",  
            "age": "44",  
            "id": 40  
        }  
    },  
    "message": "Successfully! Record ha  
s been updated."}
```



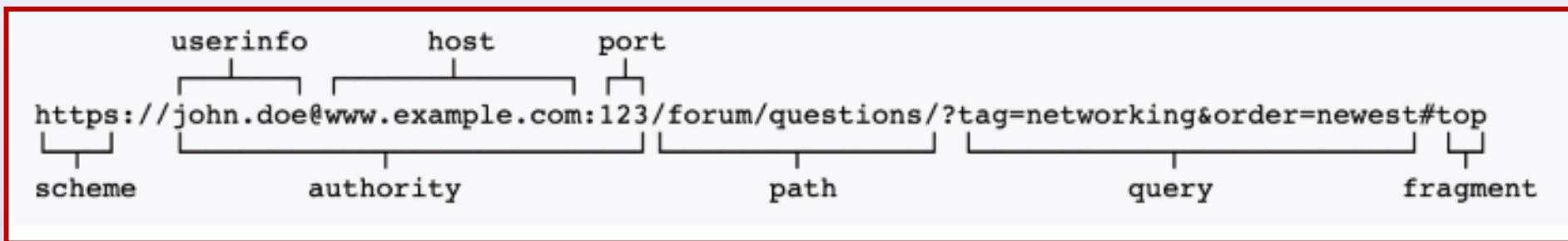
BOLUM 4

FRAMEWORK GELISTIRME

DERS 16

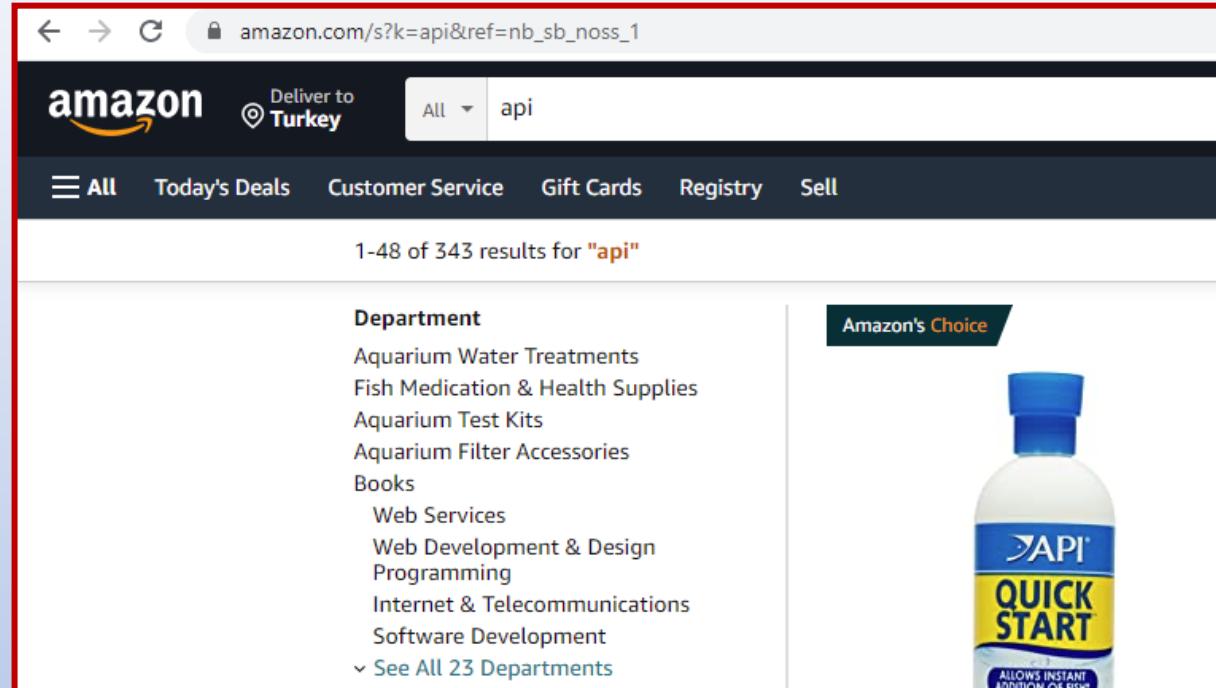
Base URL KULLANIMI

PATH VE QUERY PARAMETRELERİ

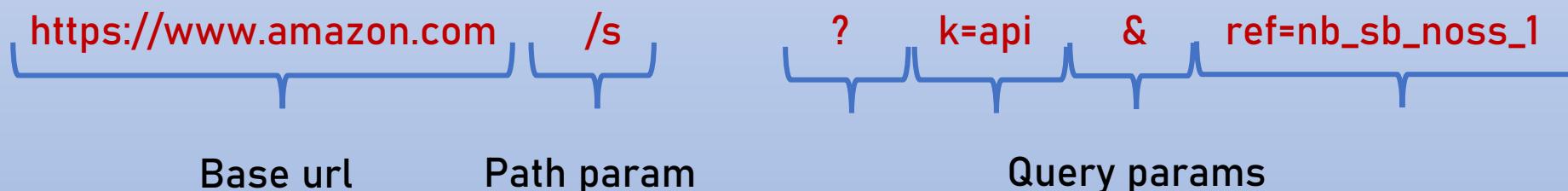


- Bir end-point'i tanımlamak veya daha ayrıntılı nesneleri üzerinde hareket etmek istiyorsanız Path Param kullanmalısınız. Ancak öğeleri sıralamak veya filtrelemek istiyorsanız, Query Parametresi kullanmalısınız. Query parametreleri, kaynakları daha iyi bir şekilde tanımlamaya yardımcı olan benzersiz özelliklere sahiptir.
- Query parametreleri URL'de "?" İşaretinin sağ tarafında görünürken, Path parametreleri soru işaretçi işaretinden önce gelir.
- URL'nin bir parçası oldukları için Path parametrelerindeki değerleri atlayamazsınız. Öte yandan, Query parametreleri URL'nin sonuna eklenir ve bu nedenle serileştirme standartları izlendiği sürece bazı değerlerin çıkarılmasına izin verebilir.
- Query parametreleri key-value şeklinde kullanılır.

PATH VE QUERY PARAMETRELERİ



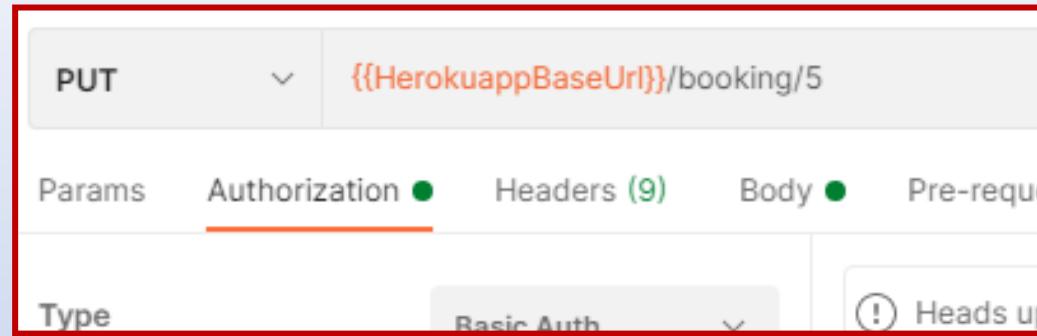
Ornegin amazon sitesine gidip "api" icin soru yaptigimizda, uygulama bizi https://www.amazon.com/s?k=api&ref=nb_sb_noss_1 adresine yonlendirmektedir



Soldaki seceneklerden Web Services'i sectigimizde base url, path param ve ilk query param ayni kalmaktadir

https://www.amazon.com/s?k=api&rh=n%3A283155%2Cn%3A377886011&dc&qid=1622368419&rnid=2941120011&ref=sr_nr_n_6

BASE URL KULLANIMI



Bir sirkette calistigimizda tum API sorgularimiz Base url ile baslayacaktir.

Her sorguda Base url'i tekrar yazmak hem zor, hem de kodlama mantigi acisindan dusuk kalitededir.

Bir framework olustururken hedeflenen temel amaclardan biri de framework'u dinamik yapmaktir.

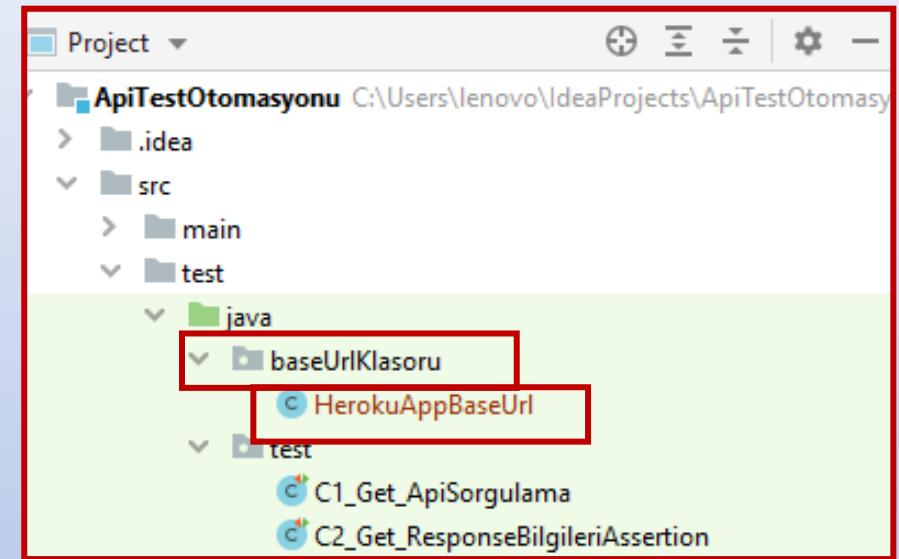
Bunu saglamak icin base url ayri bir class'da olusturulur, test yaptigimiz classlar inheritance metotlarini kullanarak base url'i bulundugu class'dan kullanirlar.

Boylece base url'de yapilacak bir degisiklikte tum class'lardaki url'leri kontrol edip duzeltmek yerine base url'in bulundugu class'da tek bir degisiklik yapmak yeterli olmaktadır.

BASE URL OLUSTURMA

- 1- Projemizde java package'i altında testlerimizi olusturdugumuz test package'i disinda bir package olusturalim.
- 2- Bu package altında her bir base url icin bir class olusturalim
- 3- Bu class altında instance bir RequestSpecification objesi olusturalim. (Bu objenin ismi genelde spec ile baslar)
- 4- Olusturdugumuz objeyi child class'lardan ulasabilmek icin protected yapalim

```
public class HerokuAppBaseUrl {  
  
    protected RequestSpecification specHerokuApp;  
  
    @Before  
    public void setup(){  
        specHerokuApp=new RequestSpecBuilder().  
            setBaseUri("https://restful-booker.herokuapp.com")  
            build();  
    }  
}
```



- 5- Her metoddan once otomatik olarak calismasi icin @Before notasyonunu kullanarak bir setup metodu olusturalim ve protected olarak olusturdugumuz objeye deger atayalim.

BASE URL KULLANIMI

BaseUrl ile PathParams Kullanimi

```
public class C11_BaseUrlKullanimi extends HerokuAppBaseUrl {  
  
    @Test  
    public void get01(){  
  
        // https://restful-booker.herokuapp.com/booking/10  
  
        specHerokuApp.pathParams(s: "pp1", o: "booking", ...objects: "pp2", 10);  
  
        Response response =given().spec(specHerokuApp).when().get( s: "{pp1}/{pp2}");  
  
        response.prettyPrint();  
    }  
}
```

1- Olusturdugumuz base url class'indaki metodlari kullanabilmek icin extends keyword ile base url class'ini inherit edelim.

2- Olusturdugumuz spec objesi ile birlikte base url'den sonra kullanilacak path parametresi 1 tane ise pathparam, path parametreleri birden fazla ise pathparams metodunu kullanalim.

3- Kullanilacak parametreler “parametre ismi”, “parametre degeri” seklinde esli olarak yazilir. Birden fazla parametre kullanacaksak bu ikililer aralarina virgul yazilarak art arda eklenebilir.

4- Response degerini hesaplarken given() metodundan sonra spec(istenenSpec) metodu yazilir ve when() metodundan sonra yazilan HTTP metodunun icine parametre isimleri { } icinde yazilir.

5- Yazilan parametrelerden once ve parametrelerin arasinda / kullanilir.

6- Yazilan parametrelerin siralamasi onemlidir. Endpoint'teki siralamaya uyulmalidir.

BASE URL KULLANIMI

C15_BaseUrlJsonPlaceHolder

Class icinde 3 Test metodu olusturun ve asagidaki testleri yapin

- 1- <https://jsonplaceholder.typicode.com/posts> endpointine bir GET request gönderdigimizde donen response'un status code'unun 200 olduğunu ve Response'ta 100 kayit olduğunu test edin
- 2- <https://jsonplaceholder.typicode.com/posts/44> endpointine bir GET request gönderdigimizde donen response'un status code'unun 200 olduğunu ve "title" degerinin "optio dolor molestias sit" olduğunu test edin
- 3- <https://jsonplaceholder.typicode.com/posts/50> endpointine bir DELETE request gönderdigimizde donen response'un status code'unun 200 olduğunu ve response body'sinin null olduğunu test edin



BOLUM 4
FRAMEWORK GELISTIRME

DERS 17

Base URL KULLANIMI 2

BASE URL KULLANIMI

<https://restful-booker.herokuapp.com/>

Base URL		Method	Endpoints	Tanim
https://restful-booker.herokuapp.com	JSON	GET	/booking	tum rezervasyonlari listele
		GET	/booking/1	id ile rezervasyon goruntule
		POST	/booking?firstname=Ali&lastname=Can&totalprice=123&depositpaid=true&additionalneeds=Wifi	yenii rezervasyon olustur
		PATCH	/booking/12	Kismi guncelleme
		PUT	/booking/11	guncelleme
		DELETE	/booking/11	silme

Tum endpoint'ler incelediginde baseUrl olarak

<https://restful-booker.herokuapp.com>

secilmesi isabetli olacaktir

BASE URL KULLANIMI

C16BaseUrlHerokuapp

Class icinde 2 Test metodu olusturun ve asagidaki testleri yapin

1- <https://restful-booker.herokuapp.com/booking> endpointine bir GET request gönderdigimizde donen response'un status code'unun 200 oldugunu ve Response'ta 12 booking oldugunu test edin

2- <https://restful-booker.herokuapp.com/booking> endpointine asagidaki body'ye sahip bir POST request gönderdigimizde donen response'un status code'unun 200 oldugunu ve "firstname" degerinin "Ahmet" oldugunu test edin

```
{  
    "firstname" : "Ahmet",  
    "lastname" : "Bulut",  
    "totalprice" : 500,  
    "depositpaid" : false,  
    "bookingdates" : {  
        "checkin" : "2021-06-01",  
        "checkout" : "2021-06-10"  
    },  
    "additionalneeds" : "wi-fi"  
}
```

BASE URL KULLANIMI

C17BaseUrlHerokuappQueryParam

Class icinde 3 Test metodu olusturun ve asagidaki testleri yapin

- 1- <https://restful-booker.herokuapp.com/booking> endpointine bir GET request gönderdigimizde donen response'un status code'unun 200 oldugunu ve Response'ta 12 booking oldugunu test edin

- 2- <https://restful-booker.herokuapp.com/booking> endpointine gerekli Query parametrelerini yazarak "firstname" degeri "Eric" olan rezervasyon oldugunu test edecek bir GET request gönderdigimizde, donen response'un status code'unun 200 oldugunu ve "Eric" ismine sahip en az bir booking oldugunu test edin

- 3- <https://restful-booker.herokuapp.com/booking> endpointine gerekli Query parametrelerini yazarak "firstname" degeri "Jim" ve "lastname" degeri "Jackson" olan rezervasyon oldugunu test edecek bir GET request gönderdigimizde, donen response'un status code'unun 200 oldugunu ve "Jim Jackson" ismine sahip en az bir booking oldugunu test edin



BOLUM 4

FRAMEWORK GELISTIRME

DERS 18

TestData CLASS KULLANIMI

TEST DATA CLASS KULLANIMI

Test Datası Nedir ? : Bir test sırasında request ile gönderilen (**request body**) veya test sonucunda donmesi beklenen dataların (**expected data ve temel response bilgileri**) tamamina Test Datası denir.

2016 yılında IBM tarafından yürütülen bir araştırmaya atıfta bulunarak, test verilerini aramak, yönetmek, sürdürmek ve oluşturmak, test uzmanlarının zamanının% 30-60'ını kapsamaktadır.

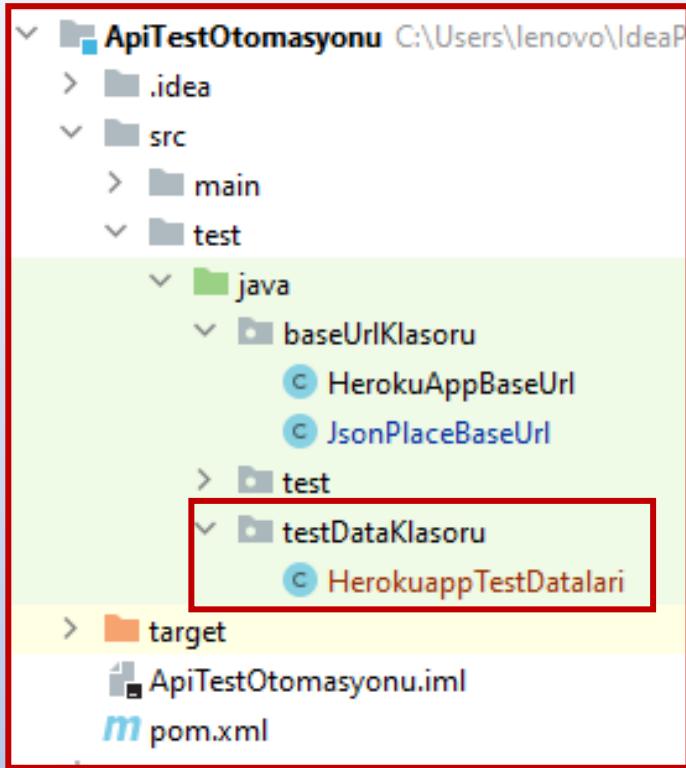
C18_Get_TestDataClassKullanimi

<https://jsonplaceholder.typicode.com/posts/22> url'ine bir GET request yolladığımızda donen response'in status kodunun 200 ve response body'sinin aşağıda verilen ile aynı olduğunu test ediniz

Response body :

```
{  
    "userId": 3,  
    "id": 22,  
    "title": "dolor sint quo a velit explicabo quia nam",  
    "body": "eos qui et ipsum ipsam suscipit aut\\nsed omnis non odio\\nexpedita ear  
        um mollitia molestiae aut atque rem suscipit\\nnam impedit esse"  
}
```

TEST DATA CLASS KULLANIMI



- 1) Test datalarimizi tutmak icin Java package'i altında bir package olusturalim
- 2) Olusturdugumuz package her endpoint icin bir class olusturup testler icin gerekli olan tum test datalarini bu class'da olusturalim.
- 3) Olusturdugumuz Class'da temel response bilgileri icin instance variable, request ve response body icin metod olusturabiliriz.

```
String basariliStatusCode = "200";
String basarisizStatusCode = "404";

public JSONObject getRequestExpectedBody() {
    JSONObject expectedDataJson=new JSONObject();
    expectedDataJson.put("userId",3);
    expectedDataJson.put("id",22);
    expectedDataJson.put("title","dolor sint quo a ve");
    expectedDataJson.put("body","eos qui et ipsum ips");

    return expectedDataJson;
}
```

TEST DATA CLASS KULLANIMI

C18_Get_TestDataClassKullanimi

<https://jsonplaceholder.typicode.com/posts/22> url'ine bir GET request yolladigimizda donen response'in status kodunun 200 ve response body'sinin asagida verilen ile ayni oldugunu test ediniz

Response body :

```
{  
    "userId": 3,  
    "id": 22,  
    "title": "dolor sint quo a velit explicabo quia nam",  
    "body": "eos qui et ipsum ipsam suscipit aut\\nsed omnis non odio\\nexpedita ear  
        um mollitia molestiae aut atque rem suscipit\\nnam impedit esse"  
}
```

TEST DATA CLASS KULLANIMI

C19_Put_TestDataClassKullanimi

<https://jsonplaceholder.typicode.com/posts/70> url'ine asagidaki body'e sahip bir PUT request yolladigimizda donen response'in status kodunun **200**, content type'inin “application/json; charset=utf-8”, Connection header degerinin “**keep-alive**” ve response body'sinin asagida verilen ile ayni oldugunu test ediniz

Request Body

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

Expected Data :

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```



BOLUM 4
FRAMEWORK GELISTIRME

DERS 19

TestData CLASS KULLANIMI 2

TEST DATA CLASS ILE EXPECTED DATA TESTİ

C20_Get_TestDataKullanimi

http://dummy.restapiexample.com/api/v1/employee/3 url'ine bir GET request gönderdigimizde donen response'un status code'unun 200, content Type'inin application/json ve body'sinin asagidaki gibi oldugunu test edin.

Response Body

```
{  
    "status": "success",  
    "data": {  
        "id": 3,  
        "employee_name": "Ashton Cox",  
        "employee_salary": 86000,  
        "employee_age": 66,  
        "profile_image": ""  
    },  
    "message": "Successfully! Record has been fetched."  
}
```

TEST DATA CLASS ILE EXPECTED DATA TESTİ

C21_Post_TestDataKullanimi

<https://restful-booker.herokuapp.com/booking> url'ine asagidaki body'ye sahip bir POST request gonderdigimizde donen response'un id haric asagidaki gibi oldugunu test edin.

Request body	Response Body
{ "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" }	{ "bookingid": 24, "booking": { "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" } }



BOLUM 5
FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DERS 20

BOLUM GIRIS ve
DE-SERIALIZATION

NICIN FARKLI TEKNIKLERE İHTİYAC DUYARIZ ?

REST API'da istenen veri türüyle işlem yapabilir, ancak JSON veri tipi ile diğer data türlerine göre çok daha düşük boyutlarda veri kullanıldığından genellikle JSONObject kullanımı tercih edilir.

Response Body :

```
{  
    "firstname": "Susan",  
    "lastname": "Smith",  
    "totalprice": 835,  
    "depositpaid": true,  
    "bookingdates": {  
        "checkin": "2020-09-07",  
        "checkout": "2021-05-28"  
    }  
}
```

Expected Data :

```
{  
    "status": "success",  
    "data": {  
        "name": "test",  
        "salary": "123",  
        "age": "23",  
        "id": 25  
    }  
}
```

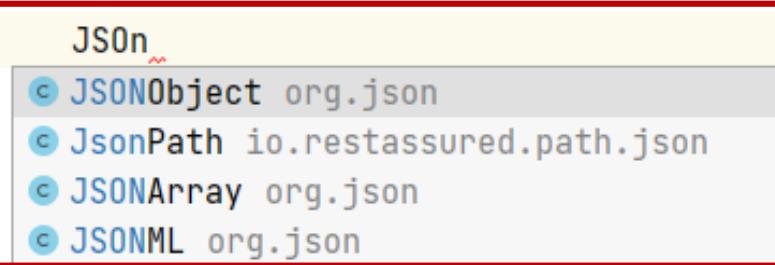
Patch Request Body :

```
{  
    "firstname": "hasan",  
    "lastname": "bayat"  
}
```

NICIN FARKLI TEKNIKLERE IHTIYAC DUYARIZ ?

JSONObject ile tüm testlerimizi yapabiliyoruz, ancak Json objesi Java objesi degildir.

Biz org.json ve io.restassured kutuphanelerini kullanarak Json objeleri ile işlem yapabiliyoruz.



İş hayatımızda tamamen Java objeleri kullanarak test yapmamız istenebilir veya karsımıza çıkacak kompleks Json datalarını API testimizde kullanmak üzere tek tek yazarak oluşturmak zor olabilir.

Bu durumda Java'dan yararlanarak ve farklı teknikler kullanarak test datalarımızı oluşturabilir ve bu dataları kullanarak testlerimizi yapabiliriz

```
1 {
2     "id": "60cefad94111ba444aec4c59",
3     "name": "api board3",
4     "desc": "",
5     "descData": null,
6     "closed": false,
7     "idOrganization": "608bb1f8c74f6f3f28e98a67",
8     "idEnterprise": null,
9     "pinned": false,
10    "url": "https://trello.com/b/iDcE0so7/api-board3",
11    "shortUrl": "https://trello.com/b/iDcE0so7",
12    "prefs": {
13        "permissionLevel": "private",
14        "hideVotes": false,
15        "voting": "disabled",
16        "comments": "members",
17        "invitations": "members",
18        "selfJoin": true,
19        "cardCovers": true,
20        "isTemplate": false,
21        "cardAging": "regular",
22        "calendarFeedEnabled": false,
23        "background": "blue",
24        "backgroundImage": null,
25        "backgroundImageScaled": null,
26        "backgroundTile": false,
27        "backgroundBrightness": "dark",
28        "backgroundColor": "#0079BF",
29        "backgroundBottomColor": "#0079BF",
30        "backgroundTopColor": "#0079BF",
31        "canBePublic": true,
32        "canBeEnterprise": true,
33        "canBeOrg": true,
34        "canBePrivate": true,
35        "canInvite": true
36    },
37    "labelNames": {
38        "green": "",
39        "yellow": "",
40        "orange": "",
41        "red": "",
42        "purple": "",
43        "blue": "",
44        "sky": "",
45        "lime": "",
46        "pink": "",
47        "black": ""
48    },
49    "limits": {
50    }
51 }
```

DE-SERIALIZATION ve MAP KULLANIMI

```
{  
    "firstname": "Susan",  
    "lastname": "Smith",  
    "totalprice": 835,  
    "depositpaid": true,  
    "bookingdates": {  
        "checkin": "2020-09-07",  
        "checkout": "2021-05-28"  
    }  
}
```

İşlem yaptığımız nesneyi, sınıfı saklamak yada transfer etmek istediğimiz formata dönüştürme işlemine **Serialization** denir.

Java objelerini API sorguları yapmak üzere Json objesine cevirmeye **Serialization** denir.

Verilen Json objesini testlerimizde kullanmak üzere Java objesine cevirmeye ise **De-Serialization** denir.

DE-SERIALIZATION ve MAP KULLANIMI

```
{  
    "firstname": "Susan",  
    "lastname": "Smith",  
    "totalprice": 835,  
    "depositpaid": true,  
    "bookingdates": {  
        "checkin": "2020-09-07",  
        "checkout": "2021-05-28"  
    }  
}
```

JSONObject key-value ikililerini kullandigi icin De-Serialization islemi icin Java'dan kullanacagimiz en uygun data turu Map'tir.

Olusturacagimiz Request body veya expected datayi direkt Map olarak olusturabiliriz.

Sorgumuz sonucunda donen response objesini De-Serialization ile Map'e cevirmek icin Gson kutuphanesinden yararlanabiliriz. (Bunun icin Gson dependency'yi pom xml'e eklemeliyiz).

```
Map<String , Object> responseMap = response.as(HashMap.class);
```

SERIALIZATION KULLANIMI (MAP'TEN JSONObject'e CEVIRME)

Map'i JSONObject'e cevirmek icin de Gson Class'indan yardim aliriz. Gson Class'indaki metodlari kullanmak icin once o Class'dan bir object olustururuz.

```
Gson gson = new Gson();  
  
String jsonFromJavaObject = gson.toJson(actualDataMap);
```

Olusturdugumuz gson objesi ile gson.toJson(actualDataMap); metodunu kullanarak, map'i JSONObject olarak kullanabilecegimiz String formatina ceviririz.

DE-SERIALIZATION ILE EXPECTED DATA TESTI

C22_Put_DeSerialization

<https://jsonplaceholder.typicode.com/posts/70> url'ine asagidaki body'e sahip bir PUT request yolladigimizda donen response'in response body'sinin asagida verilen ile ayni oldugunu test ediniz

Request Body

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

Expected Data :

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

DE-SERIALIZATION ILE EXPECTED DATA TESTI

C23_Get_DeSerialization

`http://dummy.restapiexample.com/api/v1/employee/3` url'ine bir GET request gönderdigimizde donen response'un status code'unun 200, content Type'inin application/json ve body'sinin asagidaki gibi oldugunu test edin.

Response Body

```
{  
    "status": "success",  
    "data": {  
        "id": 3,  
        "employee_name": "Ashton Cox",  
        "employee_salary": 86000,  
        "employee_age": 66,  
        "profile_image": ""  
    },  
    "message": "Successfully! Record has been fetched."  
}
```



BOLUM 5
FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DERS 21

DE-SERIALIZATION 2

DE-SERIALIZATION ILE EXPECTED DATA TESTI

C24_Post_Deserialization

<https://restful-booker.herokuapp.com/booking> url'ine asagidaki body'ye sahip bir POST request gonderdigimizde donen response'un id haric asagidaki gibi oldugunu test edin.

Request body	Response Body // expected data
{ "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" }	{ "bookingid": 24, "booking": { "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" } }



BOLUM 5
FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DERS 22

DE-SERIALIZATION 3

DE-SERIALIZATION ILE EXPECTED DATA TESTI

C25_Put_DeSerialization

<https://jsonplaceholder.typicode.com/posts/70> url'ine asagidaki body'e sahip bir PUT request yolladigimizda donen response'in response body'sinin asagida verilen ile ayni oldugunu test ediniz

Request Body

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

Expected Data :

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

DE-SERIALIZATION ILE EXPECTED DATA TESTI

C26_Get_DeSerialization

<http://dummy.restapiexample.com/api/v1/employee/3> url'ine bir GET request gönderdigimizde donen response'un asagidaki gibi oldugunu test edin.

Response Body

```
{  
    "status": "success",  
    "data": {  
        "id": 3,  
        "employee_name": "Ashton Cox",  
        "employee_salary": 86000,  
        "employee_age": 66,  
        "profile_image": ""  
    },  
    "message": "Successfully! Record has been fetched."  
}
```



BOLUM 5
FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DERS 23

POJO KULLANIMI
Plain Old Java Object

POJO ILE EXPECTED DATA TESTI

Pojo : Plain Old Java Object (Basit java objesi)

Kompleks request veya response body'lerini olusturmak uzun islem gerektirebilir.

Daha once yaptigimiz orneklerde TestData Class'lari olusturmus ve tum datalarimizi bu class'larda olusturmustuk.

Pojo kullaniminda her bir Json Objesi icin method degil Class olusturacagiz.

Pojo kullaniminda Java'daki encapsulation ozellikleri kullanilir

Kompleks Json objeleri icin otomatik olarak Pojo Class'lari olusturan web-sitelerinden yardim alabiliriz.

Basit yapidaki Json objeleri icin kendimiz de Pojo Class'lari olusturabiliriz.

```
{  
    "id": "60e17b72a9cb99719157d055",  
    "name": "API2",  
    "desc": "",  
    "descData": null,  
    "closed": false,  
    "idOrganization": "608bb1f8c74f6f3f28e98a67",  
    "idEnterprise": null,  
    "pinned": false,  
    "url": "https://trello.com/b/qPOnicXa/api2",  
    "shortUrl": "https://trello.com/b/qPOnicXa",  
    "prefs": {  
        "permissionLevel": "private",  
        "hideVotes": false,  
        "voting": "disabled",  
        "comments": "members",  
        "invitations": "members",  
        "selfJoin": true,  
        "cardCovers": true,  
        "isTemplate": false,  
        "cardAging": "regular",  
        "calendarFeedEnabled": false,  
        "background": "blue",  
        "backgroundImage": null,  
        "backgroundImageScaled": null,  
        "backgroundTile": false,  
        "backgroundBrightness": "dark",  
        "backgroundColor": "#0079BF",  
        "backgroundBottomColor": "#0079BF",  
        "backgroundTopColor": "#0079BF",  
        "canBePublic": true,  
        "canBeEnterprise": true,  
        "canBeOrg": true,  
        "canBePrivate": true,  
        "canInvite": true  
    },  
    "labelNames": {  
        "green": "",  
        "yellow": "",  
        "orange": "",  
        "red": "",  
        "purple": "",  
        "blue": "",  
        "sky": "",  
        "lime": "",  
        "pink": "",  
        "black": ""  
    },  
    "limits": {}  
}
```

POJO ILE EXPECTED DATA TESTI

Bir POJO Class olusturmak icin, 5 adima ihtiyacimiz var

- 1) Tum variable'lari "private" olarak olusturalim
- 2) Tum variable'lar icin getter() and setter() metodlari olusturalim
- 3) Tum parametreleri iceren bir constructor olusturalim
- 4) Default constructor (parametresiz) olusturalim
- 5) `toString()` metodu olusturalim

```
{  
    "id": "60e17b72a9cb99719157d055",  
    "name": "API2",  
    "desc": "",  
    "descData": null,  
    "closed": false,  
    "idOrganization": "608bb1f8c74f6f3f28e98a67",  
    "idEnterprise": null,  
    "pinned": false,  
    "url": "https://trello.com/b/qPOnicXa/api2",  
    "shortUrl": "https://trello.com/b/qPOnicXa",  
    "prefs": {  
        "permissionLevel": "private",  
        "hideVotes": false,  
        "voting": "disabled",  
        "comments": "members",  
        "invitations": "members",  
        "selfJoin": true,  
        "cardCovers": true,  
        "isTemplate": false,  
        "cardAging": "regular",  
        "calendarFeedEnabled": false,  
        "background": "blue",  
        "backgroundImage": null,  
        "backgroundImageScaled": null,  
        "backgroundTile": false,  
        "backgroundBrightness": "dark",  
        "backgroundColor": "#0079BF",  
        "backgroundBottomColor": "#0079BF",  
        "backgroundTopColor": "#0079BF",  
        "canBePublic": true,  
        "canBeEnterprise": true,  
        "canBeOrg": true,  
        "canBePrivate": true,  
        "canInvite": true  
    },  
    "labelNames": {  
        "green": "",  
        "yellow": "",  
        "orange": "",  
        "red": "",  
        "purple": "",  
        "blue": "",  
        "sky": "",  
        "lime": "",  
        "pink": "",  
        "black": ""  
    },  
    "limits": {}  
}
```

POJO CLASS ILE EXPECTED DATA TESTİ

C27_Put_PojoClass

<https://jsonplaceholder.typicode.com/posts/70> url'ine asagidaki body'e sahip bir PUT request yolladigimizda donen response'in response body'sinin asagida verilen ile ayni oldugunu test ediniz

Request Body

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

Expected Data :

```
{  
    "title": "Ahmet",  
    "body": "Merhaba",  
    "userId": 10,  
    "id": 70  
}
```

POJO CLASS ILE EXPECTED DATA TESTİ

C28_Post_Pojo

<https://restful-booker.herokuapp.com/booking> url'ine asagidaki body'ye sahip bir POST request gonderdigimizde donen response'un id haric asagidaki gibi oldugunu test edin.

Request body	Response Body // expected data
<pre>{ "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" }</pre>	<pre>{ "bookingid": 24, "booking": { "firstname": "Ahmet", "lastname": "Bulut", "totalprice": 500, "depositpaid": false, "bookingdates": { "checkin": "2021-06-01", "checkout": "2021-06-10" }, "additionalneeds": "wi-fi" } }</pre>

POJO CLASS ICİN CONVERTOR KULLANMA

jsonschema2pojo

Star 5,509 Tweet

Generate Plain Old Java Objects from JSON or JSON-Schema.

```
1 {  
2     "bookingid": 38,  
3     "booking": {  
4         "firstname": "Ahmet",  
5         "lastname": "Bulut",  
6         "totalprice": 500,  
7         "depositpaid": false,  
8         "bookingdates": {  
9             "checkin": "2021-06-01",  
10            "checkout": "2021-06-10"  
11        },  
12        "additionalneeds": "wi-fi"  
13    }  
14 }
```

Package pojos

Class name PojoHerokuapp

Source type:

JSON Schema JSON
 YAML Schema YAML

Annotation style:

Jackson 2.x Gson
 Moshi None
 Generate builder methods
 Use primitive types
 Use long integers
 Use double numbers
 Use Joda dates
 Include getters and setters
 Include constructors
 Include `hashCode` and `equals`
 Include `toString`
 Include JSR-303 annotations
 Allow additional properties
 Make classes serializable
 Make classes parcelable
 Initialize collections

Property word delimiters: - _

Response Body // expected data

```
{  
    "bookingid": 24,  
    "booking": {  
        "firstname": "Ahmet",  
        "lastname": "Bulut",  
        "totalprice": 500,  
        "depositpaid": false,  
        "bookingdates": {  
            "checkin": "2021-06-01",  
            "checkout": "2021-06-10"  
        },  
        "additionalneeds": "wi-fi"  
    }  
}
```

Preview

Zip

CONVERTOR KULLANARAK POJO ILE EXPECTED DATA TESTİ

C29_Get_Pojo

<http://dummy.restapiexample.com/api/v1/employee/3> url'ine bir GET request gönderdigimizde donen response'un asagidaki gibi oldugunu test edin.

Response Body

```
{  
    "status": "success",  
    "data": {  
        "id": 3,  
        "employee_name": "Ashton Cox",  
        "employee_salary": 86000,  
        "employee_age": 66,  
        "profile_image": ""  
    },  
    "message": "Successfully! Record has been fetched."  
}
```

POJO ILE EXPECTED DATA TESTİ

C28_Post_Serialization

<https://api.trello.com/1/boards/?key={{Trello api key}} & token={{Trello Token}}&name=API2> url'ine bir post request gönderdigimizde donen response'un yandaki body'ye sahip oldugunu test ediniz

```
{  
    "id": "60e17b72a9cb99719157d055",  
    "name": "API2",  
    "desc": "",  
    "descData": null,  
    "closed": false,  
    "idOrganization": "608bb1f8c74f6f3f28e98a67",  
    "idEnterprise": null,  
    "pinned": false,  
    "url": "https://trello.com/b/qPOnicXa/api2",  
    "shortUrl": "https://trello.com/b/qPOnicXa",  
    "prefs": {  
        "permissionLevel": "private",  
        "hideVotes": false,  
        "voting": "disabled",  
        "comments": "members",  
        "invitations": "members",  
        "selfJoin": true,  
        "cardCovers": true,  
        "isTemplate": false,  
        "cardAging": "regular",  
        "calendarFeedEnabled": false,  
        "background": "blue",  
        "backgroundImage": null,  
        "backgroundImageScaled": null,  
        "backgroundTile": false,  
        "backgroundBrightness": "dark",  
        "backgroundColor": "#0079BF",  
        "backgroundBottomColor": "#0079BF",  
        "backgroundTopColor": "#0079BF",  
        "canBePublic": true,  
        "canBeEnterprise": true,  
        "canBeOrg": true,  
        "canBePrivate": true,  
        "canInvite": true  
    },  
    "labelNames": {  
        "green": "",  
        "yellow": "",  
        "orange": "",  
        "red": "",  
        "purple": "",  
        "blue": "",  
        "sky": "",  
        "lime": "",  
        "pink": "",  
        "black": ""  
    },  
    "limits": {}  
}
```