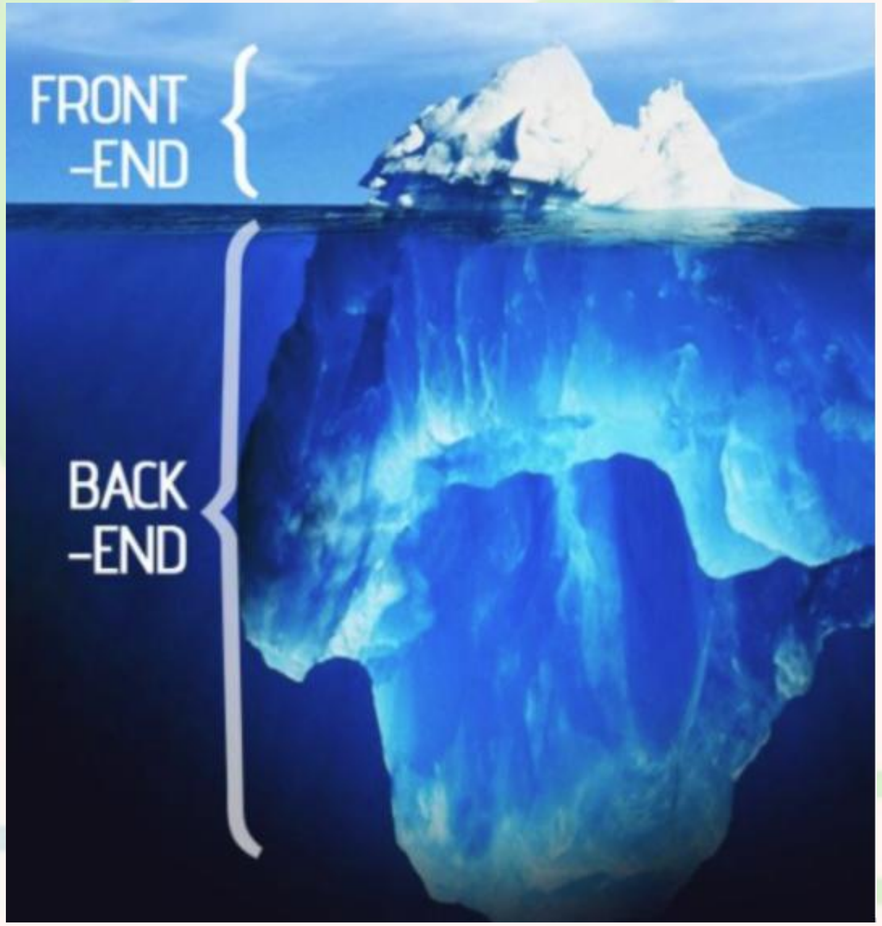
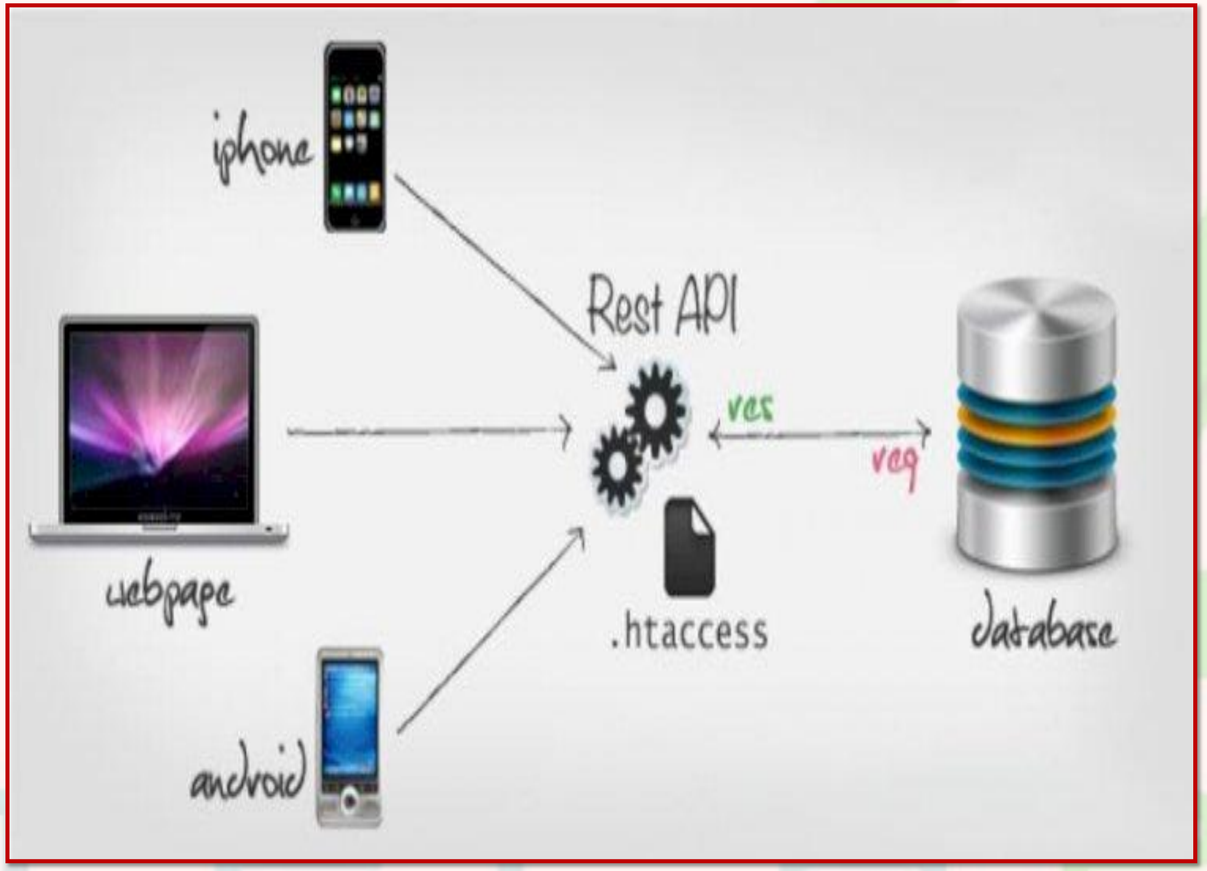
API(Application Programming Interface)
Uygulama Programlama Arayüzü

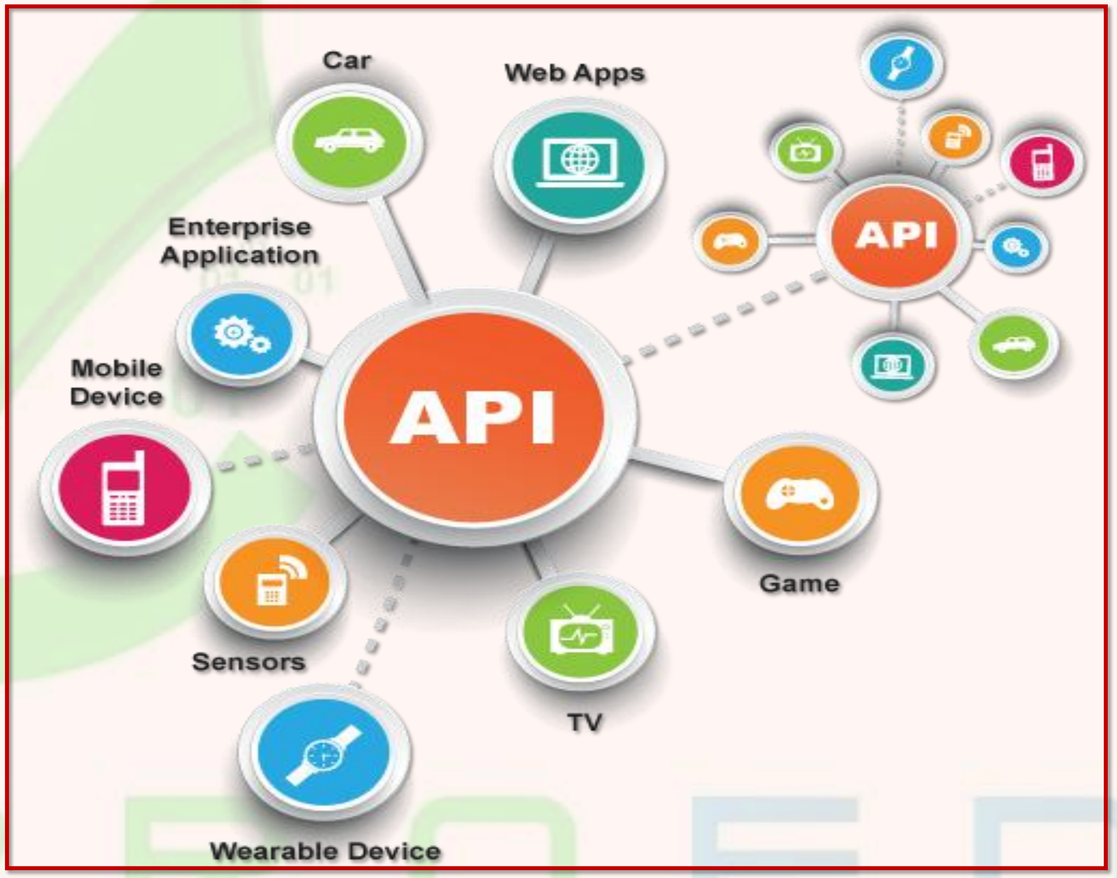


API NEDİR?

- API, bir uygulamanın işlevlerine dışarıdan veya uzaktan erişilip bu işlevlerin kullanılmasını sağlayan arayüzdür.
- API, farklı uygulamalar veya aynı uygulamanın parçaları arasındaki bir arayüz veya iletişim protokolüdür.

- Örneğin; Selenium WebDriver, .jar tipi bir API'dir. Tarayıcılarla konuşmamızı sağlar. Bir iletişim kanalıdır. JDBC kitaplığı, veritabanları ile iletişim kurmamızı sağlayan .jar tipi API'dir.


iphone
Rest API
webpage
res
req
.htaccess
Database
android


Car
Web Apps
Enterprise Application
Mobile Device
API
API
Sensors
Game
TV
Wearable Device

API

- API’in 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
Nasil Calisir ?

Client

1- Request to API

Corba, Pizza, Kola

API
Garson

2- Request to Server

Musteri Corba, Pizza, Kola istiyor

Database/Server

3- Response from Server

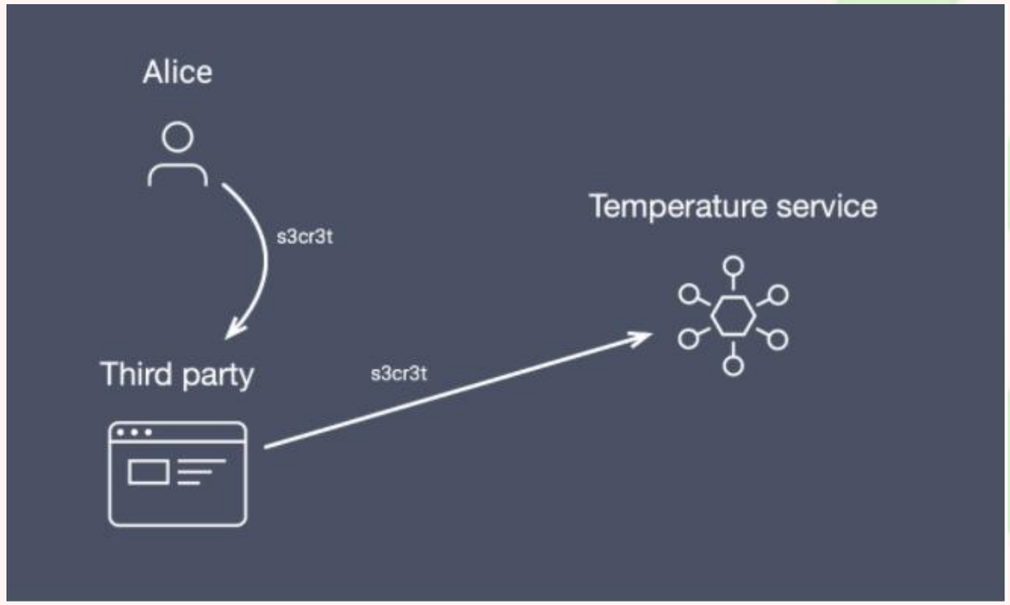
Siparis hazir

4- Response to client

Siparisiniz…

API
Nasil Calisir ?

API
Nasil Calisir ?



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

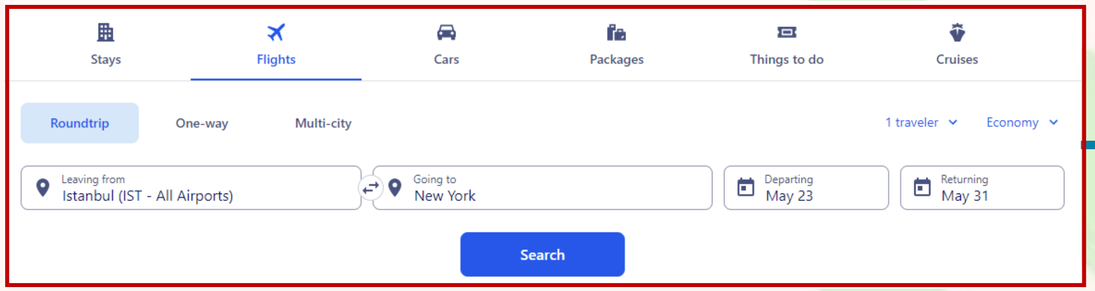
Alice ayrıca, sıcaklık verilerini okumak, sıcaklıkları bir grafik üzerinde çizmek ve diğer hizmetlerden gelen verilerle çapraz referans yapmak için üçüncü taraf bir uygulamaya erişim izni vermek istiyor.

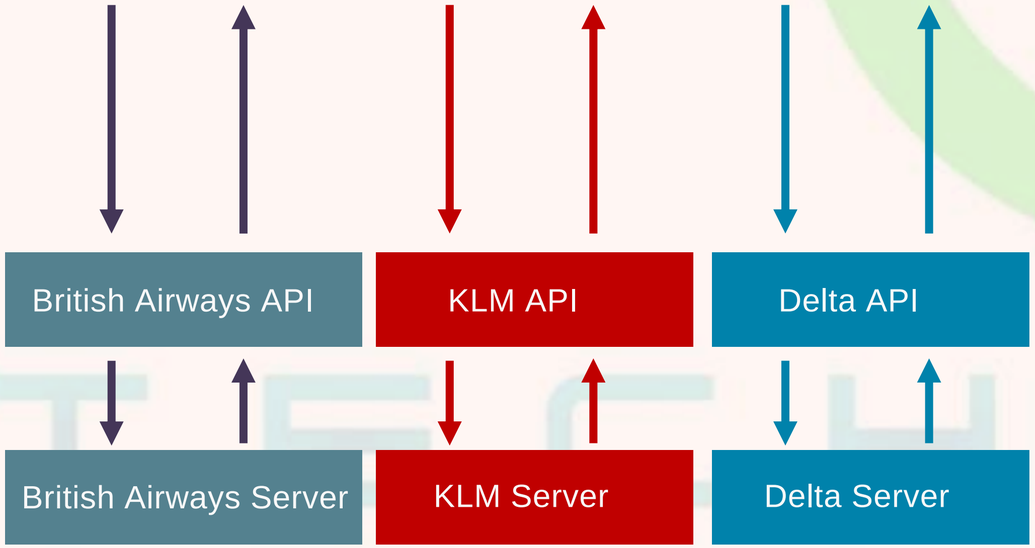
Sıcaklık hizmeti, sıcaklık verileriyle bir API ortaya çıkarır, bu nedenle üçüncü taraf uygulamasının verilere oldukça kolay bir şekilde erişebilmesi gerekir.

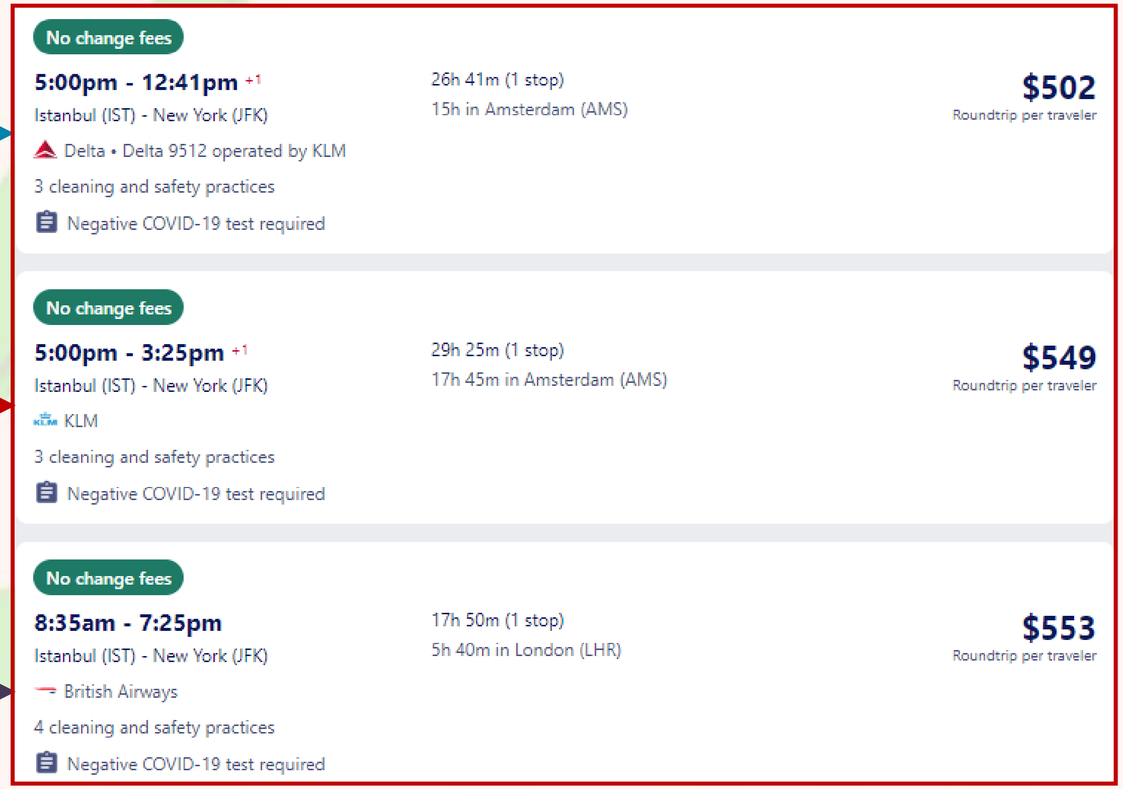
Ayrica uygulamada yalnızca Alice'in verilerinin kullanılabilmesi icin datanin kisisellestirilmesi gerekir.

API
Nasil Calisir?

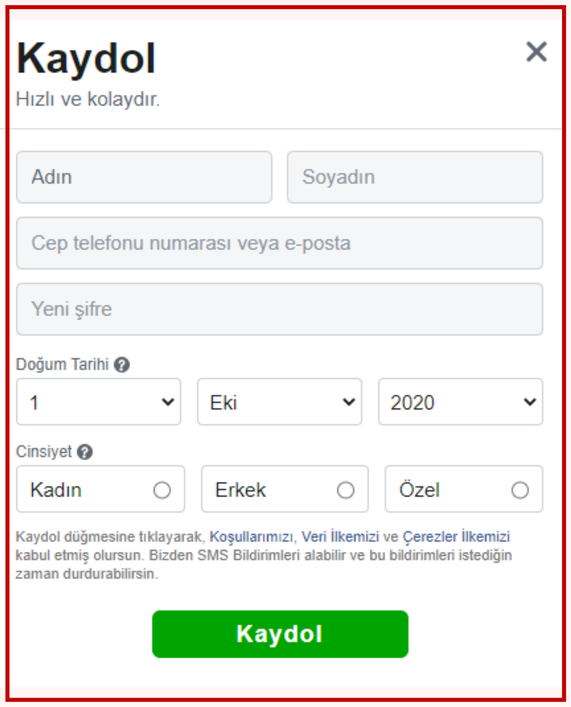
Expedia



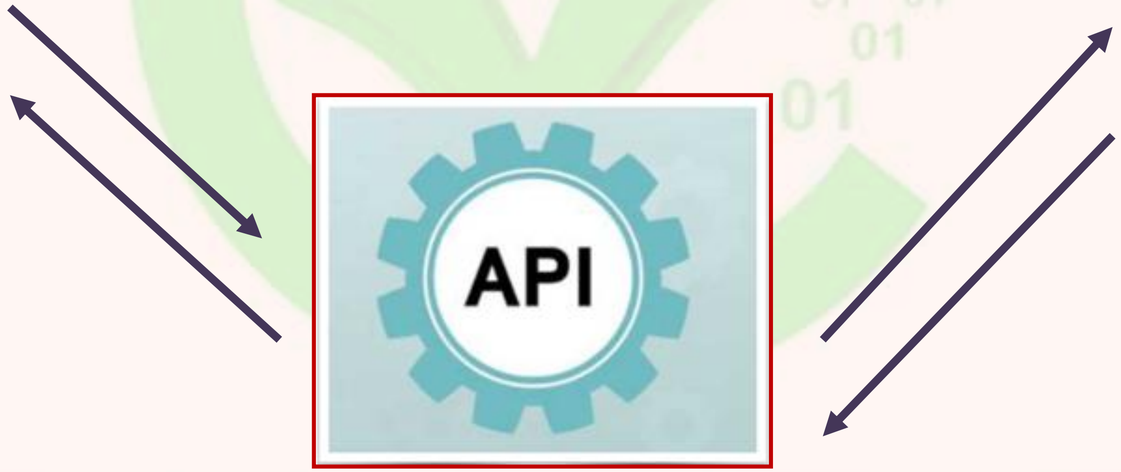




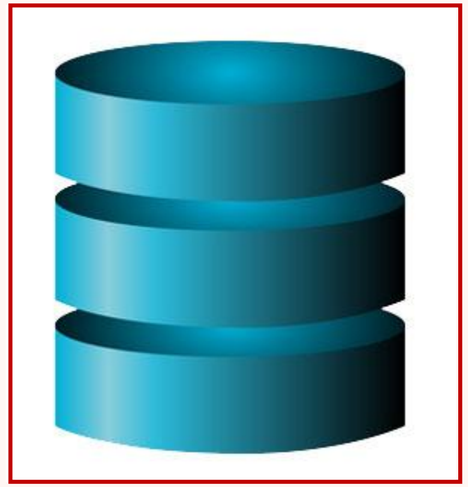
API Testi Nedir?



User Interface



API



Database

API testi uygulamanin goruntusu ile ilgilenmez, uygulamanin fonksiyon, performans ve guvenirlik acisindan beklentilere uygun calisip calismadigini kontrol eder

API vs Web Services

Ikisi de uygulamalar arasinda iletisim saglar.

Web Service : Web servis, elektronik cihaz tarafından başka bir elektronik cihaza sunulan, World Wide Web(www) üzerinden birbirleriyle iletişim kuran yapıların bütününe verilen isimdir. 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.

Http Nedir?

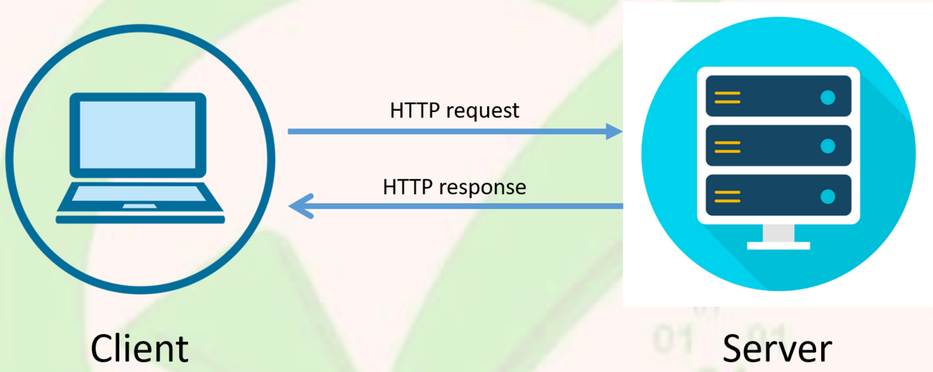
HTTP : Hyper Text Transfer Protocol

HTTP protokolü istemci (PC) ile sunucu (server) arasındaki veri alışverişi 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.

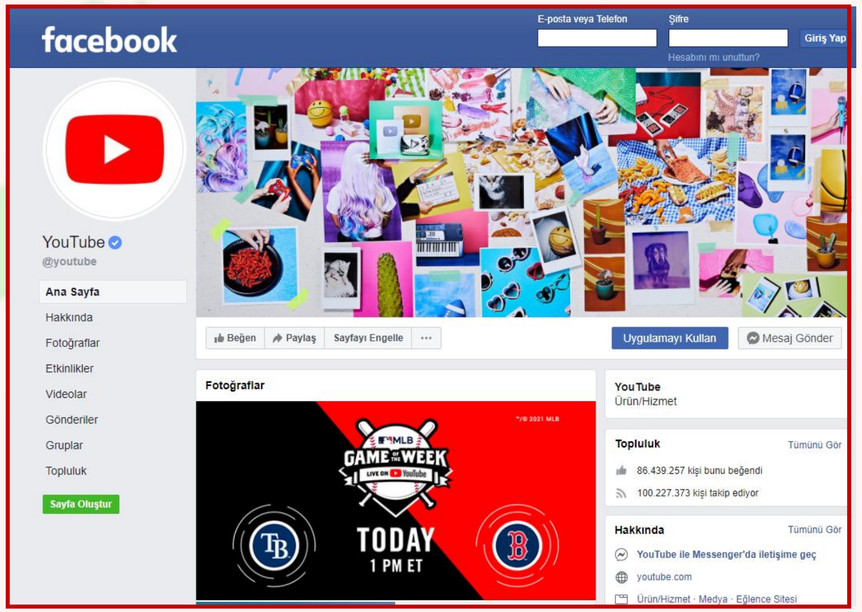
Client ve Server arasindaki tum iletisim request ve response'lar ile olur

Request & Response



http://www. facebook.com/youtube

UI



Request & Response API

API TESTING DE NEYİ DOĞRULUYORUZ.

API Testinde, bilinen verilerle API'ye bir istek göndeririz ve yanıtı analiz ederiz.

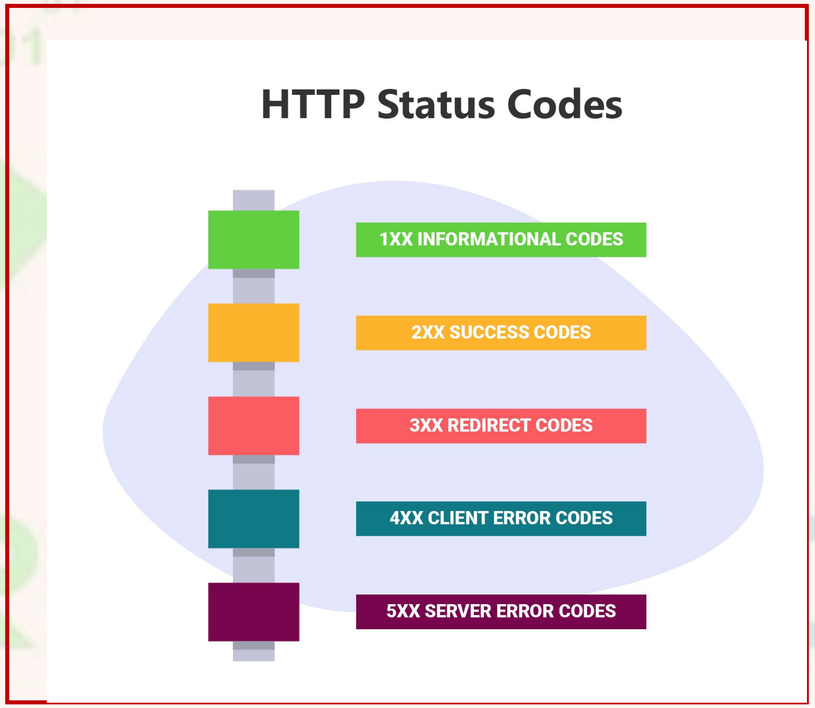
1- HTTP STATUS CODE
2- DATA

Http Durum Kodlari

İstemci bir sunucu içeriğine HTTP kullanarak ulaşmaya çalıştığında sunucu yanıtın durumunu belirten bir sayısal kod gönderir.
Bazı durumlarda HTTP durum kodu (HTTP Status Code) istemcinin tarayıcısında da gösterilebilir
Örn; 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



Genel Http 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 Durum Kodlari

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 Durum Kodlari

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

Http Durum Kodlari

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 gerekli	
408	İstek zaman aşamına uğradı	
409	Conflict	(Hatlar) 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 unsatifiable	
417	Expectation failed	
422	Unprocessable entity	
423	Locked	
424	Method failure	

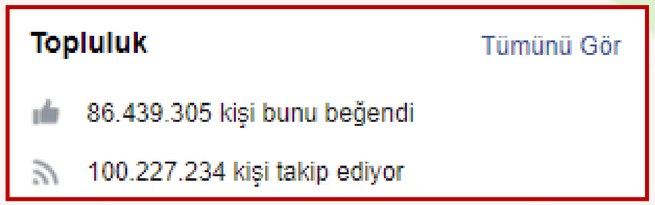
Http Durum Kodlari

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

Request & Response

http://graph.facebook.com/youtube

API




```
{
   "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

http://graph.facebook.com/youtube

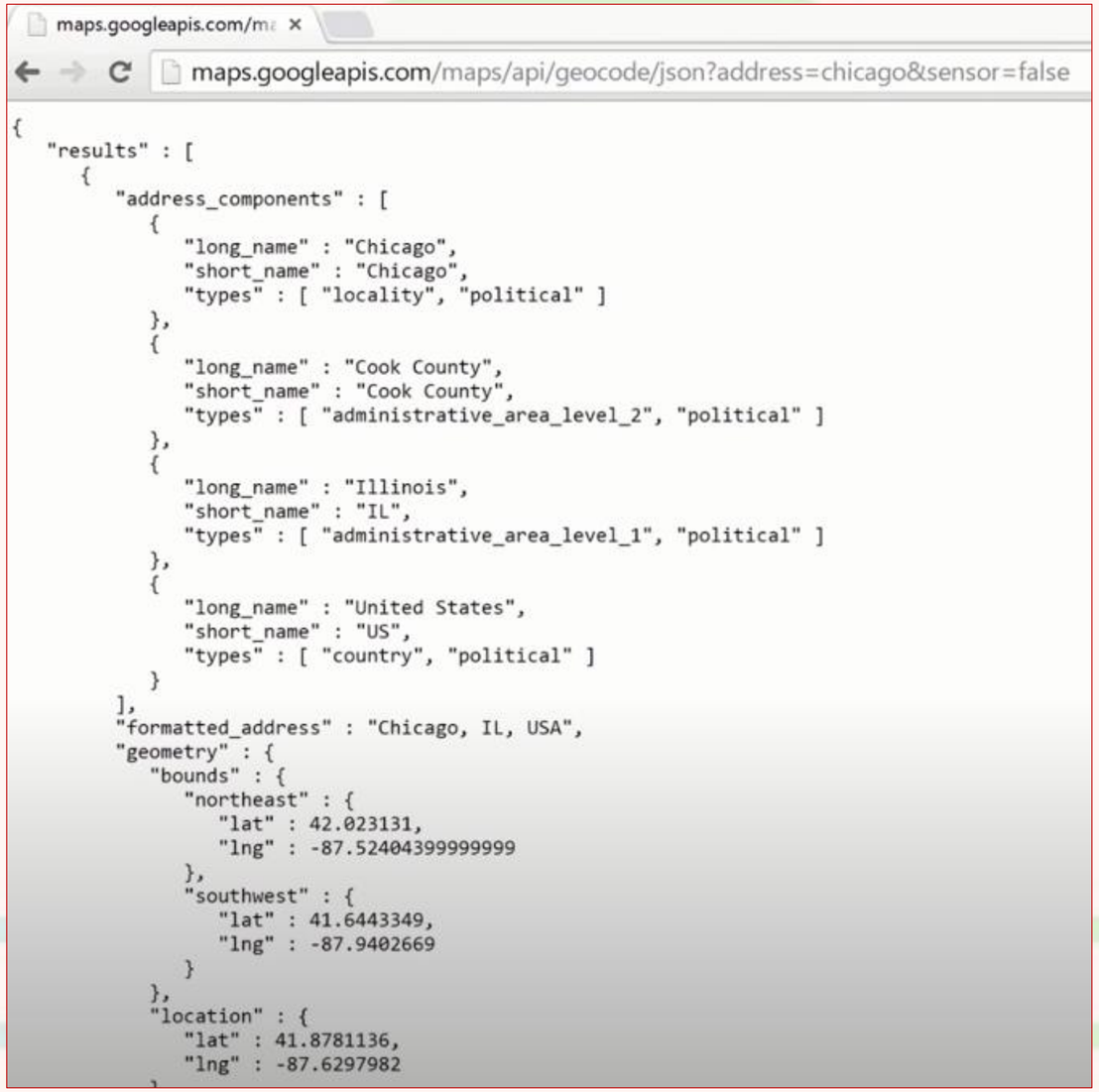
API


```
{
   "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 PROTOKOLLERİ

SOAP

REST

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 biri SOAP protokolü diğeri ise REST'dir

API Protokolleri

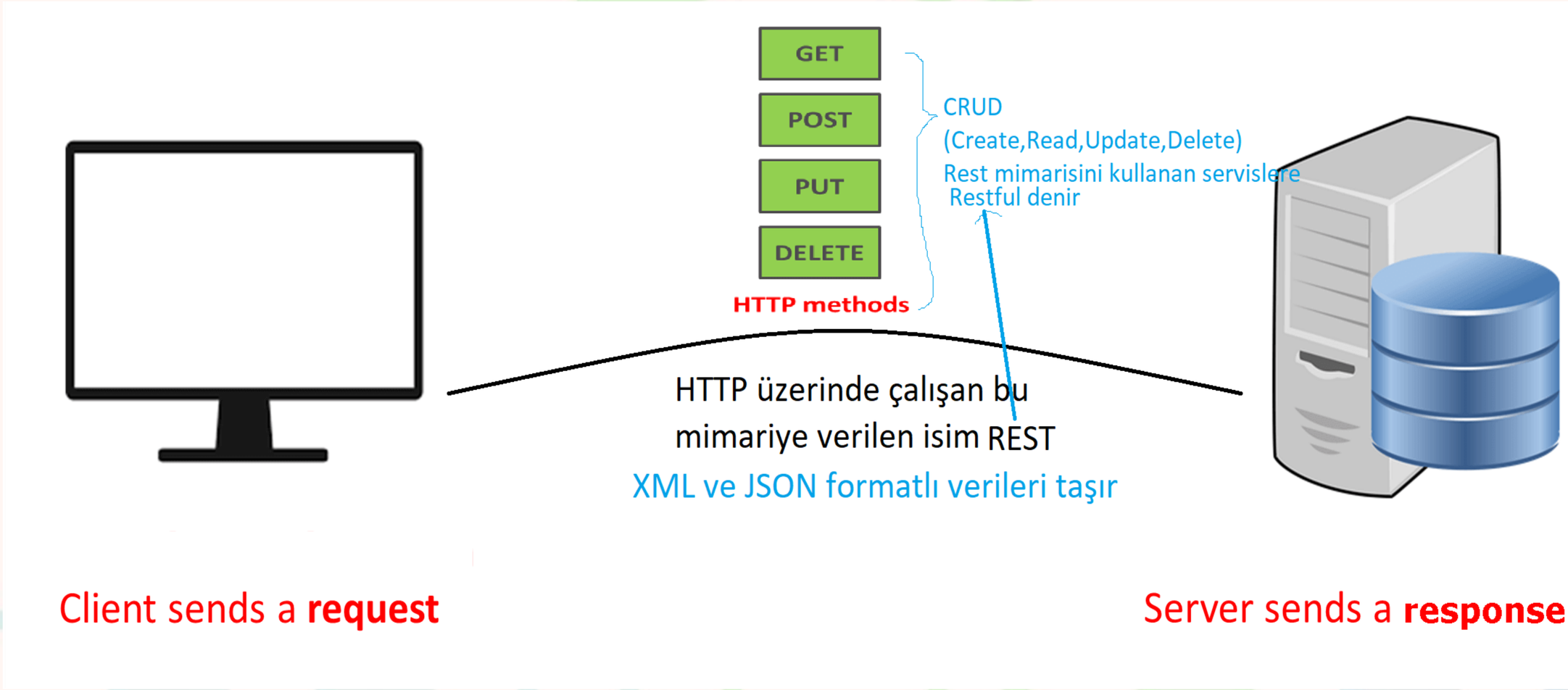
SOAP (Simple Object Access Protocol- Basit Nesne Erişim Protokolü) uygulamalar ile web servislerin bilgi aktarımını sağlayan XML tabanlı bir protokoldür. Yani web servise giden bilgi XML olarak gönderilir, web servis bu bilgiyi yorumlar ve sonucunu XML olarak geri döndürür. XML, makine ve insan tarafından okunabilir şekilde tasarlanmıştır. SOAP tabanlı bir web servisin, gönderilen XML verisini nasıl yorumlayacağının tanımlanması gerekir. Bu web servis tanımlaması WSDL(Web Service Description Language- Web Servisleri Tanımlama Dili) standardı ile yapılır.

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

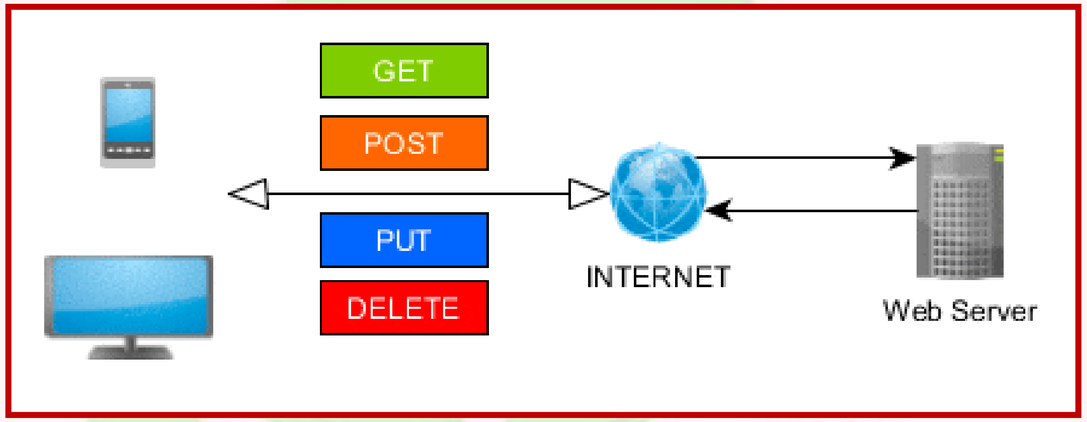
```
<?xml version="1.0" encoding=
<definitions name="AktienKurs
  targetNamespace="http://loc
  xmlns:xsd="http://schemas.xmlsoap.or
  xmlns="http://schemas.xmlsoap.org/wsd
  <service name="AktienKurs">
    <port name="AktienSoapPort" binding
      <soap:address location="http://loc
    </port>
    <message name="Aktie.HoleWert">
      <part name="body" element="xsd:Tra
    </message>
    …
  </service>
</definitions>
```

WSDL

REST


GET
POST
PUT
DELETE
HTTP methods
CRUD
(Create,Read,Update,Delete)
Rest mimarisini kullanan servislere
Restful denir
HTTP üzerinde çalışan bu
mimariye verilen isim REST
XML ve JSON formatlı verileri taşır
Client sends a **request**
Server sends a **response**

API Protokolleri



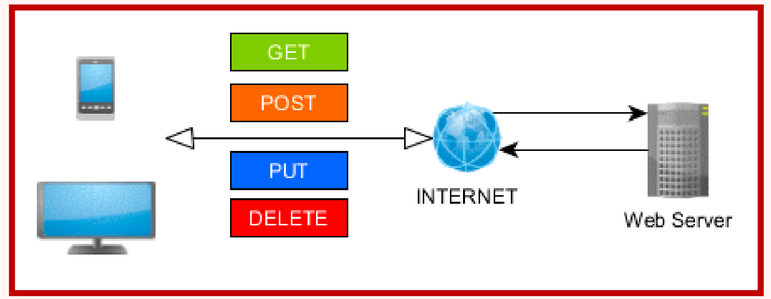
REST (Representational State Transfer), REST mimarisinde ise işlemler resource kavramıyla yapılır. Resource URI (Uniform Resource Identifiers) ile tanımlanır. Yani REST'te SOAP'ta olduğu gibi XML yardımıyla metodlar çağırılmaz bunun yerine o metodu çağıracak URI'ler ile web servise HTTP protokolüyle istek yapılır. Böylece işlemler tamamen HTTP metodları üzerinden yapılır.

Örneğin, bir web servisin metodunu SOAP ile "getProductName" şeklinde çağırırken REST ile "/products/name/1" URI'si ile çağırabiliriz. Ayrıca RESTin döndürdüğü veri tipinin de XML olması zorunlu değildir JSON (Java Script Object Notation) , XML, TXT, HTML gibi istenen veri türünde değer döndürülebilir.

API Protokolleri

HTTP metodların REST ile kullanımı;

REST tabanlı web servislerde HTTP metodlarına özel anlamlar yüklenir ve böylece web servise bir HTTP isteği geldiği anda metod çalıştırılmış olur. Bu durumda HTTP metodlarının REST ile nasıl kullanılacağı önemlidir.



Ornek : http://example.com/resources/item/17

GET: Adresi verilen nesneyi döndürmek için kullanılır. Bu metot kullanıldığında kayıtlarda bir değişiklik yapılmaz

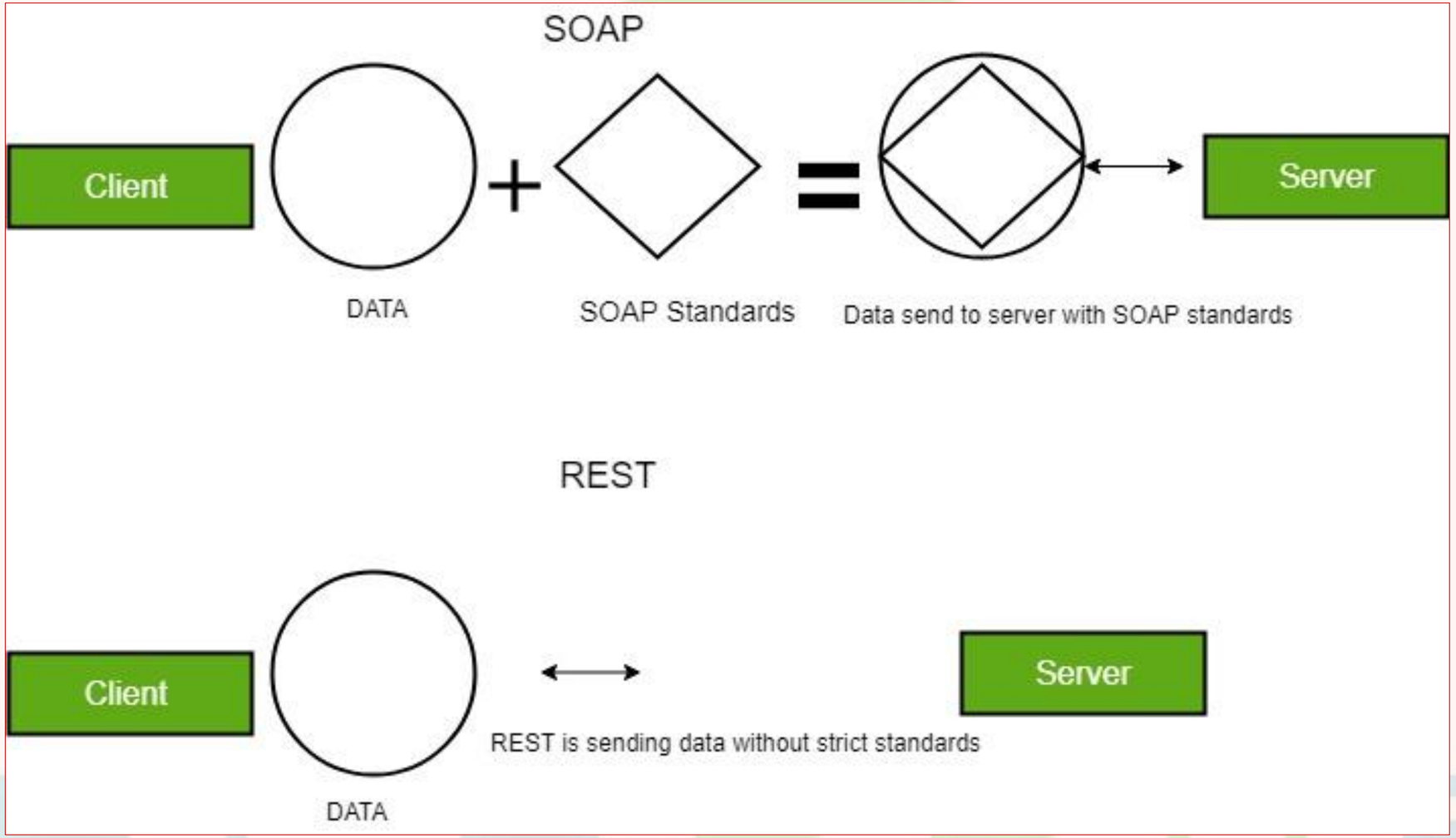
PUT: Var olan bir nesneyi değiştirmek için veya eğer yoksa yeni bir tane oluşturmak için kullanılır.

POST: Yeni bir nesne oluşturmak için kullanılır. Her seferinde yeni bir nesne oluşturur. PUT ile yapılan bir işlem POST ile de yapılabilir fakat aralarındaki fark tarayıcıların bu iki metodu farklı yorumlayıp iki metot için farklı tepkiler verebilmesidir.

DELETE: Adresi verilen nesneyi silmek için kullanılır.

Rest Vs SOAP

SOAP	REST
SOAP bir protokoldür	HTTP protokolünü kullanan bir mimaridir
WDSL ile tasarlama yapmak gerektiğinden kullanması daha zor	HTTP methodları ile tasarlandığı için kullanması daha kolay
Yalnızca XML format kullanır.	XML , JSON , HTML , TXT format kullanır
Önbelliği okuyamaz (can not be cached)	Önbelliği okur (can be cached)
Rest e göre daha yavaş	Soap' a göre daha hızlı
Finansal,iletişim ve ödeme noktalarında kullanılır	Sosyal medya, Web Chat , Mobil uygulamalar da kullanılır
Rest e göre daha güvenlidir.	


SOAP
Client
Server
DATA
SOAP Standards
Data send to server with SOAP standards
REST
Client
Server
REST is sending data without strict standards
DATA

Use in Technology driven sectors

REST

- Social Media
- Web Chat
- Mobile

SOAP

- Financial
- Telecommunication
- Payment Gateways

REST Request Genel Olarak Nelerden Oluşur?

En yalın haliyle bir HTTP isteğini aşağıdaki bilgileri taşır;

Request-Line: HTTP isteğin türü, hangi url'e yapılacağı ve http/https protokolü bilgisi

Header: Yapılan isteği niteleyen ve isteğe ait temel bilgileri içeren parametreleri taşır. Gönderilmesi zorunlu değildir, bir ve birden fazla header parametresi gönderebiliriz.

Body: Eğer POST,PUT,PATCH gibi API üzerinden bazı kayıt ve işlemler yapılmasını istiyorsak, bu bilgileri de isteğimizin Body alanında göndeririz. Restful bir API'da JSON olarak bu bilgileri gönderilmesi tercih edilir.

API TESTING
DAY 02

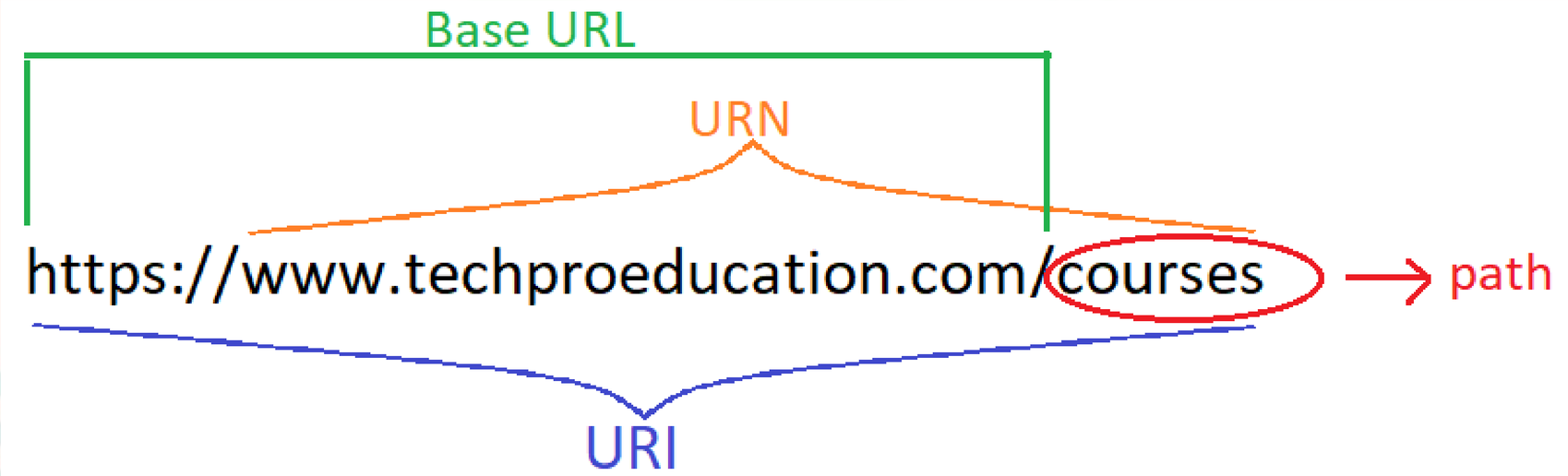
- **GEÇEN DERSTEN HATIRLADIKLARIMIZ**

Endpoints

Endpoint kaynaga nasil erisebilecegimizi gosteren URI (Uniform Resource Identifiers)'lara denir. Bir API olusturdugunuzda kullanicilarin ulasabilmesi icin bu endpoint'i ve kullanilabilecek Http method'larini kullanicilara bildirmemiz gerekir.

URI gunluk hayatta kullandigimiz URL (Uniform Resource Locator)'a benzer.

URL kullanicinin gorebilecegi ve etkilesimde bulunacagi bir web sayfasini ifade ederken URI o sayfanin icerdigi bilgiyi ifade eder.



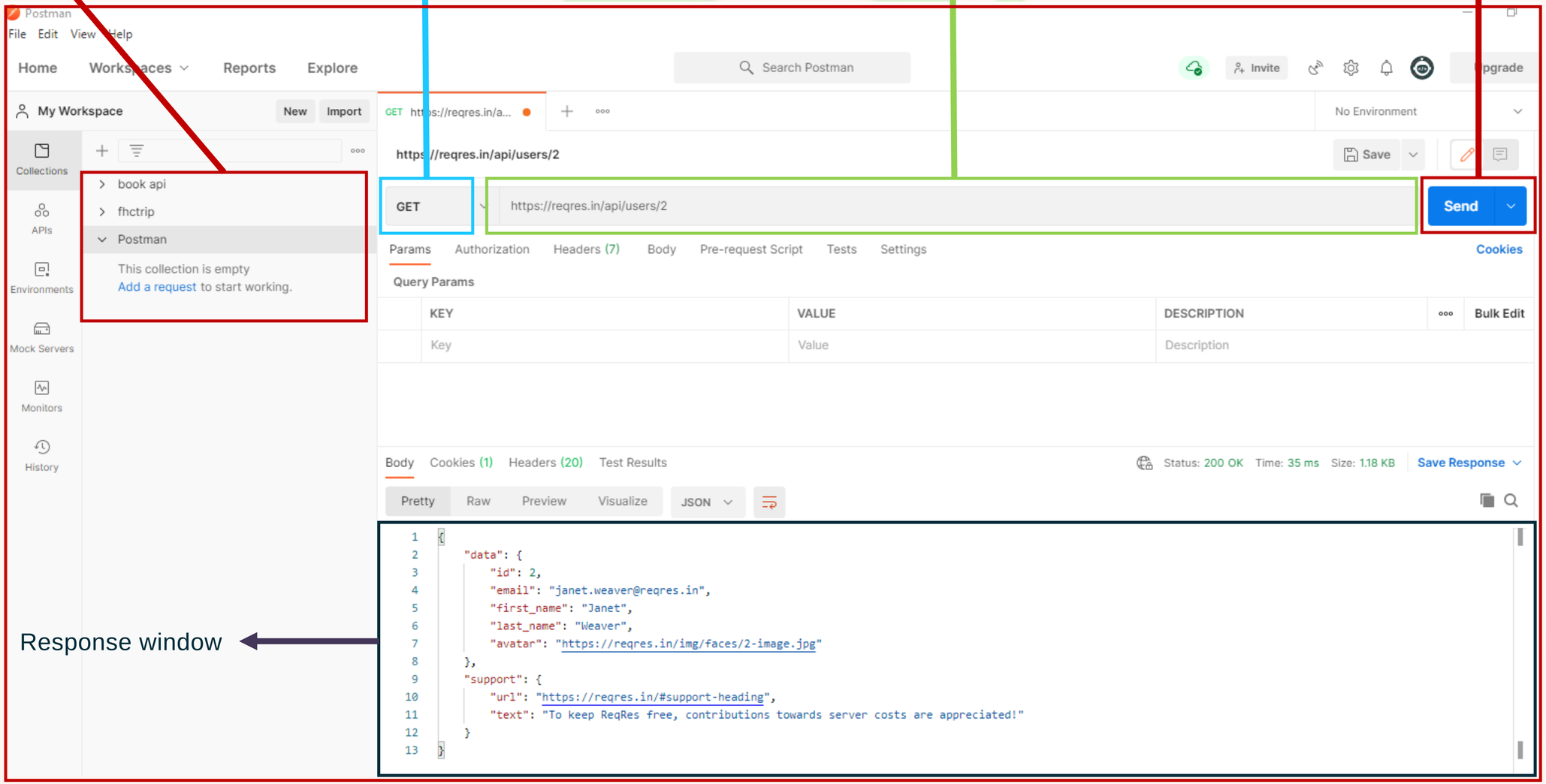
POSTMAN

Requestleri kaydetmek ve kullanmak icin collection olusturulabilir .

Request method

Request URI Satiri

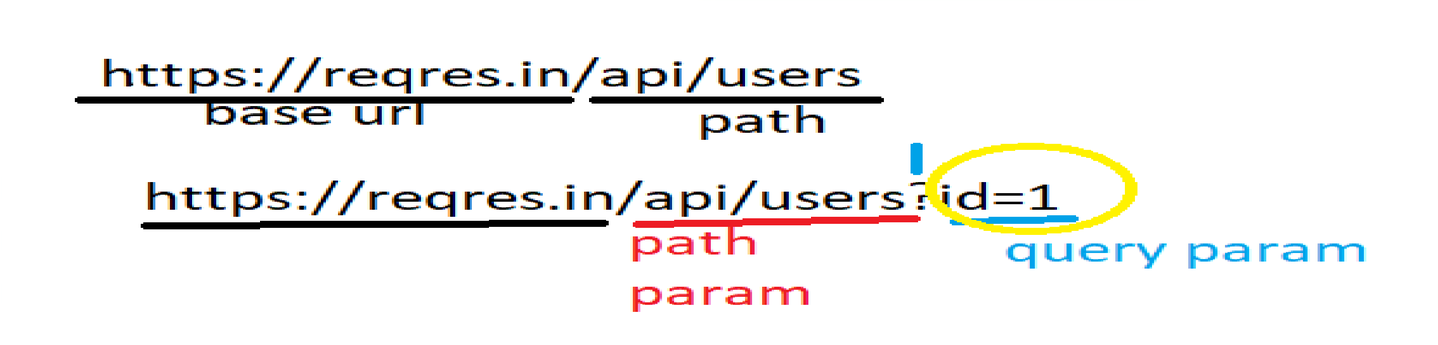
Send Butonu



Response window

Path ve Query Parametreleri

- Bir kaynağı tanimlamak veya daha ayrıntılı nesneleri üzerinde hareket etmek istiyorsaniz 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 işareti işaretinden önce gelir.
- URL'nin bir parçası oldukları için Path parametrelerindeki değerleri atlayamazsınız.
- Query parametreleri key-value seklinde kullanilir.



REGRES.IN -------API TESTING

BASE URL : https://reqres.in

YAPILACAK İŞLEMLER

GET

POST

PUT

PATCH

DELETE

Swagger Documents

Swagger Nedir?

Web API geliştirmede en önemli ihtiyaçlardan biri dokümantasyon ihtiyacıdır. API methodlarının ne işe yaradığı ve nasıl kullanıldığının dokümantasyon içeresinde anlaşılır olması gerekir.

Api dokümantasyonunu el emeği ile yazmak hem zordur hemde güncel tutması imkansızdır. Bir biçimde bu dokümantasyonu güncel olarak üretmek gerekir. Burada imdadımıza swagger yetişiyor.

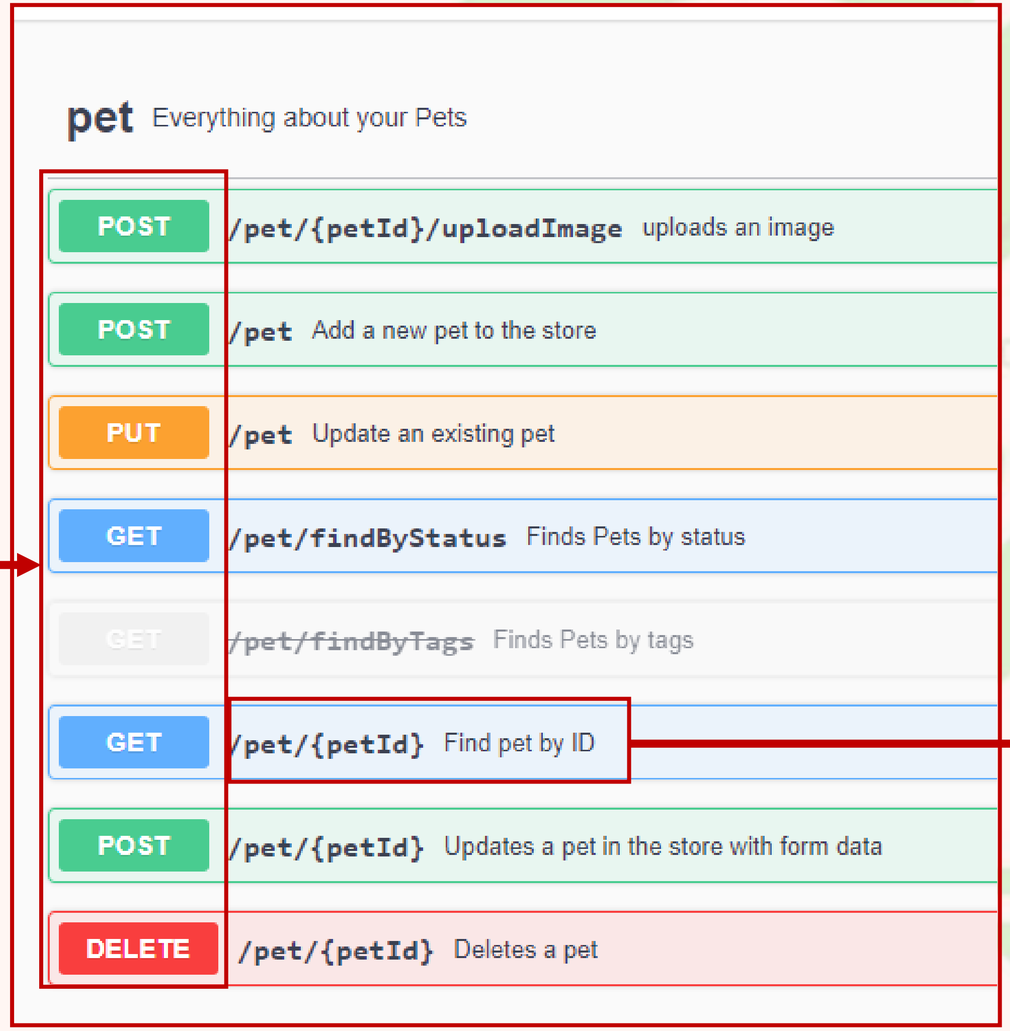
Swagger'ın önemli bir amacı RestApi ler için bir arayüz sağlamaktır. Bu insanların kaynak koda erişmeden RestApi lerin özelliklerini görmesine, incelemesine ve anlamasına olanak sağlar.

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

Swagger Documents

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

Kullanilabilecek
Http Method'lari



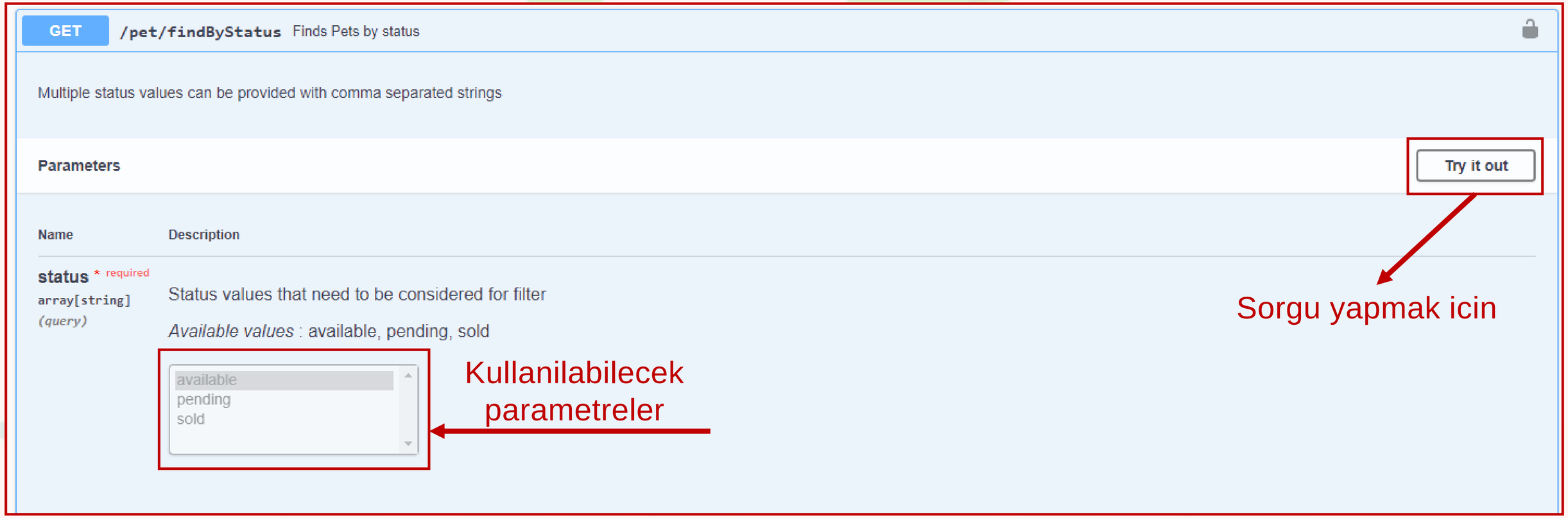
SWAGGER var olan bir API'yi nasil kullanacagimizi gosteren bir dokumandir.

Swagger, yeni gittigimiz bir sehrin gezilecek yerlerini ve nerede ne bulabilecegimizi gosteren bir harita gibidir.

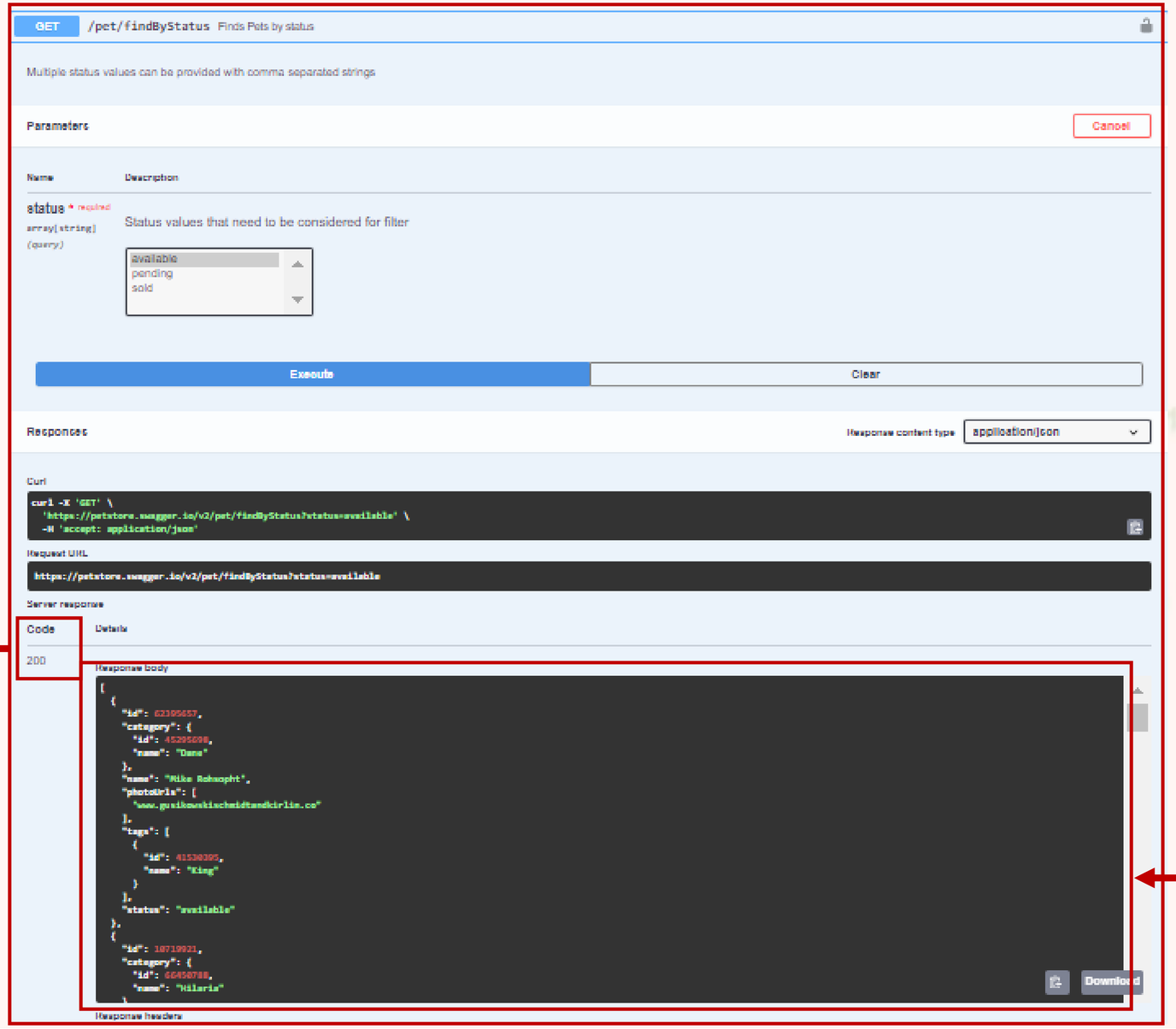
Her bir method icin
Yazilmasi gerekli olan parametreler
Ve method'un islev aciklamasi

Endpoints / Swagger Documents

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



Swagger Documents



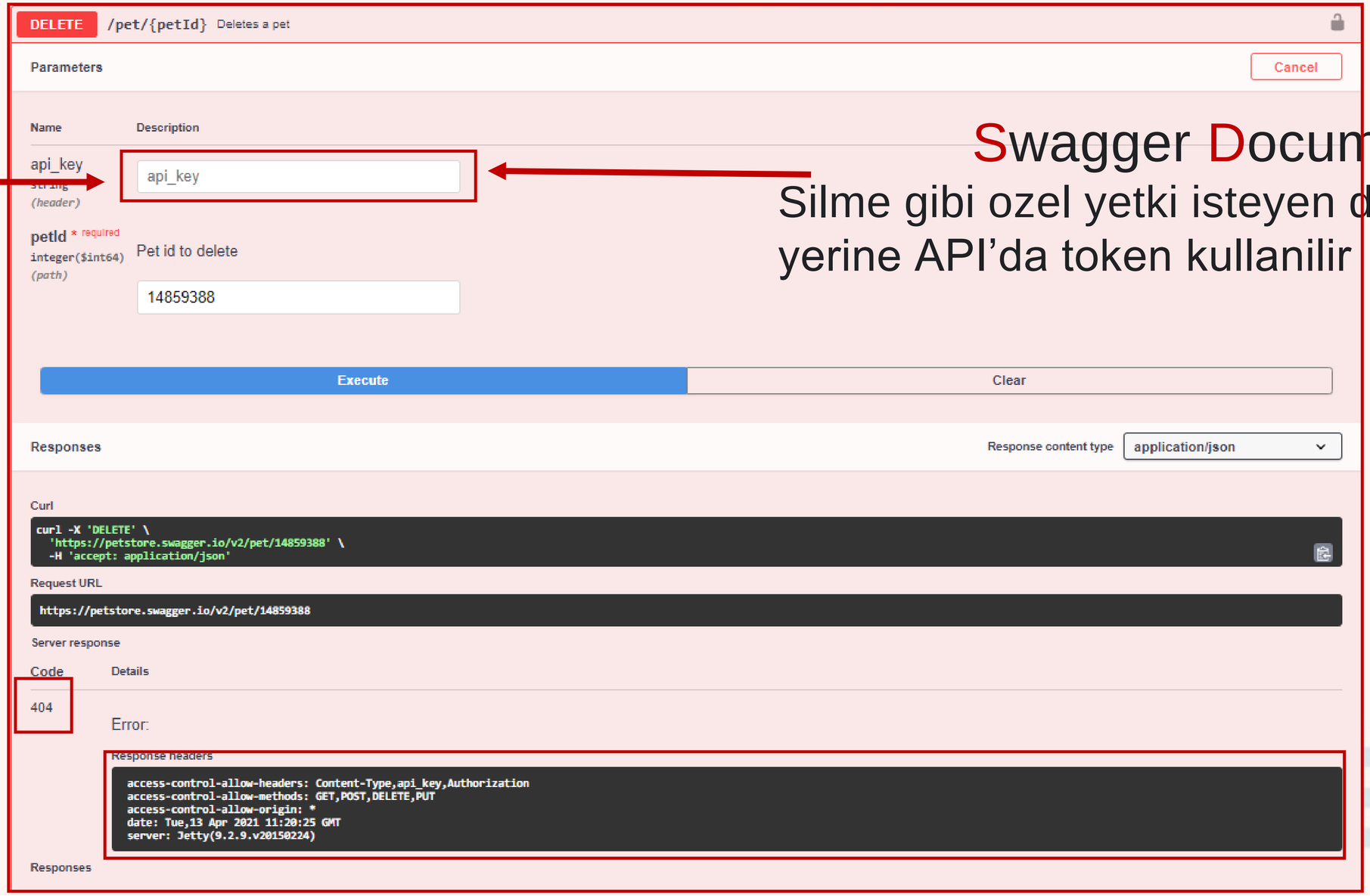
Durum Kodu

Response

Swagger Documents

Silme gibi ozel yetki isteyen durumlarda sifre yerine API'da token kullanilir

Token ?



Durum Kodu

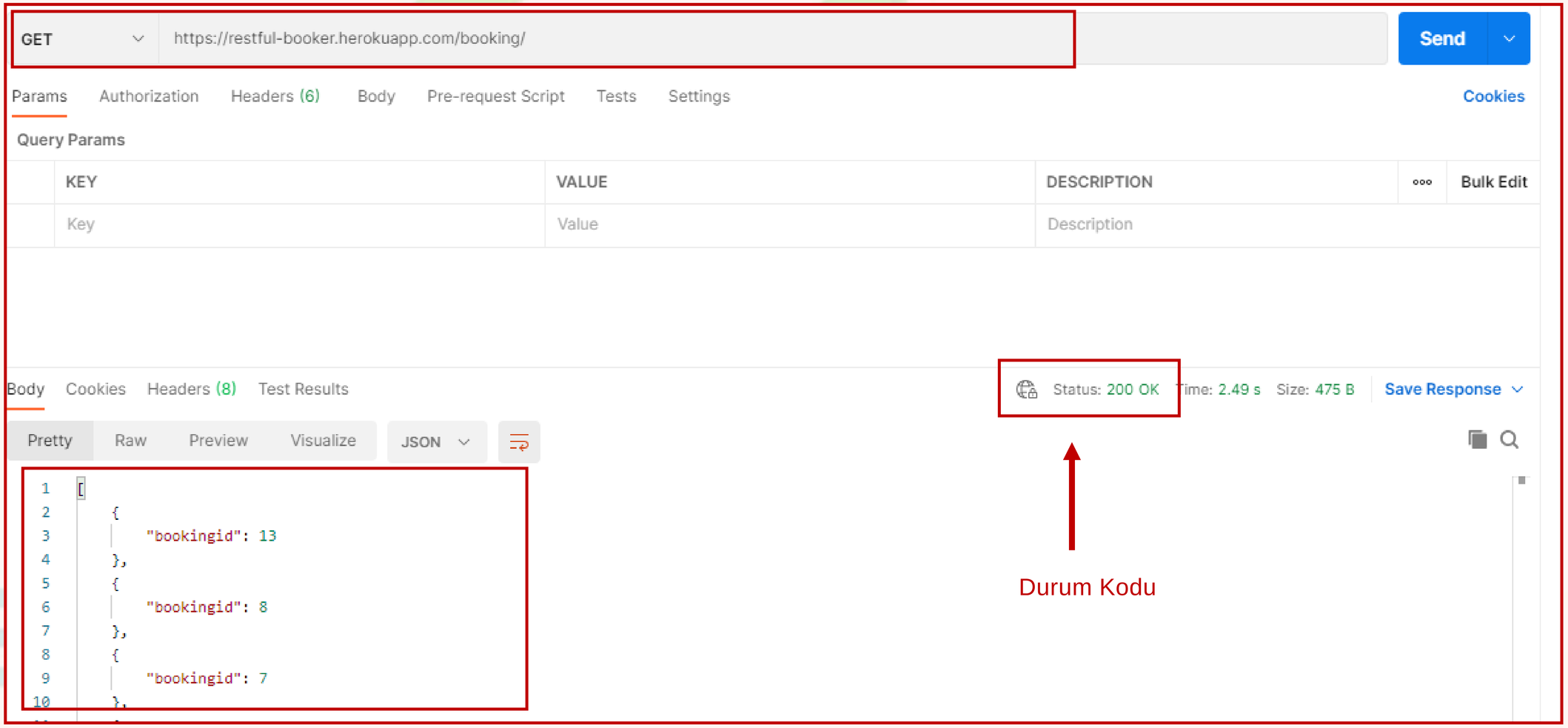
Response

Postman, uzun kodlara ihtiyaç duymadan API'ları paylaşmak, test etmek, dokümante etmek, goruntulemek için kullanılan bir araçtır .

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	yeni rezervasyon olustur
		PATCH	/booking/12	Kismi guncelleme
		PUT	/booking/11	guncelleme
		DELETE	/booking/11	silme

Get Method

Http Method + Request



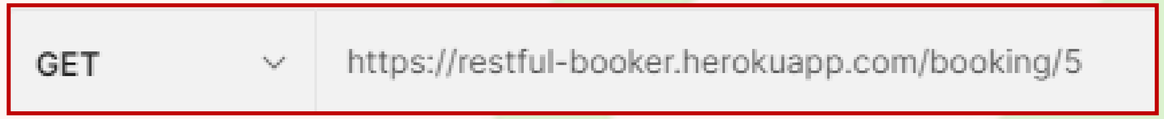
Response

Durum Kodu

POSTMAN

Get Method

Http Method + Request



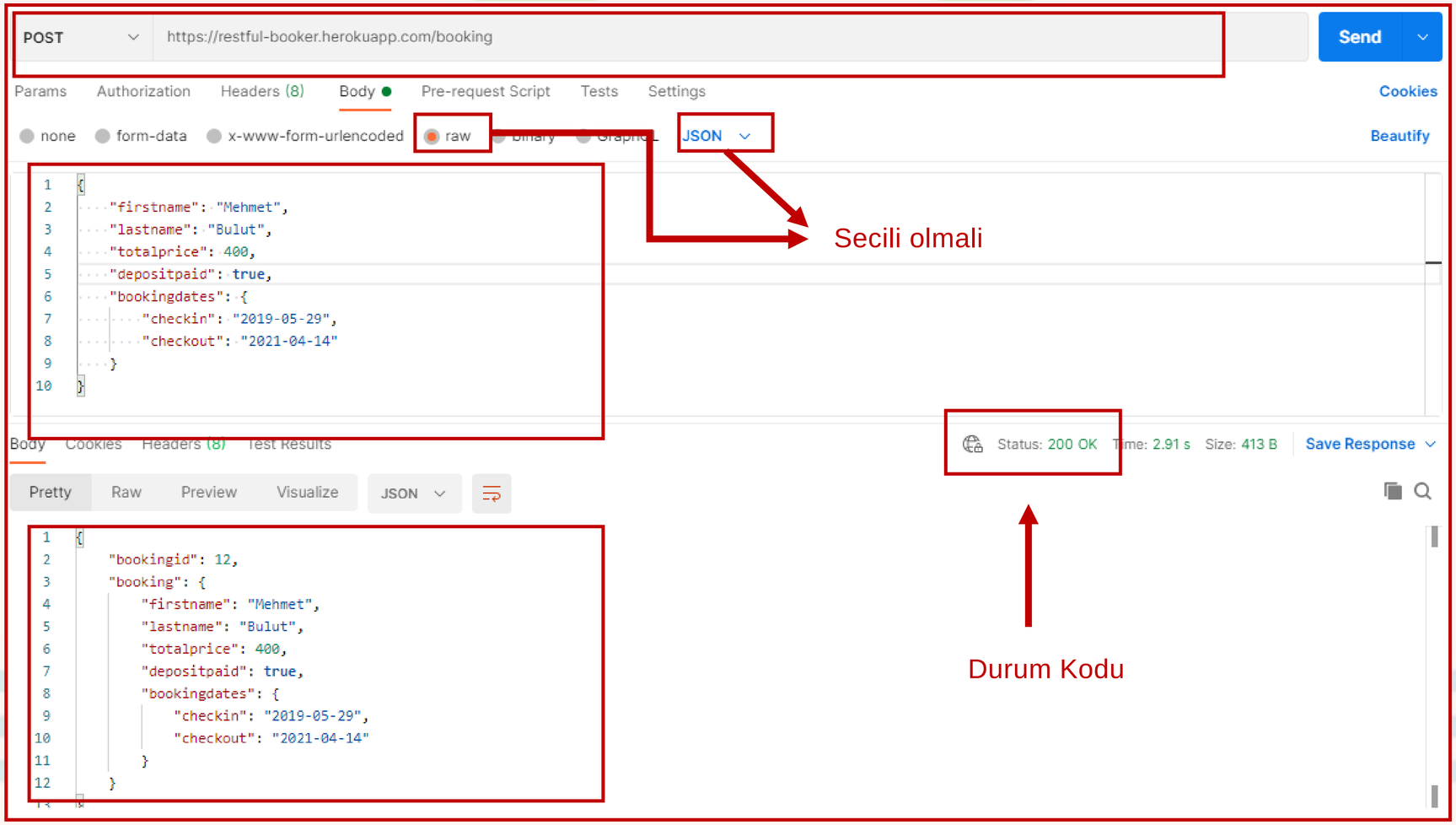
Response

Post Method 1

Http Method + Request

Request Body

Response



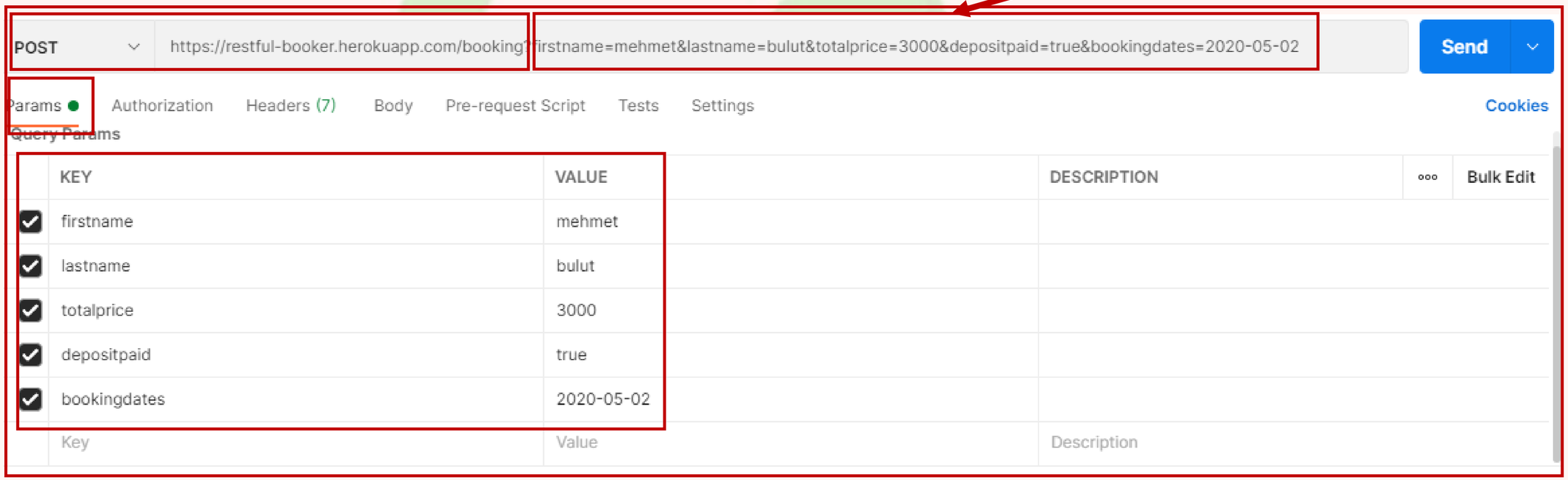
Post Method 2

Girdigimiz degerleri Base URI'ye parametre olarak ekliyor

Base URI

Params

Girilen degerler



POSTMAN

Put Method

https://restful-booker.herokuapp.com/booking/3

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "firstname": "Mary",
    "lastname": "Jackson",
    "totalprice": 798,
    "depositpaid": false,
    "bookingdates": {
        "checkin": "2018-02-24",
        "checkout": "2019-11-08"
    }
}
```

https://restful-booker.herokuapp.com/booking/3

PUT https://restful-booker.herokuapp.com/booking/3

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

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
{
    "firstname": "Hasan",
    "lastname": "Yasa",
    "totalprice": 798,
    "depositpaid": false,
    "bookingdates": {
        "checkin": "2018-02-24",
        "checkout": "2019-11-08"
    }
}
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text

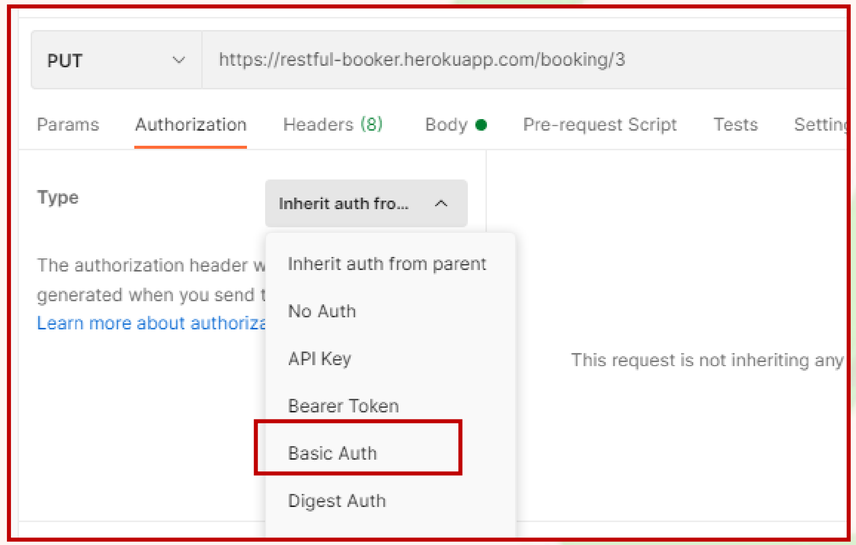
```
Forbidden
```

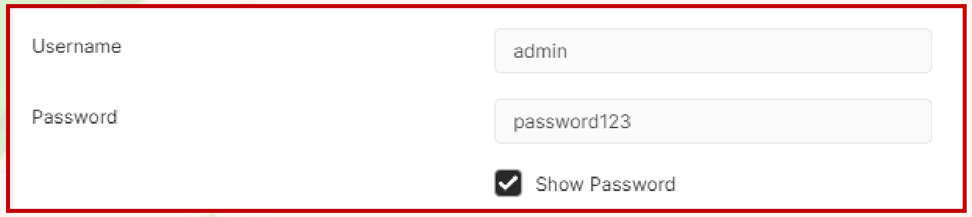
Status: 403 Forbidden

Durum Kodu

POSTMAN

Put Method





```
{
   "store" : {
      "book" : [
         {
            "category" : "reference",
            "author" : "Nigel Rees",
            "title" : "Sayings of the Century",
            "price" : 8.95
         },
         {
            "category" : "fiction",
            "author" : "Evelyn Waugh",
            "title" : "Sword of Honour",
            "price" : 12.99
         },
         {
            "category" : "fiction",
            "author" : "Herman Melville",
            "title" : "Moby Dick",
            "isbn" : "0-553-21311-3",
            "price" : 8.99
         },
         {
            "category" : "fiction",
            "author" : "J. R. R. Tolkien",
            "title" : "The Lord of the Rings",
            "isbn" : "0-395-19395-8",
            "price" : 22.99
         }
      ],
      "bicycle" : {
         "color" : "red",
         "price" : 19.95
      }
   },
   "expensive" : 10
}
```

JsonPath

JsonPath, Json Format ile verilen bir dataya ulasmak veya “manipulate” etmek icin kullanilir.

- $ isareti Json dokumanindaki tum node’lari verir
- Child bolumlere ulasmak icin (.) kullanilabilir. store.book bize tum kitaplari verir
- Belirli bir kitaba ulasmak icin array oldugu icin index kullanabiliriz. store.book[1] , Birden fazla kitaba ulasmak istersek virgule indexleri yazabiliriz. store.book[1,3]
- 2.kitabin price bilgisine ulasmak icin store.book[1].price kullanabiliriz
- Son kitaba ulasmak icin index olarak -1 kullanabiliriz.
- Tum kitaplarin yazarlarini listelemek icin store.book[*].author kullanabiliriz

https://jsonpath.herokuapp.com/

API Testing

```xml
<dependencies>
    <!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
    <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>

</dependencies>
```

IntelliJ ile API testleri yapabilmek icin POM Xml'e io-rest-assured ve junit dependency'lerinin yuklenmesi gerekir

API testlerini yaparken Gherkin dilini kullaniriz.

Given : Istenen Endpoint sorgusundan once yapilacak gereklilikleri ifade eder.

When: Kullanicinin aksiyonunu belirtir

Then: Ciktilari ifade eder.(Assert islemleri genelde then ile yapilir)

And: Coklu islem yapilacaksa kullanilir.

API Testing

GetRequest01:

https://restful-booker.herokuapp.com/booking/3 adresine bir request gonderildiginde donecek cevap(response) icin

- HTTP status kodunun 200
- Content Type'in Json
- Ve Status Line'in HTTP/1.1 200 OK

Oldugunu test edin.

response.prettyPrint(); → Response body i consolda görüntülememizi sağlar

API Testing

GetRequest02:

https://restful-booker.herokuapp.com/booking url'ine
accept type'i "application/json" olan GET request'i yolladigimda
gelen response'un
status kodunun 200
content type'inin "application/json" oldugunu test edin

https://restful-booker.herokuapp.com/booking/1001 url'ine
accept type'i "application/json" olan GET request'i yolladigimda
gelen response'un
status kodunun 404 oldugunu
ve Response body'sinin "Not Found" icerdigini
ve Response body'sinin "API" icermedigini test edin

response.asString() response'u String'e cevirir

API Testing

GetRequest03:

https://restful-booker.herokuapp.com/booking/7 url'ine
accept type'i "application/json" olan GET request'i yolladigimda
gelen response'un
status kodunun 200
ve content type'inin "application/json"
ve firstname'in "Sally"
ve lastname'in "Ericsson"
ve checkin date'in 2018-10-07"
ve checkout date'in 2020-09-30 oldugunu test edin

body("key", Matchers.*equalTo*("Value")) : key olarak verilen degisken'in degerinin value'ya esit olup olmadigini kontrol eder

API Testing

GetRequest04:

https://restful-booker.herokuapp.com/booking/5 url'ine
accept type'i "application/json" olan GET request'i yolladigimda
gelen response'un
status kodunun 200
ve content type'inin "application/json"
ve firstname'in “Jim"
ve totalprice’in 600
ve checkin date'in 2015-06-12"oldugunu test edin

API Testing

GetRequest05:

http://dummy.restapiexample.com/api/v1/employees url'ine

accept type'i "application/json" olan GET request'i yolladigimda

gelen response'un

status kodunun 200

ve content type'inin "application/json"

ve employees sayisinin 24

ve employee'lerden birinin "Ashton Cox"

ve gelen icinde 21, 61, ve 23 degerlerinden birinin oldugunu test edin

body("data.id", Matchers.hasSize(" value ")) : key olarak verilen degisken'in sayisinin value'ya esit olup olmadigini kontrol eder.

body("data.employee_name",Matchers.hasItem(" value ")) : key olarak verilen degisken'in aldigi degerlerin icinde value var mi diye kontrol eder. Value birden fazla ise hasItems kullanip value'lari vigulle yanyana yazilabilir

API Testing

GetRequest06:

https://jsonplaceholder.typicode.com/todos/123 url'ine
accept type'i "application/json" olan GET request'i yolladigimda
gelen response'un
status kodunun 200
ve content type'inin "application/json"
ve Headers'daki "Server" in "cloudflare"
ve response body'deki "userId"'nin 7
ve "title" in "esse et quis iste est earum aut impedit"
ve "completed" bolumunun false oldugunu test edin

Bir Utilities package olusturalim, icinde her bir baseURL icin bir TestBase class'i olsun, hangi baseURL'i kullanmak istersek onun child'ini olusturup testlerimizi yapalim

TestBase Class'i Olusturma

```java
public class TestBaseJsonplaceholder {

    protected RequestSpecification spec01;
    // Sadece child classlarin ulasmasini istiyorum

    @Before
    public void setup01(){
        spec01 = new RequestSpecBuilder().
                            setBaseUri("https://jsonplaceholder.typicode.com").
                            build();
    }
}
```

```java
public void get01(){
    spec01.pathParams( s: "name", o: "todos",  ...objects: "id",123);
    Response response=given().spec(spec01).when().get( s: "/{name}/{id}");
}
```

API Testing

GetRequest07:

https://restful-booker.herokuapp.com/booking/5 url'ine bir request yolladigimda

HTTP Status Code'unun 200

ve response content type'inin "application/JSON" oldugunu

ve response body'sinin asagidaki gibi oldugunu test edin

```
{"firstname": Sally,
  "lastname": "Smith",
  "totalprice": 789,
  "depositpaid": false,
  "bookingdates": {   "checkin": "2017-12-11",
                      "checkout":"2020-02-20" }
}
```

response.jsonPath(); methodu JsonPath class'indan obje ureterek response uzerinden JsonPath class'indaki methodlari kullanmamizi saglar

API Testing

GetRequest08:
http://dummy.restapiexample.com/api/v1/employees url'inde bulunan

1) Butun calisanlarin isimlerini consola yazdıralim
2) 3. calisan kisinin ismini konsola yazdıralim
3) Ilk 5 calisanin adini konsola yazdiralim
4) En son calisanin adini konsola yazdiralim

API Testing

GetRequest09:
http://dummy.restapiexample.com/api/v1/employees

url ine bir istek gönderildiğinde,
status kodun 200,
gelen body de,
5. çalışanın isminin "Airi Satou" olduğunu ,
6. çalışanın maaşının "372000" olduğunu ,
Toplam 24 tane çalışan olduğunu,
"Rhona Davidson" ın employee lerden biri olduğunu
"21", "23", "61" yaşlarında employeeler olduğunu test edin

Oluşturduğumuz json içerindeki değerlere ulaşabilmek için json methodlarını kullanırız. json.getString , json.getInt , json.getList , json.getBoolean gibi…..

API Testing

GetRequest10:
http://dummy.restapiexample.com/api/v1/employees
url ine bir istek gönderildiğinde
Dönen response un
Status kodunun 200,
1)10'dan büyük tüm id'leri ekrana yazdırın ve
10'dan büyük 14 id olduğunu,
2)30'dan küçük tüm yaşları ekrana yazdırın ve
bu yaşların içerisinde en büyük yaşın 23 olduğunu
3)Maası 350000 den büyük olan tüm employee name'leri ekrana yazdırın ve
bunların içerisinde "Charde Marshall" olduğunu test edin

DE-Serialization

Java’da oluşturduğumuz bir nesneyi veya sınıfı, saklamak yada transfer etmek istediğimiz formata dönüştürme işlemine Serialization denir. Bunun tam tersi duruma ise De-serialization denir. Yani API ‘da dönen response’ u Map,List, List of Map, Set gibi Java objelerine çevirme işlemidir. Bu işlemi yapabilmek için aşağıdaki kütüphaneleri kullanmak gerekir.

```
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.6</version>
</dependency>
```

```
<dependency>
    <groupId>org.codehaus.jackson</groupId>
    <artifactId>jackson-mapper-asl</artifactId>
    <version>1.9.13</version>
</dependency>
```

API Testing

GetRequest11:

https://jsonplaceholder.typicode.com/*todos/2* *url ‘ine istek gönderildiğinde,*

Dönen response un

Status kodunun 200, dönen body de,

"completed": değerinin false

"title”: değerinin “quis ut nam facilis et officia qui”

"userId" sinin 1 ve header değerlerinden

"Via" değerinin “1.1 vegur” ve

"Server" değerinin “cloudflare” olduğunu test edin…

API Testing

GetRequest12:

https://restful-booker.herokuapp.com/booking/1 url ine bir istek gönderildiğinde dönen response body nin

```
{
  "firstname": "Eric",
  "lastname": "Smith",
  "totalprice": 555,
  "depositpaid": false,
  "bookingdates": {
      "checkin": "2016-09-09",
      "checkout": "2017-09-21"
  }
} gibi olduğunu test edin.
```

API Testing

GetRequest13:

http://dummy.restapiexample.com/api/v1/employees url ine bir istek gönderildiğinde Status kodun 200 olduğunu,
5. Çalışan isminin "Airi Satou" olduğunu , çalışan sayısının 24 olduğunu,
Sondan 2. çalışanın maaşının 106450 olduğunu
40,21 ve 19 yaslarında çalışanlar olup olmadığını
11. Çalışan bilgilerinin

```
{
"id":"11"
"employee_name": "Jena Gaines",
"employee_salary": "90560",
"employee_age": "30",
"profile_image": "" }
```

} gibi olduğunu test edin.

API Testing

GetRequest14:

http://dummy.restapiexample.com/api/v1/employees url ine bir istek gönderildiğinde
Status kodun 200 olduğunu,
En yüksek maaşın 725000 olduğunu,
En küçük yaşın 19 olduğunu,
İkinci en yüksek maaşın 675000
olduğunu test edin.

API Testing

PostRequest01:

http://dummy.restapiexample.com/api/v1/create url ine, Request Body olarak

```
{
            "name":"Ahmet Aksoy",
        "salary":"1000",
        "age":"18",
        "profile_image": ""
}
```

gönderildiğinde, Status kodun 200 olduğunu ve dönen response body nin ,

```
{
    "status": "success",
        "data": {
            “id”:…
    },
    "message": "Successfully! Record has been added."
}
```

olduğunu test edin

JSONObject

JSONObject'i post yaparken java collectionları kullanmak yerine kullanabiliriz. JSONObject class indan bir obje oluşturarak kullanılır. JSONObjectlerde type belitrmediğimiz için type casting işlemi yapmayız…

JSONObject kullanabilmek için aşağıdaki kütüphaneyi eklemek gerekir.

Request gönderilirken body içerisinde toString metodu kullanılması gerekiyor..

```
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20190722</version>
</dependency>
```

PostRequest02:

https://restful-booker.herokuapp.com/booking url ine, Request Body olarak

```
{              "firstname": "Selim",
               "lastname": "Ak",
               "totalprice": 11111,
               "depositpaid": true,
               "bookingdates": {
                   "checkin": "2020-09-09",
                   "checkout": "2020-09-21"
                }
}gönderildiğinde,  Status kodun 200 olduğunu ve dönen response body nin ,
          "booking": {
                "firstname": " Selim ",
                "lastname": " Ak ",
                "totalprice":  11111,
                "depositpaid": true,
                 "bookingdates": {
                      "checkin": "2020-09-01",
                      "checkout": " 2020-09-21”
                  },
          }
olduğunu test edin
```

PostRequest03:

https://jsonplaceholder.typicode.com/todos URL ine aşağıdaki body gönderildiğinde,

```
    }
    "userId": 55,
    "title": "Tidy your room",
    "completed": false
  }
```

Dönen response un Status kodunun 201 ve response body nin aşağıdaki gibi olduğunu test edin

```
  {
    "userId": 55,
    "title": "Tidy your room",
    "completed": false,
    "id": …
   }
```

PutRequest01:

https://jsonplaceholder.typicode.com/todos/198 URL ine aşağıdaki body gönerdiğimde

```
{
    "userId": 21,
    "title": "Wash the dishes",
    "completed": false
}
```

Dönen response un status kodunun 200 ve body kısmının aşağıdaki gibi olduğunu test edin

```
{
  "userId": 21,
  "title": "Wash the dishes",
  "completed": false,
  "id": 198
}
```

PatchRequest01:

https://jsonplaceholder.typicode.com/todos/198 URL ine aşağıdaki body gönderdiğimde

```
{

    "title": "API calismaliyim"

}
```

Dönen response un status kodunun 200 ve body kısmının aşağıdaki gibi olduğunu test edin

```
{
  "userId": 10,
  "title": "API calismaliyim"
  "completed": true,
  "id": 198
}
```

DeleteRequest01:

http://dummy.restapiexample.com/api/v1/delete/2 bir DELETE request gönderdiğimde

Dönen response un status kodunun 200 ve body kısmının aşağıdaki gibi olduğunu test edin

```
{
  "status": "success",
  "data": "2",
  "message": "Successfully! Record has been deleted"
}
```

POJO Class –Plain Old Java Object

Bir API ‘a POST, PUT ,PATCH request gönderirken, Göndermek istediğimiz Request Body i ya da response dan dönen cevabı(tüm methodlar için), test ederken kullanmak amacıyla önceden oluşturmamız gerekir. Bu dataları farklı yöntemler kullanarak oluşturabiliriz. Bu yöntemlerden biri de POJO classlardır.

Biz POJO’lar yardımı ile oluşturacağımız datalar için kalıplar oluştururuz. Bu kalıpları Java’da encapsulation yöntemi ile oluştururuz. Bu yapıyı şu adımları takip ederek yapabiliriz.

1- json objesindeki herbir key değeri için private türünde bir değişken tanımlanır.
2- Tanımlanan tüm değişkenlerin GETTER/SETTER methodları oluşturulur.
3- Default constructor oluşturulur.
4- Tanımlanan tüm değişkenleri içeren parametreli constructor oluşturulur.
5- Tanımlanan tüm değişkenleri içeren toString methodu oluşturulur.

PostRequestWithPojo01:

https://jsonplaceholder.typicode.com/todos url ‘ine bir request gönderildiğinde
Request body {

```
"userId": 21,
"id": 201,
"title": "Tidy your room",
"completed": false
```

}
Status kodun 201, response body ‘nin ise
{

```
"userId": 21,
"id": 201,
"title": "Tidy your room",
"completed": false
```

} *olduğunu test edin.*

PostRequestWithPojo02:

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

url'ine aşağıdaki request body gönderildiğinde,

```
{
        "firstname": "Selim",
        "lastname": "Ak",
        "totalprice": 15000,
        "depositpaid": true,
        "bookingdates": {
            "checkin": "2020-09-09",
            "checkout": "2020-09-21"
        }
}
```

Status kodun 200 ve dönen response 'un

```
{
    "bookingid": 11,
    "booking": {
        "firstname": "Selim",
        "lastname": "Ak",
        "totalprice": 15000,
        "depositpaid": true,
        "bookingdates": {
            "checkin": "2020-09-09",
            "checkout": "2020-09-21"
        }
    }
}
```

olduğunu test edin

GetRequestWithPojo01:

http://dummy.restapiexample.com/api/v1/employee/1 url 'ine bir get request gönderildiğinde , dönen response 'un,

Status kodunun 200 ve response body'nin


```
{
"status": "success",
"data": {
"id": 1,
"employee_name": "Tiger Nixon",
"employee_salary": 320800,
"employee_age": 61,
"profile_image": ""
},
"message": "Successfully! Record has been fetched."
}
```


Olduğunu test edin

Serialization İşlemi

Java objesini json a çevirme işlemidir

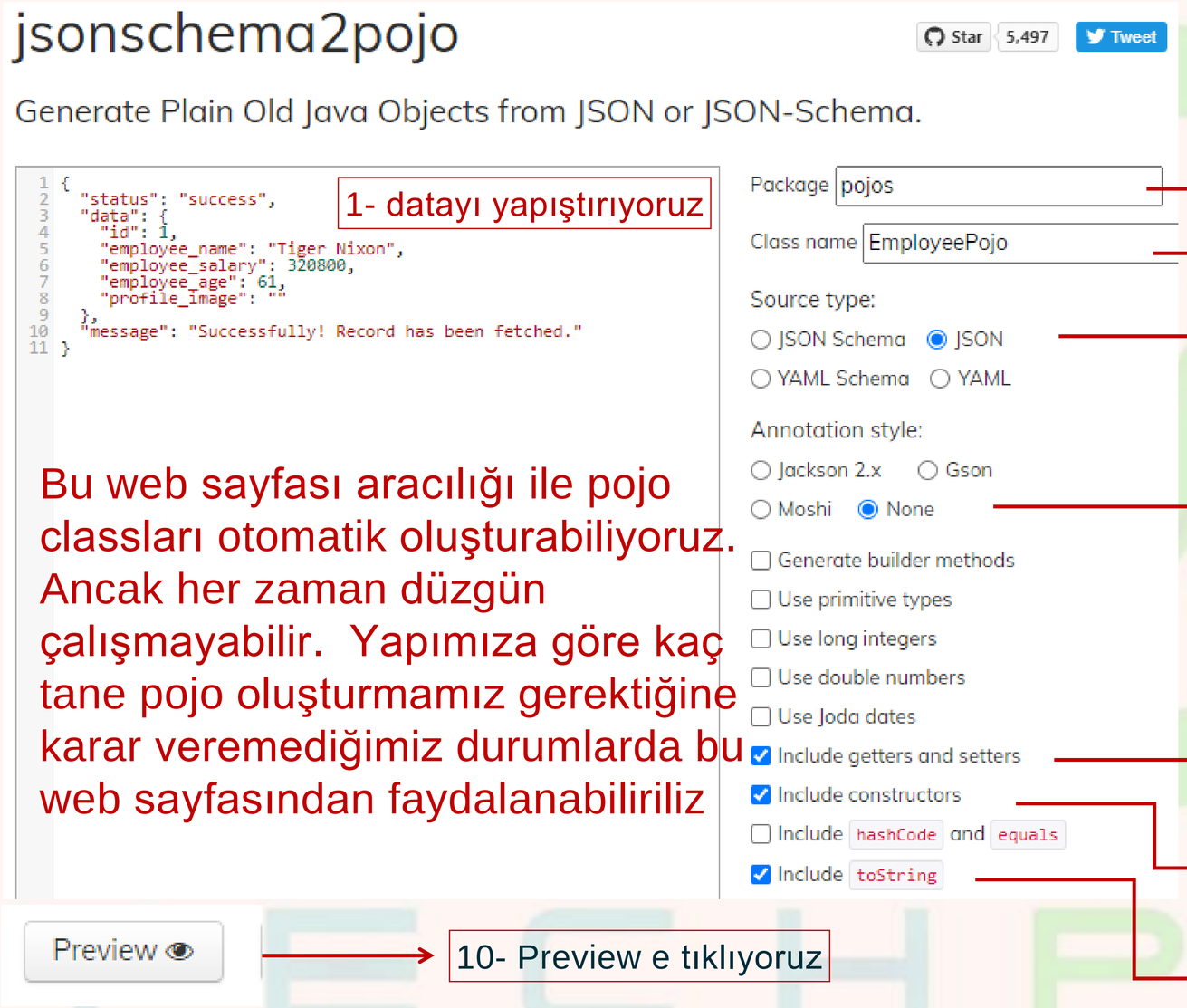
Gson gson=new Gson(); → *Gson classından bir obje üretilir*

→*Oluşturulan obje üzerinden toJson metodu yardımıyla java objesi json formatına dönüştürülür..*

String jsonFromJava=gson.toJson(actualDataMap);

System.*out*.println(jsonFromJava); → Ekranda yazdırma

https://www.jsonschema2pojo.org/



1- datayı yapıştırıyoruz

Bu web sayfası aracılığı ile pojo classları otomatik oluşturabiliyoruz. Ancak her zaman düzgün çalışmayabilir. Yapımıza göre kaç tane pojo oluşturmamız gerektiğine karar veremediğimiz durumlarda bu web sayfasından faydalanabiliriliz

2-Projemizdeki package adını yazıyoruz

3-Projemizdeki class adını yazıyoruz

4- Kaynak data tipini seçiyoruz

6- Getter ve setter methodlar için burayı işaretliyoruz

7- Parametreli ve default constructor oluşturmak için burayı işaretliyoruz

8- toString methodunu sayfamıza dahil etmek için bu seçeneği işaretliyoruz.

10- Preview e tıklıyoruz

https://www.jsonschema2pojo.org/

Preview

Copy to Clipboard

----------pojos.Data.java----------

```java
package pojos;

import javax.annotation.Generated;

@Generated("jsonschema2pojo")
public class Data {

private Integer id;
private String employeeName;
private Integer employeeSalary;
private Integer employeeAge;
private String profileImage;

/**
* No args constructor for use in serialization
*
*/
public Data() {
}

/**
*
* @param employeeName
* @param employeeAge
* @param id
* @param profileImage
* @param employeeSalary
*/
public Data(Integer id, String employeeName, Integer employeeSalary, Integer employeeAge, String profileImage) {
super();
this.id = id;
this.employeeName = employeeName;
this.employeeSalary = employeeSalary;
this.employeeAge = employeeAge;
this.profileImage = profileImage;
}
```

Böylece belirttiğimiz json objesinin POJO classlarını oluşturmuş oluyoruz

Object Mapper

Json datasını java objelerine çevirme işlemidir.
Çevireceği uygun java objesini kendi belirler. De-Serialization yapar.
Object Mapper kullanmak için aşağıdaki kütüphanenin pom.xml'e yüklü olması gerekir.

```
<dependency>
    <groupId>org.codehaus.jackson</groupId>
    <artifactId>jackson-mapper-asl</artifactId>
    <version>1.9.13</version>
</dependency>
```

Object Mapper

Json datalarını Java objelerine çevirmek için utilities altında bir reusable method içeren bir class oluşturabiliriz…

```
public class JsonUtil {
    private  static ObjectMapper mapper;
    static{
        mapper=new ObjectMapper();
    }
    public static <T> T convertJsonToJava(String json,Class<T> cls){

        T javaResult= null;
        try {
            javaResult = mapper.readValue(json, cls);
        } catch (IOException e) {
            System.err.println("json datası javaya dönüştürülemedi”);
        }
        return javaResult;
    }
}
```

GetRequestWithObjectMapper01:

https://jsonplaceholder.typicode.com/todos/198 url'ine bir get request gönderildiğinde,
Dönen response 'un status kodunun 200 ve body kısmının

```
{
    "userId": 10,
    "id": 198,
    "title": "quis eius est sint explicabo",
    "completed": true
}
```

Olduğunu Object Mapper kullanarak test edin

GetRequestWithObjectMapper02:

https://restful-booker.herokuapp.com/booking/2 url'ine bir get request gönderildiğinde, status kodun 200 ve response body'nin

```
{
    "firstname": "Mark",
    "lastname": "Wilson",
    "totalprice": 284,
    "depositpaid": false,
    "bookingdates": {
        "checkin": "2016-08-10",
        "checkout": "2018-06-22"
    }
}
```

Olduğunu Object Mapper kullanarak test edin