

API Nedir ?

Application Programming Interface

Uygulama Programlama Arayüzü

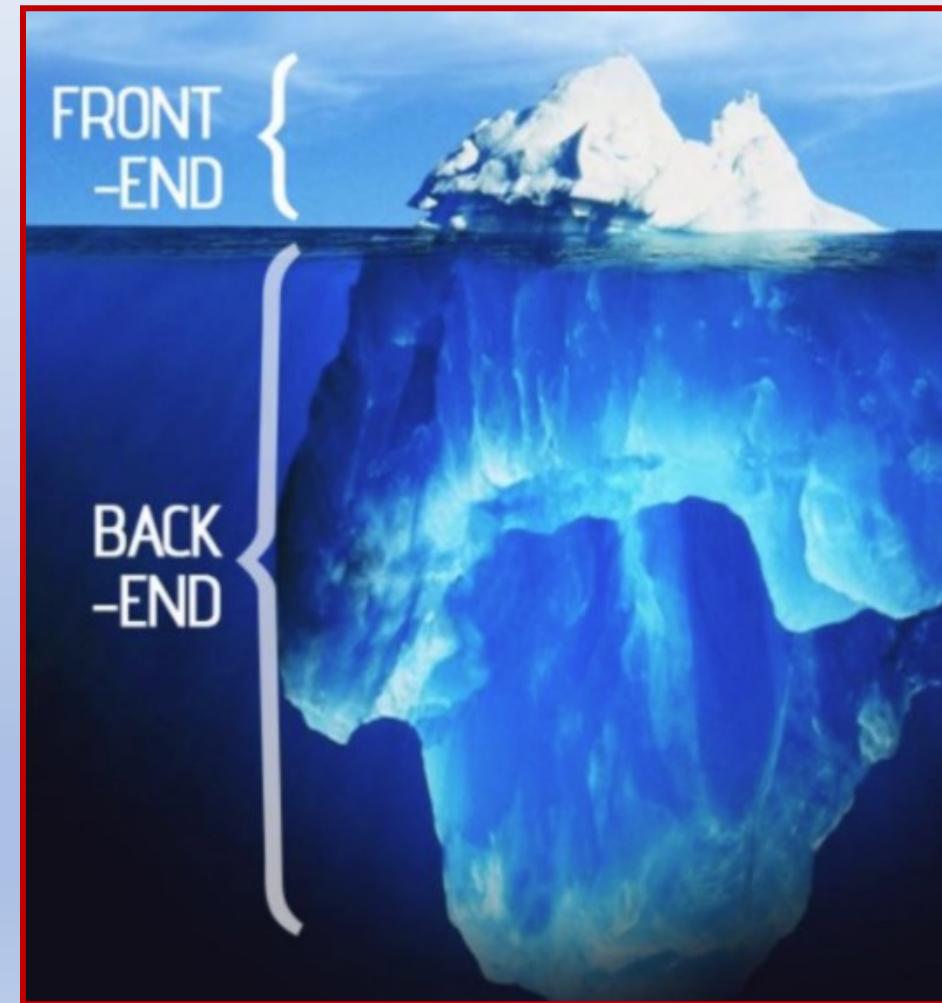
BOLUM 1 – Temel API kavramlari

BOLUM 2 – Postman ile manuel
API sorgulama

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

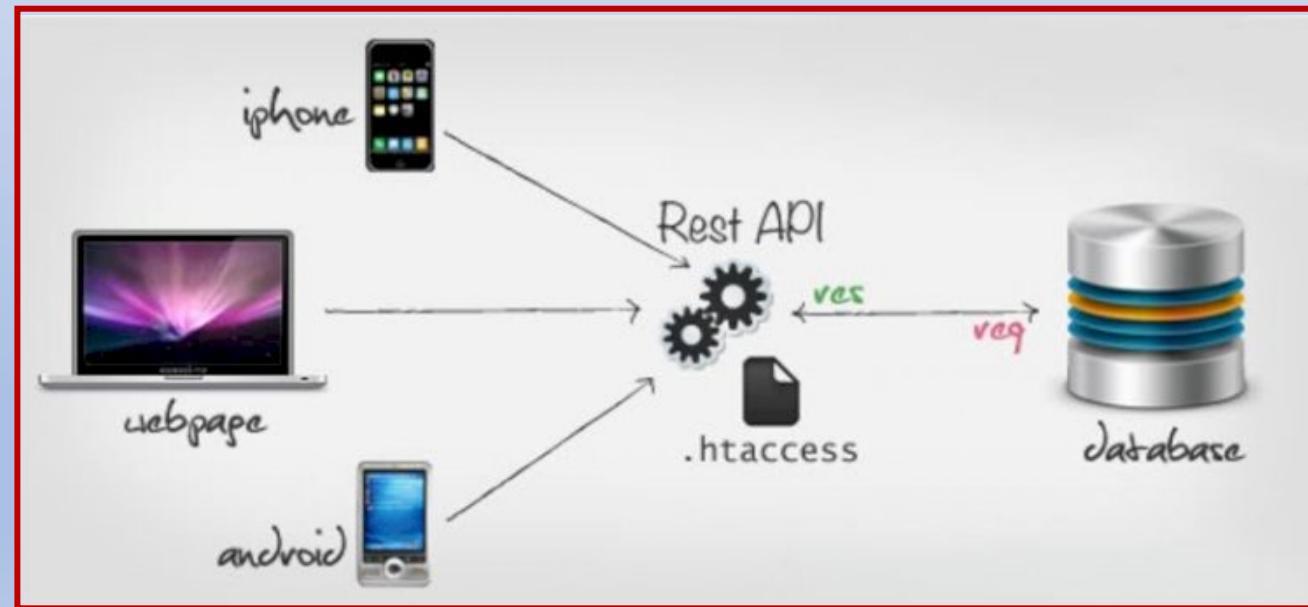
- URI
- URL

11- Swagger dokumani

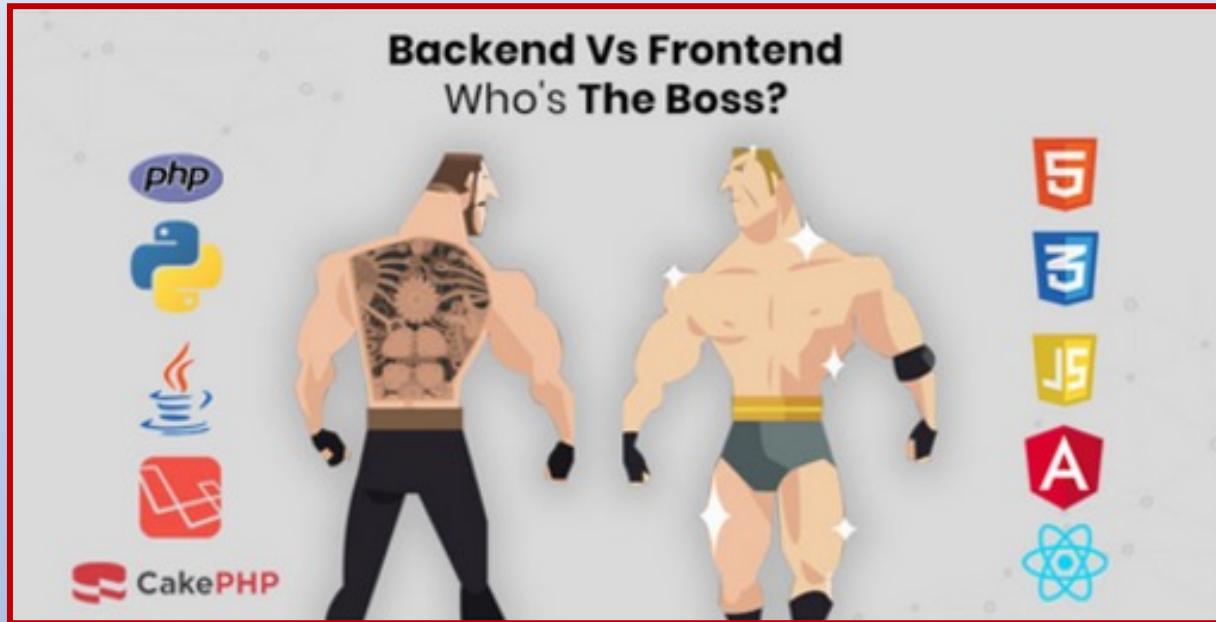
- Nasıl kullanılır
- Neler Yapılabilir
- Token kullanımı

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(Istek)'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 ?



Client

1- Yemek siparisi

Request to API



Garson
API

4- Yemeklerin servis edilmesi

Response to client



Server

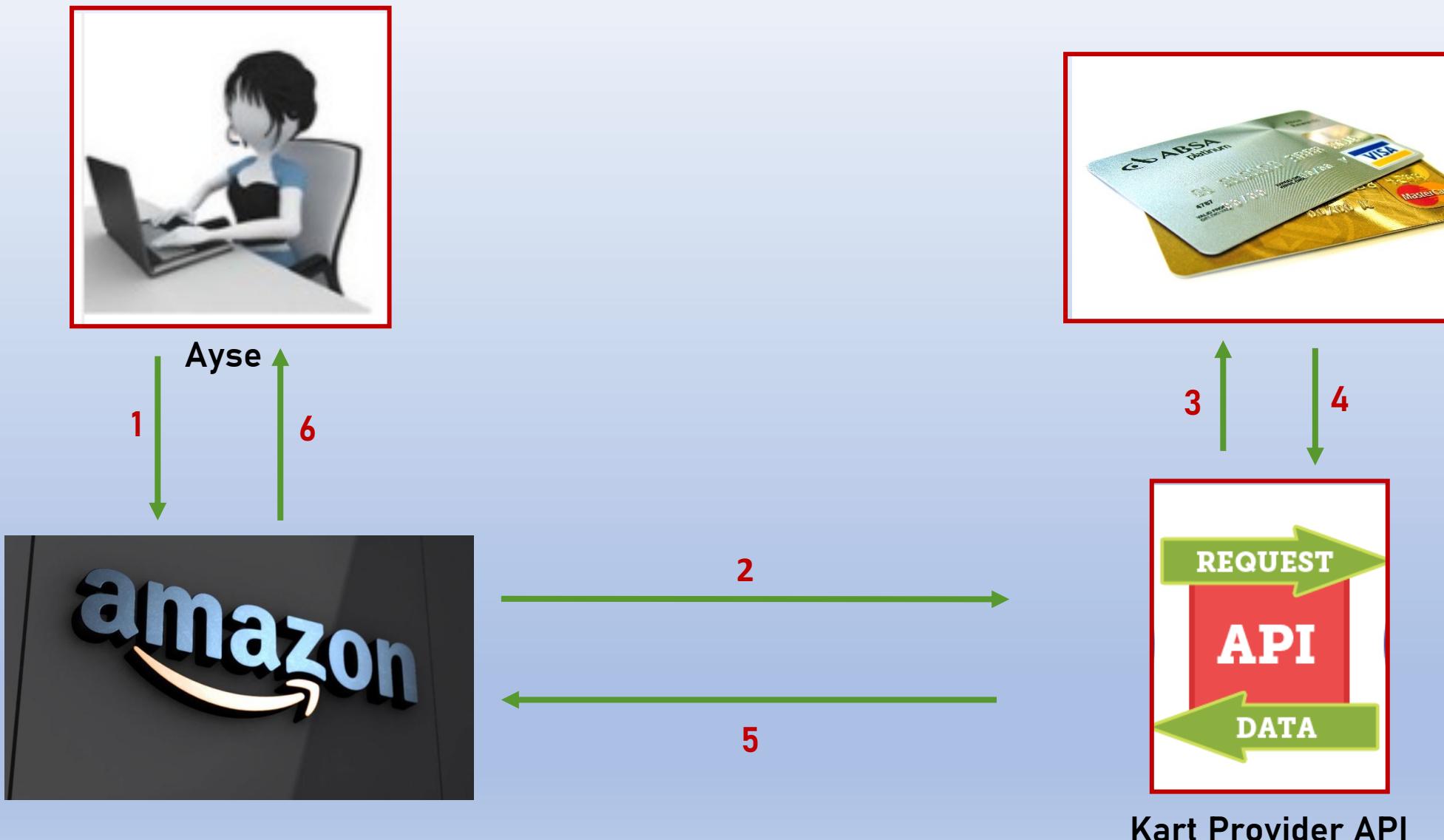
2- Siparisin mutfaga
iletilmesi

Server'da sorgulama

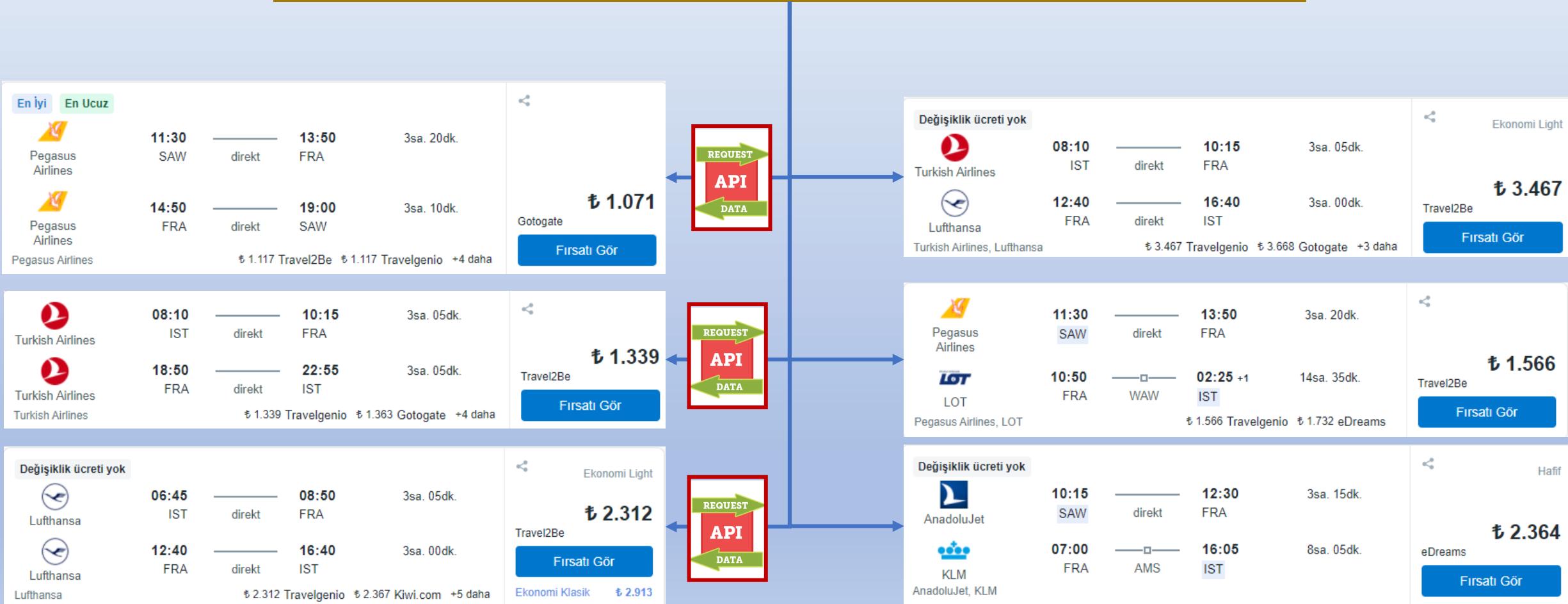
3- Siparisin garsona teslimi

Server'dan sorğu
sonucunu alma

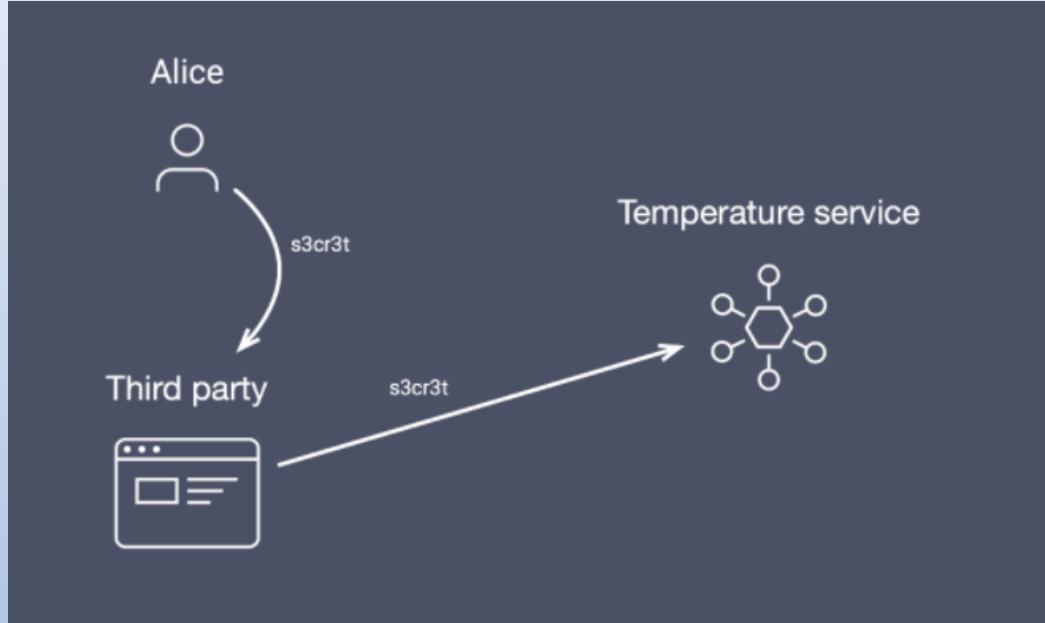
API Nasil Calisir ?



API Nasıl Çalışır ?



API Nasil Calisir ?



Alice'in evinin mevcut iç ortam sıcaklığını ayarlayabilecegi bir iklimlendirme sistemi var. Bu sistem farkli odalara soguk/sicak hava verme ozelligine sahip.

Alice ayrıca, oda sicakliklarinin istedigi gun ve saatlerde istedigi seviyede tutulmasi icin third party bir cihaz kullanmaktadır.

Bu cihaz odalardaki isi bilgisi ile Alice'in istedigi sicaklik bilgilerini karsilastirarak iklimlendirme sisteminin odalarda farkli zamanlarda acip kapanmasini saglamaktadir.

Bu cihazin farkli odalardan gelen sicaklik bilgilerini birbirinden ayirt edebilmesi ve iklimlendirme sisteminin farkli odalardaki vanalarini acip kapatabilmesi gereklidir.

API'da uretebilecegimiz bir key ile verileri ozellestirebilir ve istenirse erisim kontrolu yapilabilir.

API Nasıl Çalışır ?



Kaydol

Hızlı ve kolaydır.

Adın Soyadın

Cep telefonu numarası veya e-posta

Yeni şifre

Doğum Tarihi 27 Kas 2022

Cinsiyet Kadın Erkek Özel

Hizmetimizi kullanan kişiler senin iletişim bilgilerini Facebook'a yüklemiş olabilir. [Daha fazla bilgi al.](#)

Kaydol düğmesine tıklayarak, Koşullarımı kabul etmiş olursun. Gizlilik İlkemizde verilerin nasıl topladığımız, kullandığımız ve paylaştığımız hakkında ve Çerezler İlkemizde cerezleri ve benzer teknolojileri nasıl kullandığımız hakkında bilgi alabilirsin. Bizden SMS bildirimleri alabilirsin ve bunu istediğiniz zaman durdurabilirsin.

Kaydol

UI

Bir uygulamanın frontend ve backend'i arasında data alışverişi için de API kullanılır.

Kullanıcıdan alınan tüm bilgiler, kullanım sırasındaki sorgulamalar, görüntüleme talepleri



Kayıt tamamlandı veya kayıt yapılamadı bilgisi, kullanıcının görüntülemek istediği bilgiler...



**Backend
(database)**

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 protokoller SOA ve Rest

API Nasıl Kullanılır ?

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

Kategoriler	Mesaj
Tümü	Oyun
Ücretsiz API'ler	Resim Boyutlandırma
AI Yapayzeka	Seyahat
Akaryakıt	Sinema
Corona	Spor
E-Ticaret	Sözlük
Eczane	Website
Ekonomi	Whatsapp
Haberler	Yaşam
Hal Fiyatları	Çeviri
Harita	
Hava Durumu	
IP Adres	
Instagram	
Kitap	

Whatsapp Business API
Official WhatsApp Business API. WhatsApp'ın resmi API'sidir.

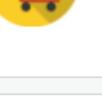
Whatsapp Business Sandbox
WhatsApp Business API'yi test edebilmeniz için sandık gibi bir ortam sunar.

COVID-19 Koronavirüs İstatistik API
Dünya genelindeki COVID-19 virüsüne etkilerini DÜ olarak getiren servis.

Instagram DM API
Unofficial Instagram DM API kullanarak HTTP istekle servis.

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	 IMDb API IMDb sitesinde isimle yada id ile arama yaparak film
 Haberler API 7 farklı ülkeden, ülkelerin anadillerine göre getirilen haberleri bulabileceğiniz API'ler. İsterseniz RSS linki	 Araç Tanıma API Resimlerdeki araçların sayısını, marka, model, renk bilgisi
 Hava Durumu API Dünyanın her yerindeki hava durumunu günlük	 Plaka Tanıma API Resimdeki araçların plakalarını yakalayın.
 Faiz Oranları API Bankaların genel faiz oranları ve farklı kredi türlerinin bilgileri de ulaşabilirsiniz.	 Nesne Tanıma API Resimde neler olduğunu, hangi nesnelerin bulunduğu
 Duygu Analiz API Resmin ya da yazının içерdiği duyguları derecelendirme	 E-Ticaret Ürün Öneri API Kullanıcının bulunduğu e-ticaret sitesinde bakmış olduğu
 Namaz Vakitleri API İstediğiniz şehrin namaz vakitlerini getiren, se	 Çiplaklılık Algılama API Resimlerin çiplaklık içerip içermediğini öğrenin.

API Nasıl Kullanılır ?

END POINTS

▼ GET /dutyPharmacy

Şehirlerdeki gün nöbetçi olan eczaneleri getiren servis.

PARAMETRELER

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

Shell Go Node Javascript Java Python Ruby Csharp Swift Ocaml

```
curl --request GET \
--url 'https://api.collectapi.com/health/dutyPharmacy?ilce=%C3%87ankaya&il=Ankara' \
--header 'authorization: apikey 700IIqcDndqPfwSEQBrke7:63xbVXUVHLL0CeD3nubi40' \
--header 'content-type: application/json'
```

1- Kullanılacak End Point

2- Kullanılacak HTTP metodu ve parametreler

3- Zorunlu alanlar

4- Ornek request ve varsa authorization turu ve apikey



Nöbetçi Eczane API

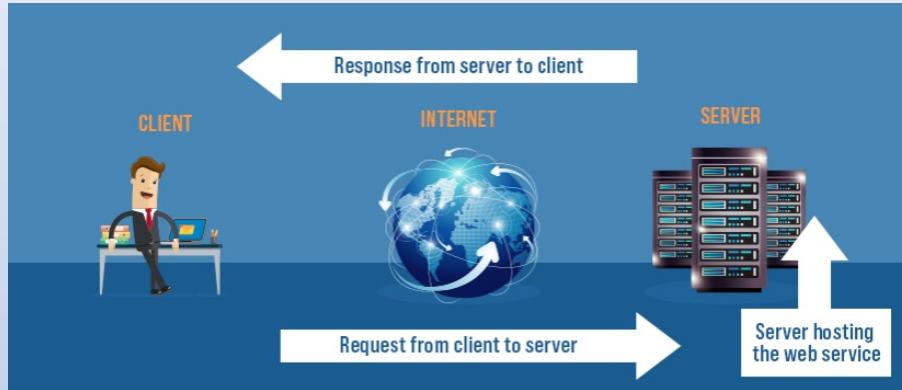
Response

```
{
  "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"
    },
    ...
  ]
}
```

Gonderecegimiz request'e karsilik gelecek Response(cevap)

API vs Web Services

Ikisi de uygulamalar arasında iletisim saglar. Aralarindaki birtek fark vardır.



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.

Http Nedir ?

HTTP : Hyper Text Transfer Protocol

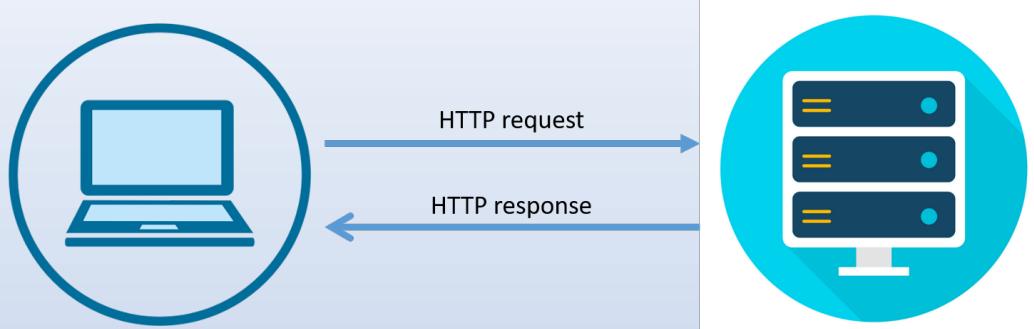
HTTP protokolü istemci (PC) ile sunucu (server) arasındaki alışveriş kurallarını belirler.



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



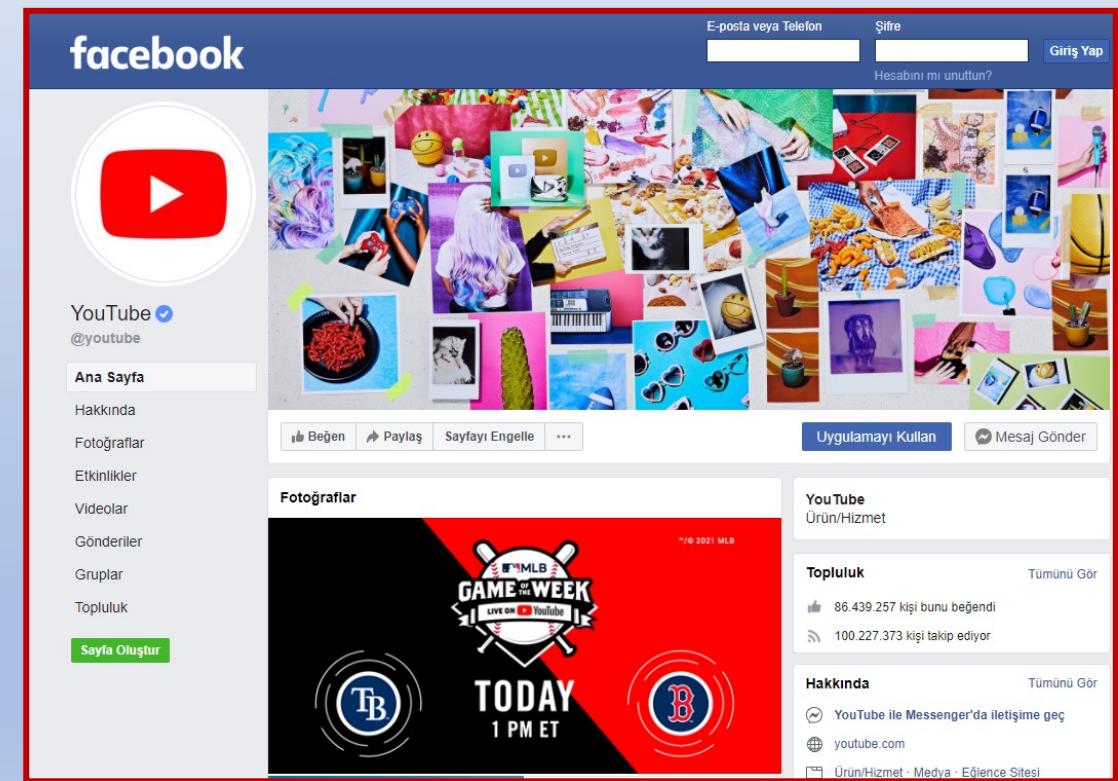
Client

Server

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

Request

UI



Response

Request & Response API

http://graph.facebook.com/youtube

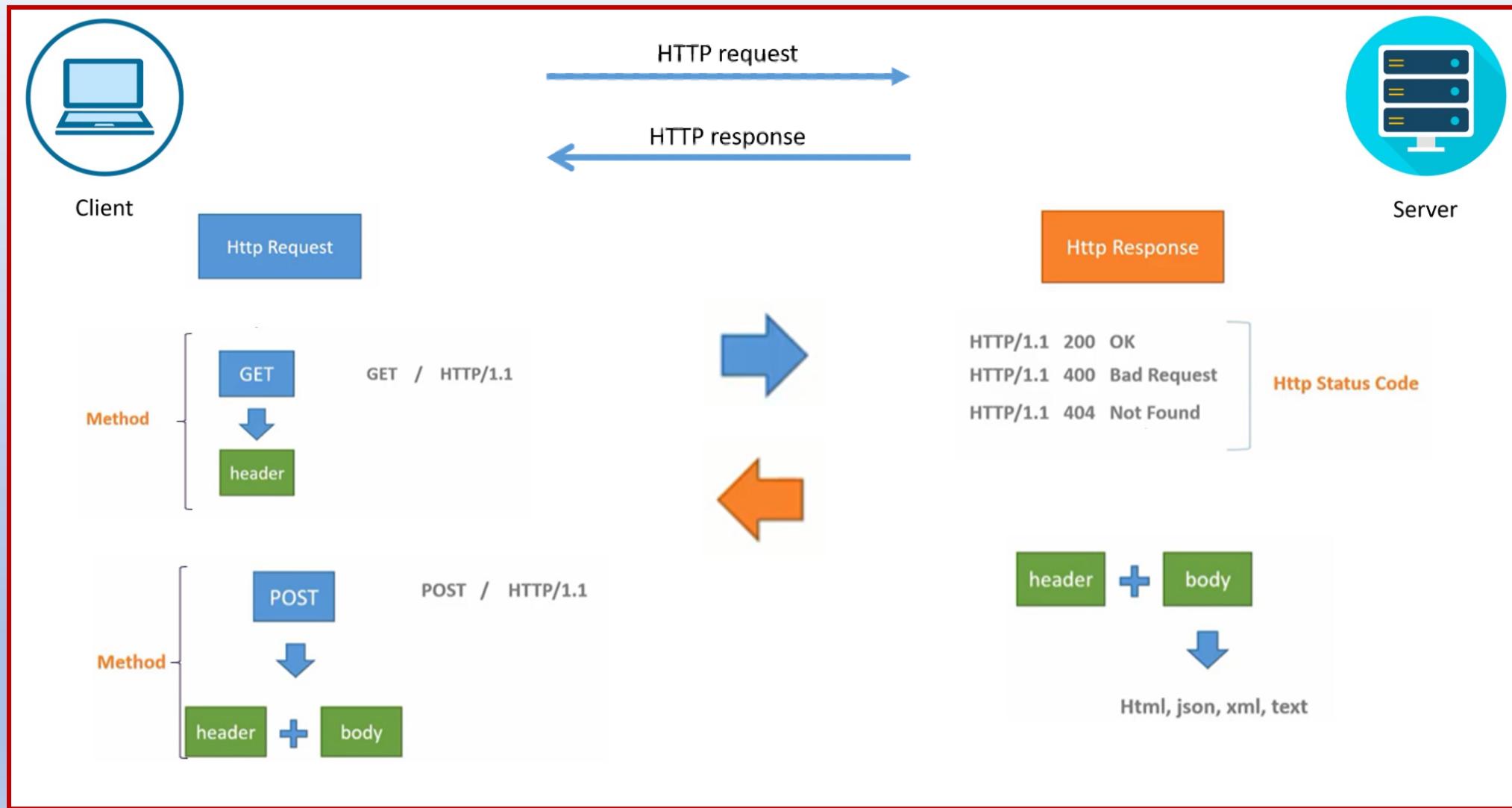
Request

API

The diagram illustrates a request to the Facebook Graph API. On the left, a blue bracket labeled "Request" encloses the URL "http://graph.facebook.com/youtube". A red arrow labeled "API" points from this request to a red-bordered box on the right, which contains the JSON response.

```
{  
  "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 uygulaması gibi API'da da bir guvenlik kontrolu vardır.

Ozel kullanici adi ve sifre istenen alanlara girmek icin TOKEN almanız gereklidir

Request & Response API

Sonuc olarak

API'da işlem yapmak için uygulamalar arasındaki iletişim saglayacak kurallar net olarak belirlenmiş olmalıdır

Request

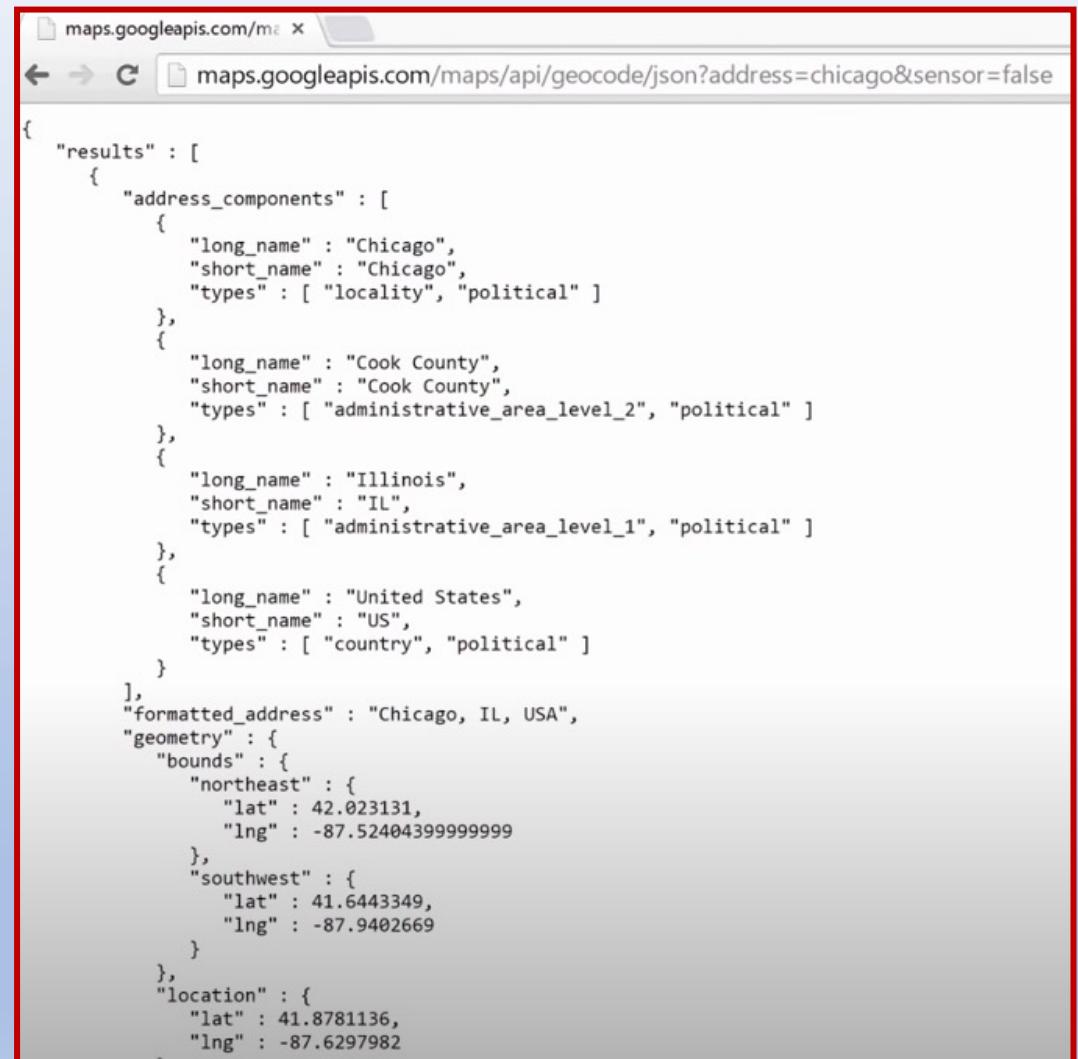
Bir uygulamadan herhangi bir bilgi istenecekse bu bilgilere ulaşabilmek için istek sahiplerinin gondermesi gereken request net olarak belirlenmelidir

- 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

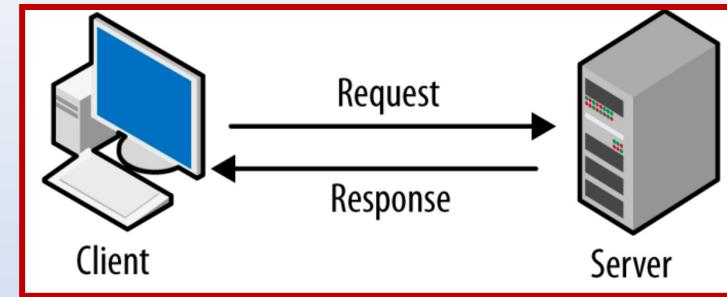


The screenshot shows a browser window with the URL `maps.googleapis.com/maps/api/geocode/json?address=chicago&sensor=false`. The page displays a JSON object representing the geocoding results for the address "Chicago". The JSON structure is as follows:

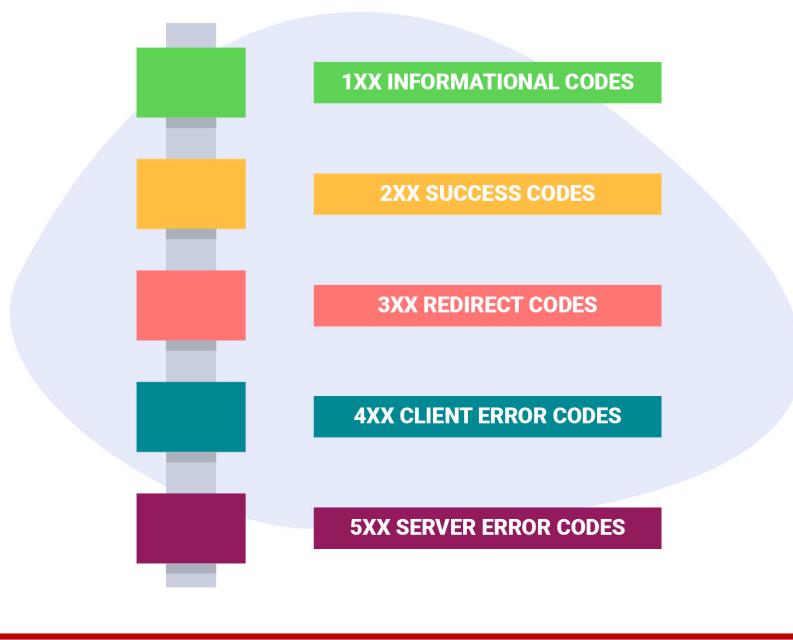
```
{  
  "results": [  
    {  
      "address_components": [  
        {  
          "long_name": "Chicago",  
          "short_name": "Chicago",  
          "types": ["locality", "political"]  
        },  
        {  
          "long_name": "Cook County",  
          "short_name": "Cook County",  
          "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 verdigi yanitta, yanitin durumunu belirten bir sayisal kod da gönderir.

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

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 durum kodlari
- 7. HTTP request ve Response
- 8. HTTP metotlari
 - GET
 - PUT
 - POST
 - PATCH
 - DELETE

- 9. Endpoints
 - URI
 - URL
- 10. Swagger dokumani
 - Nasil kullanilir
 - Neler Yapilabilir
- 11. API protokolleri
 - SOAP
 - Rest

Request & Response API

Sonuc olarak

API'da işlem yapmak için uygulamalar arasındaki iletişim saglayacak kurallar net olarak belirlenmiş olmalıdır

Request

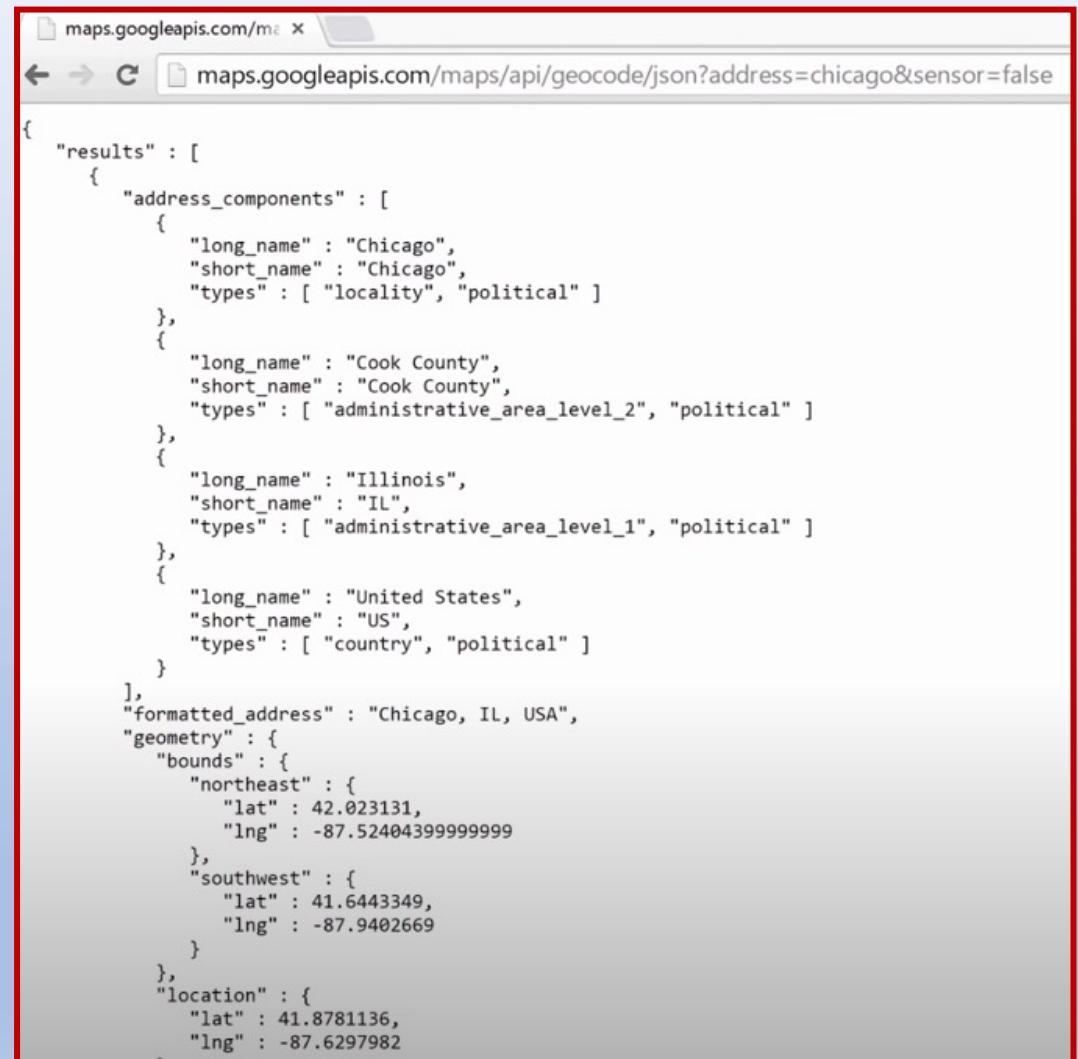
Bir uygulamadan herhangi bir bilgi istenecekse bu bilgilere ulaşabilmek için istek sahiplerinin gondermesi gereken request net olarak belirlenmelidir

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

Response

Yapılan request'e verilen cevap da net ve anlaşılır olmalıdır

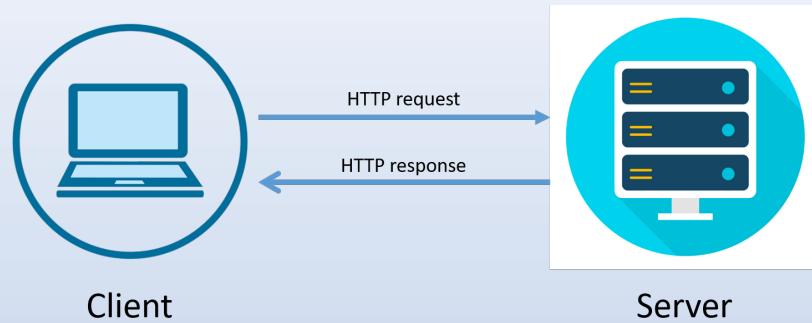
- 1- Status Code
- 2- Body



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 is as follows:

```
{  
  "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  
        }  
      }  
    }  
  ]  
}
```

Request / Response Body



Client

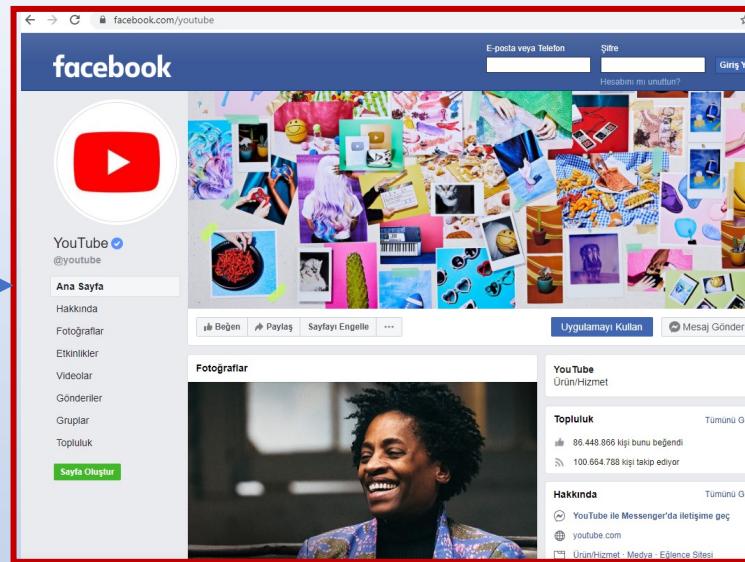
Server

facebook.com/youtube

Request

UI

Html



Response
(Kullanıcının
Gordüğü)

```
<!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

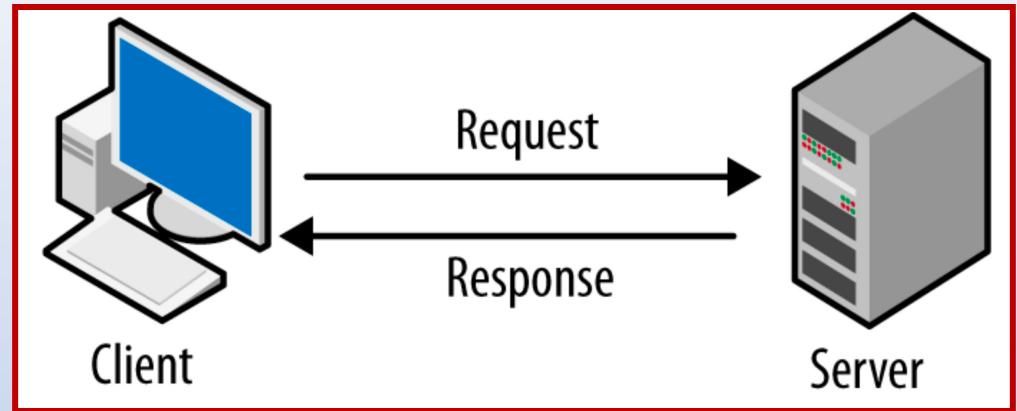
Bir sunucuya request gonderildiginde istegimizi sunucuya dogru sekilde anlatabilmemiz ve donen Response'u dogru sekilde anlamlandırmamız icin kullanılan 4 temel data formatı vardır.

- 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 arasında data alisverisi yapmak (Request/Response) icin en çok tercih edilen formattir.

JSON formatindaki bir data icin uc temel bolum vardir.

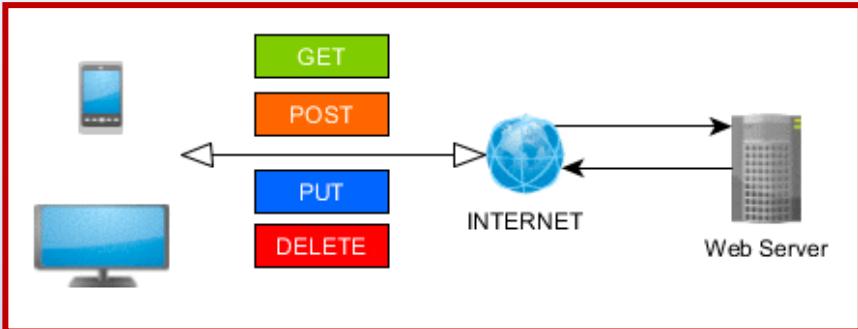
1- **Suslu parantezler** : JSON formatindaki bir datanin nerede baslayip, nerede bittigini gosterir. Ihitiyac oldugunda NESTED (ic ice) JSON datalari olusturulabilir

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

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

Keys ve **Values** arasında : kullanilir.

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 onemlidir.

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

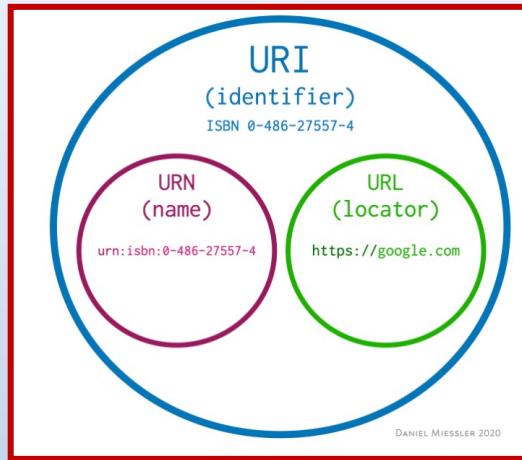
POST : Yeni bir nesne oluşturmak için kullanılır. Her seferinde yeni bir nesne oluşturur. Yeni bir nesne oluşturacağımız için nesne için gerekli tüm bilgileri Request'e yazmamız gereklidir

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

PATCH : Var olan nesnenin istediğimiz belirli bilgilerini değiştirmek için kullanılır.

DELETE : Adresi verilen nesneyi silmek için kullanılır. Sileceğimiz nesneyi Request'te net olarak belirtmemiz gereklidir.

ENDPOINTS



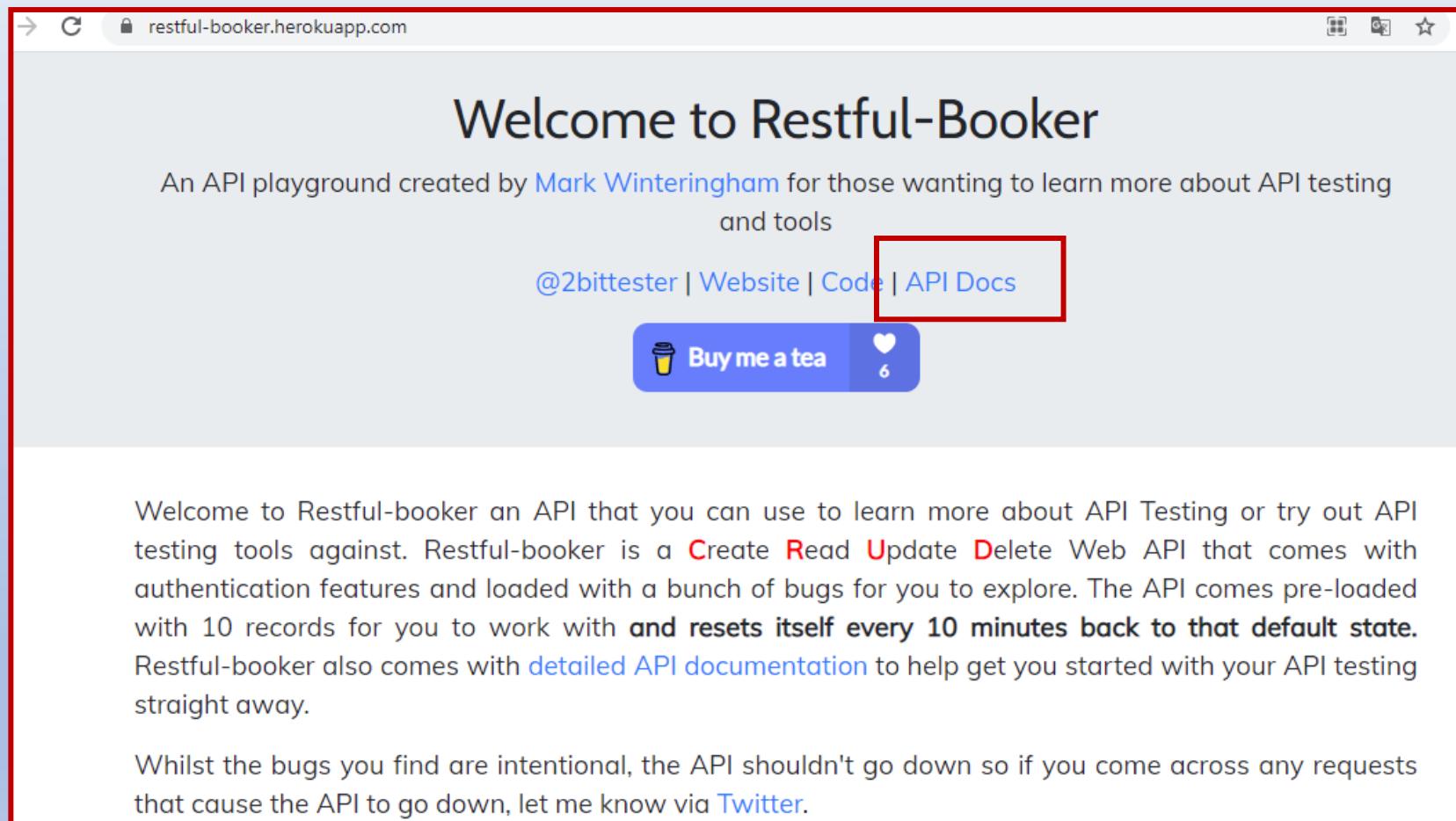
Endpoint kaynaga nasıl erişebileceğimizi gösteren URI (Uniform Resource Identifiers)'lara denir. Bir API oluşturduğunuzda kullanıcıların ulaşabilmesi için bu endpoint'i ve kullanılabilecek Http method'larını kullanıcılarla bildirmemiz gereklidir.

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

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

ENDPOINTS

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



ENDPOINTS

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

The screenshot shows a section of an API documentation page for a POST endpoint. The endpoint URL is <https://restful-booker.herokuapp.com/auth>. The response body contains a curl command to demonstrate how to use the endpoint.

Header

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

Request body

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

Success 200

Alan	Tip	Açıklama
token	String	Token to use in future requests

1 Yapılacak işlem ve kullanılacak HTTP metot

2 Kullanılacak Endpoint

3 Gondereceğimiz datanın formatı (Json)

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

5 Gondereceğimiz 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'
```

API

API Testing

DERS 4

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 durum kodlari
- 7. HTTP request ve Response
- 8. HTTP metotlari
 - GET
 - PUT
 - POST
 - PATCH
 - DELETE

9. Endpoints

- URI
- URL

10. Swagger dokumani

- Nasil kullanilir
- Neler Yapilabilir

11. API protokoller

- SOAP
- Rest

Swagger DOCUMENTS

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



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

The screenshot shows the Swagger API documentation for the "pet" resource. The title is "pet Everything about your Pets". Below it are four 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. The 'pet' resource is described as 'Everything about your Pets'. Below it, a list of operations is provided:

- 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 (highlighted with a red box)
- 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- Kullanılabilecek Http Method'lari

3- Her bir method için yazılması gereklili olan parametreler ve method'un işlev açıklaması
<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.

The screenshot shows the Swagger UI interface for the `/pet/findByStatus` endpoint. At the top, there's a blue button labeled "GET" and the endpoint URL `/pet/findByStatus` followed by the description "Finds Pets by status". To the right of the URL is a lock icon. Below the URL, a note says "Multiple status values can be provided with comma separated strings". A "Parameters" section follows, containing a table with two columns: "Name" and "Description". The first parameter is "status * required", described as "array[string] (query)" with the note "Status values that need to be considered for filter". Below this, it says "Available values : available, pending, sold" and shows a dropdown menu with three options: "available", "pending", and "sold". To the right of the parameters is a "Try it out" button, which is also highlighted with a red box. The entire screenshot is framed by a thick red border.

1- Kullanilabilecek parametreler

2- Swagger dokumanindan ornek API sorgusu yapmak icin

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": 66450788, "name": "Hilaria"}]
```

Download

Response headers

Swagger DOCUMENTS

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

1- Durum kodu (Status code)

2- Response body

Swagger DOCUMENTS

DELETE /pet/{petId} Deletes a pet

Parameters

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

api_key: api_key

petId: 14859388

Execute Clear

Responses

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/>

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.

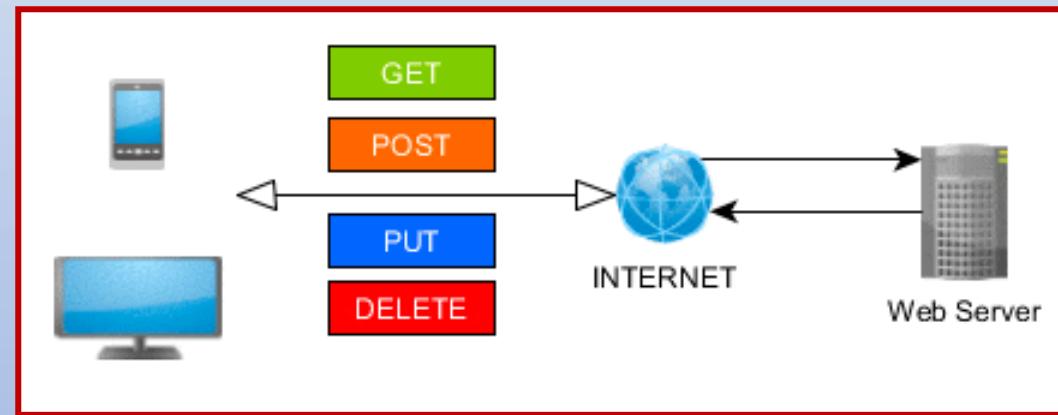
```
<customer>
    <customer_id> 1001 </customer_id>
    <customer_name> Mark Star </customer_name>
</customer>
```

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 metodу çağıracak URI'ler ile web servise HTTP protokölyüyle 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.
- İki de HTTPS destekler, SOAP için WS-SECURITY adlı bir eklienti mevcuttur.
- Güvenlik açısından SOAP daha gelişmiştir çünkü hazır yapılar bulunmaktadır.
- Dokümantasyon bakımından SOAP daha gelişmiştir ve daha fazla kaynak bulunmaktadır.

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 içeren 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.



DERS 5

BOLUM 2

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

- 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA**

- 3- FARKLI ENDPOINTLER KULLANMA**



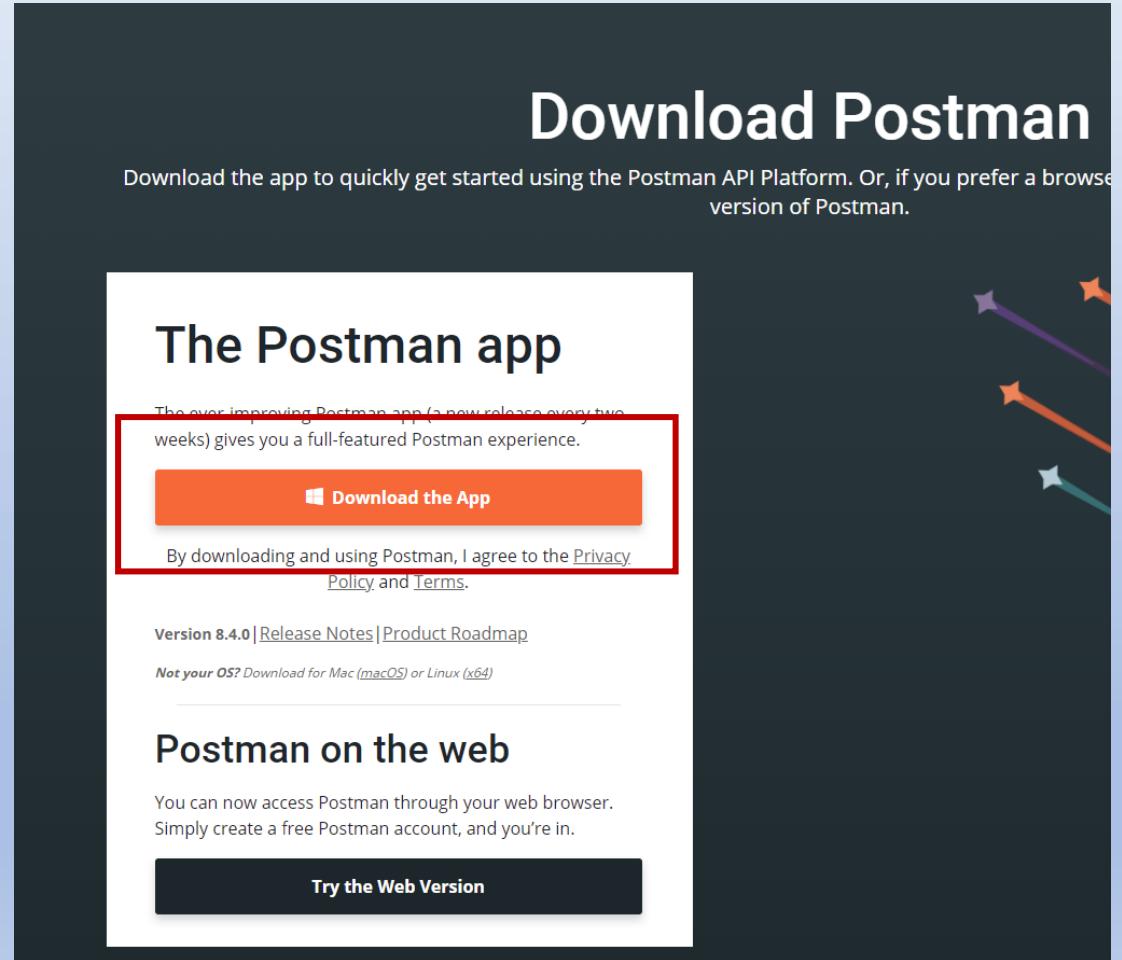
<https://www.postman.com/downloads/>

Postman nedir ?

Postman API sorusu 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 kolaylaştıracaktır.

Postman ile yaptığınız sorguları oluşturacağınız 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" and provides instructions to download the app or use the browser version. Below that, a large section is titled "The Postman app" with a sub-section about the improvements in the app. It features a prominent orange "Download the App" button. A note below the button states: "By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#)". At the bottom of this section, there are links for "Version 8.4.0 | Release Notes | Product Roadmap" and a note for users not using Mac or Linux. Another section titled "Postman on the web" describes how users can access Postman through their web browser by creating a free account. It includes a "Try the Web Version" button.



COLLECTION OLUSTURMA

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

- 1**: A red box highlights the "Collections" icon in the left sidebar.
- 2**: A red box highlights the "New" button in the top right corner of the main workspace.
- 3**: A red box highlights the "Collections" section in the main workspace, which displays a message: "You don't have any collections". Below it, there's a note: "Collections let you group related requests, making them easier to access and run." and a "Create Collection" button.

1- Collection sekmesini secelim

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

The screenshot shows the "Collections" section with a "New Collection" button highlighted by a red box. A red box also highlights the "3" icon next to the "New Collection" button.

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

The screenshot shows the "Collections" section with a "New Collection" entry renamed to "4" (highlighted by a red box). The "4" entry is also highlighted by a red box. The interface includes tabs for "Overview", "Authorization", "Pre-request Script", "Tests", and "Variables".

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



A screenshot of the Postman application interface. On the left, there's a sidebar with icons for Collections (highlighted with a red box), APIs, Environments, Mock Servers, and Monitors. Below the sidebar, a message says "You don't have any collections". A "Create Collection" button is visible. The main area shows a "Collections" section with a plus sign and a "New" button. A "Import" button is also present. In the center, there's a search bar and a "GET Untitled Request" card. The card has tabs for "Overview", "Params", "Authorization", "Headers (6)", "Body", "Pre-request Script", "Tests", and "Settings". The "Headers" tab is selected. Below the card, there's a table for "Query Params" with columns for KEY, VALUE, and DESCRIPTION. At the bottom right of the card, there are "Save" and "Send" buttons. The "Send" button is highlighted with a red box. The entire interface is framed by a large red border. Red numbers from 1 to 9 are overlaid on the interface to indicate specific features:

- 1: New Collection
- 2: History
- 3: HTTP Method (GET)
- 4: Request URL input field
- 5: Cookies tab
- 6: Request Name (Untitled Request)
- 7: Save button
- 8: Request Headers tab
- 9: Query Params table

- 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 sekemeleri, yeni bir request icin + kullanilabilir
- 9- Request icin detay ekleyebilecegimiz sekmeler



Untitled Request

GET Enter request URL

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

Query Params

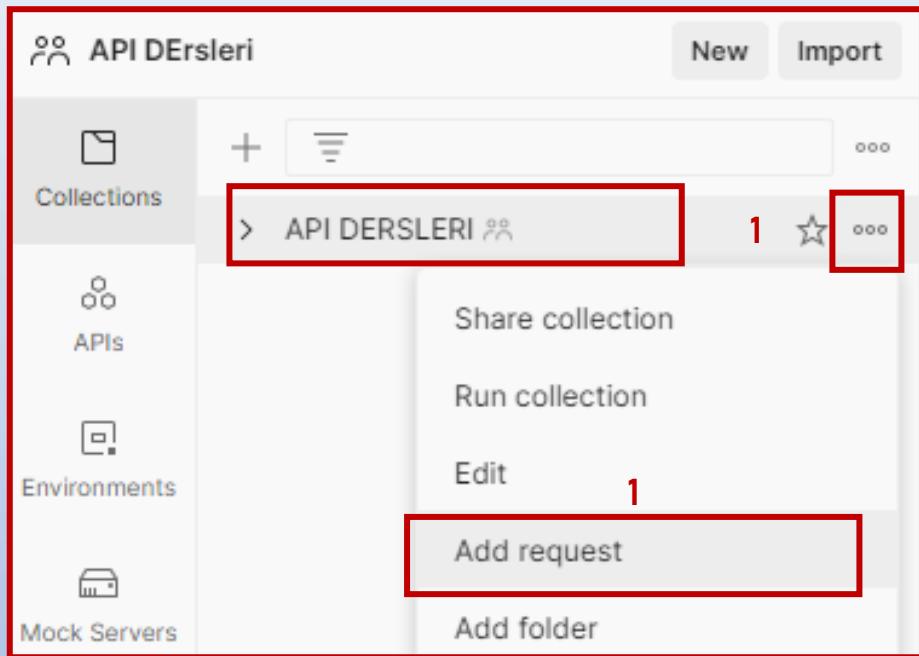
	KEY	VALUE
	Key	Value

A screenshot of the Postman interface showing the 'Params' tab selected. It displays a table for 'Query Params' with one row: 'Key' under 'KEY' and 'Value' under 'VALUE'. Other tabs shown include 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'.

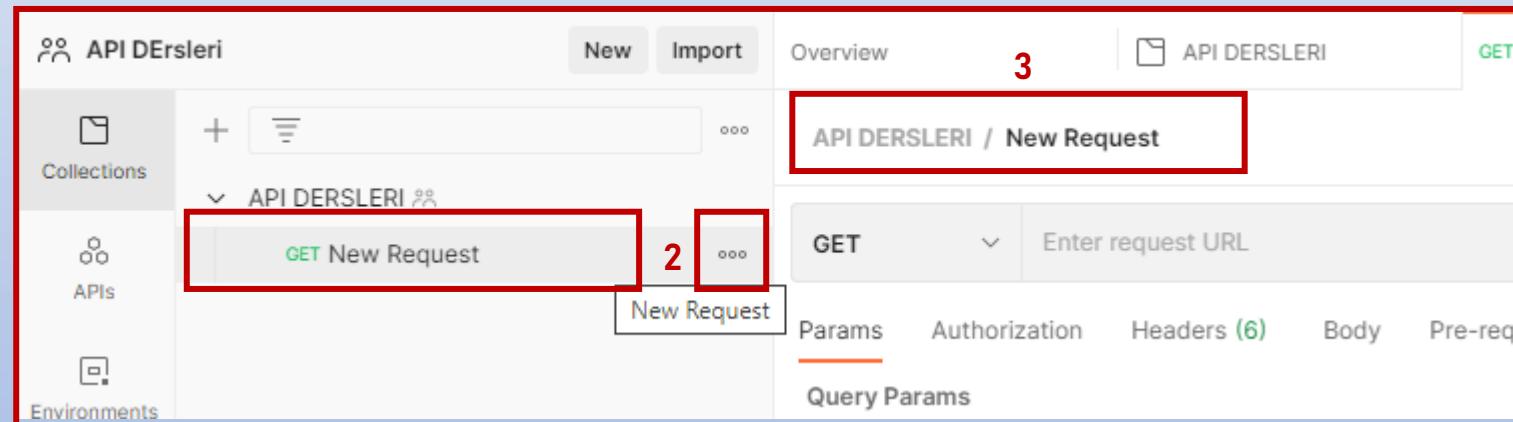
- 1- **Params:** gerekli parametreleri key, value olarak gönderilebilir.
- 2- **Authorization :** API'lere erişmek için yetkilendirme işlemleri için kullanılır, bir kullanıcı adı / şifre ya da bearer token vs bu kısımdan set edilir.
- 3- **Headers :** ihtiyacınıza göre header parametreleri buradan gönderilir, content type JSON gibi.
- 4- **Body :** Bir istekteki ayrıntıların özelleştirebildiği 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 bölüm API'yi test edeceğimiz test scriptlerinin yazıldığı kısımdır.



REQUEST OLUSTURMA



- 1- Mouse ile **API DERSLERI** olarak isimlendirdigimiz collection'in uzerine gelelim, cikan ... 'ya basalim ve acilan DROPDOWN 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'i yeniden yasmak yerine istersek bunu bir **VARIABLE** olarak tanimlayabilir ve ihtiyac duyduğumuzda kolayca kullanabiliriz.

VARIABLE OLUSTURMA



The screenshot shows the Postman interface with a red box highlighting the 'Variables' section. Inside this box, four numbered steps are indicated:

- 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

The 'Variables' section displays a table with one row:

VARIABLE	INITIAL VALUE	CURRENT VALUE
<input checked="" type="checkbox"/> herokuapBaseUrl	https://restful-booker.herokuapp.com	https://restful-booker.herokuapp.com

- 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

The screenshot shows the Postman interface with a red box highlighting the 'Params' section of a GET request. The 'Params' section contains two variables: '\$randomHexColor' and 'herokuapBaseUrl'. The variable '\$randomHexColor' is highlighted with a blue box and labeled 'G'. The variable 'herokuapBaseUrl' is highlighted with an orange box and labeled 'C'. The 'Params' section also includes columns for 'KEY' and 'Key'.

- 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 numbered steps:

1. Collections panel: A collection named "Herokuapp" is selected.
2. Request list: A request titled "Herokuapp / Tum Rezervasyonlari listeleme" is listed.
3. Request details: The request method is "GET" and the URL is "[{HerokuappBaseUrl}]/booking ...".
4. Request settings: The "Authorization" tab is selected under "Params".
5. Save button: The "Save" button is highlighted.
6. Send button: The "Send" button is highlighted.
7. Response body: The response body is displayed as JSON, showing a list of bookings with booking 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 sorğu oluşturalım
- 2- Request ismini degistirelim
- 3- HTTP metodunu secelim
- 4- EndPoint'i yazalim
- 5- Save butonu ile yaptığımız request'i kaydedelim.
- 6- Send butonu ile request'i çağıralım.
- 7- Response body'sinin geldigini kontrol edelim



POSTMAN

DERS 6

BOLUM 2

- 1- POSTMAN GENEL TANITIMI, VARIABLE OLUSTURMA,
GET VE POST REQUEST OLUSTURMA**
- 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA**
- 3- FARKLI ENDPOINTLER KULLANMA**



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 gonderelim.

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.



PUT REQUEST OLUSTURMA

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

- 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
- 8 - Save butonu ile yaptigimiz request'i kaydededelim.
- 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:

- 1: Collection dropdown, 'Herokuapp' selected.
- 2: Request URL: {{(HerokuappBaseUrl)}}/booking/1
- 3: Method: PUT
- 4: Headers: Content-Type: application/json
- 5: Body tab selected.
- 6: Raw tab selected.
- 7: Request body JSON:

```
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: Save button.
- 9: Send button.
- 10: Response body: "1 Forbidden"
- 11: Status: 403 Forbidden

- 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
- 8- Save butonu ile yaptigimiz request'i kaydededelim.
- 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



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 navigation is a sidebar with sections for 'Collections', 'APIs', 'Environments', 'Mock Servers', 'Monitors', and 'History'. The main workspace shows a collection named 'Herokuapp' with several requests listed. A specific PUT request for '/booking/1' is selected. The 'Authorization' tab in the header is highlighted with a red box. In the body section, the 'Type' dropdown is set to 'Basic Auth' (also highlighted with a red box). Below it, there are fields for 'Username' (containing 'admin') and 'Password' (containing 'password123'), both of which are also highlighted with a red box. A note above these fields says: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.' A 'Send' button is visible at the top right.

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



POSTMAN

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 is selected, targeting the URL `{{HerokuappBaseUrl}}/booking/1`. The "Body" tab is active, showing a JSON payload:

```
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 pane shows the results of the request. The status bar indicates "Status: 200 OK". The response body is also displayed in JSON format, matching the original payload:

```
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 }
```

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 - Collections list
- 2 - Request title: Herokuapp / Kismi guncelleme
- 3 - Method: PATCH
- 4 - Endpoint: {{HerokuappBaseUrl}}/booking/10
- 5 - Body tab selected
- 6 - Raw tab selected
- 7 - Request body content:

```
1
2   ...
3   ...
4
```
- 8 - Save button
- 9 - Send button
- 10 - Response body content:

```
1
2   "firstname": "Hasan",
3   "lastname": "Kova",
4   "totalprice": 529,
5   "depositpaid": false,
6   "bookingdates": {
7     "checkin": "2021-04-11",
8     "checkout": "2021-05-09"
9   }
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 section: The "Authorization" param is selected.
4. Request URL: The URL is set to `DELETE {{(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 "1 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 sorğu 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 kaydededelim.
- 7- Send butonu ile request'i çağırıralım.



<https://jsonplaceholder.typicode.com>

The screenshot shows the JSONPlaceholder homepage within a Postman interface. The URL in the address bar is <https://jsonplaceholder.typicode.com>. The page title is "JSONPlaceholder". In the top right corner, there is a navigation bar with links: "Guide" (which is highlighted with a red box), "Sponsor this project", "Blog", and "My JSON Server". The main content area features a large heading "**{JSON} Placeholder**". Below it, a subheading reads "Free fake API for testing and prototyping." and "Powered by [JSON Server](#) + [LowDB](#)". A small note at the bottom states "As of Dec 2020, serving ~1.8 billion requests each month."

- 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

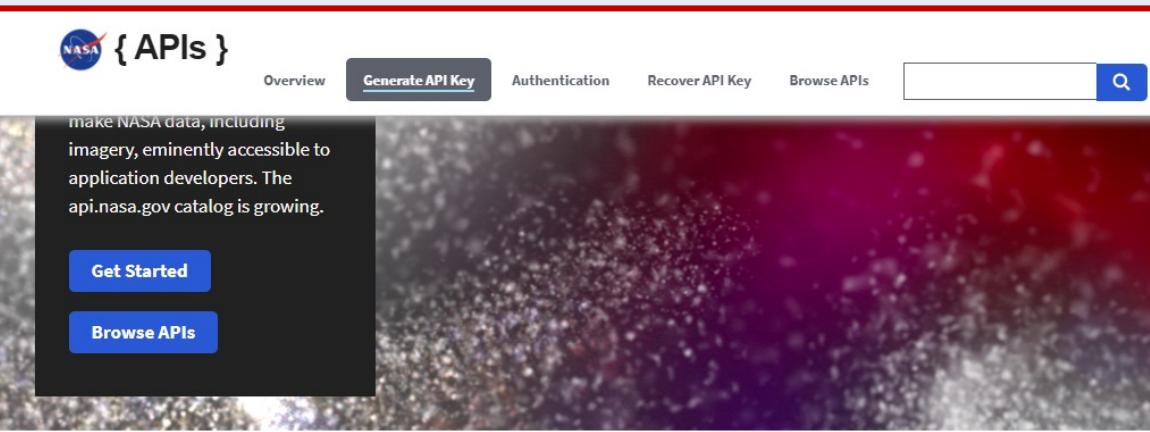


POSTMAN

DERS 7

BOLUM 2

- 1- POSTMAN GENEL TANITIMI, VARIABLE OLUSTURMA,
GET VE POST REQUEST OLUSTURMA**
- 2- AUTHORIZATION ILE PUT, PATCH VE DELETE REQUEST
OLUSTURMA**
- 3- FARKLI ENDPOINTLER KULLANMA**



The screenshot shows the NASA APIs homepage. At the top, there's a navigation bar with links for "Overview", "Generate API Key" (which is highlighted in blue), "Authentication", "Recover API Key", and "Browse APIs". Below the navigation is a search bar with a magnifying glass icon. The main content area features a dark background with a starry space image. On the left, there's a sidebar with the NASA logo and the text: "make NASA data, including imagery, eminently accessible to application developers. The api.nasa.gov catalog is growing." It contains two buttons: "Get Started" and "Browse APIs".

Generate API Key

Sign up for an application programming interface (API) key to access and use web services available on the Data.gov developer network.

* Required fields

* First Name

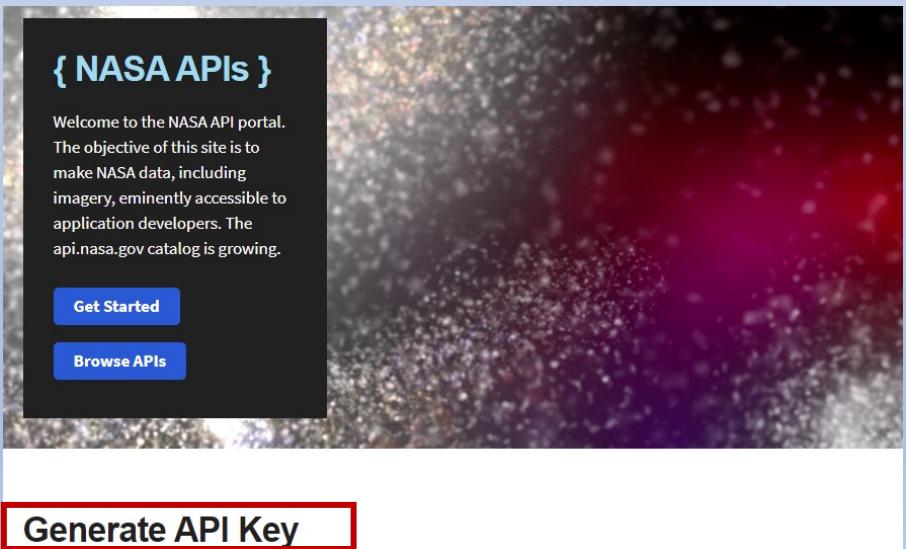
* Last Name

* Email

Application URL (optional):

Signup

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



The screenshot shows a generated API key page. It features a dark background with a starry space image. At the top, 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 contains two buttons: "Get Started" and "Browse APIs". Below this, there's a large red box containing the text "Generate API Key".



1- Üst bölümde bulunan Browse APIs sekmesini seçelim

A screenshot of the NASA API homepage. At the top, there is a navigation bar with the NASA logo, a search bar containing '{ APIs }', and four tabs: 'Overview', 'Generate API Key', 'Authentication', and 'Recover API Key'. The 'Browse APIs' tab is highlighted with a red box. Below the navigation bar, the page title 'Browse APIs' is displayed, along with a search bar and a blue 'Search' button. A list of NASA APIs is shown, each with a '+' sign to its right.

Browse APIs	
<input type="text"/> <button>Search</button>	
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 end-point 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 sorgu :

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 secebilir istediginiz tarihi girerek gunun astronomy fotografini alabilir ve uygulamalarinizda kullanabilirsiniz

APOD: Astronomy Picture of the Day

APOD

ornek sorgu :

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

Mesaj
Oyun
Resim Boyutlandırma
Seyahat
Sinema
Spor
Sözlük
Website
Whatsapp
Yaşam
Çeviri

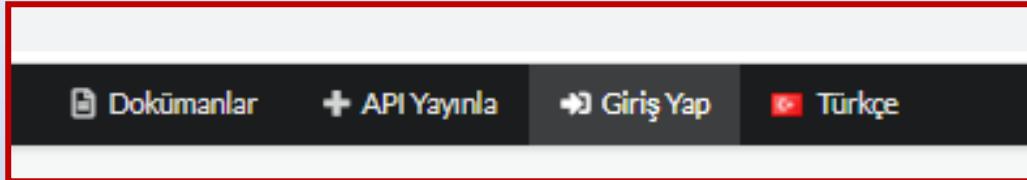
Whatsapp Business API Official WhatsApp Business API. WhatsApp'ın resmi API'sidir.
Whatsapp Business Sandbox WhatsApp Business API'yi test edebilmeniz için sandık gibi bir ortam sunar.
COVID-19 Koronavirus İstatistik API Dünya genelindeki COVID-19 virüsünün etkilerini DÜ olarak getiren servis.
Instagram DM API Unofficial Instagram DM API kullanarak HTTP istekle servis.
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 getirilen haberleri bulabileceğiniz API'ler. İsterseniz RSS linki
Hava Durumu API Dünyanın her yerindeki hava durumunu günlük olarak bulabileceğiniz API'ler.
Faiz Oranları API Bankaların genel faiz oranları ve farklı kredi türlerinin bilgileri de ulaşabilirsiniz.
Şans Oyunları API Son sayısal ve süper lotoda kazanan numaraları bulabileceğiniz API'ler.
IP Adresi API İp adresini girerek lokasyon bilgisi, şehir bilgisi,ISP bilgisi
Namaz Vakitleri API İstediğiniz şehrin namaz vakitlerini getiren, se
Çıplaklılık Algılama API Resimlerin çıplaklı içeriği içermediğini öğrenin.
IMDB API IMDB sitesinde isimle yada id ile arama yaparak film
Araç Tanıma API Resimlerdeki araçların sayısını, marka, model, renk bilgilerini bulabileceğiniz API'ler.
Plaka Tanıma API Resimdeki araçların plakalarını yakalayın.
Nesne Tanıma API Resimde neler olduğunu, hangi nesnelerin bulunduğu konumu bulabileceğiniz API'ler.
Duygu Analiz API Resim ya da yazının içerdığı duyguları dereceleriley bulabileceğiniz API'ler.
E-Ticaret Ürün Öneri API Kullanıcının bulunduğu e-ticaret sitesinde bakmış olduğu ürünlerin önerilerini bulabileceğiniz API'ler.

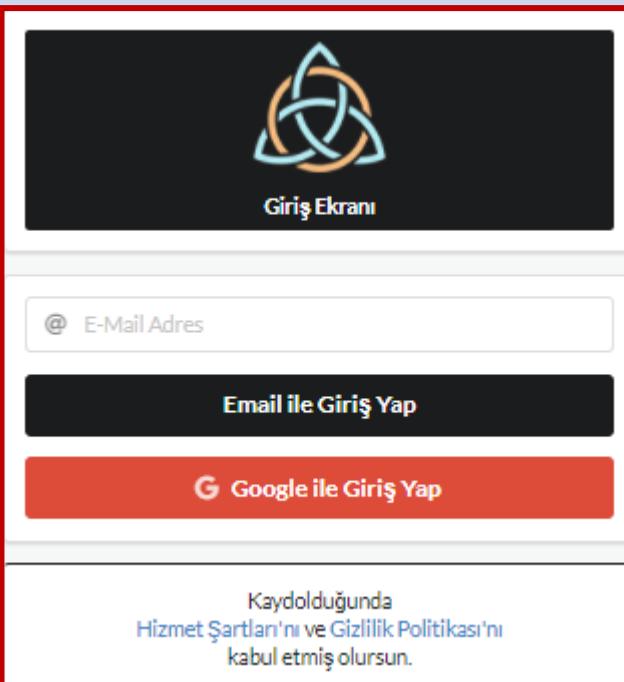
<https://collectapi.com/tr/>



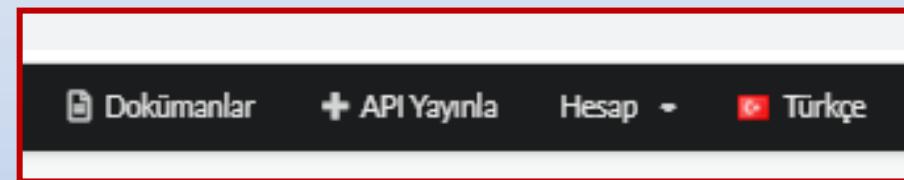
1- Ekranın sağ üst tarafındaki **Giris Yap** sekmesine gidelim



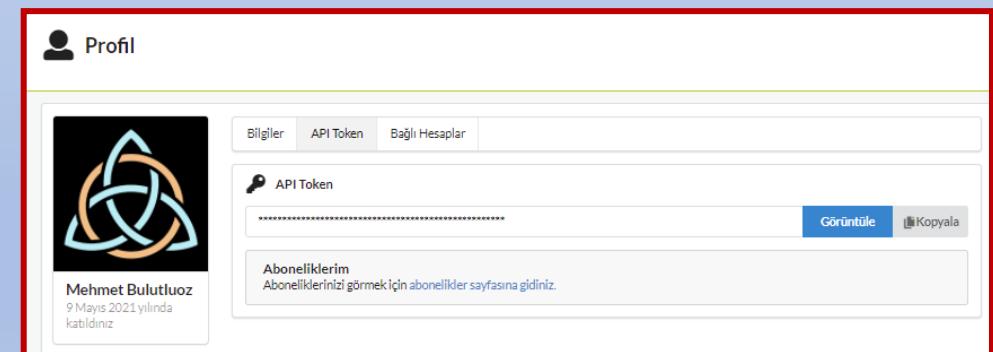
2- Dilediginiz yontemle sisteme giriş yapalim



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



4- API Token'i gorunur hale getirelim ve kopyalayalim





Akaryakıt Fiyatları API

Şehirlere göre farklı akaryakıt istasyonlarındaki benzin ve dizel fiyatlarına ulaşabilirsiniz.

Görüntüle >

Sorguyu yaparken Headers menusunde “**authorization**” isminde yeni bir header olusturalım ve değer olarak siteden aldığımız “**apikey token**” i yazalım

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
<input checked="" type="checkbox"/> district	kadikoy	
<input checked="" type="checkbox"/> city	istanbul	
<input checked="" type="checkbox"/> content-type	application/json	
<input type="checkbox"/> authorization	apikey 7o0llqcDndqPfWsEQBrke7:63xbVxUvHLL0CeD3nubi...	

Key Value Description

Body Cookies (1) Headers (21) Test Results Status: 200 OK Time: 5.01 s Size: 1.45 KB Save Response

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    }
  ]
```



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 / Nobetci 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: apikey 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
```



DERS 8

BOLUM 3
API TEST OTOMASYONU

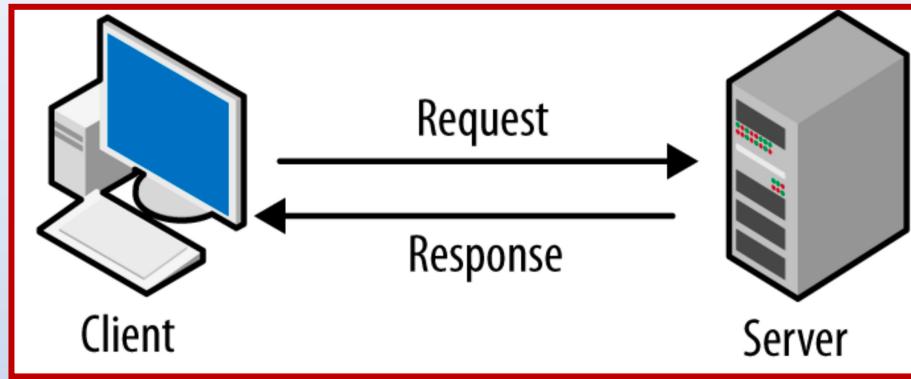
API TESTING NEDIR ?

MANUEL VE OTOMASYON TEST KARSILASTIRMASI

INTELLIJ FRAMEWORK OLUSTURULMASI

ILK API SORGUSU YAPIP, RESPONSE'U YAZDIRMA

API TESTING NEDIR ?



Her API sorgusu Client ile Server arasında bir iletisim demektir.

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

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 donmesi gereken Response belirlenir (Expected Data)
- 3- Request gönderilip, request karşılığında Server'in dondurdugu Response kaydedilir (Actual Data)
- 4- Expected Data ile Actual Data karşılaştırılır (Assertion) fark varsa rapor edilir.

API TESTING NEDİR ?

<https://jsonplaceholder.typicode.com/posts> uri'ina

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

Json oblesi kullanilarak yapılan bir POST request sonucunda,
Server'in dondurecegi Response'in

- status kodunun 201,
- title'inin Ahmet,
- body'sinin "Merhaba"
- Content type'inin application/json
- server isimli header degerinin "cloudflare",
- response suresinin 500 ms'den az oldugunu test edin

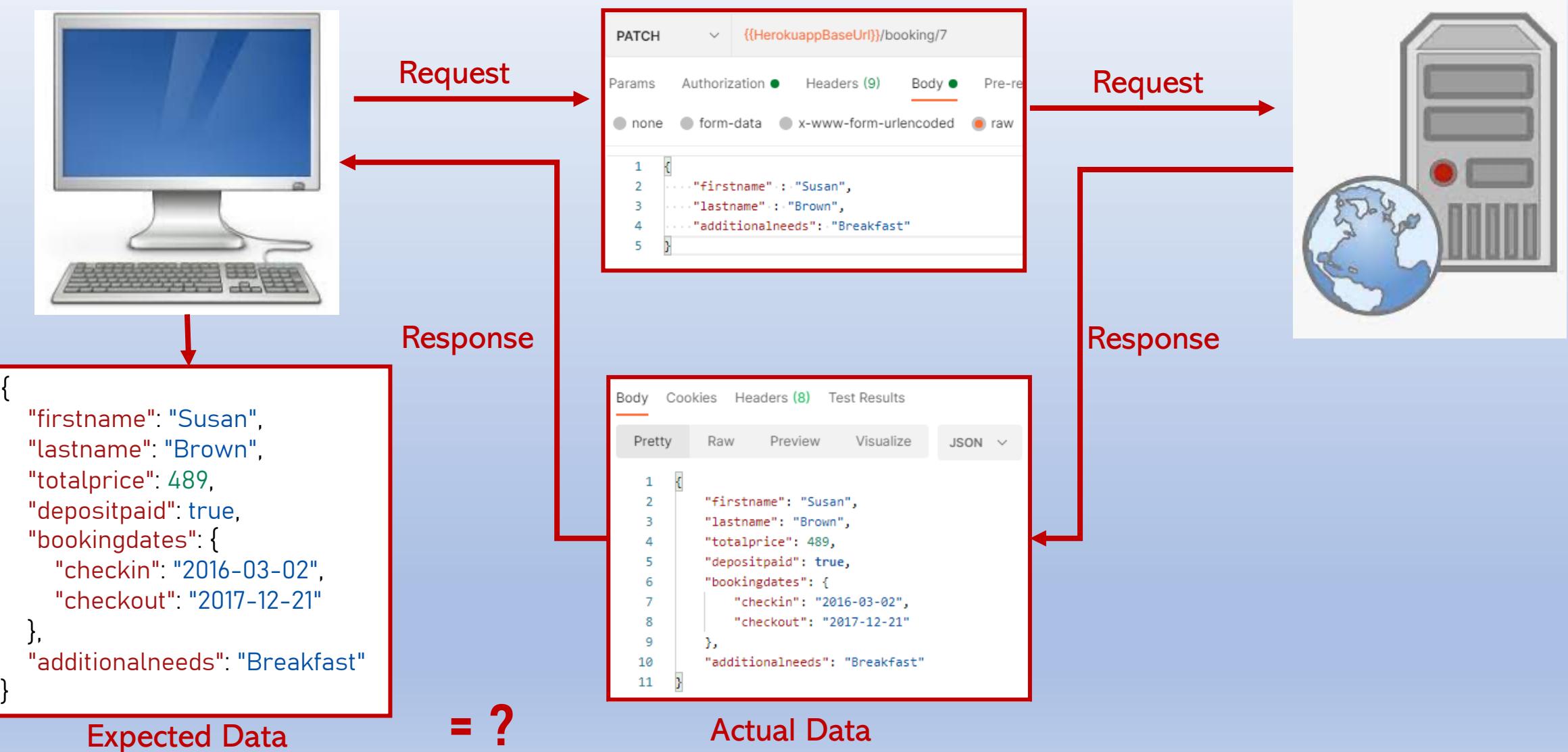
The screenshot shows the Postman interface with a red border around the request area. The method is set to POST, the URL is {{JsonplaceBaseUrl}}/posts, and the Body tab is selected. The body content is a JSON object:

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

The screenshot shows the Postman interface with a red border around the response area. The Body tab is selected, showing the response body as a JSON object:

```
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

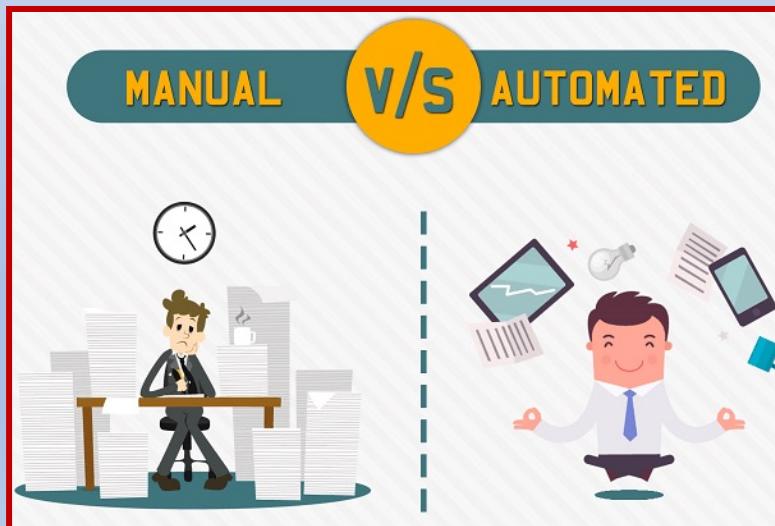
= ?

```
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

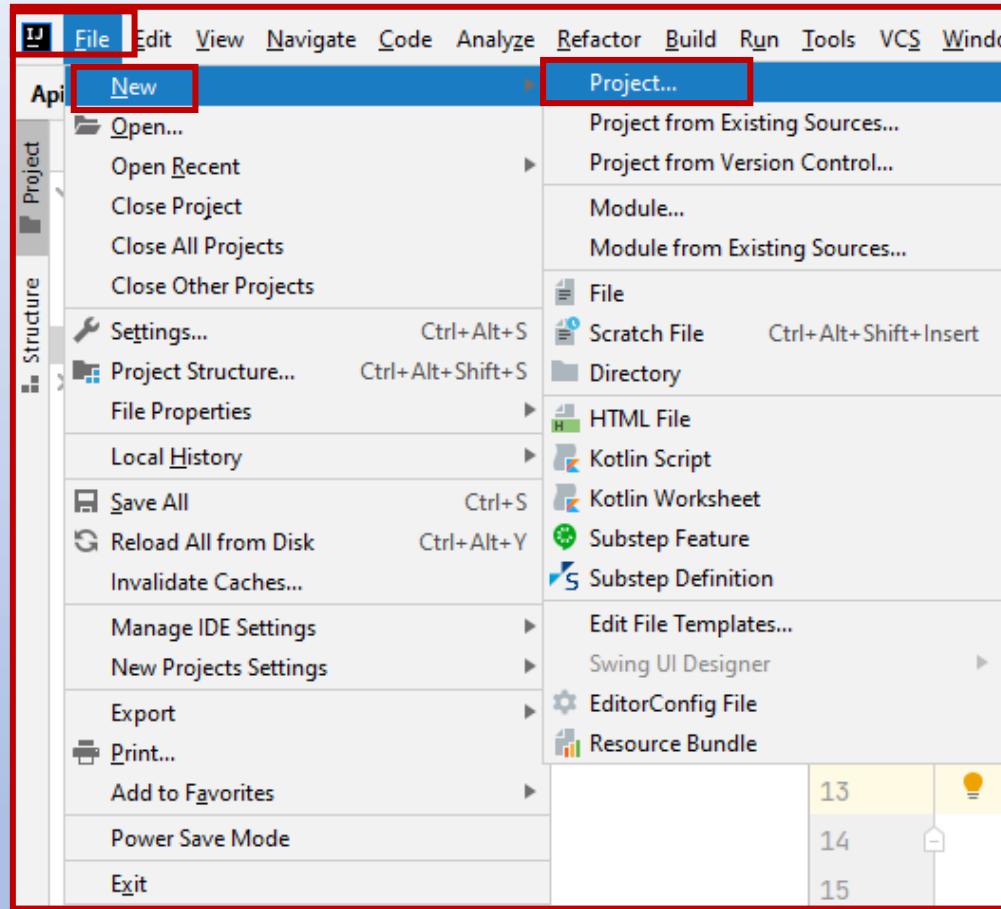


Request'in yazilan otomasyon kodlari ile Server'a gonderilmesi ve donen Response'in Expected Data karsilastirilip raporlanmasinin da yazilan kodlarla yapilmasidir.

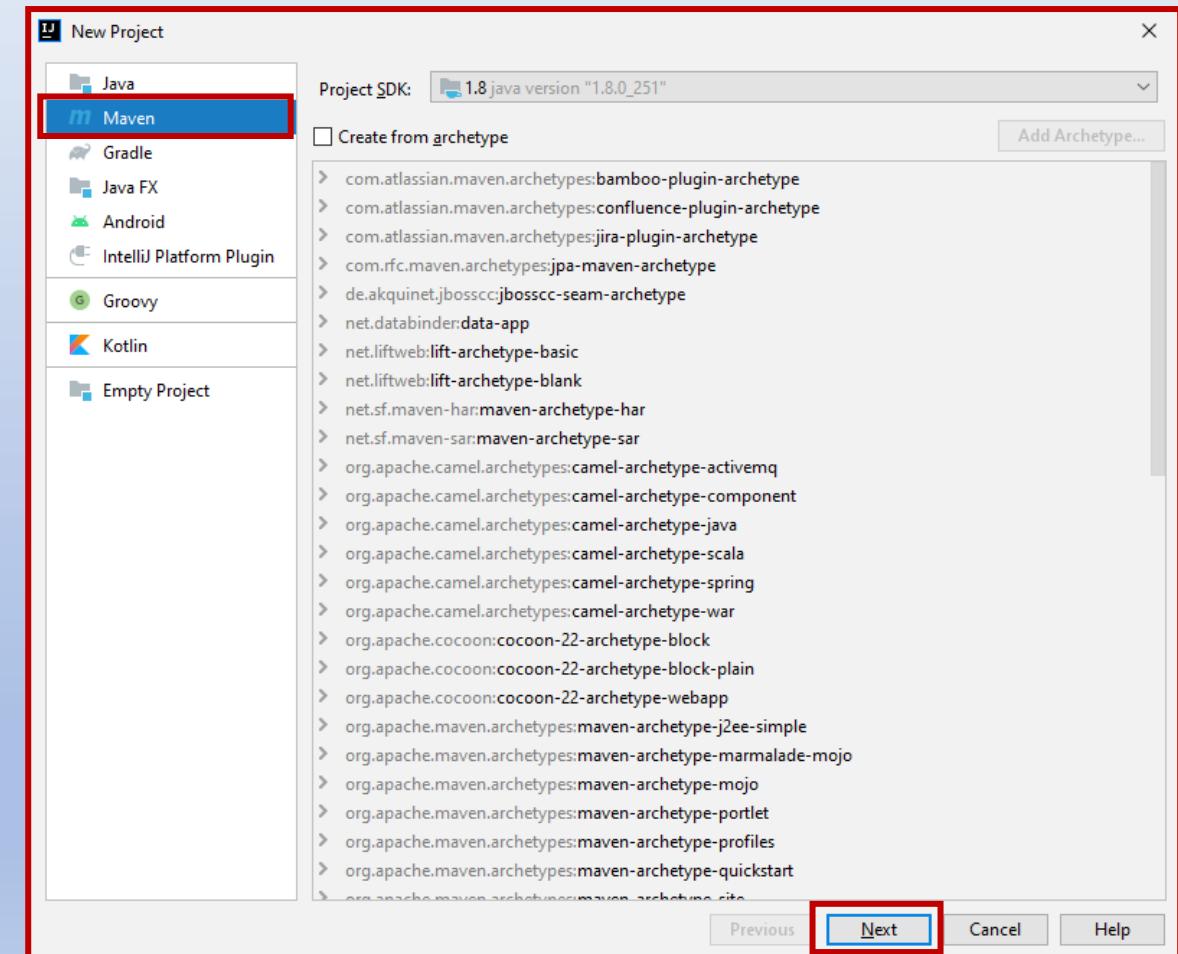
Tool olarak IntelliJ ide ve kutuphane olarak Selenium kullanacagiz

INTELLIJ FRAMEWORK OLUSTURMA

1- File-New-Project secelim

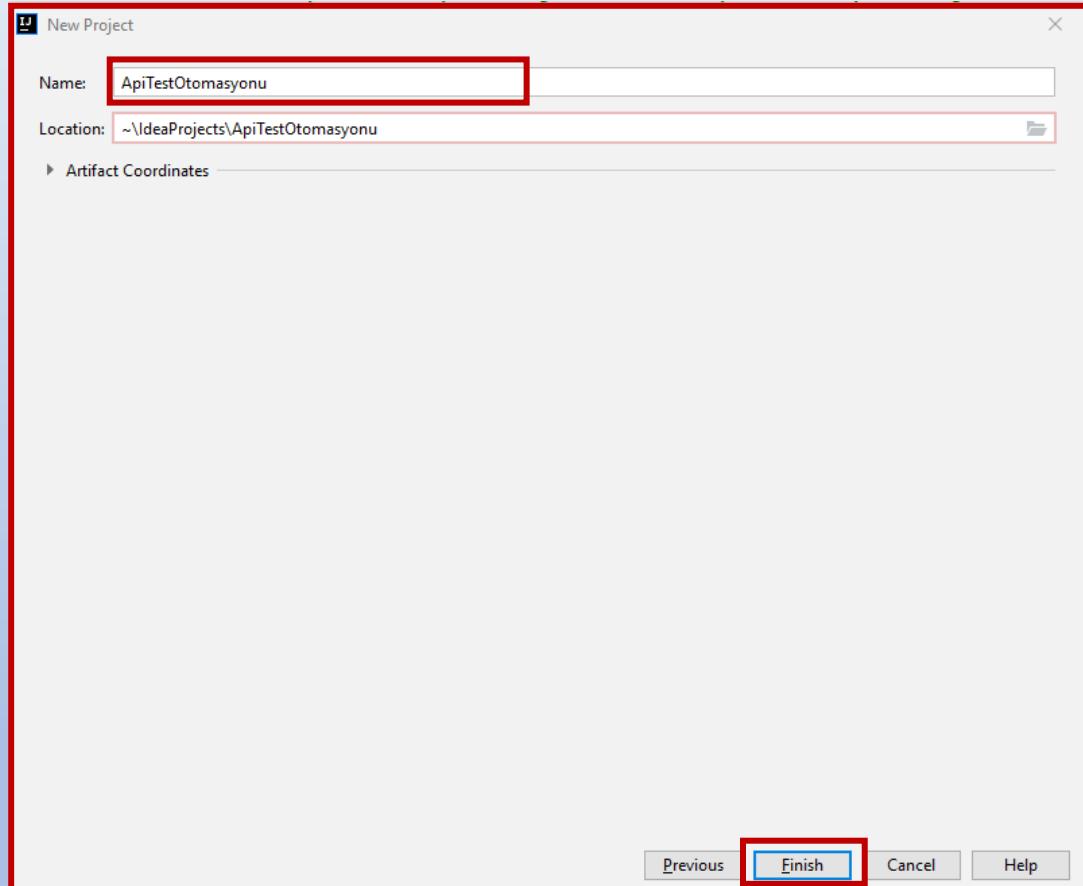


2- Maven'i secip next'e basalim



INTELLIJ FRAMEWORK OLUSTURMA

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



4- pom.xml'i secelim

The screenshot shows the 'Project Structure' dialog for the 'ApiTestOtomasyonu' project. The 'pom.xml' file is selected in the 'Project' view. The right-hand panel displays the XML content of the 'pom.xml' file, which includes project metadata like groupId, artifactId, version, and properties.

```
<?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

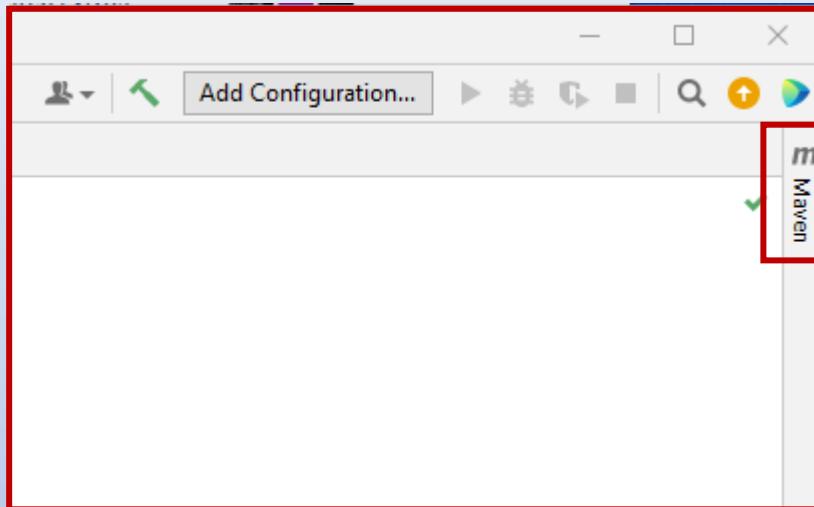
5- properties'den sonraki isaretli kisma sagdaki dependies'i ekleyelim

```
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>
```

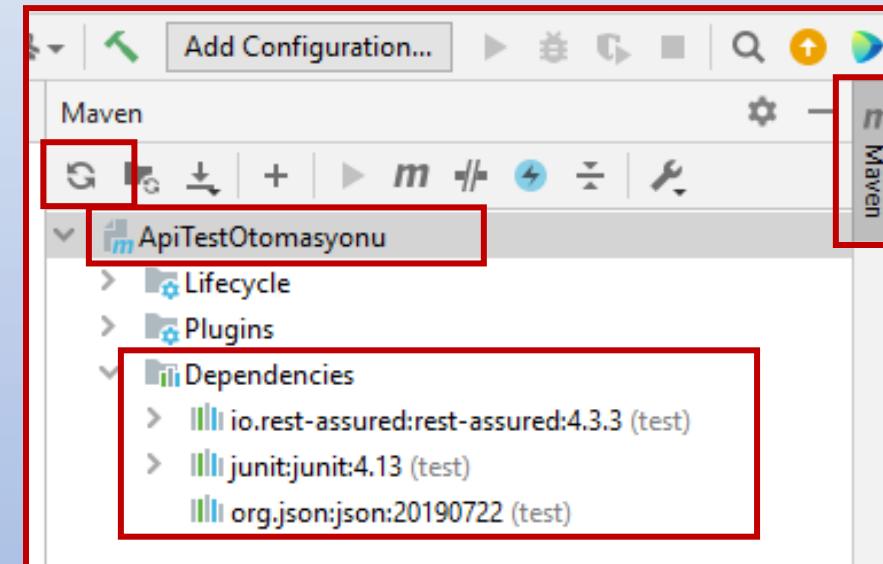
```
<dependencies>
    <dependency>
        <groupId>io.rest-assured</groupId>
        <artifactId>rest-assured</artifactId>
        <version>5.2.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20220320</version>
    </dependency>
</dependencies>
```

INTELLIJ FRAMEWORK OLUSTURMA

6- Ekranin sag ust kosesindeki Maven yazisina tiklayalim



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

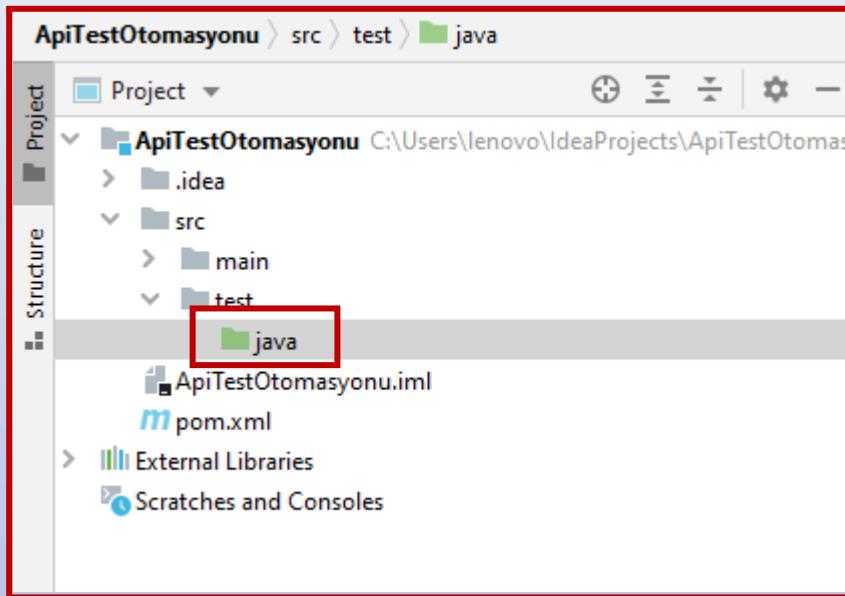


8- Eger kutuphaneler gorunmuyorsa yenile butonuna basin ve kutuphaneler gorunur oluncaya kadar bekleyin

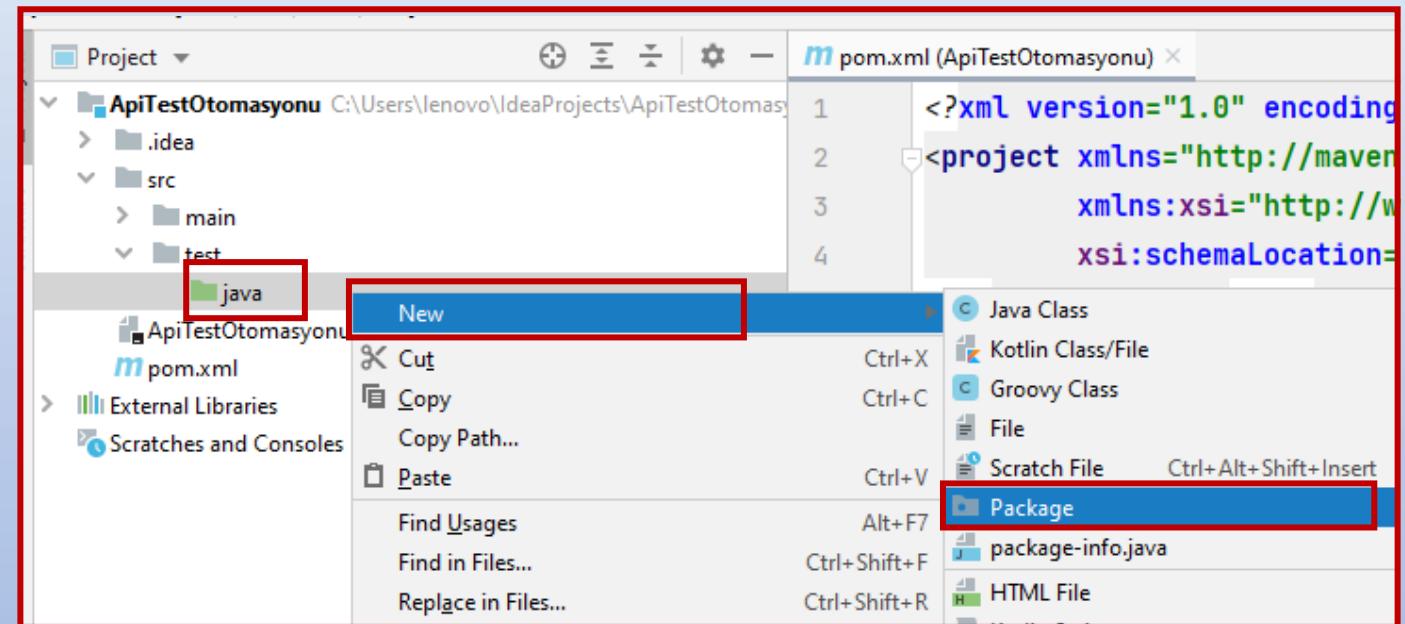
9- Kutuphaneler gorunur oldugunda Maven yazisina basarak acilir pencereyi kapatin

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 uzerinde sag klik yapip New – Java Class 'i secelim ve class ismi olarak C1_Get_ApiSorgulama yazalim

OTOMASYON ILE GET SORGUSU YAPMA

C1_Get_ApiSorgulama

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

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 gerçeklestirdigimiz islemleri ifade eder

then : Response degerlerini degerlendirmek icin yapılan islemleri ifade eder

and : Birbirine bagli islemleri ifade eder



DERS 9
BOLUM 3
API TEST OTOMASYONU

OTOMASYON ILE GET REQUEST YAPMA
RESPONSE BILGILERININ MANUEL TEST EDİLMESİ
OTOMASYON ILE GET SORGUSU RESPONSE BILGILERININ TEST EDİLMESİ

OTOMASYON ILE GET SORGUSU YAPMA

C02_Get_ApiResponseBilgileriGoruntuleme

<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 gerçeklestirdigimiz 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 kullanabilecegimiz 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()

C03_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");



DERS 10
BOLUM 3
API TEST OTOMASYONU

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

OTOMASYON ILE PUT / POST VE PATCH SORGUSU YAPMA

The screenshot shows the Postman application interface. The title bar says "jsonplaceholder / Yeni kayit olustur". The method is set to "POST" and the URL is "{{JsonplaceBaseUrl}}/posts". The "Body" tab is selected, showing a JSON object with five lines of code:

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

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.

llerleyen 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 içinde bulunan variable isimleridir.
 - 3- **Values** : JSON datalari içinde bulunan variable'lara atanan degerlerdir.
- Keys** ve **Values** arasında : kullanılır.

JSON kullandigi key – value yapisi 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 acisindan esnektir.

JSON OBJECT OLUSTURMA

C04(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

C05_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()

C06_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



DERS 11
BOLUM 3
API TEST OTOMASYONU

API SORGULARINDA RESPONSE BODY'NIN TEST EDILMESI

GET REQUEST BODY BILGILERINI ASSERT YAPMA

C07_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().

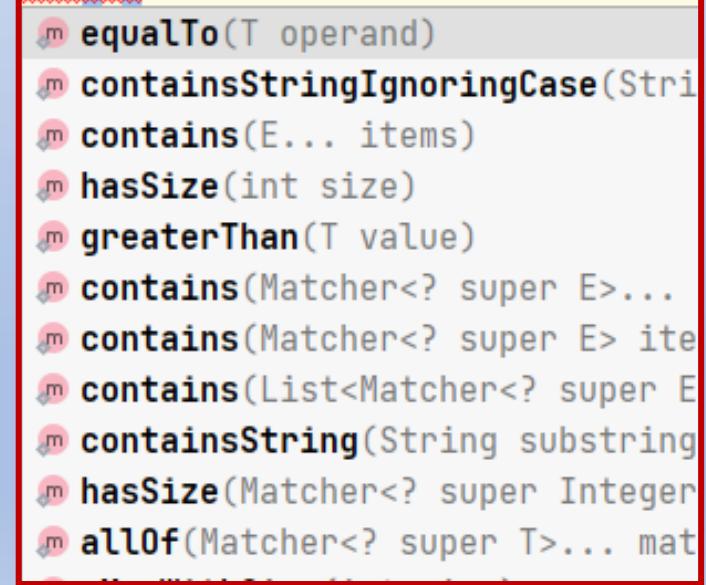
body("title", Matchers.equalTo("optio dolor molestias sit")).

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 shown 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.

POST REQUEST BODY BILGILERINI ASSERT YAPMA

C08_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));
```



DERS 12
BOLUM 3
API TEST OTOMASYONU

API SORGULARINDA RESPONSE BODY TEST EDILIRKEN
KOD TEKRARLARINDAN KURTULMA

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C09_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

C09_Get_BodyTekrarlardanKurtulma

Su ana kadar yaptigimiz haliyle assertion;

response.

then().

assertThat().

statusCode(**200**).

contentType(MediaType.**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

C09_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(ContentType.**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

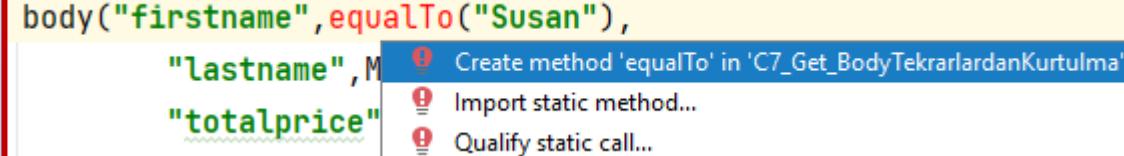
C09_Get_TestYaparkenTekrarlardanKurtulma

Matchers yazilarindan kurtulmak icin

- 1) Matchers yazilarindan birini silin

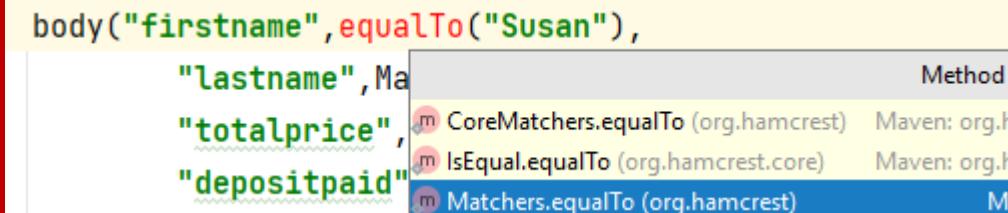
```
body("firstname", equalTo("Susan"))
```

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



```
body("firstname", equalTo("Susan")),
    "lastname", M
        "totalprice"
    "depositpaid"
```

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



```
body("firstname", equalTo("Susan")),
    "lastname", Ma
    "totalprice",
    "depositpaid"
```

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

GET REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN TEKRARLARDAN KURTULMA

C09_Get_TestYaparkenTekrarlardanKurtulma

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

C09_Get_TestYaparkenTekrarlardanKurtulma

Matchers yazılarından kurtulmak için

response.

then().

assertThat().

statusCode(**200**).

contentType(MediaType.**JSON**).

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

 "lastname", equalTo("Wilson"),

 "totalprice", equalTo(**643**),

 "depositpaid", equalTo(**false**),

 "additionalneeds", equalTo(**null**));



DERS 13

BOLUM 3
API TEST OTOMASYONU

KOMPLEKS JSON OBJELERI VE JSONPath KULLANIMI

JSONPath KULLANIMI

C10(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

C10_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);
```

C10_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":"53  
2-555 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/>



DERS 14

BOLUM 3
API TEST OTOMASYONU

API RESPONSE BODY BILGILERI TESTLERINDE
JsonPATH KULLANIMI

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C11_Post(jsonPathleBodyTesti

<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" : "2023-01-10",  
        "checkout" : "2023-01-20"  
    },  
    "additionalneeds" : "wi-fi"  
}
```

donen Response'un,
status code'unun 200,
ve content type'inin application-json,
ve response body'sindeki
"firstname"in,"Ahmet",
ve "lastname"in, "Bulut",
ve "totalprice"in,500,
ve "depositpaid"in,false,
ve "checkin" tarihinin 2023-01-10
ve "checkout" tarihinin 2023-01-20
ve "additionalneeds"in,"wi-fi"

oldugunu test edin

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C11_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 olusuyorsa JsonPath yöntemleri kullanılarak assertion yapılabilir.

response.

```
then().  
assertThat().  
statusCode(200).  
contentType(MediaType.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

C12_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.

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



DERS 15

BOLUM 3
API TEST OTOMASYONU

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'ı 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

C13_Get_ExpectedDataOlusturma

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

Response body :

```
{  
    "userId": 3,  
    "id": 22,  
    "title": "dolor sint quo a velit explicabo quia nam",  
    "body": "eos qui et ipsum ipsam suscipit autsed omnis non odioexpedita ear  
        um mollitia molestiae aut atque rem suscipitnam 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"));
```



DERS 16

BOLUM 3
API TEST OTOMASYONU

Junit ASSERT
(Hard Assert)

POST REQUEST, RESPONSE BODY BILGILERINI ASSERT YAPARKEN JSONPATH KULLANMA

C14_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"  
    }  
}
```



DERS 17

BOLUM 3
API TEST OTOMASYONU

TestNG SOFT ASSERT

TestNG ASSERT

1. HARD ASSERT

JUnit'te Öğrendiğimiz Assertion Türü

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

Assert methodları kullanıldığında, bir assertion FAILED olursa execution durur, test method'un 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 calistirdığınızda başka bir assertion FAILED olabilir, her seferinde bir hatayı gormus ve düzeltmis olursunuz. Testteki hataları düzeltmek için bu yöntem çok uygun olmayabilir.

2. SOFT ASSERT (VERIFICATION)

Eğer softAssert başarısız olursa test method'unun geri kalanını 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'ları yapalım

```
softAssert.assertTrue( );
```

- 3.Adım: SoftAssert'in durumu raporlamasını isteyelim

```
softAssert.assertAll( );
```

SOFT ASSERT İLE EXPECTED DATA TESTİ

C15_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."  
}
```



DERS 18

BOLUM 3
API TEST OTOMASYONU

TestNG SOFT ASSERT

SOFT ASSERT İLE EXPECTED DATA TESTİ

C16_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 has been updated."}
```

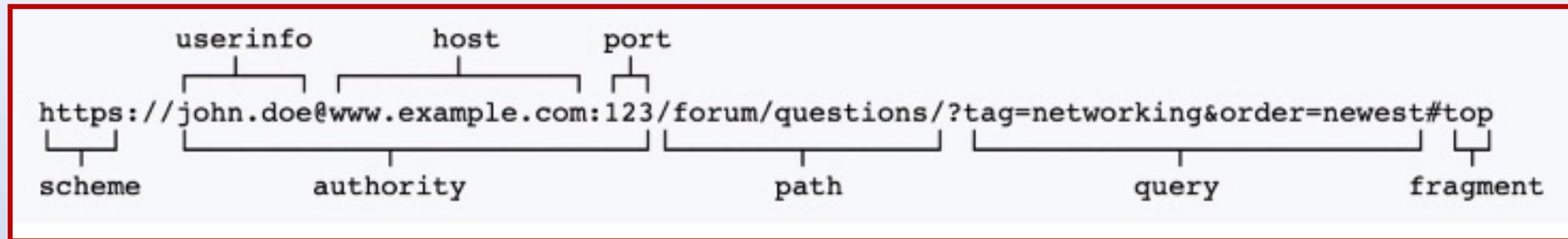


DERS 19

BOLUM 4
FRAMEWORK GELISTIRME

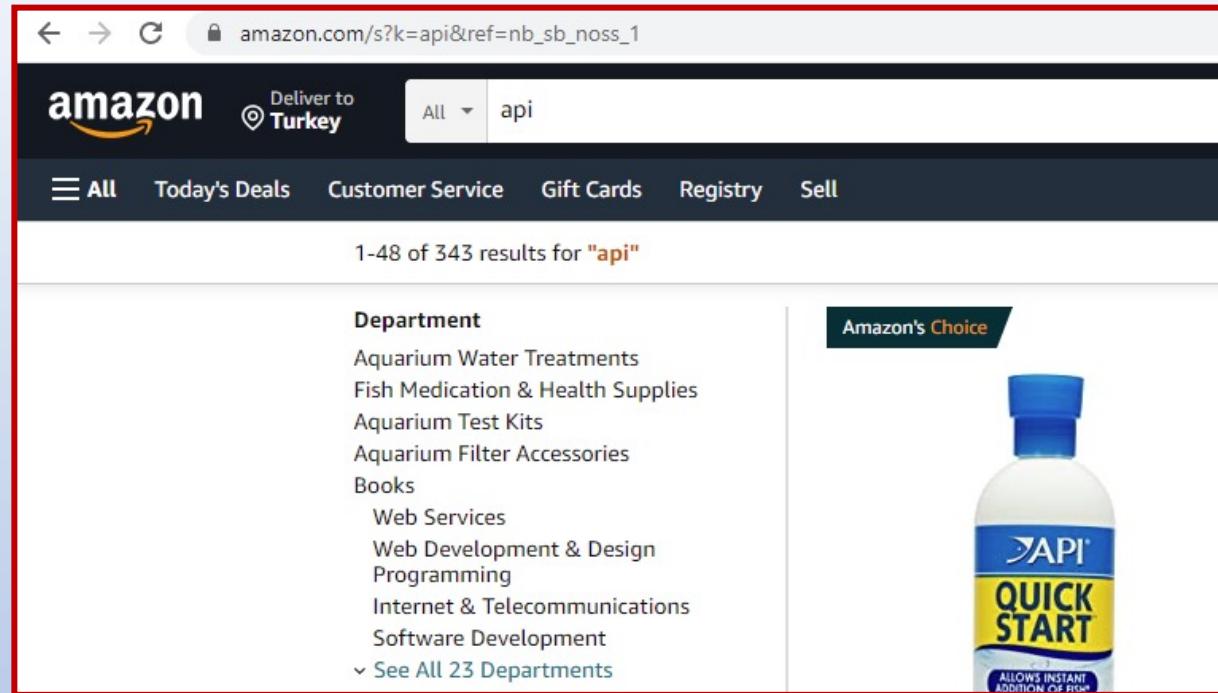
Base URL KULLANIMI

PATH VE QUERY PARAMETRELERİ

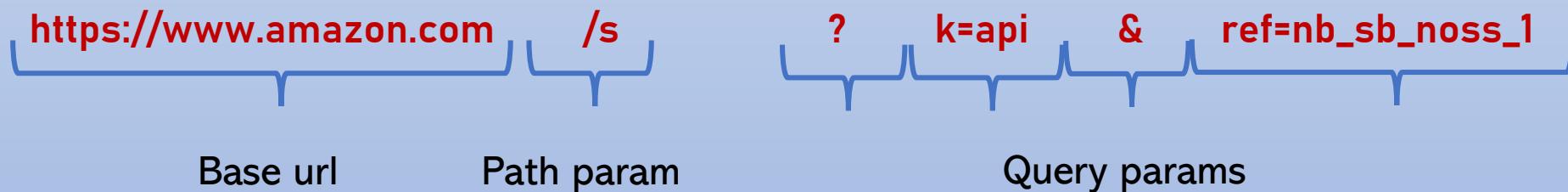


- Bir end-point'i tanımlamak veya daha ayrıntılı nesneleri üzerinde hareket etmek istiyorsanız Path Param kullanmalısınız. Ancak öğeleri sıralamak veya filtrelemek istiyorsanız, Query Parametresi kullanmalısınız. Query parametreleri, kaynakları daha iyi bir şekilde tanımlamaya yardımcı olan benzersiz özelliklere sahiptir.
- Query parametreleri URL'de "?" işaretinin sağ tarafında görünürken, Path parametreleri soru 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



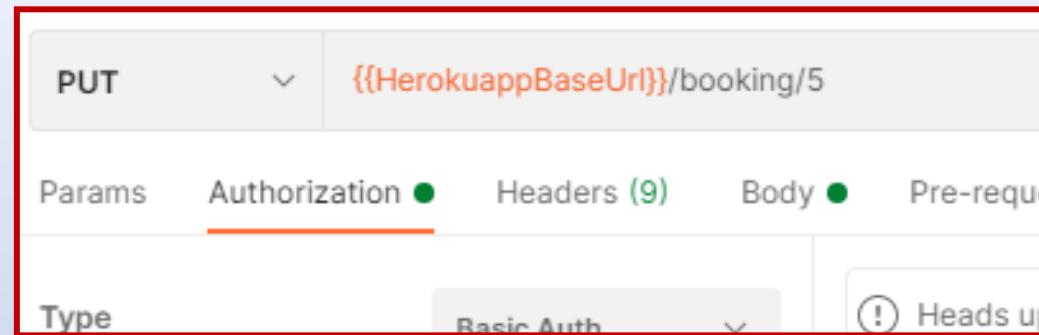
Ornegin amazon sitesine gidip "api" icin sorgu yaptigimizda, uygulama bizi https://www.amazon.com/s?k=api&ref=nb_sb_noss_1 adresine yonlendirmektedir



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.

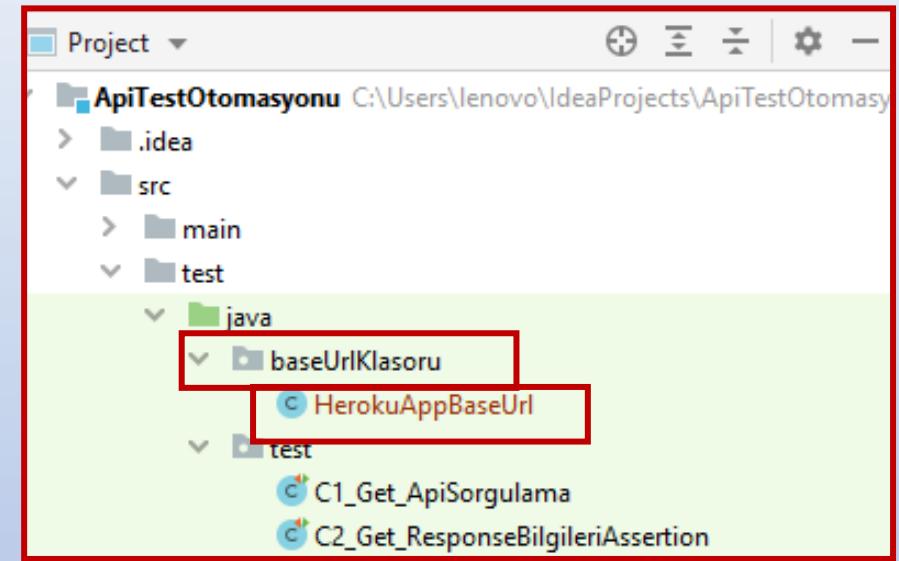
Bunu saglamak icin base url ayri bir class'da olusturulur, test yaptigimiz classlar inheritance metotlarini kullanarak base url'i bulundugu class'dan kullanirlar.

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 metodу 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.

C17_BaseUrlDummyRestapi

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



DERS 20

BOLUM 4
FRAMEWORK GELISTIRME

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



DERS 21

BOLUM 4
FRAMEWORK GELISTIRME

Base URL KULLANIMI 2

BASE URL KULLANIMI

C19_BaseUrlHerokuapp

Class icinde 2 Test metodu olusturun ve asagidaki testleri yapin

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

- 2- <https://restful-booker.herokuapp.com/booking> endpointine yandaki body'ye sahip bir POST request gonderdigimizde 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

C21_BaseUrlHerokuappQueryParam

Class icinde 2 Test metodu olusturun ve asagidaki testleri yapin

- 1- <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

- 2- <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



DERS 22

BOLUM 4
FRAMEWORK GELISTIRME

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**) tamamına 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.

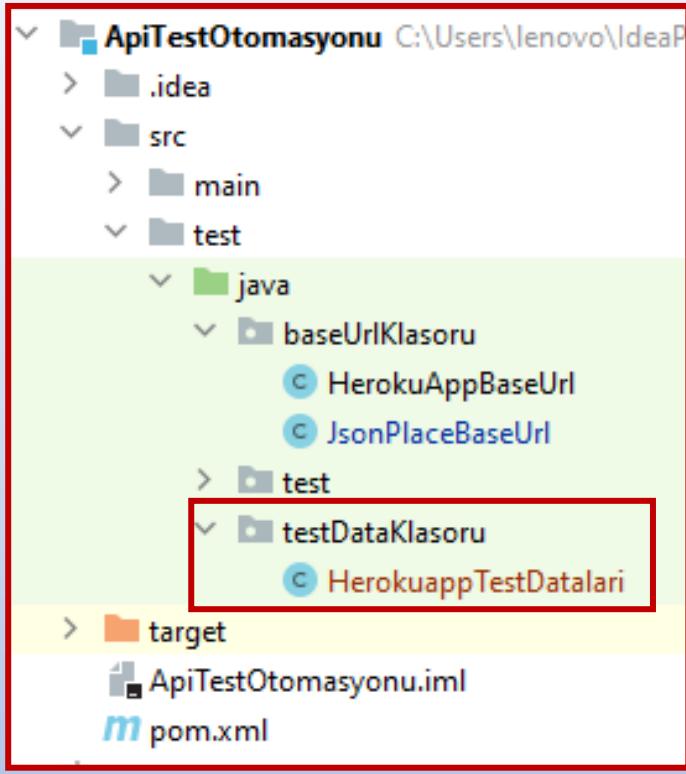
C23_Get_TestDataClassKullanimi

<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 datarini bu class'da olusturalim.
- 3) Olusturdugumuz Class'da temel response bilgileri icin instance variable, request ve response body icin metot 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

Java'da Class Uyelerini Kullanma

Java ile olusturulan bir proje'de farkli class'daki class uyelerine erisim ve kullanma farkli yontemlerle yapilabilir.

1- inheritance (Miras)

kullandigimiz Class'i extends anahtar kelimesi (keyword) ile istedigimiz Class'in child'i yapabiliriz.

Bu durumda object olusturmaya gerek kalmadan Parent class'a ulasabilir ve oradaki variable ve methodlari kullanabiliriz. (BaseUrl class'lari gibi)

Inheritance ile variable ve method kullanirken static keyword'e dikkat etmek gerekir. Static olarak tanimlanmis bir variable veya method static olmayan method icinden kullanilamaz.

```
public class Okul {  
    String okulIsmi="Yildiz Koleji";  
    static int ogrenciSayisi=120;  
  
    public void okulMethod(){  
        System.out.println("Yildiz Koleji");  
    }  
}
```

```
public class Ogrenci extends Okul {  
  
    public void ogrenciMethod(){  
        System.out.println(okulIsmi);  
        okulMethod();  
  
        System.out.println(ogrenciSayisi);  
    }  
}
```

extends

TEST DATA CLASS KULLANIMI

Java'da Class Uyelerini Kullanma

2- Object olusturarak

Bir class'dan obje olusturarak istedigimiz class'a ulasabilir ve o class'daki variable ve methodlari object'imizi aracılıgiyla kullanabiliriz

ornek: Servis class'indan Okul class'ina ulasmak istiyorsak

- Okul class'indan bir obje olustururuz
- obje uzerinden variable veya method'lara ulasabiliriz

3- Static Class Uyeleri :

Eger kullanacagimiz variable veya method static ise object olusturmadan direk class ismi ile variable veya method'a ulasabiliriz.

```
public class Okul {  
    String okulIsmi="Yildiz Koleji";  
    static int ogrenciSayisi=120;  
  
    public void okulMethod(){  
        System.out.println("Yildiz Koleji");  
    }  
}
```

```
public class Servis {  
    public static void main(String[] args) {  
        Okul okulObj = new Okul();  
        System.out.println(okulObj.okulIsmi);  
        okulObj.okulMethod();  
  
        System.out.println(okul.ogrenciSayisi);  
    }  
}
```

TEST DATA CLASS KULLANIMI

C23_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 oldugunutest 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"  
}
```



DERS 23

BOLUM 4
FRAMEWORK GELISTIRME

TestData CLASS KULLANIMI

TEST DATA CLASS KULLANIMI

C24_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": 4,  
  "id": 40,  
  "title": "enim quo cumque",  
  "body": "ut voluptatum aliquid illo tenetur nemo sequi quo facilis\\nipsum rem optio mollitia  
quas\\nvoluptatem eum voluptas qui\\nunde omnis voluptatem iure quasi maxime voluptas nam"  
}
```

TEST DATA CLASS KULLANIMI

C25_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  
}
```



DERS 24

BOLUM 4
FRAMEWORK GELISTIRME

TestData CLASS KULLANIMI 3

TEST DATA CLASS ILE EXPECTED DATA TESTİ

C26_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."  
}
```



DERS 25

BOLUM 4
FRAMEWORK GELISTIRME

TestData CLASS KULLANIMI 4

TEST DATA CLASS ILE EXPECTED DATA TESTİ

C27_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

```
{  
    "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"  
    } }  
}
```



DERS 26

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

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 direk 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 yardım alırız. Gson Class'ındaki metodları kullanmak için önce o Class'dan bir object oluştururuz.

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

Oluşturduğumuz gson objesi ile gson.toJson(actualDataMap); metodunu kullanarak, map'i JSONObject olarak kullanabileceğimiz String formatına çeviririz.

DE-SERIALIZATION ILE EXPECTED DATA TESTİ

C28_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  
}
```



DERS 27

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DE-SERIALIZATION 2

DE-SERIALIZATION ILE EXPECTED DATA TESTİ

C29_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."  
}
```



DERS 28

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

DE-SERIALIZATION 3

DE-SERIALIZATION ILE EXPECTED DATA TESTİ

C30_Post_Deserialization

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



DERS 29

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

POJO KULLANIMI 1
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 TESTI

C31_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  
}
```



DERS 30

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

POJO KULLANIMI 2
Plain Old Java Object

POJO CLASS ILE EXPECTED DATA TESTİ

C32_Post_Pojo

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



DERS 31

BOLUM 5
FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

POJO KULLANIMI 3
Plain Old Java Object

POJO CLASS ICIN 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

C33_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."  
}
```



DERS 32

BOLUM 5

FARKLI DATA TURLERI ve TEKNIKLER KULLANMA

POJO KULLANIMI 4
Plain Old Java Object

POJO ILE EXPECTED DATA TESTI

C34_Post_Pojo

<https://api.openweathermap.org/data/2.5/weather?q=London&appid=f4ffe3b2ef1fcb3600ab1d7fbc88c2f0> url'ine

bir post request gonderdigimizde donen response'un asagidaki body'ye sahip oldugunu test ediniz

```
{
  "coord": {
    "lon": -0.1257,
    "lat": 51.5085
  },
  "weather": [
    {
      "id": 803,
      "main": "Clouds",
      "description": "broken clouds",
      "icon": "04d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 279.13,
    "feels_like": 275.22,
    "temp_min": 277.57,
    "temp_max": 280.03,
    "pressure": 1023,
    "humidity": 82
  },
  "visibility": 10000,
  "wind": {
    "speed": 6.17,
    "deg": 250
  },
  "clouds": {
    "all": 56
  },
  "dt": 1675241066,
  "sys": {
    "type": 2,
    "id": 2075535,
    "country": "GB",
    "sunrise": 1675237170,
    "sunset": 1675270107
  },
  "timezone": 0,
  "id": 2643743,
  "name": "London",
  "cod": 200
}
```



DERS 33

BOLUM 6
CUCUMBER ILE BDD KULLANIMI

Cucumber ve BDD Genel Bilgilendirme

CUCUMBER FRAMEWORK

Feature:

✓ 24

@Api

Scenario:

```
Given Kullanici "jPHBaseUrl" base URL'ini kullanir
Then Path parametreleri icin "posts/44" kullanir
And jPH server a GET request gonderir ve testleri yapmak icin response degerini kaydeder
Then jPH respons'da status degerinin 200
And jPH respons'da content type degerinin "application/json; charset=utf-8"
Then jPH GET respons body'sinde "userId" degerinin Integer 5
And jPH GET respons body'sinde "title" degerinin String "optio dolor molestias sit"
```

Cucumber **BDD** (behaviour driven development / Davranış tabanlı geliştirme) yaklaşımı ile kullanılan açık kaynak kodlu bir kütüphanedir.

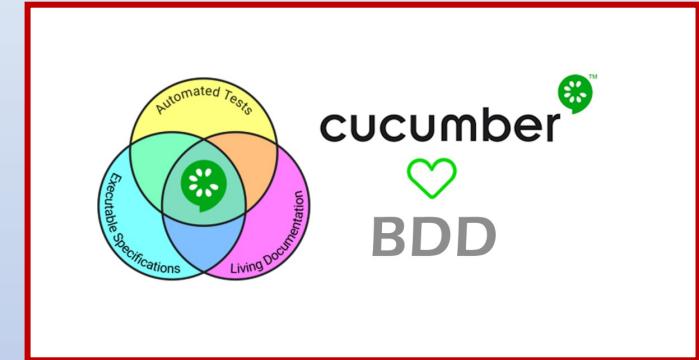
Agile metodolojisinde, insanlar uygulamanın işlevsellliğini geliştirmek için birlikte çalışmak zorundadır. Cucumber **development team**'deki herkesin test case'leri anlayabilmesini saglar.

CUCUMBER FRAMEWORK

Anlasilabilir **Gherkin** Language nedeniyle BDD'ye uygun sekilde Cucumber ile otomasyon kodlari yazilabilir.

BDD kod yazimina baslamadan once kullanici davranışlarini belirlemeyi esas alir.

- Kullanicinin atacagi adimlar belirlenip Gherkin Language anahtar kelimeleri ile .feature uzantili bir Feature olusturulur.
- Daha sonra kullanici davranışları için uygun step definition(test adim)'ları olusturulur.



Gherkin Language 4 keyword'e sahiptir. Bu key-word'ler arasında kod acisindan hic bir farklilik yoktur ancak test adimlarının daha anlasılır olması icin uygun kelimelerin kullanılması daha guzel olur.

Given anahtar kelimesi ile ön koşul yani başlangıç durumu tanımlanır,

When, And anahtar kelimeleri ile olayı

Then anahtar kelimesi ile de sonuç tanımlanır.

CUCUMBER FRAMEWORK PROJE OLUSTURMA

1. Create Project: File => New => Project => Select maven => click next

2. Name: cucumber_api => finish

3. Add Dependencies =>

Selenium-java,

cucumber java,

cucumber junit

rest assured

json

4. Click Maven => click  “Enable auto-reload after any changes” (Reload)

5. Java'ya sag click yapip asagidaki paketleri olusturalim

a. utilities

b. hooks

c. runners (test case'leri calistirmak ve control etmek icin kullanacagiz)

d. stepdefinitions (kodlarimizi burada olusturacagiz)

CUCUMBER FRAMEWORK PROJE OLUSTURMA

- 6- Utilities paketi altinda ConfigReader Class'ini olusturalim
- 7- Projeye sag click yapip configuration.properties dosyasi olusturalim
- 8- test paketi altinda yeni bir klasor olusturalim : resources
- 9- resources klasoru altinda yeni bir klasor olusturalim : features (Java kodu icermeyen dosyalari buraya koyacagiz)
- 10- features'a sag clik yapip dosya olusturalim amazonsearch.feature
- 11- cucumber for Java plugin'i intelliJ'e ekleyelim (settings/Plugins)
MAC => IntelliJ Idea->Preference->Plugins->Marketplace->Type Cucumber for Java->Install->Restart



DERS 34

BOLUM 6
CUCUMBER ILE BDD KULLANIMI

Cucumber ve BDD ile GET Request



C35_Get_Bdd_JsonPlaceHolder

```
Given Kullanici "jPHBaseUrl" base URL'ini kullanir
Then Path parametreleri icin "posts/44" kullanir
And jPH server a GET request gonderir ve testleri yapmak icin response degerini kaydeder
Then jPH respons'da status degerinin 200
And jPH respons'da content type degerinin "application/json; charset=utf-8"
Then jPH GET respons body'sinde "userId" degerinin Integer 5
And jPH GET respons body'sinde "title" degerinin String "optio dolor molestias sit"
```

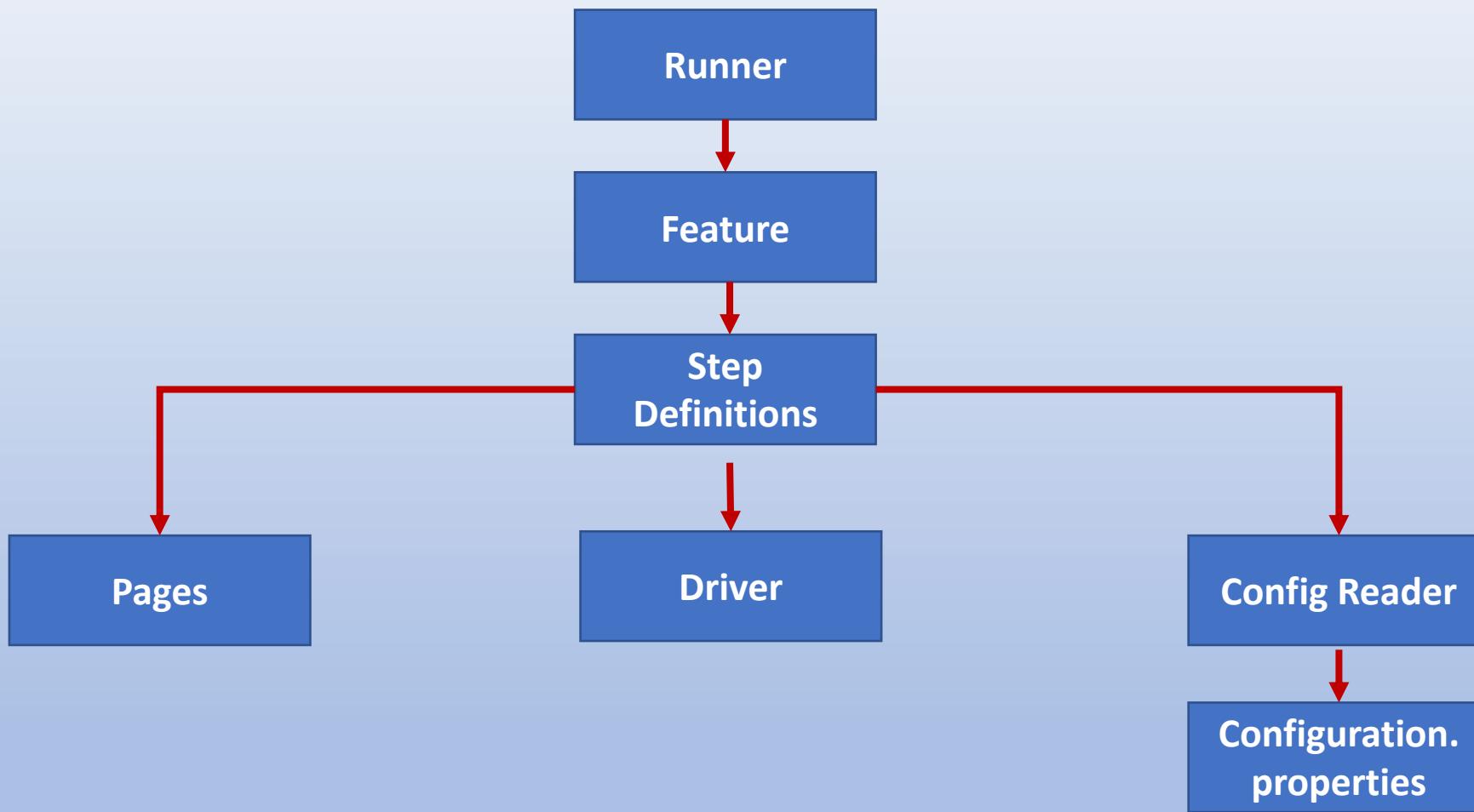


DERS 35

BOLUM 6
CUCUMBER ILE BDD KULLANIMI

Cucumber ve BDD ile POST Request

CUCUMBER FRAMEWORK CALISMA SEMASI





C36_Post_Bdd_JsonPlaceHolder

```
Given Kullanici "jPHBaseUrl" base URL'ini kullanir
Then Path parametreleri icin "posts/70" kullanir
And POST request icin "Ahmet","Merhaba",10 70 bilgileri ile request body olusturur
And jPH server a POST request gonderir ve testleri yapmak icin response degerini kaydeder
Then jPH respons'da status degerinin 200
And jPH respons'da content type degerinin "application/json; charset=utf-8"
And jPH respons daki "Connection" header degerinin "keep-alive"
Then response attribute degerlerinin "Ahmet","Merhaba",10 70
```