

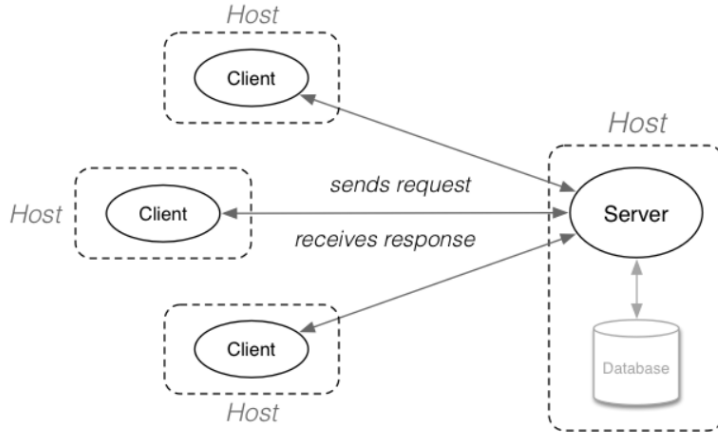
Client-server Architecture

Ağ üzerinde iki farklı rolde cihaz bulunur.

client: istemci

server: sunucu

Bu mimaride bir server'a bağlı client cihazlar server üzerinden yetki kontrollerinden geçerek verilere ulaşır. Client'ler server'dan bağımsız hareket edebilir ama veri paylaşım durumlarda yetkiyi server'dan alacaktır.



Bir tarayıcıdan internet sitesine gitmek istersek, tarayıcıya girmek için kullandığımız cihaz client, isteği alıp işleyen arama motoru da server rolünde olur. İstek karşılanırsa cevabı gönderilir eğer karşılanamazsa neden karşılanamadığının sebebini belirten bir cevap iletir.

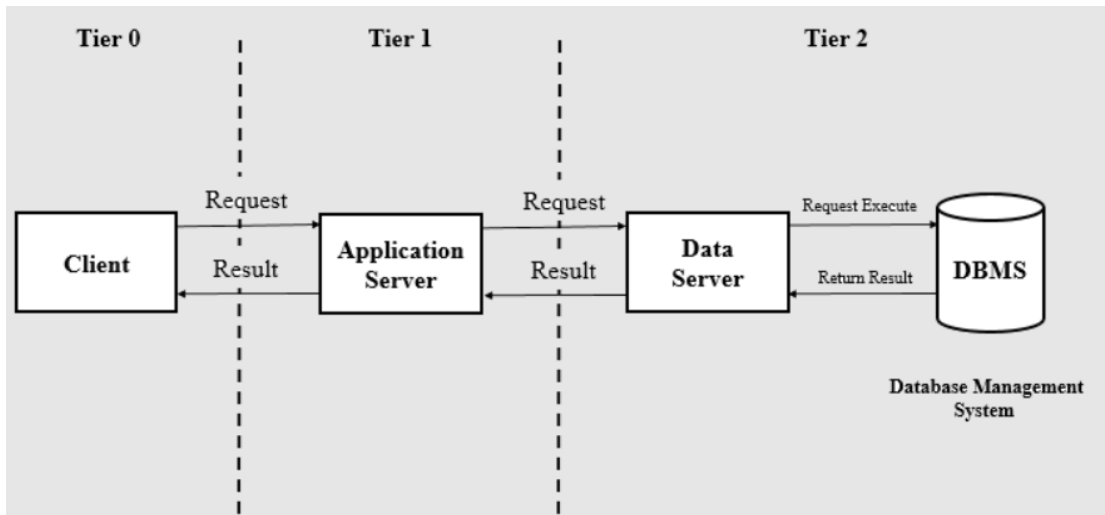
Client-server mimarisi 2 ve 3 katmanlı olarak 2 sınıfa ayrılır.

2 Katmanlı(2-tier)

Client doğrudan server ile iletişime geçebilir bu durum güvenlik sorunları oluşturabilir. Bunun için SSL(Secure Socket Layer) kullanılır.

3 Katmanlı(3-tier)

Client ile server arasında uygulama katmanı bulunur, istek ve talepleri iletir bu durum güvenliği de sağlar.



Server Kullanım Alanları

Uygulama(Application) Sunucuları: Uygulama ya da yazılımın çalıştırılması için kullanılır.

Web Sunucuları: Çok sayıda kullanıcıya eş zamanlı hizmet verebilen web siteleri dosyalarının bulunduğu server'dır. Kullanıcıya tarayıcılar aracılığıyla ulaşır.

FTP Sunucuları: Dosya göndermeyi ve almayı sağlar. Kesinti durumunda kaldığı yerden devam etmeyi sağlar.

DNS Sunucuları: IP adreslerinin yönlendirildiği adresleri kayıtlı tutar ve alan adlarını çözümler.

Ev Sunucuları: Kişisel dosya depolama, yayın yapma gibi amaçlar için kullanılır.

Veritabanı Sunucuları: Veritabanı yönetim sistemi için kullanılır. Master-slave modeline göre senkronize edilir. Ana(Master) veritabanına zarar geldiğinde yedeği bulunan slave veritabanı ile kurtarılabilir.

Sanal Sunucular: Birbirinden bağımsız ve izole sistemler için kullanılır.

Oyun Sunucuları: Online oyunlar için kullanılır.

File(Dosya) Sunucuları: Ortak dosya paylaşımları için kullanılır.

Mail(E-posta) Sunucuları: E-posta hizmetleri için kullanılır.

Proxy(Vekil) Sunucuları: Ağ üzerindeki trafiği azaltmak ve yanıt süresini hızlandırmak için kullanılır. Yoğun talep alan sayfaları ön belleğe alarak client'a bu veriyi gönderir ve bekleme süresi azalır. İnternete erişim sırasında kullanılan ara sunucudur.

Serverların kişisel bilgisayarlardan farkı haftalarca kesintisiz çalışması gerektiği için donanımsal olarak daha verimli olması gerektiğidir.

Web Services

HTTP veya HTTPS protokolü kullanarak hizmet sağlayan yapıların bütünüdür. Platform bağımsız tüm cihazlara veri göndermeyi sağlar. Kullandığımız uygulamalardaki verilere hem tarayıcıdan hem mobil uygulamadan erişmeyi web servis sağlar.

Yazılımlar birbirleri ile web servisler aracılığı ile iletişim kurarlar.

Web Servis Roller

Service Provider

Servis uygulanır ve internete hazır hale gelir.

Service Requestor

Bir ağ bağlantısı oluşturulur sonrasında XML bağlantısı oluşturulup web servis kullanılır.

Service Registry

Mevcut ve yeni hizmetlerin bulunmasını sağlar.

REST

REST: Representational State Transfer - Temsili Durum Transferi

Web servise ekleme yapılarak restful api bulunmuş onun da üzerine ekleme yapılarak web servis ortaya çıkmıştır.

Client-server arasındaki haberleşmede kullanılan HTTP protokolü ile çalışan yazılım mimarisidir. Client-server arasında XML ve JSON veriler üzerinden haberleşme sağlanır.

XML: eXtenbsible Markup Language(Genişletilebilir İşaretleme Dili)

JSON: JavaScript Object Notation(JavaScript Nesne Gösterimi)

Bir servisin RESTful olması için gerekenler:

Client-server yapısı: Sadece kendilerine gelen istekleri değerlendirir böyle daha basit çalışır. Tarayıcının sunucudan haberi yoktur.

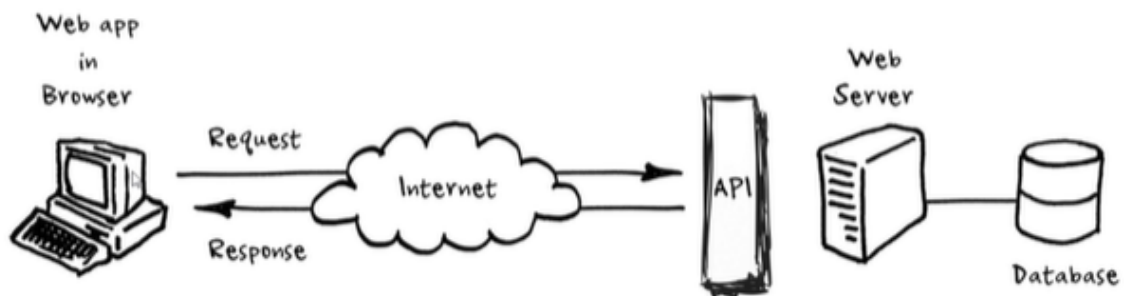
Stateless: Server tarafında client ile ilgili bilgi tutulmaz. Client tarafından gerçekleştiren requestlerde server'ın response dönebilmesi için gerekli bilgiler taşınır yani client server'a request gönderirken bir token ya da kimlik bilgisi göndermelidir.

Cacheable: Veri tutarlılığı için önemlidir.

Layered System: Client server'ın hangi katmanına bağlandığını bilemez burada amaç load-balancing'dir.

Uniform Interface: Client ve server'ın birbirinden bağımsız olmasıdır.

Code on Demand(opsiyonel): Response olarak Client-side script gönderilir.



Temel kavramlar

Resource: API'den elde edilen veri parçasıdır

Request: API'ye yapılan çağrıdır.

Request'in temel olarak 4 bileşeni vardır.

endpoint/route: sorgu oluşturabilmek için kullanılan url'lerdir. Restful API'lere URL üzerinden erişilir. Sorgulamalar url'ler üzerinden yapılır. Sonunda "/" olduğunda endpoint anlamında geliyor. Sorgu parametreleri, "?"'den sonra başlar. ?query1=value1&query2=value2 şeklindedir.

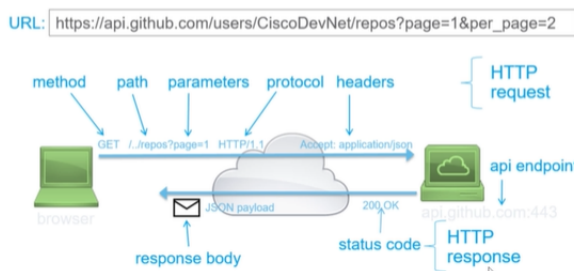
HTTP method: programlama dilleri için bir standarttır.

API mimarisi her isteğin hangi HTTP metodunu kullandığını tutar ve bildirir

HTTP header: veri formatı gibi bilgiler burada belirlenir. Hem client hem server'a bilgi sağlar. Headerlar Property-value pairs şeklindedir.

data/body/message: sunucuya veri göndermek için kullanılır. POST, PUT, PATCH ve DELETE requestlerinde kullanılır.

Anatomy of a REST API query



Response: API'ye yapılan request sonucu, API'den dönen sonuçtur.

Restful HTTP Metotları:

GET: Verileri listeler ve görüntüler.

POST: Veriyi ekleyebilir veya günceller.

PUT: Veriyi günceller.

PATCH: Verinin bir bölümünü günceller.

DELETE: Veriyi siler.

CRUD işlemlerinde karşılıkları:

C -> Create -> POST

R -> Read -> GET

U -> Update -> PUT

D -> Delete -> DELETE

Diğer HTTP methodları

Head: GET ile aynı işlemi yapar ama sadece status line ve header bölümlerini aktarır.

Connect: Verilen uri tarafından tanımlanan sunucuya bir tunnel oluşturur.

Options: Hedef resource için iletişim seçeneklerini açıklar.

Trace: Hedef resource giden yol boyunca message loop-back testi gerçekleştirir.

Message-loop back test: Sinyalleşmenin başarılı olup olmadığını kontrol etmeye yarayan testtir.

SOAP(Simple Object Access Protocol)

İletişim protokolüdür.

TCP protokolü üzerinden işlemler yürütülür.

Platform ve dil bağımsızdır.

Uygulamalar arası iletişimi sağlar.

WSDL(Web Hizmetleri Açıklama Dili) kullanır. SOAP isteklerinin nasıl olması gerektiğini açıklar. Mesaj iletir.

SOAP Yapısı

Envelope: Servis istek ve cevapların bilgisini içerir. XML root elemanıdır. Header, Body, Fault alanlarını içerir.

Header: meta-data(bir kaynağın verileri) bilgilerini iletir.

Body: istekte ve cevapta mesaj adı ve parametreleri barındırır.

Fault: istek sonucunda hata varsa hata mesajını içerir.

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://elif.com/2021/09/soap-envelope"
  SOAP-ENV:encodingStyle = "http://elif.com/2021/09/soap-encoding">

  <SOAP-ENV:Header>
    ...
  </SOAP-ENV:Header>
```

```

<SOAP-ENV:Body>
  ...
  ...
  <SOAP-ENV:Fault>
    ...
    ...
  </SOAP-ENV:Fault>
  ...
</SOAP-ENV:Body>
</SOAP_ENV:Envelope>

```

WSDL(Web Service Description Language)

SOAP isteklerini oluşturmak için kullanılan standarttır. XML biçiminde hazırlanır. Web servisi tanımı işlemler, mesaj formatları, ağ bilgileri gibi bilgilerini içerir. Servisin uygun olup olmadığını tanımlar.

Web servisi tanım belgesinin içerdiği temel elemanlar:

Types: Kullanılacak veri tipini tanımlar.

Message: Kullanılacak mesajları tanımlar.

PortType: İçerilen işlemleri ve mesajları tanımlar.

Binding: İşlem ve mesajlarda da kullanılacak veri formatlarını tanımlar.

Port: Web adresinden oluşan servisi tanımlar.

Service: Kullanılan portları tanımlar.

WSDL Yapısı

```

<definitions>
  <types>
    tip
  </types>

  <message>
    mesaj
  </message>

  <portType>
    <operation>
      işlem
    </operation>
  </portType>

  <binding>
    binding
  </binding>

  <service>
    servis
  </service>
</definitions>

```

Javada JRE kütüphanesinde bulunan “wsimport” komutu ile SOAP servisi WSDL’e dönüştürülebilir.

RESTful ve SOAP Arasındaki Farklar

SOAP *protokol*, REST *mimaridir*.

SOAP protokol olduğu için REST'i kullanamaz. REST SOAP web servisini kullanabilir.

JAX-WS, SOAP web servis için kullanılan java API'dir. **JAX-RS**, RESTful web servisleri için kullanılır.

Güvenlik: SOAP'da güvenlik sağlamak daha kolaydır.

Desteklenen Veri Tipi: REST JSON, XML, TEXT veri tipini destekler ama SOAP sadece XML'i destekler.

Hız: REST SOAP'a göre daha hızlıdır.

Entegrasyon: REST SOAP'a göre daha kolay entegre edilir.

WDSL: REST SOAP gibi WDSL gerektirmez.

Kütüphane kullanımı: REST'de ekstra bir kütüphaneye ihtiyaç duyulmaz.

URL: SOAP sabit URL ile iletişim sağlarken REST ise değişken URL ile metotlar üzerinden iletişim sağlar.

SAMPLE: Common operations on news item object	SOAP approach	RESTful approach
	CreateNewsItem(string id, string title)	/news.svc/{id} HTTP METHOD: PUT
	UpdateNewsItem(string id)	/news.svc/{id} HTTP METHOD: PUT
	GetNewsItem(string id, string title)	/news.svc/{id} HTTP METHOD: GET
	GetNewsItems()	/news.svc/ HTTP METHOD: GET
	DeleteNewsItem(string id)	/news.svc/{id} HTTP METHOD: DELETE

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

Spring Boot ile Restful Web Servis Oluşturmak

```
C:\Users\DELL>java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)
```

-start.spring.io sitesinden spring için dosya indirilir.

The screenshot shows the Spring Initializr web application interface. The browser address bar displays 'start.spring.io'. The interface is divided into several sections:

- Project:** Includes radio buttons for 'Maven Project' (selected) and 'Gradle Project'.
- Language:** Includes radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'.
- Spring Boot:** Includes radio buttons for versions: '2.6.0 (SNAPSHOT)', '2.6.0 (M2)', '2.5.5 (SNAPSHOT)', '2.5.4' (selected), and '2.4.11 (SNAPSHOT)'. There is also a '2.4.10' option.
- Project Metadata:** Includes input fields for 'Group' (com.elif), 'Artifact' (webservices), 'Name' (webservices), and 'Description' (java web services). The 'Package name' is com.elif.webservices.
- Packaging:** Includes radio buttons for 'Jar' (selected) and 'War'.
- Java:** Includes radio buttons for versions: '16' (selected), '11', and '8'.
- Dependencies:** Includes a button 'ADD DEPENDENCIES... CTRL + B' and a section for 'Spring Web' with a 'WEB' tag and a description: 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'
- Buttons:** At the bottom, there are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'.

-Greeting.java sınıfını oluşturulur.

```
package com.elif.webservices;

public class Greeting {
    private final long id;
    private final String content;

    public Greeting(long id,String content) {
        this.id = id;
        this.content = content;
    }

    public long getId() {
        return id;
    }

    public String getContent() {
        return content;
    }
}
```

-istekleri karşılayıp cevap verecek olan RestContoller oluşturulur.

```
package com.elif.webservices;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {
    private static final String template = "Hello, %s";
    private final AtomicLong counter = new AtomicLong();

    @GetMapping("/greeting")
    public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String name) {
        return new Greeting(counter.incrementAndGet(), String.format(template, name));
    }
}
```

@RestController: class'ın rest controller olduğunu belirten anotasyondur.

@GetMapping: HTTP GET isteklerini belirtmek için kullanılmıştır.

@RequestMapping: Tüm istekler için kullanılan genel anotasyondur. GET için kullanılması gerekirse; @RequestMapping(method = GET) olarak kullanılır.

@RequestParam: name için parametre atamak için kullanılmıştır.

MVC controller işe RESTful web servis controller'ı arasındaki fark, HTTP response gövdesinin oluşturulmasıdır. RESTful'da nesne oluşturulur ve döndürülür. Veriler HTTP yanıtına doğrudan JSON olarak yazılır.

Projenin çalışması için:

```
package com.elif.webservices;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class WebservicesApplication {
    public static void main(String[] args) {
        SpringApplication.run(WebservicesApplication.class, args);
    }
}
```

@SpringBootApplication: @Configuration, @EnableAutoConfiguration, @ComponentScan anotasyonlarını içerir.

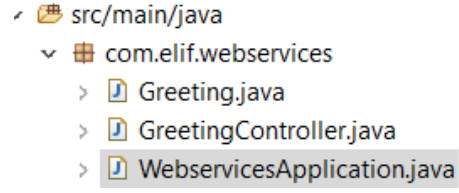
@Configuration: Spring konfigürasyon sınıfı vardır ve bu sınıfta bir ya da birden fazla bean tanımlaması yapılabileceğini belirtir.

Spring bean: Spring de yeniden kullanılabilir objeler olarak kullanılabilir.

@EnableAutoConfiguration: Class path ayarlarına, diğer beanlere ve özellik ayarlarına bean eklemeyi sağlar.

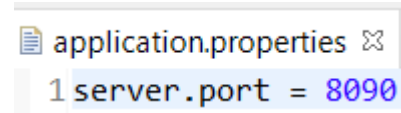
@ComponentScan: Uygulama başladığında bileşenleri, yapılandırmaları, servisleri ve controller'ın bulunmasını sağlar.

Dosya yapısı:



-Port değiştirmek

default port 8080'dir.



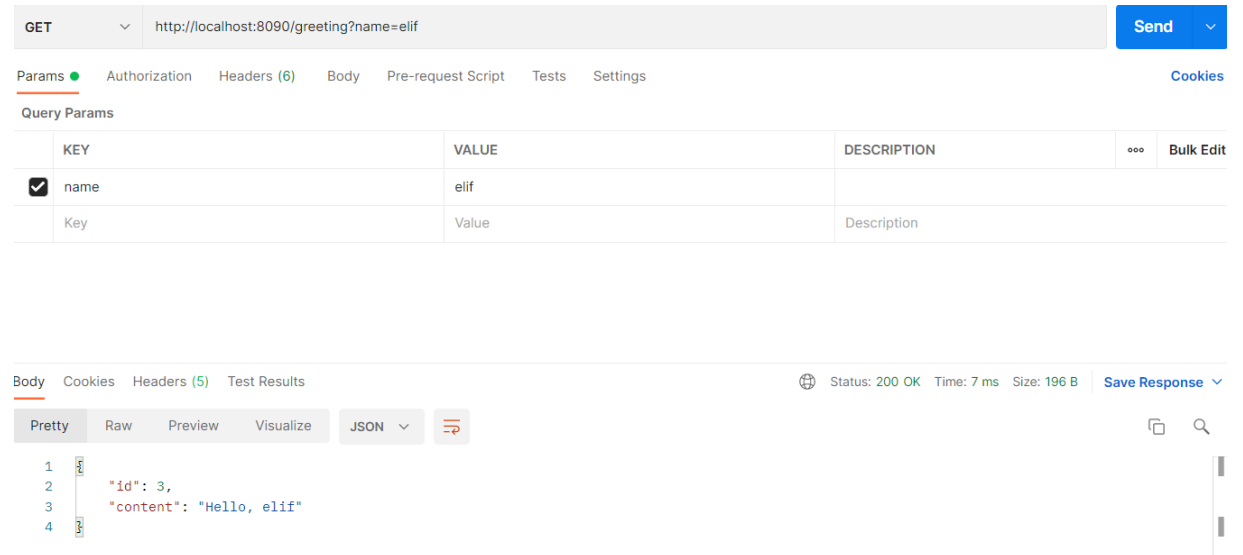
-Browser üzerinden görüntülemek



```
{"id":2,"content":"Hello, World"}
```

Postman üzerinde get methodu isteği göndermek

http://localhost:8090/greeting?name=elif

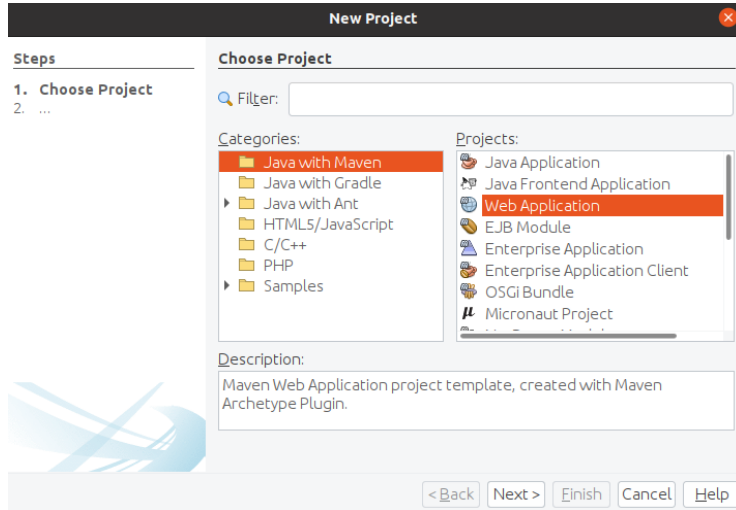


Java Web Servis

-java versiyon kontrolü

```
java@ubuntu:~$ java -version
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~20.04-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

-yeni Web Application projesi oluřturmak



-server ve java ee versiyon seçmek

glassfish server versiyonu 4.1'dir.



-proje oluřtuktan sonra Web Service eklemek



New Web Service

Steps

1. Choose File Type
2. Name and Location

Name and Location

Web Service Name:

Project:

Location:

Package:

☒ Create Web Service from Scratch

-web service işlem ekleme

Add Operation

Name:

Return Type:

Parameters Exceptions

Name	Type	Final
num1	int	<input type="checkbox"/>
num2	int	<input type="checkbox"/>

```
/**
 * Web service operation
 */
@WebMethod(operationName = "sumTwoNumber")
public int sumTwoNumber(@WebParam(name = "num1") int num1, @WebParam(name = "num2") int num2) {
    return num1+num2;
}
```

-proje clean and build işleminden sonra çalıştırılır

```
Starting GlassFish Server
GlassFish Server is running.
In-place deployment at /home/java/NetBeansProjects/CalculatorJavaWebService/target/CalculatorJavaWebService-1.0-SNAPSHOT
GlassFish Server, deploy, null, true
```

-web servisi test etme

Calculator Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract int com.elif.calculatorjavawebsevice.Calculator.sumTwoNumber(int,int)

sumTwoNumber (,)

-SOAP Request ve SOAP Response'ların görüntülenmesi

Start Page x Method invocation trace x +

localhost:18513/CalculatorJavaWebService/Calculator?Tester

sumTwoNumber Method invocation

Method parameter(s)

Type	Value
int	48
int	37

Method returned

int : "85"

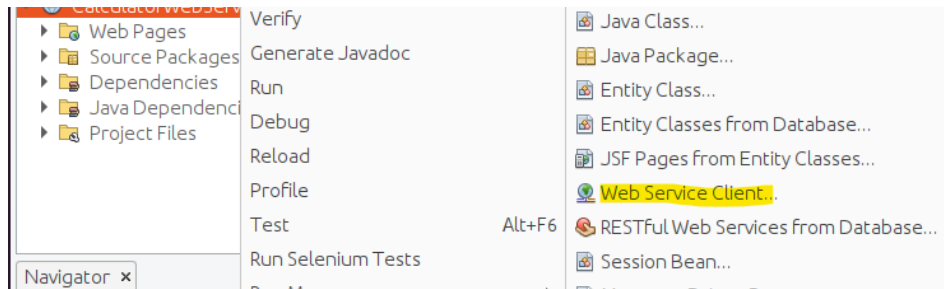
SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sumTwoNumber xmlns:ns2="http://calculatorjavawebsevice.elif.com/">
      <num1>48</num1>
      <num2>37</num2>
    </ns2:sumTwoNumber>
  </S:Body>
</S:Envelope>
```

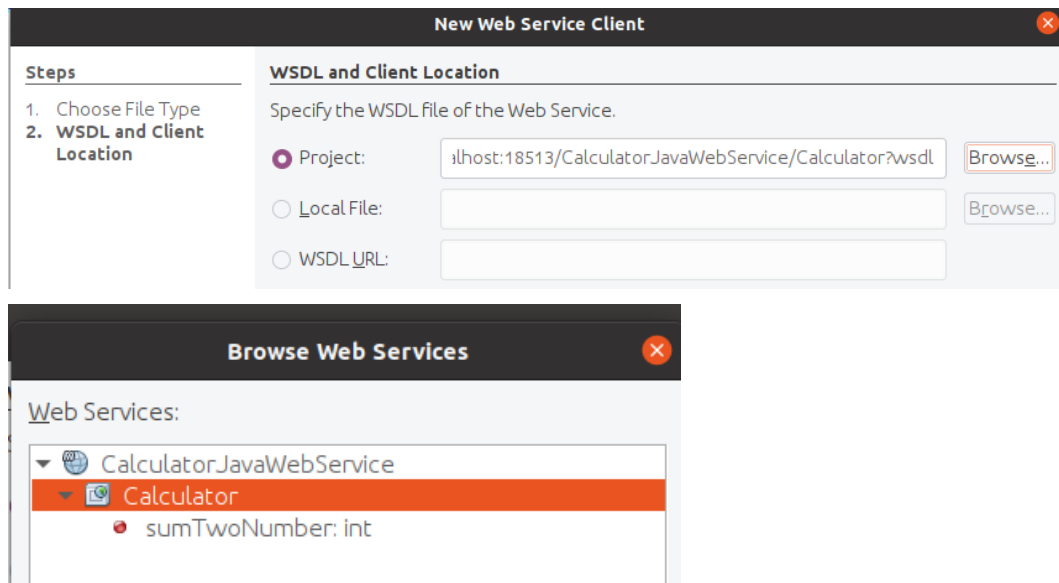
SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body>
    <ns2:sumTwoNumberResponse xmlns:ns2="http://calculatorjavawebsevice.elif.com/">
      <return>85</return>
    </ns2:sumTwoNumberResponse>
  </S:Body>
</S:Envelope>
```

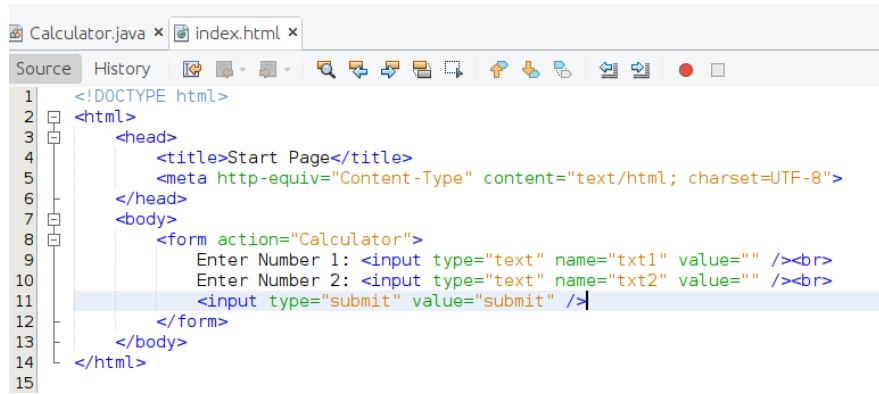
-Web Servis'i Client'a bağlamak yeni web application projesi oluşturulur.



-bağlanılacak web servis seçilir

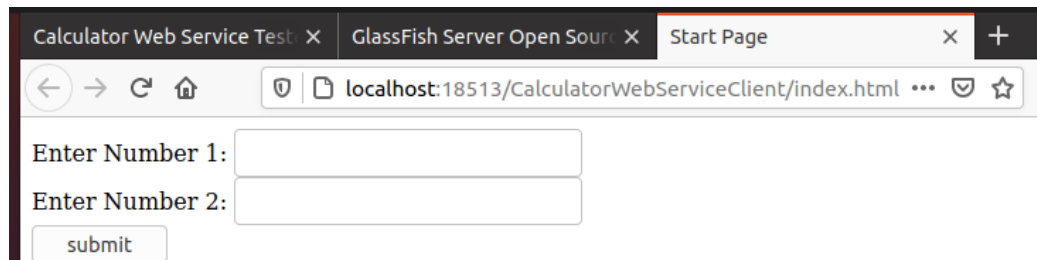


-html sayfasını düzenleme

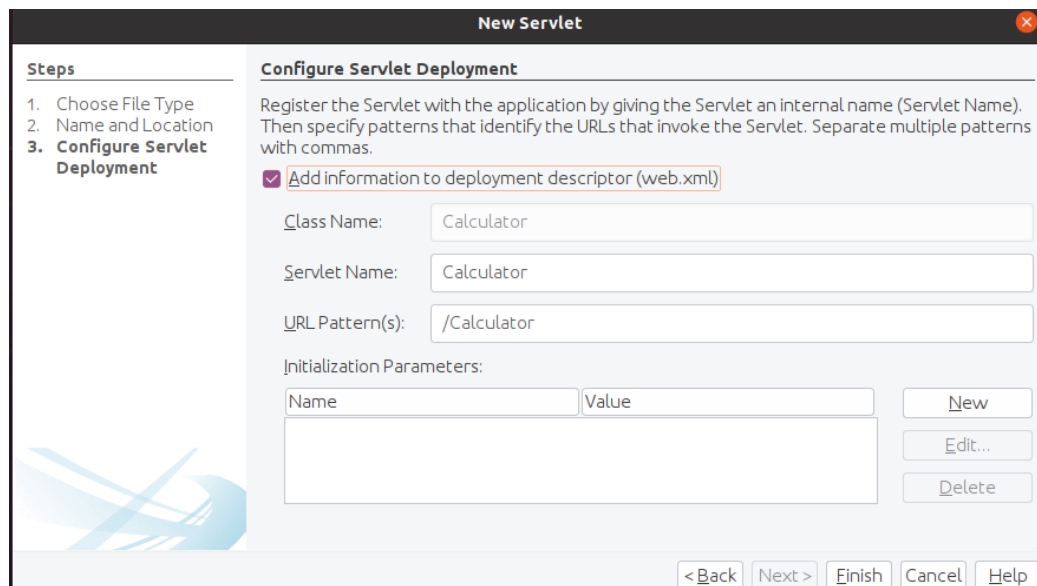
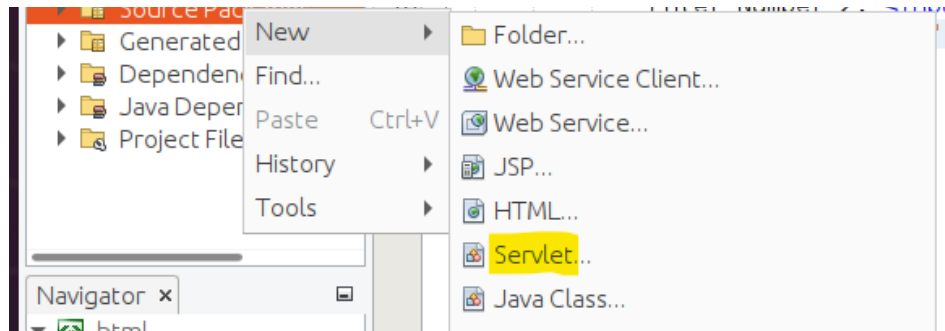


```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Start Page</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6   </head>
7   <body>
8     <form action="Calculator">
9       Enter Number 1: <input type="text" name="txt1" value="" /><br>
10      Enter Number 2: <input type="text" name="txt2" value="" /><br>
11      <input type="submit" value="submit" />
12    </form>
13  </body>
14 </html>
15
```

-sayfanın çıktısı



-servlet oluşturun işlemleri yaptırmak



-servlet'in düzenlenmesi

Calculator_Service: client oluşturulduktan sonra otomatik oluşan class'tır.

```
HttpServlet methods. Click on the + sign on the left to edit the code.
private int sumTwoNumber(int num1,int num2){
    Calculator_Service service = new Calculator_Service();
    com.elif.calculatorjavaweb.service.Calculator port = service.getCalculatorPort();
    return port.sumTwoNumber(num1, num2);
}

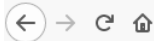
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        int num1, num2;
        num1 = Integer.parseInt(request.getParameter("txt1"));
        num2 = Integer.parseInt(request.getParameter("txt2"));
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Calculator</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>" + sumTwoNumber(num1, num2) + "</h1>");
        out.println("</body>");
    }
}
```

-sonuç

Enter Number 1: 48

Enter Number 2: 37

submit



localhost:18513/CalculatorJavaWebServiceClient/Calculator?txt1=48&txt2=37

85

-postman

http://localhost:18513/CalculatorJavaWebServiceClient/Calculator?txt1=48&txt2=37

POST http://localhost:18513/CalculatorJavaWebServiceClient/Calculator?txt1=48&txt2=37

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

Query Params

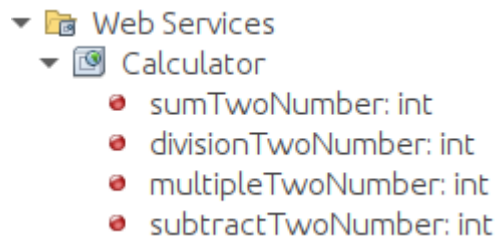
KEY	VALUE
<input checked="" type="checkbox"/> txt1	48
<input checked="" type="checkbox"/> txt2	37
Key	Value

body Cookies Headers (5) Test Results

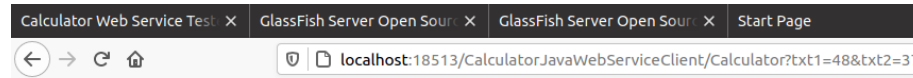
Pretty Raw Preview Visualize HTML

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <title>Servlet Calculator</title>
6 </head>
7
8 <body>
9   <h1>85</h1>
10 </body>
11
12 </html>
```

-başka operationlar eklemek



-sayfanın çıktısı



Result Sum of numbers: 85

Result Subtract of numbers: 11

Result Multiple of numbers: 1776

Result Division of numbers: 1

API(Application Programming Interface)

Web ya da işletim sistemi üzerinde çalışan desktop uygulamaları için kullanılır. Dışarıya açılmış olan veri altyapısıdır. Bir uygulamanın sahip olduğu özelliklere dışarıdan erişim sağlanabilmesidir. Uygulamalara özellik eklemek istendiğinde uygulamayı baştan yazmaya gerek olmaz.

Bu özellikler kullanılırken uygulamanın kendisine ihtiyaç duyulmaz. Bir API kullanırken güvenlik sınırları ve loglamalar vardır.

API'nin Türleri

Open API/ Public API: Kullanımı herkese açıktır.

Internal API/ Private API: Yalnızca dahili sistemler tarafından kullanımdadır.

Partner API: Şirket dışı ama herkese açık olmayan kullanımlar içindir.

Composite API: Birden çok veri veya hizmet API'sini birleştirir.

REST ve SOAP API'lerin nasıl sunulacağını açıklar.

API nasıl çalışır?

Client uygulaması bilgi almak için bir request(API call) başlatır.

Bu istek URI(Uniform Resource Identifier)'de işlenir.

İstekten sonra API programa ya da web server'a çağrı yapar.

Server, istenen bilgileri API'ye yanıt olarak gönderir.

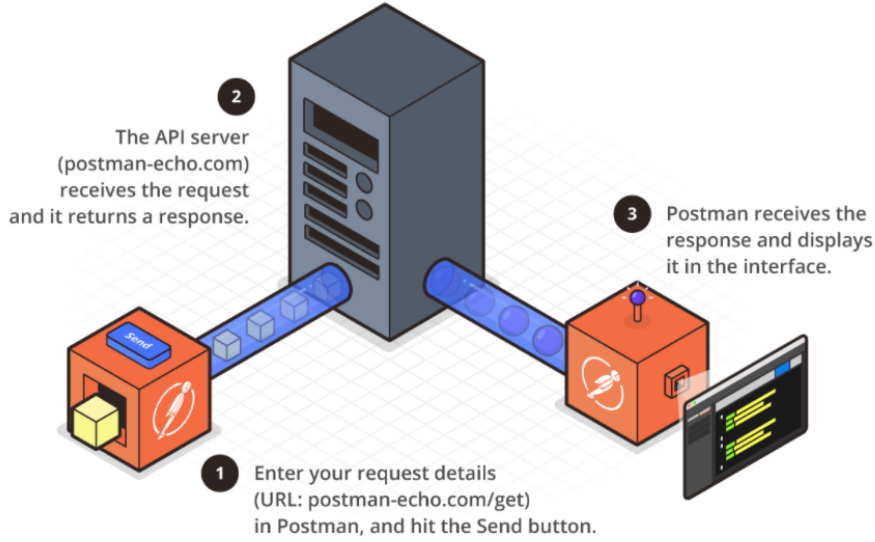
API ilk istekte bulunan uygulamaya cevabı aktarır.

Postman

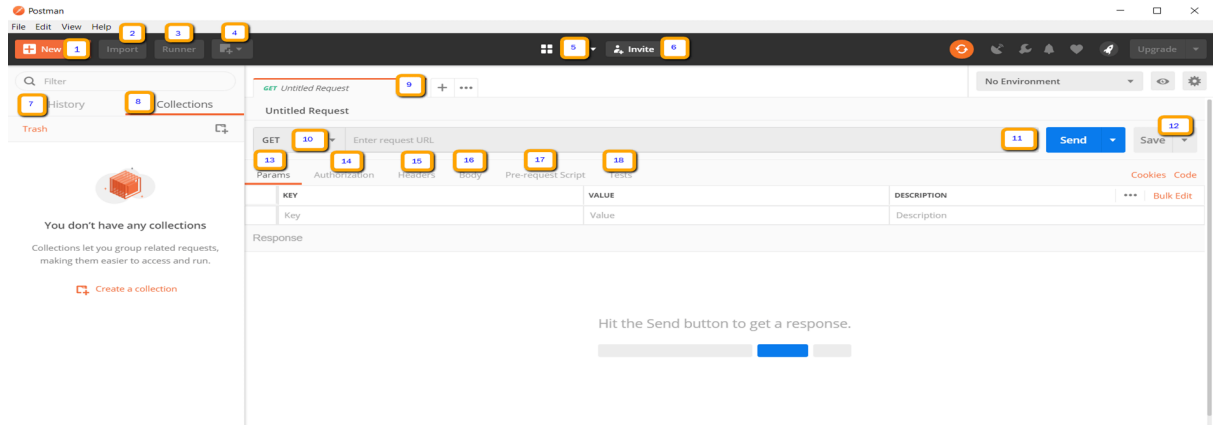
API farklı uygulama yazılımlarının etkileşimini sağlar.

Postman ile API'ler test edilebilir.

RESTful API'lerinden veri almak için kullanılır.



Postman Kullanımı



1. New: Yeni request, collection ya da environment oluşturmayı sağlar.

2. Import: Collection ya da environment'i içe aktarmayı sağlar. Dosya, klasör, link ya da metin ile gerçekleştirilebilir.

3. Runner: Otomasyon testleri burada yapılır.

4. Open New: Yeni sayfa açmak için kullanılır.

5. My Workspace: Bireysel ya da takım olarak çalışabilmek için workspace oluşturur.

- 6. Invite:** Takım üyelerini davet etmek için kullanılır.
- 7. History:** Geçmiş requestler'e bakmak için kullanılır.
- 8. Collections:** Testleri düzenlemek için kullanılır. Alt klasörlere sahip olabilir.
- 9. Request Tab:** Üzerinde çalışılan request'i gösterir.
- 10. HTTP Request:** Postman API testing için kullanılacak method seçilir.
- 11. Request URL:** Endpoint olarak da bilinir, API'nin iletişim kuracağı yerin bağlantısı burada belirlenir.
- 12. Save:** Request üzerinde yapılan değişiklikleri kaydetmeye yarar.
- 13. Params:** İstekler için kullanılan key-value değerler burada yazılır.
- 14. Authorization:** API'lere erişmek için gerekli olan yetkilendirme burada yapılır.
- 15. Headers:** Headerlar burada belirlenir.
- 16. Body:** POST request yapılırken istek ayrıntıları burada belirlenebilir.
- 17. Pre-request Script:** İstekten önce yürütülecek scriptlerdir. Testlerin doğru ortamlarda çalıştırılmasını sağlamaya yarar. Bazı değerler hesaplandıktan sonra request oluşturmak pre-request script'e örnektir.
- 18. Tests:** Gelen verilerin test edilmesi ve requestlerin başka requestlerden aldığı verilerle çalışmasını sağlar. İstek sırasında çalışır.
- Mock Server:** sahte server oluşturmayı sağlar. Client tarafını test etmeye yarar



Response Code'lar

5'e ayrılmıştır.

-1xx: Bilgi

-2xx: Başarılı

-3xx: Yönlendirme

-4xx: Client Hatası

-5xx: Server Hatası

200 OK: Gönderilen isteğin başarılı olduğunu gösterir.

201 Create: Client tarafından resource yaratıldığını gösterir.

202 Accepted: Client tarafından yaratılan resource süresi uzun sürecek ve asenkron olarak devam ederse bu sonuç döndürülür. Sonuç başarılı olmayabilir. Controller'lar 202 kodunu kullanabilir.

204 No Content: Client'ın gönderdiği request de server'ın body göndermediğini gösterir.

301 Moved Permanently: Client'ın istek gönderdiği URI'nin kullanılmadığını gösterir.

Response'ta bu kodla birlikte header location alanında yeni URI'ye yönlendirme yapılır.

304 Not Modified: 204 koduna benzer. Client resource'nin en güncel halindedir.

400 Bad Request: Genel 4xx kodudur.

401 Unauthorized: Korumalı resource için gerekli doğrulama olmadan girilme çalıştığını gösterir.

403 Forbidden: Authorzation sağlanan kullanıcının erişim izni olmadığına döndürülür.

404 Not Found: Client'ın istediği resource'nin bulunamadığını gösterir.

405 Method Not Allowed: İstek yapılan URI'nin o methodu desteklemediğini gösterir.

406 Not Acceptable: Gönderilen istek de header'da Accept alanında yazılan medya tipinde çıktı veremediğini gösterir.

409 Conflict: Client'ın server'a gönderdiği istekte resource'u olmaması gereken bir duruma getirmeye çalıştığını gösterir. Null değer yapmaya çalışmak buna örnektir.

500 Internal Server Error: Server hataları için genel koddur. Server kaynaklı hataları gösterir.

Level 500

500 : Internal Server Error

503 : Service Unavailable

501 : Not Implemented

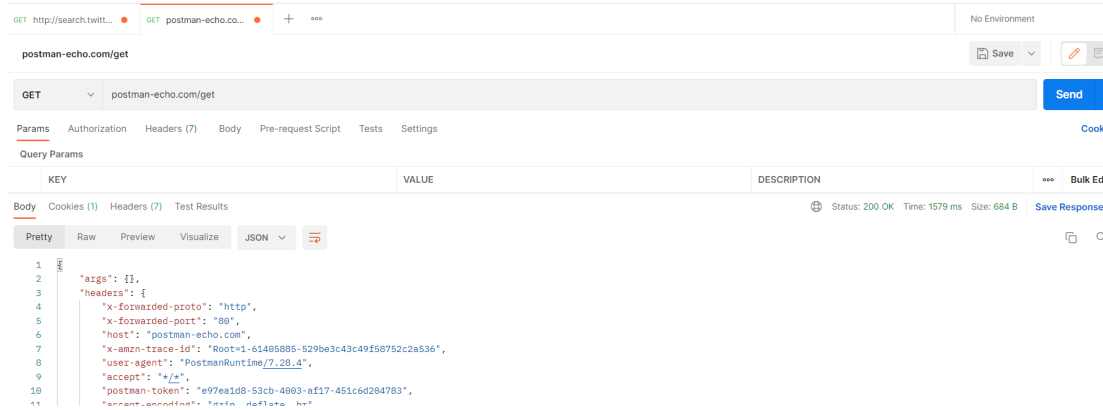
504 : Gateway Timeout

599 : Network timeout

502 : Bad Gateway

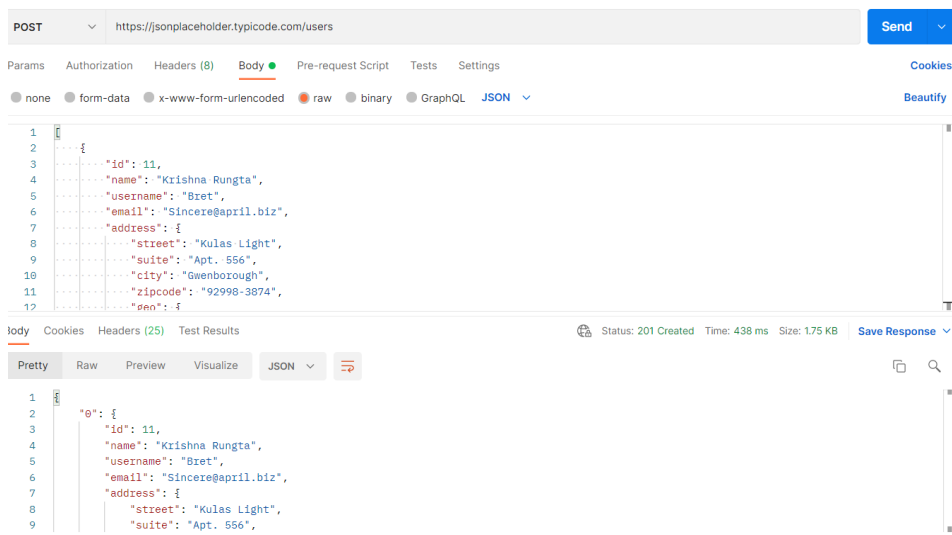
Postman request oluşturmak

İsteğin hangi method ve URL ile yapılacağı seçildikten sonra send butonuna basılır. Gelen output JSON, XML, HTML gibi formatlara dönüştürülebilir.



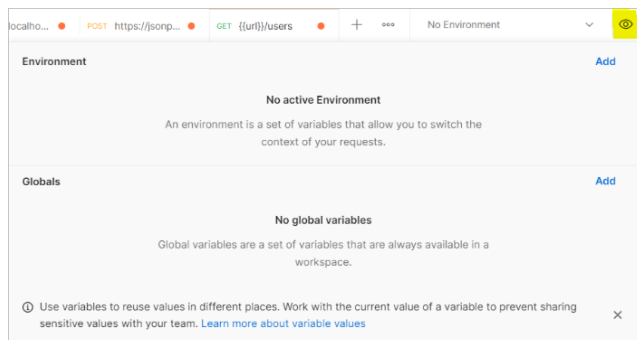
Postman POST Methodu

status kodu 201'dir. body kısmından post için parametre girilir.



Requestleri parametrize etmek

Tekrarlama durumundan kaçınmayı sağlar.



Globals			Save	Export	1
Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals					
VARIABLE	INITIAL VALUE	CURRENT VALUE		Persist All	Reset All
<input checked="" type="checkbox"/> url	https://jsonplaceholder.typicode.com	https://jsonplaceholder.typicode.com			

eklenen global değişken {{url}}/users olarak kullanıldı.

GET

{{url}}/users

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (25)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 1,
3   "name": "Leanne Graham",
4   "username": "Bret",
5   "email": "Sincere@april.biz",
6   "address": {
7     "street": "Kulas Light",
8     "suite": "Apt. 556",
9     "city": "Austin",
10    "zipcode": "78701-8203",
11    "geo": {
12      "lat": 37.31,
13      "lng": -122.23
14    }
15  }
16 }
```

Postman Test Oluşturmak

Arayüzden test kısmına geldikten sonra sağdan code snippetleri eklenebilir.

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
```

istek çağrıldıktan sonra test sonucu da gelir. request'in status'u 200 olduğu için testten geçmiştir.

Body

Cookies

Headers (25)

Test Results (1/1)

Status: 200 OK

All

Passed

Skipped

Failed

PASS

Status code is 200

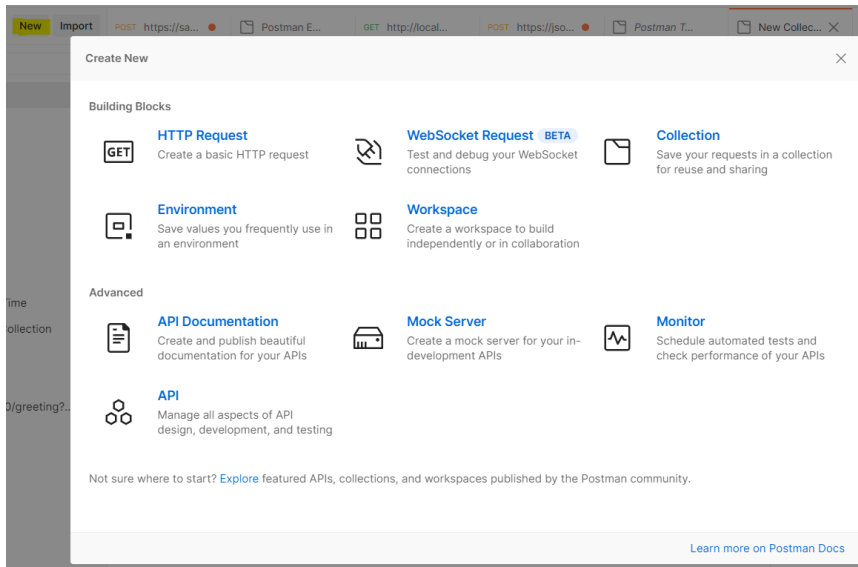
Response body: json value check testi

```
pm.test("Check if user with id1 is Leanne Graham", function () {  
  var jsonData = pm.response.json();  
  pm.expect(jsonData[0].name).to.eql("Leanne Graham");  
});
```

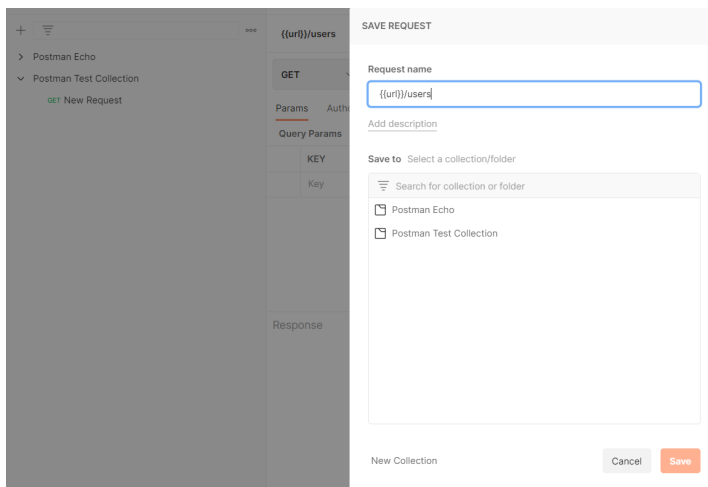
ilk gelen data'nın name'i "Leanne Graham" olduğu için testten geçer.

PASS Check if user with id1 is Leanne Graham

Postman Collection



Oluşturan collection'a request eklenebilir.

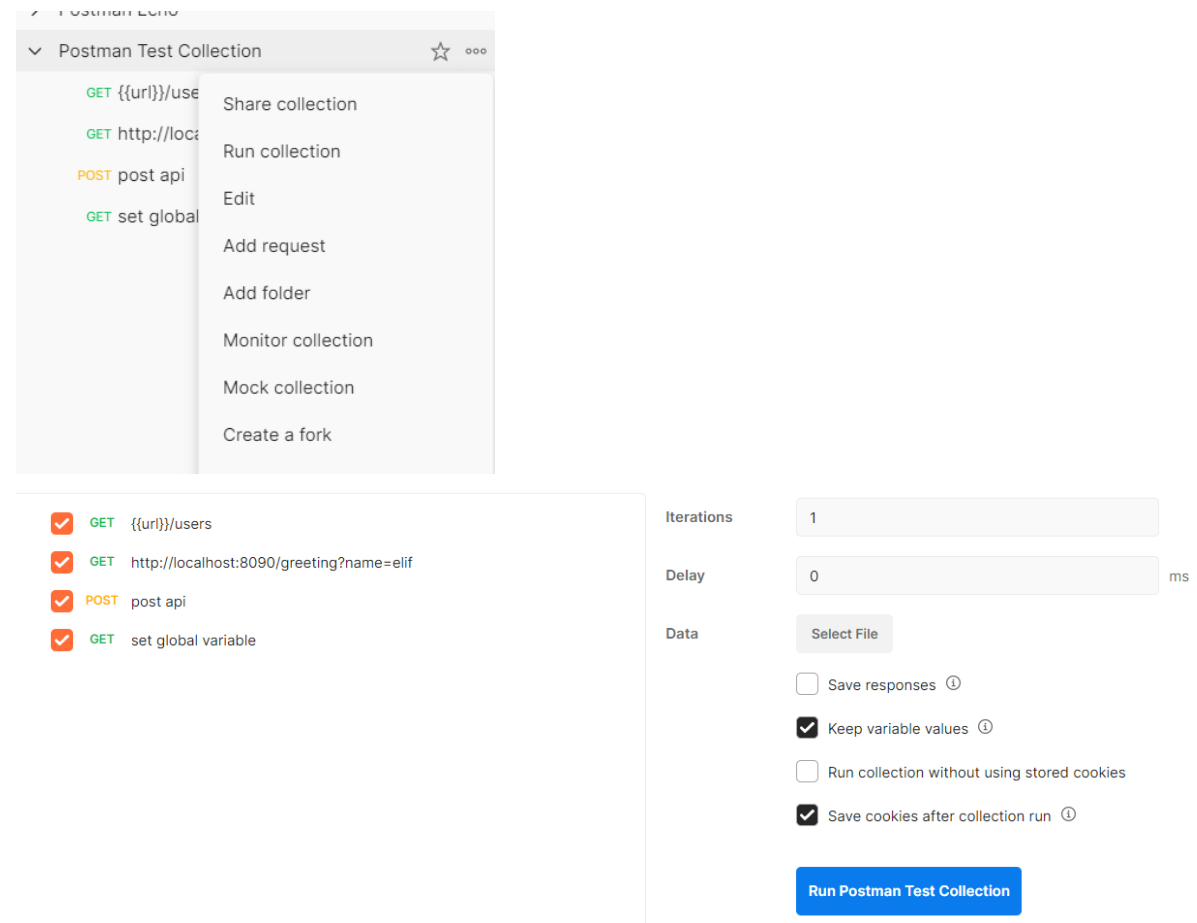


Aynı collection içindeki tüm requestlere aynı test oluşturabilir, kendini tekrar etmekten kaçınılmış olur. Collection edit -> test kısmından yapılır.



Postman Runner

İlgili collection'u seçip seçeneklerden run collection dendiğinde postman runner ile testleri tüm requestler için aynı anda yapmayı sağlar.



-Testlerin çalışması:

Postman Test Collection No Environment, just now

[View Summary](#) [Run Again](#) [New](#) [Export Results](#)

All Tests Passed (6) Failed (0)

Iteration 1

GET	{{url}}/users	{{url}}/users	/ {{url}}/users	200 OK	23 ms	6.785 KB
	Pass	Check if user with id1 is Leanne Graham				
	Pass	Status code is 200				
	Pass	Status code is 200				
GET	http://localhost:8090/greeting?name=elif	http://localhost:8090/greeting?name=elif	/ http://localhost:8090/greeting?name=elif	200 OK	13 ms	196 B
	Pass	Status code is 200				
POST	post api	https://httpbin.org/post	/ post api	200 OK	1501 ms	958 B
	Pass	Status code is 200				
GET	set global variable	https://httpbin.org/uuid	/ set global variable	200 OK	143 ms	282 B
	Pass	Status code is 200				

RUN SUMMARY

		1
▶	GET {{url}}/users	3 0
▶	GET http://localhost:8090/greeting?name=elif	1 0
▶	POST post api	1 0
▶	GET set global variable	1 0

Requestler arası veri aktarımı

Get methodu ile global variable set edilir.

```
var jsonData = pm.response.json();  
pm.globals.set("uuid", jsonData.uuid);
```

Postman Echo

GET https://httpbin.org/... GET https://httpbin.org/... + ...

No Environment

Environment [Add](#)

No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

Globals [Edit](#)

VARIABLE	INITIAL VALUE	CURRENT VALUE
url	https://jsonplaceholder.typicode.com	https://jsonplaceholder.typicode.com
uuid		95fc7002-d0c4-4ad6-952b-88e36c9b5504

https://httpbin.org/uuid

GET https://httpbin.org/uuid

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

```
1 var jsonData = pm.response.json();  
2 pm.globals.set("uuid", jsonData.uuid);
```

Set edilen global variable post methodu ile kullanılır.

"id": "{{uidd}}" olarak kullanılmıştır.

The image shows a Postman interface for a POST request. The URL is `https://httpbin.org/post`. The request body is a JSON object with the following fields:

```
1 {
2   "name": "elif",
3   "email": "elif@example.com",
4   "id": "{{uidd}}"
5 }
```

The response is displayed in the Test Results tab, showing the following JSON structure:

```
13 {
14   "User-Agent": "PostmanRuntime/7.28.4",
15   "X-Amzn-Trace-Id": "Root=1-6141d5d6-296e76194ed3b04a2efba571",
16   "json": {
17     "email": "elif@example.com",
18     "id": "95fc7002-d0c4-4ad6-952b-88e36c9b5504",
19     "name": "elif"
20   },
21   "origin": "94.55.115.230",
22   "url": "https://httpbin.org/post"
23 }
```


Kaynakça

- <https://www.teknologweb.com/ag-mimarileri-nedir>
- <https://www.webhostuzmani.com/server-client-nedir/>
- <https://blog.kmk.net.tr/yazilimlarin-tanismasi-web-servis-nedir>
- <https://ceaksan.com/tr/web-service-nedir>
- <https://www.yusufsezer.com.tr/web-servis-dersleri/>
- <https://medium.com/@kdrcandogan/web-servis-soap-verest-93930908a465>
- https://www.tutorialspoint.com/webservices/what_are_web_services.htm
- <https://medium.com/kodgemisi/rest-api-tasarim-rehberi-2f004a87426d>
- <https://datatracker.ietf.org/doc/html/rfc3986#section-6.2.2.1>
- <http://burcualtinok.com.tr/blog/rest-api-nedir/>
- <https://www.postman.com/postman/workspace/published-postman-templates/documentation/631643-f695cab7-6878-eb55-7943-ad88e1ccfd65?ctx=documentation>
- <https://medium.com/android-türkiye/java-ile-restapi-olusturup-web-service-json-forma-tinda-basmak-80de5da946ad>
- <https://www.cleo.com/blog/knowledge-base-web-services>
- <https://ceaksan.com/tr/rest-soap-api-nedir>
- <https://subscription.packtpub.com/book/application-development/9781788992992/4/ch04lvl1sec26/implementing-client-server-architectures-with-spring>
- <https://www.argenova.com.tr/api-nedir>
- <https://github.com/craigsdennis/intro-to-apis-course/blob/master/course-notes.md>
- <https://github.com/vdespa/introduction-to-postman-course/blob/main/course-notes.md>
- <https://medium.com/huawei-developers-tr/postman-nedir-83eeaa5ed6ac>
- <https://www.freecodecamp.org/news/how-to-test-and-play-with-web-apis-the-easy-way-with-postman/>
- <https://medium.com/better-practices/ui-testing-with-postman-df713eb9788c>
- <https://github.com/craigsdennis/intro-to-apis-course/blob/master/course-notes.md>
- <https://github.com/vdespa/introduction-to-postman-course/blob/main/course-notes.md>
- <https://medium.com/kodcular/postman-apiler-arasi-veri-aktarimi-dc557afb8f0e>
- <https://metinalniacik.medium.com/javada-anotasyon-olusturma-2dd47967dad5>
- <https://medium.com/@fsengul081/spring-boot-ile-restful-web-servis-olusturma-65ea340ad9c6>
- <http://benimzamanim.com/soap-ve-rest-farklari>
- <https://www.guru99.com/postman-tutorial.html>
- <https://www.ibm.com/cloud/learn/api>