1. HTML Nedir?

◈ Açılımı:

HTML, İngilizce "HyperText Markup Language" ifadesinin kısaltmasıdır. Türkçesi: "Üst Düzey Metin İşaretleme Dili" olarak çevrilebilir.

Ne İşe Yarar?

HTML, bir web sayfasının iskeletini oluşturur.

Web Sayfalarındaki Rolü

Bir web sayfası, HTML olmadan var olamaz. Sayfanın başlığı, paragrafları, listeleri, tabloları, görselleri ve butonları gibi tüm öğeler HTML etiketleriyle tanımlanır.

Temel HTML Örnek:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Benim İlk Web Sayfam</title>
  </head>
  <body>
    <h1>Merhaba Dünya!</h1>
    <p>Bu, HTML ile yazılmış bir paragraftır.</p>
    <a href="https://www.google.com">Google'a git</a>
  </body>
</html>
```

2. CSS Nedir?

Açılımı:CSS, İngilizce “Cascading Style Sheets”, yani Türkçesiyle “Basamaklı Stil Sayfaları” anlamına gelir.

Ne İşe Yarar?

CSS, HTML ile oluşturulan web sayfalarının görsel görünümünü ve düzenini şekillendirmek için kullanılır.

HTML ile İlişkisi

CSS tek başına kullanılmaz; mutlaka HTML veya benzeri bir işaretleme diliyle birlikte çalışır.

```
.uyari {
  color: red;
  font-weight: bold;
}
```

Lütfen bu alanı boş bırakmayınız!

Görsel Tasarıma Katkısı

CSS sayesinde:

Web sayfası mobil cihazlara uyumlu hale getirilebilir (responsive tasarım).

Arka planlara resim veya geçiş efekti eklenebilir.

Butonlara hover (üzerine gelince) animasyonları yapılabilir.

Grid ve Flexbox gibi yapılarla düzenli, modern ve profesyonel bir görünüm elde edilir.

Temel CSS Örneği:

```
body {
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}

h1 {
  color: navy;
  text-align: center;
}

p {
  font-size: 16px;
  line-height: 1.5;
}
```

3. Endpoint Nedir?

Temel Tanım

Endpoint, yazılım geliştirme dünyasında özellikle API (Application Programming Interface) mimarilerinde kullanılan bir kavramdır ve Türkçesiyle "uç nokta" anlamına gelir.

Yazılım ve API Mimarilerinde Anlamı

Endpoint ile İstemci-Sunucu İletişimi Nasıl Kurulur?

İstemci, belirli bir işlem (örneğin kullanıcı bilgisi çekme) için uygun endpoint'e HTTP isteği gönderir.

Bu istek genellikle şu bilgileri içerir:

URL (örnek: /api/products/)

HTTP metodu (GET, POST…)

Gerekirse veri (örneğin JSON formatında kullanıcı bilgileri)

Sunucu, gelen isteği karşılar, işleme koyar ve ilgili cevabı döner (örneğin 200 OK, 404 Not Found, 201 Created gibi HTTP durum kodları ile birlikte).

GET /api/products/5

```
{
  "id": 5,
  "name": "Kablosuz Mouse",
  "price": 129.99,
  "inStock": true
}
```

4. ORM (Object Relational Mapping) Nedir?

Açılımı ve Tanımı

ORM, İngilizce "Object Relational Mapping" ifadesinin kısaltmasıdır. Türkçesi "Nesne-İlişkisel Eşleme" olarak çevrilebilir.

ORM, yazılım geliştiricilerin veritabanlarıyla doğrudan SQL sorguları yazmadan çalışmasını sağlayan bir tekniktir. Nesne yönelimli programlama ile ilişkisel veritabanları arasındaki farkı köprüleyen bir katmandır.

Ne İşe Yarar?

SQL bilmeden veritabanı işlemleri yapılabilir.

Kodun okunabilirliği ve sürdürülebilirliği artar.

Yazılım daha güvenli ve test edilebilir hale gelir.

Nasıl Bir Köprü Kurar?

Geliştirici, bir sınıf tanımlar:

```
public class Urun {
    public int Id { get; set; }
    public string Ad { get; set; }
    public decimal Fiyat { get; set; }
}
```

ORM aracı, bu sınıfı otomatik olarak veritabanındaki bir tabloya eşleştirir:

Sınıf adı → Tablo adı (Urun → Urunlar)

Özellikler → Sütunlar (Ad, Fiyat vs.)

SQL sorgusu yazmak yerine:

```
context.Urunler.Add(new Urun { Ad = "Kulaklık", Fiyat = 299 });
context.SaveChanges();
```

ORM arka planda şu SQL sorgusunu üretir:

```
INSERT INTO Urunler (Ad, Fiyat) VALUES ('Kulaklık', 299);
```

Popüler ORM Kütüphaneleri

ORM Aracı	Kullanıldığı Teknoloji
Entity Framework	.NET / C#
Hibernate	Java
Sequelize	Node.js
SQLAlchemy	Python
Django ORM	Django / Python
TypeORM	TypeScript
Doctrine	PHP

5. HTTP Metodları

HTTP (HyperText Transfer Protocol), istemci (örneğin tarayıcı veya mobil uygulama) ile sunucu arasındaki iletişimi yöneten bir protokoldür. Bu iletişim sırasında kullanılan "HTTP metodları", istemcinin sunucuya ne yapmak istediğini belirtir. Yani bu metodlar, bir veriyi istemek mi, göndermek mi, güncellemek mi, yoksa silmek mi istediğimizi açıklar.

1. GET

Ne işe yarar?

Sunucudan veri istemek için kullanılır. Genellikle veri okuma amaçlıdır.

Ne zaman kullanılır?

Bir ürün listesi, kullanıcı bilgisi, blog yazısı gibi verilerin istemciye gösterilmek istendiği durumlarda.

Örnek:GET /api/urunler/12 HTTP/1.1

2. POST

Ne işe yarar?

Sunucuya yeni veri göndermek için kullanılır. Genelde veri oluşturma işlemlerinde tercih edilir.

Ne zaman kullanılır?

Yeni bir kullanıcı kaydı oluşturmak, ürün eklemek gibi işlemlerde.

Örnek:POST /api/kullanicilar HTTP/1.1

Body: { "ad": "Ahmet", "email": "ahmet@mail.com" }

3. PUT

Ne işe yarar?

Var olan bir veriyi tamamen güncellemek için kullanılır.

Ne zaman kullanılır?

Bir ürünün adını, fiyatını veya tüm bilgilerini topluca güncellemek gerektiğinde.

Örnek:PUT /api/urunler/12 HTTP/1.1

Body: { "ad": "Yeni Ürün", "fiyat": 99 }

4. DELETE

Ne işe yarar?

Sunucudaki bir veriyi silmek için kullanılır.

Ne zaman kullanılır?

Kullanıcı, sipariş, ürün gibi kayıtların sistemden silinmesi gerektiğinde.

Örnek:DELETE /api/urunler/12 HTTP/1.1

5. OPTIONS

Ne işe yarar?

Belirli bir endpoint’in hangi HTTP metodlarını desteklediğini öğrenmek için kullanılır.

Ne zaman kullanılır?

Özellikle CORS (Cross-Origin Resource Sharing) ayarlarında, tarayıcı önce bu isteği gönderip sunucunun hangi işlemlere izin verdiğini kontrol eder.

Örnek:OPTIONS /api/kullanicilar HTTP/1.1

6. HEAD

Ne işe yarar?

GET ile aynıdır, ancak yalnızca başlık bilgilerini getirir. Yanıt gövdesi (body) yoktur.

Ne zaman kullanılır?

Sadece içeriğin var olup olmadığını kontrol etmek için. Örneğin bir dosya mevcut mu diye.

Örnek:HEAD /api/urunler/12 HTTP/1.1

7. PATCH

Ne işe yarar?

Var olan bir verinin bir kısmını güncellemek için kullanılır. PUT'tan farkı tüm veriyi değil sadece değişen alanları günceller.

Ne zaman kullanılır?

Örneğin sadece bir kullanıcının email adresi değiştirilmek istenirse.

Örnek:PATCH /api/kullanicilar/5 HTTP/1.1

Body: { "email": "yeni@mail.com" }

8. CONNECT

Ne işe yarar?

HTTP üzerinden bir tünel kurmak için kullanılır. Genellikle HTTPS bağlantılarında kullanılır.

Ne zaman kullanılır?

Proxy sunucular üzerinden şifreli bağlantı oluşturmak gerektiğinde.

Örnek:

Web tarayıcısı bir HTTPS sitesine bağlanırken CONNECT metodunu kullanır: CONNECT www.ornek.com:443 HTTP/1.1

9. TRACE

Ne işe yarar?

HTTP isteğinin sunucuya nasıl ulaştığını izlemek için kullanılır. Genellikle hata ayıklamada kullanılır.

Ne zaman kullanılır?

Geliştiriciler, HTTP isteğinin herhangi bir proxy veya sunucu tarafından değiştirilip değiştirilmediğini kontrol etmek için TRACE kullanır.

Örnek:TRACE /api/kullanicilar HTTP/1.1

6. Swagger ve Postman Nedir?

Yazılım geliştirirken, özellikle RESTful API oluşturma ve test etme sürecinde, geliştiricilerin işini kolaylaştıran bazı araçlara ihtiyaç vardır. Bu noktada öne çıkan iki araç: Swagger ve Postman'dır.

Swagger Nedir?

Tanım:

Swagger (şimdiki adıyla OpenAPI), API'lerin nasıl çalıştığını açık bir şekilde belgelemek, tasarlamak, test etmek ve doğrulamak için kullanılan bir araç setidir.

Amacı:

API'nin ne yaptığını, hangi endpoint'lerin bulunduğunu, hangi HTTP metodlarının desteklendiğini ve her endpoint'in ne tür veri beklediğini geliştiricilere ve testçilere açıklamak için kullanılır.

Swagger'ın Kullanıldığı Yerler:

API dökümantasyonu hazırlamak.

Backend ile frontend ekipleri arasında ortak bir API anlaşması oluşturmak.

Canlı olarak API'yi test etmek (arayüzden GET, POST istekleri göndermek).

Örnek Swagger Arayüzü:paths:

```
  /api/urunler/{id}:
    get:
      summary: Ürün bilgisi getirir
      parameters:
        - in: path
          name: id
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: Başarılı
```

Postman Nedir?

Tanım:

Postman, API’lere istek (request) göndermek, cevapları (response) görmek ve otomatik test senaryoları yazmak için kullanılan bir masaüstü ve web tabanlı araçtır.

Amacı:

Geliştiricilerin API’leri manuel veya otomatik olarak test etmelerine olanak sağlar. Hem frontend hem backend geliştiriciler tarafından kullanılır.

Postman’ın Kullanıldığı Yerler:

API’ye manuel olarak GET, POST, PUT, DELETE vb. istekler göndermek.

Token ile authentication işlemlerini test etmek.

Test senaryoları yazmak (örneğin: "Kullanıcı eklendiğinde HTTP 201 dönmeli").

Örnek İstek:

Method: POST

URL: http://localhost:5000/api/urunler

Body:

```
{
  "ad": "Kahve Makinesi",
  "fiyat": 899.99
}
```

Gerçek API Test Örneği

Diyelim ki bir kitaplık uygulamasında, yeni bir kitap eklemek istiyorsun.

Swagger Üzerinden Test:

POST /api/kitaplar endpoint'ini Swagger UI arayüzünden bul.

Body kısmına aşağıdaki JSON'u yaz:

```
{
  "kitapAdi": "1984",
  "yazar": "George Orwell"
}
```

Execute tuşuna basarak isteği gönder.

Sağda çıkan yanıta göre işlemin başarılı olup olmadığını gör.

Postman Üzerinden Test:

Postman’ı aç → Yeni istek oluştur.

Method: POST

URL: http://localhost:5000/api/kitaplar

Body → Raw → JSON:

```
{
  "kitapAdi": "1984",
  "yazar": "George Orwell"
}
```