Temel HTTP Method Çeşitleri

- **GET:** Get Metodu sunucudaki veri çekme işlemlerinde kullanılır. Sunucunun istek sırasında herhangi bir veri istememesi durumlarında kullanılır.

```
@GET("posts")
fun getPosts(): Call<List<Post>>
```

- **POST:** Bu method ile sunucuya veri yazdırılabilir, veriye ulaşmak için ise sunucunun bizden istediği parametreler gönderilir ve veriye ulaşılabilir. Veri hem URL hem de mesaj gövdesinde gönderilebilir. Bu yüzden güvenlik gerekçeleri nedeniyle POST yöntemi kullanılması öneriliyor. Yeni bir kaynak oluşturma, mevcut kaynağı güncelleme, veri çekme işlemleri örnek verilebilir

```
@POST("posts")
fun createPost(@Body post: Post): Call<Post>
```

- **PUT:** Bu metod ile servis sağlayıcı üzerindeki bir kaynağı güncelleyebilirsiniz. Hangi kaynağı güncelleyecekseniz o kaynağın id'sini göndermek zorunludur.

```
@PUT("posts/{id}")
fun putPost(@Path("id") id: Int, @Body post: Post):
Call<Post>
```

- **DELETE:** Bu metod ile sunucudaki herhangi bir veriyi silebilirsiniz.

```
@DELETE("posts/{id}")
fun deletePost(@Path("id") id: Int): Call<Void>
```

Parametre Annotasyonları

- **Query:** Retrofit, isteklere yönelik sorgu parametrelerini tanımlamak için @Query ek açıklamasını kullanır. Sorgu parametreleri metot parametrelerinden önce tanımlanır. Annotation'da, URL'ye eklenecek sorgu parametresi adını iletiriz

  ```
  @GET("/api/users")
  fun getUsers(
                  @Query("per_page") pageSize: Int,
                  @Query("page") currentPage: Int
  ): Call<UsersApiResponse>
  ```

- **Path:** Retrofit 2'de yol parametreleri @Path ek açıklaması ile gösterilir. Ayrıca yöntem parametrelerinden önce gelirler. Bir URL yolu segmentinde değiştirme olarak adlandırılırlar. Yol parametreleri null olamaz

  ```
  @GET("/api/users/{id}")
  fun getUser(@Path("id") id: Long): Call<UserApiResponse>
  ```

- **Field:** Verileri form-urlencoded olarak gönderir. Bunun için yönteme eklenmiş bir @FormUrlEncoded ek açıklaması gerekir. Field parametresi yalnızca POST ile çalışır. Field zorunlu bir parametre gerektirir. Field'ın isteğe bağlı olduğu durumlarda, bunun yerine @Query kullanabilir ve null değer geçebiliriz.

  ```
  @FormUrlEncoded
  @POST("/")
  suspend fun example(
          @Field("name") name: String,
          @Field("occupation") occupation:
  String):Response<ResponseBody>
  ```

- **Body: N**esneleri istek gövdesi olarak gönderir.

  @POST("users/register")
  suspend fun register(@Body registerRequest: RequestBody): String

Bazı HTTP Durum Kodları

- **100:** Bu durum kodu, sunucunun isteği hala işlemekte olduğunu ve tamamlanmasını beklediğini belirtir. Bu durumda, sunucu belirli bir süre boyunca işlemi sürdürebilir ve daha sonra sonuç döndürebilir

- **200:** İsteğin başarıyla işlendiği ve tarayıcı ile sunucu arasında her şeyin yolunda olduğu anlamına gelir. Bu durumda isteğin istenen kaynakla ilgili doğru sonuçlarla döndürüldüğü anlaşılır.

- **300:** Bu durum kodu, sunucunun birden fazla olası kaynağa sahip olduğunu ve tarayıcının seçim yapması gerektiğini ifade eder. Sunucu, farklı kaynakları tarayıcıya sunar ve tarayıcı, hangi kaynağı kullanmak istediğini seçer.

- **401:** Bu durum kodu, kullanıcının erişmek istediği kaynağın güvenlik doğrulaması gerektirdiğini ve kullanıcının geçerli kimlik doğrulama bilgilerine sahip olmadığını belirtir. Kullanıcı, kimlik doğrulama yapmadan kaynağa erişemez.

- **403:** Bu durum kodu, kullanıcının ilgili kaynağa erişimin yasaklandığını belirtir. Kullanıcı, güvenlik veya izin sorunları nedeniyle kaynağa erişemez.

- **404:** Bu durum kodu, istenen kaynağın sunucuda bulunmadığını ifade eder. En sık karşılaşılan HTTP durum kodlarından biridir ve genellikle "Sayfa Bulunamadı" hatası olarak bilinir.

- **500:** Bu durum kodu, sunucudaki bir sorun nedeniyle isteğin tamamlanamadığını ifade eder. Sunucu tarafında bir hata veya başka bir sorun olduğunda bu durum kodu döndürülür.