Retrofit, Android ve Java için popüler bir HTTP istemci kütüphanesidir. Bu kütüphane, RESTful API'lerle iletişim kurmak için kullanılır ve çeşitli HTTP metodlarını ve durum kodlarını destekler.
Retrofit içinde kullanılan HTTP metodları ve bunların kullanım alanları:

1. GET: Sunucudan veri almak için kullanılır. Örneğin, bir kaynağı almak veya bir sorguyu gerçekleştirmek için kullanılır.
- Sunucudan veri almaya kullanılır.
- Veri okuma işlemleri için idealdir.
- Cacheleme için uygundur.

Cacheleme: GET metoduyla alınan veriler cache'e kaydedilebilir ve sonraki isteklerde internet bağlantısı olmadan kullanılabilir.

2. POST: Sunucuya veri göndermek için kullanılır. Örneğin, yeni bir kayıt oluşturmak veya bir formu göndermek için kullanılır.
- Sunucuya veri göndermeye kullanılır.
- Yeni veri oluşturma veya güncelleme işlemleri için kullanılır.
- Cacheleme için uygun değildir.

3. PUT: Sunucuda var olan bir kaydı güncellemek için kullanılır. Tüm kaydı değiştirmek için veya belirli alanları güncellemek için kullanılabilir.
- Mevcut bir kaynağı güncellemek için kullanılır.
- Tam güncelleme işlemleri için kullanılır.

4. DELETE: Sunucudan bir kaydı silmek için kullanılır. Belirli bir kaydı veya kaynakları silmek için kullanılabilir.
- Bir kaynağı sunucudan silmek için kullanılır.
- Kalıcı veri silme işlemleri için kullanılır.

En çok kullanılan Retrofit annotasyonları ve bunların interface içindeki kullanımları ve açıklamaları şunlardır:

1. @GET: Bir GET isteği yapmak için kullanılır. Kullanımı genellikle bir metodun üzerine yazarak yapılır.GET metodu ile bir endpoint'e istek gönderir.

```
@GET("endpoint")
Call<Response> getEndPoint();
```

2. @POST: Bir POST isteği yapmak için kullanılır.POST metodu ile bir endpoint'e veri gönderir.

```
@POST("endpoint")
Call<Response> postEndPoint(@Body RequestBody body);
```

3. @PUT: Bir PUT isteği yapmak için kullanılır.PUT metodu ile bir endpoint'teki kaynağı günceller.

```
@PUT("endpoint/{id}")
Call<Response> putEndPoint(@Path("id") int id, @Body RequestBody body);
```

4. @DELETE: Bir DELETE isteği yapmak için kullanılır.DELETE metodu ile bir endpoint'teki kaynağı siler.

```
@DELETE("endpoint/{id}")
Call<Void> deleteEndPoint(@Path("id") int id);
```

5. @Path: Dinamik olarak URL'yi oluşturmak için kullanılır. Değişken parçaları URL'ye ekler.(URL'deki parametreleri belirler.)

6. @Query: Query parametreleri eklemek için kullanılır.

7. @Body: İstek gövdesini belirtmek için kullanılır.(Gönderilecek veriyi belirler.)

8. @Header: İstek başlıklarını belirtmek için kullanılır.

En çok kullanılan HTTP durum kodları şunlardır:

1. ***200 OK:*** İstek başarılı oldu ve istenen kaynak başarıyla alındı veya işlendi.

2. ***500 Internal Server Error:*** Sunucu bir istekte bulunurken bir hata ile karşılaştı ve bu nedenle isteği tamamlayamadı.

3. ***401 Unauthorized:*** İstemci, yetkilendirme gerektiren bir kaynağa erişmeye çalıştı, ancak kimlik doğrulaması başarısız oldu veya kimlik doğrulaması yapılmadı.

4. ***403 Forbidden:*** İstemci, kaynağa erişim iznine sahip değil veya kaynağa erişmeye çalışmak yasaklandı.

5. ***404 Not Found:*** İstenen kaynak bulunamadı.

Diğer Önemli Kodlar:

- **301 Moved Permanently:** Kaynak kalıcı olarak yeni bir yere taşındı.
- **302 Found:** Kaynak geçici olarak yeni bir yere taşındı.
- **400 Bad Request:** İstekte hata var.
- **405 Method Not Allowed:** İstekte kullanılan metod bu endpoint için geçerli değil.
- **503 Service Unavailable:** Sunucu şu anda kullanılabilir değil.

Bu durum kodları, sunucunun yanıtının isteğin durumunu tanımlamasına yardımcı olur. Başarılı veya başarısız bir işlemin gerçekleştiğini ve gerekirse uygun işlem yapılması gerektiğini belirtirler.