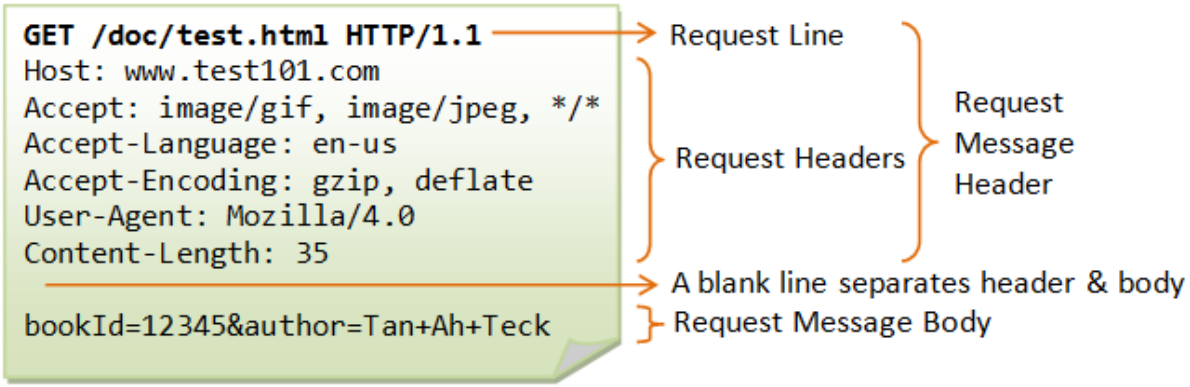
ÖDEV-6

Retrofit içerisinde kullanılan HTTP Method çeşitleri

Anroid geliştirme sürecinde en çok tercih edilen http isteklerini yönetmek ve Restful API'larla bağlantı kurmak için Retrofit kütüphanesi kullanılmaktadır. Retrofit kullanıcılara, servis interfacelerini ve istek yapılandırmalarını ayrı dosyalarda tanımlama şansı sunar bu sayede Retrofit modüler yapı sağlayarak öne çıkmaktadır. Bununla birlikte kullanım kolaylığıyla açık kaynaklı networking kütüphanesi sunmaktadır.

HTTP Methodları

Kaynağa istekte bulunurken yapılacak işlemi bir fiil ile belirtiyoruz. Resim.1'deki Request line de gösterilen GET fiili http methoduna denk gelmektedir. Bu method sayesinde ulaştığımız kaynakta yapmak istediğimiz işlemleri belirtiriz.



Resim.1

GET: En çok kullanılan http methodudur. Kaynaktan verileri almak ve listelemek için kullanılır. Kullanıcıların tamamını listelemek istediğimizde GET işlemini kullanırız. Bir başka kullanım alanı da id'si 1 olan kullanıcıya ait bilgiyi görmek istediğimizde de GET methodunu kullanırız.

POST: Kaynakta yeni bir entety-veri oluşturmak için kullanılır. Örnek olarak bir uygulamada yeni bir kullanıcı oluşturmak için o kullanıcıya ait bilgileri girerek POST methodu ile göndeririz. Bu sayede o kaynakta yeni bir kullanıcı bilgisi eklenmiş olur.

PUT: Belirli bir kaynaktaki verinin tamamını veya bir kısmını değiştirilmek için kullanılan methodtur. Önemli bir nokta da şudur; PUT methodunu gönderdiğimizde gönderilen mesajın body'sinde verilerin tamamını göndermek gerekmektedir. Örnek olarak var olan kullanıcı da bir değişiklik yapmak istiyoruz. Bu kullanıcının name ve age şeklidne iki alanı bulunmaktadır.Age'i 23 olarak değiştirmek istiyorum. Yalnızca {"age": 40} bu bilgiyi göndermek soruna yol açacaktır. { "name": "Şevval", "age": 30} şeklinde tüm alanlara ait bilgiler gönderilerek değiştirme işlemi yapılmalıdır. Çünkü PUT methodu değiştirmek istenilen veriyi silip yerine yeni bir veri yazar.

PATCH: Belirli bir kaynaktaki verilerin bir kısmının değiştirilmesi için kullanılan metodtur. PUT methodundaki örnekte Age i değiştirmek istiyorum. Yalnızca age alanını {"age": 50} değiştirip gönderebiliriz. PATCH methodu değişiklik yapmak istenilen yerdeki değeri değiştirir.

DELETE: Belirli bir kaynaktaki verilerin silinmesi için kullanılan methodtur.

Retrofit İçerisinde Kullanılan Annotationslar

HTTP requestlerine yönelik GET, POST, PUT gibi annotationlar ve @Query, @Path, @Body,@Patch, @Header gibi özel annotationlar bulunmaktadır.

@GET("users") annotation'ı, GET HTTP metodunu belirtir. Bu metodun users kaynağına yapılacağını belirtir. Tüm kullanıcıları getirmek için bu istek kullanılacaktır.

```
interface ApiService {

    // Tüm kullanıcıları getirmek için GET isteği
    @GET("users")
    fun getUsers(): Call<List<User>>
```

@GET("users/{id}") annotation, belirtilen URL adresine kullanıcı ID'sine {id} göre gerçekleştirileceğini belirtir.*@Path* /users/{id} ile request atılmak istenildiğinde @Path kullanılarak id verisi setlenebilir.

```
// Belirli bir kullanıcıyı ID'ye göre getirmek için GET isteği
@GET("users/{id}")
fun getUserById(@Path("id") userId: Int): Call<User>
```

@POST("users") annotation, belirtilen URL adresine yeni bir kullanıcı oluşturulmasını sağlar. *@Body* annotasyonu, POST isteği sırasında gönderilen verilerin HTTP isteğinin gövdesine eklenmesini sağlar. createUser fonksiyonu, kullanıcı nesnesini HTTP isteğinin gövdesine ekler ve bu isteği belirtilen URL'ye gönderir.

```
// Yeni bir kullanıcı oluşturmak için POST isteği
@POST("users")
fun createUser(@Body user: User): Call<User>
```

@PUT("users/{id}") annotation, belirtilen kullanıcı ID'sine göre kullanıcının güncellenmesini sağlar.

```
 // Var olan bir kullanıcıyı güncellemek için PUT isteği
 @PUT("users/{id}")
 fun updateUser(@Path("id") userId: Int, @Body user: User): Call<User>
```

@PATCH e kullanıcının kısmen güncellenmesini sağlamaktadır.

```
// Var olan bir kullanıcının kısmen güncellenmesi için PATCH isteği
@PATCH("users/{id}")
fun partialUpdateUser(@Path("id") userId: Int, @Body userPartial: UserPartial)
```

@DELETE annotasyonu, kullanıcının silinmesini sağlar.

```
// Belirli bir kullanıcıyı ID'ye göre silmek için DELETE isteği
@DELETE("users/{id}")
fun deleteUser(@Path("id") userId: Int): Call<Void>
```

@Header atılacak olan requestde headerde iletilmesi gereken herhangi bir bilgi varsa kullanılır. Kimlik doğrulama için bu annotasyonu kullanarak isteğe "Authorization" başlığı eklenebilir.

```
@GET("user")
fun getUser(@Header("Authorization") authToken: String): Call<User>
```

@Query annotasyonu, sorgu parametreleri eklemek için kullanılır. Endpointe bağlı olarak iletilecek olan query parametrelerinde kullanılır. Bir kullanıcının adını filtrelemek için kullanım şekli:

```
@GET("users")
fun getUsersByName(@Query("name") name: String): Call<List<User>>
```

HttpStatus Kodlar

Sunucunun tarayıcıya verdiği üç haneli cevaplar HTTP durum kodları ya da HTTP status codes olarak adlandırılır. İlgili durum kodları bazen bir hata olduğunu bildirirken bazen de sayfanın herhangi bir sorun olmadan açıldığını ifade edebilir. Bu nedenle HTTP durum kodlarının daima hata olarak değerlendirilmesi doğru değildir.

1xx: Tarayıcı tarafından gönderilen isteğin sunucuya ulaştığını ve işlemin başladığını bildiren bilgilendirme kodlarını ifade eden durum kodlarıdır.

2xx: Tarayıcı tarafından gönderilen isteğin sunucuya ulaştığını, anlaşıldığını ve başarılı olduğunu ifade eden durum kodlarıdır.

3xx: Erişilmek istenen kaynağın başka bir kaynağa taşındığını ve bir yönlendirmenin söz konusu olduğunu ifade eden durum kodlarıdır.

4xx: İsteğin yerine getirilemediğini, ilgili web sayfasına ya da web sitesine ulaşılamadığını ifade eden durum kodlarıdır.

5xx: Tarayıcı tarafından gönderilen isteğin başarıyla sunucuya ulaştığını fakat sunucu tarafındaki sorunlar nedeniyle isteğin yerine getirilemediğini ifade eden durum kodlarıdır.

En çok kullanılan HttpStatus kodlar

200 Durum Kodu (Başarılı)
İsteğin başarıyla işlendiği ve tarayıcı ile sunucu arasında her şeyin yolunda olduğu anlamına gelir.

301 Durum Kodu (Kalıcı Yönlendirme)
Kaynağın kalıcı olarak farklı bir yere taşındığını belirtir. Tarayıcı, gelecekte bu kaynağı yeni konumunda bulabilir ve isteklerini doğrudan oraya yönlendirebilir.

302 Durum Kodu (Geçici Yönlendirme)
Tarayıcının Post, Put ve Delete methodlarıylq talep ettiği kaynağın başka bir URL'de bulunduğunu belirtir.

403 Durum Kodu (Erişim İzni Sorunu)
Kullanıcının ilgili kaynağa erişimin yasaklandığını belirtir.

404 Durum Kodu (Bulunamadı)
İstenen kaynağın sunucuda bulunmadığını ifade eder. En sık karşılaştığımız HTTP durum kodlarından biridir. Genellikle "Sayfa Bulunamadı" hatası olarak bilinir.