ÖDEV 6

Retrofit, REST tabanlı bir web servisi aracılığıyla veri almayı ve yüklemeyi kolay ve hızlı hale getiren bir kütüphanedir.

Retrofit bu işlemleri yaparken bazı metodlar kullanır. Bu metodlardan birkaçı GET, POST, PUT ve DELETE metodlarıdır.

GET Metodu:

GET metodu, sunucudan veri almak için kullanılır. Genellikle veritabanından veri çekmek veya bir kaynağın detaylarını görüntülemek için kullanılır.

Kullanım Alanları:

- Bir liste veya belirli bir öğeyi almak.
- Arama sonuçlarını getirmek.
- Statik veya dinamik içerikleri almak.

NOT: Verilen örneklerde base url olarak https://dummyjson.com/ kullanılmıştır.

Kullanım örneği:

```
interface IDummyService {
    @GET("auth/me")
    fun getUser(@Header("Authorization") token: String): Call<UserModel>
}
```

POST Metodu:

POST metodu, sunucuya yeni bir veri göndermek veya sunucuda bir işlem yapmak için kullanılır. Genellikle veritabanına yeni kayıt eklemek için kullanılır.

Kullanım Alanları:

- Yeni kullanıcı oluşturmak.
- Form verisi göndermek.
- Bir kaynağa bağlı alt kaynak oluşturmak.

Kullanım örneği:

```
interface IDummyService {
    @POST("auth/login")
    fun userLogin(@Body userLogin : UserLoginModel) : Call<UserModel>
}
```

PUT Metodu:

PUT metodu, mevcut bir kaynağı güncellemek veya değiştirmek için kullanılır. Belirli bir kaynağın tamamen yenilenmesini sağlar.

Kullanım Alanları:

- Kullanıcı bilgilerini güncellemek.
- Bir öğenin tüm detaylarını değiştirmek.

Kullanım örneği:

```
interface IDummyService {
    @PUT("products/{id}")
    suspend fun updateProductTitle(
        @Path("id") id: Int,
        @Body product: ProductUpdateRequest
    ): Call<ProductResponse>
}
```

Gerekli dataclass tanımlamaları yapıldığında kodumuz çalışacaktır.

DELETE Metodu

DELETE metodu, sunucudan bir kaynağı silmek için kullanılır.

Kullanım Alanları:

- Kullanıcı hesabını silmek.
- Bir öğeyi veya kaynağı kaldırmak.

Kullanım örneği:

```kotlin
interface IDummyService {
    @DELETE("products/{id}")
    fun deleteProduct(@Path("id") id: Int): Call<Unit>
}
```

Retrofit içerisinde en çok kullanılan: Annotations'lar

Annotations (anotasyonlar), Java ve Kotlin programlama dillerinde kullanılan özel işaretlerdir. Bu işaretler, sınıflar, metodlar, değişkenler ve diğer programlama elemanları hakkında ek bilgiler sağlar. Anotasyonlar, derleyiciye veya çalışma zamanı ortamına belirli davranışları dikte etmek için kullanılır. Retrofit gibi kütüphanelerde, anotasyonlar özellikle HTTP isteklerinin yapılandırılması ve yönetilmesi için yaygın olarak kullanılır.

@Body

@Body anotasyonu, HTTP isteğinin gövdesini belirtmek için kullanılır. Genellikle POST ve PUT isteklerinde kullanılır. Gövde verisi JSON formatında olur ve sunucuya gönderilir.

```kotlin
interface IDummyService {
    @PUT("products/{id}")
    suspend fun updateProductTitle(
        @Path("id") id: Int,
        @Body product: ProductUpdateRequest
    ): Call<ProductResponse>
}
```

Bu örnekte, ProductUpdateRequest sınıfından gelen product nesnesini JSON formatına dönüştürüp, bu dönüştürülmüş veriyi gövde kısmında gönderiyor.

@Header

@Header anotasyonu, HTTP isteği başlıklarını belirtmek için kullanılır. Dinamik veya sabit başlıklar eklemek için kullanılır. Özellikle kimlik doğrulama veya içerik tipi gibi başlıkları eklemek için kullanışlıdır.

```
interface IDummyService {
    @GET("auth/me")
    fun getUser(@Header("Authorization") token: String): Call<UserModel>
}
```

Bu örnekte, başlıkta yer alan token ile sunucuya isteği göndererek doğrulama işlemini gerçekleştiriyoruz.

@Field ve @FormUrlEncoded

@Field anotasyonu, HTTP isteği gövdesinde form verisi göndermek için kullanılır. @FormUrlEncoded anotasyonu ile birlikte kullanılır ve form verisini URL kodlanmış şekilde gönderir.

```
interface IDummyService {
    @FormUrlEncoded
    @POST("users/edit")
    fun updateUser(@Field("id") id: Int, @Field("name") name: String): Call<UserModel>
}
```

Bu fonksiyon, id ve name alanlarını URL kodlanmış form verisi olarak POST isteği gövdesine ekler.

@Query

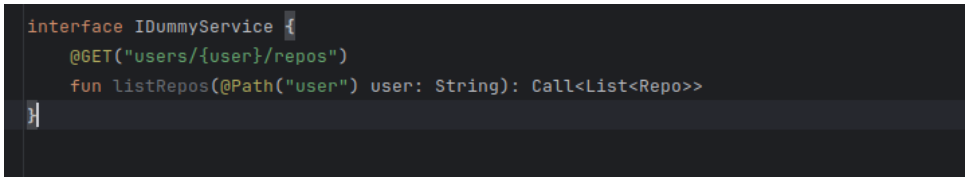
@Query anotasyonu, URL'nin sorgu parametrelerini belirtmek için kullanılır. Genellikle isteğe bağlı parametreler veya URL'nin sonuna eklenmesi gereken parametreler için kullanılır.

```
interface IDummyService {
    @GET("users")
    fun listUsers(@Query("page") page: Int): Call<List<User>>
}
```

Bu örnekte, listUsers fonksiyonu, "users" endpoint'ine bir GET isteği gönderir. @Query("page") parametresi, URL'nin sonuna "?page=value" şeklinde eklenir. Örneğin, listUsers(2) çağrısı "users?page=2" URL'sine bir istek gönderir.

@Path

@Path anotasyonu, URL'nin belirli bir kısmının dinamik olarak değiştirilmesini sağlar. Bu, özellikle bir kaynağın kimliğine göre veya başka bir dinamik değere göre URL'lerin oluşturulması gerektiğinde kullanışlıdır.



{user}: Bu, URL'nin dinamik bir parçasıdır ve @Path("user") anotasyonu ile belirtilen değeri alır. Fonksiyon çağrılırken, user parametresi bu segmentin yerine geçer.

En çok kullanılan HttpStatus kodlar nedir ve ne şe yarar.

HTTP durum kodları, istemci ve sunucu arasındaki iletişimin sonucunu belirten üç haneli sayılardır. Her kodun belirli bir anlamı vardır ve genellikle beş kategoriye ayrılırlar: 1xx (Bilgi), 2xx (Başarı), 3xx (Yönlendirme), 4xx (İstemci Hatası) ve 5xx (Sunucu Hatası). İşte en çok kullanılan HTTP durum kodlarından bazıları ve ne işe yaradıkları:

200 OK

İstek başarıyla tamamlandı ve sunucudan istenen kaynak döndürüldü.

Kullanım Alanları:

Başarılı GET, POST, PUT veya DELETE istekleri.

400 Bad Request

İstek kötü biçimlendirilmiş veya geçersiz.

Kullanım Alanları:

İstemciden gelen hatalı istekler.

401 Unauthorized

İstek, kimlik doğrulama gerektiriyor ve geçerli kimlik bilgileri sağlanmamış.

Kullanım Alanları:

Kimlik doğrulama gerektiren işlemler.

403 Forbidden

İstemcinin, istenen kaynağa erişim izni yok.

Kullanım Alanları:

Yetkisiz erişim girişimleri.

404 Not Found

İstenen kaynak bulunamadı.

Kullanım Alanları:

Geçersiz URL'ler.

500 Internal Server Error

Sunucuda beklenmedik bir hata oluştu.

Kullanım Alanları:

Genel sunucu hataları.

503 Service Unavailable

Sunucu şu anda kullanılamıyor (aşırı yüklenmiş veya bakımdan geçiyor olabilir).

Kullanım Alanları:

Sunucu geçici olarak hizmet veremediğinde.

504 Gateway Timeout

Sunucu, başka bir sunucudan yanıt alırken zaman aşımına uğradı.

Kullanım Alanları:

Geçiş sunucularındaki zaman aşımı hataları.

Mustafa Koçer

Kaynakça:

- https://square.github.io/retrofit/
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- https://medium.com/@9cv9official/what-are-get-post-put-patch-delete-a-walkthrough-with-javascripts-fetch-api-17be31755d28