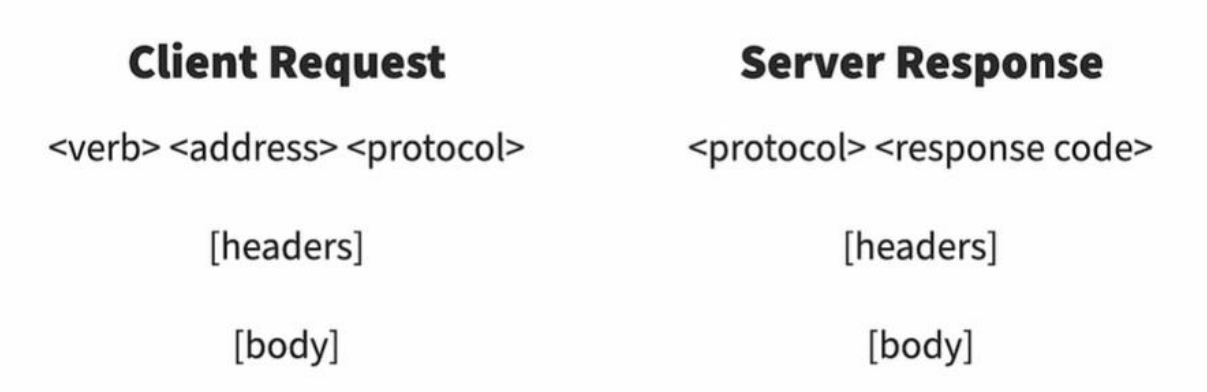
Retrofit

Retrofit, Android ve Java için RESTful web servislerinin(hizmetlerinin) kullanımını kolaylaştırmayı amaçlayan, Square (Dagger, Okhttp) tarafından geliştirilen, tür açısından güvenli bir HTTP istemcisidir(client). Uygulamanız ile REST API Web servisiniz arasındaki iletişimi Retrofit aracılığı ile sağlayabilirsiniz.

HTTP Method Çeşitleri:

HTTP (Hypertext Transfer Protocol) istemci ile host için ihtiyaç duyulan protokolleri sağlar. Bu protokoller bilgiye nasıl erişileceğini, bilginin nasıl transfer edileceğini, gösterileceğini ve belirli bir komut geldiğinde nasıl hareket edileceğini belirler.
HTTP mehodları, bir istemcinin (örneğin internet uygulamanızın) bir web hizmetinin yürütülmesini talep ettiği prosedürler için standartlaştırılmış formatlardır. İstemci isteği ve sunucu yanıtı şeklini alırlar. İstek, üç bölümden oluşan bir istek satırıyla başlar; get veya post gibi bir HTTP method, göreceli adres, temel adres gerekli header olarak eklenir ve kullanılan protokol genellikle HTTP/1.1'dir. Daha sonra sunucu ana bilgisayarı, yetkilendirme ve içerik linki gibi bilgileri içeren içerik başlıklarına sahiptir. Sonraki boş bir satır ve ardından isteğe bağlı bir istek mesajı gövdesi gelir. HTTP sunucusu yanıtı, ilk satırın bir protokol sürümünü ve ardından isteğin sonucunu açıklayan yanıt kodunu içermesi dışında benzer bir formatı izler. Yanıtta istek adresi ve HTTP methodu yok. Herhangi bir HTTP yönteminin önemli bir kısmı, bu isteğin ilişkili methodlarıdır. Methodların seçenekleri şunlardır: get, head, post, put, patch, delete, trace, options ve connect. Ancak Retrofit yalnızca ilk altı methodu destekler.



Bu Methodlar Nedir ve Kullanım Alanları Nelerdir?

Her yöntemin, istek yöntemini ve ilgili URL'yi sağlayan bir HTTP annotationına sahip olması gerekir.

GET

Sunucudan veri istemek için kullanılır. Hedef kaynağın durumunun bir temsilini aktarmasını ister. GET istekleri yalnızca verileri almalı ve başka bir etkisi olmamalıdır. Bir web sayfasını görüntülemek ,bir API'den veri almak, bir dosya indirmek gibi kullanım alanları vardır. Örneğin, bir web sitesine girdiğinizde, sunucuya bir GET isteği gönderirsiniz ve sunucu size web sayfasının HTML kodunu geri gönderir.

Endpointler proje süresince kullanılacak olan ve farklı sınıflardan da erişimi kolay olacak şekilde interface yapısına uygun şekilde ayarlanır. Ek olarak, yapılan API çağrılarında dönüş verisi her zaman Call<Article> gibi parametreleştirilmiş bir Call<T> nesnesidir.

ApiInterface.java

```java
import info.yazilimdersi.sampleretrofit.model.Article;
import retrofit2.Call;
import retrofit2.http.GET;

public interface ApiInterface {
    @GET("articles")
    Call<List<Article>> getArticles();
}
```

POST

Sunucuya yeni veri göndermek için kullanılır. Bu metodla istek parametreleri hem URL içinde hem de mesaj gövdesinde gönderilebilir. Bir formu göndermek, yeni bir kullanıcı kaydetmek, yeni bir ürün oluşturmak gibi kullanım alanları vardır. Örneğin, bir e-posta gönderdiğinizde, sunucuya bir POST isteği gönderirsiniz ve sunucu e-postayı alıcının posta kutusuna gönderir.

```kotlin
@POST("/posts")
suspend fun createPost(
    @Body user: User
):Response<User>
```

PUT

Bu metodu kullanarak servis sağlayıcı üzerindeki kaynağı (kaynağın ID'sini göndermek şartıyla) güncelleyebilirsiniz veya yeni bir kaynak oluşturabilirsiniz. Bir kullanıcının bilgilerini güncellemek, bir ürünün fiyatını güncellemek, bir blog yazısını güncellemek gibi kullanım alanları vardır. Örneğin bir blog yazısını düzenlediğinizde, sunucuya bir PUT isteği gönderirsiniz ve sunucu blog yazısını günceller.

```kotlin
interface APIService {
    // ...

    @PUT("/api/users/2")
    suspend fun updateEmployee(@Body requestBody: RequestBody): Response<ResponseBody>

    // ...
}
```

DELETE

Bu metod ile sunucu üzerindeki herhangi bir veri silinebilir. Bu sebeple, Delete Metod'nun kullanılabilir olması güvenlik açısından problem oluşturabilir. Bir kullanıcıyı silmek, bir ürünü silmek, bir blog yazısını silmek gibi kullanım alanları vardır. Örneğin, bir dosyayı

sildiğinizde, sunucuya bir DELETE isteği gönderirsiniz ve sunucu dosyayı siler.

```
interface APIService {
    // ...

    @DELETE("/typicode/demo/posts/1")
    suspend fun deleteEmployee(): Response<ResponseBody>

    // ...
}
```

HEAD

Bir kaynağın başlık bilgilerini almak için kullanılır. GET metoduna benzer, fakat bu metodu kullanarak istek yaptığımız zaman dönen cevapta mesaj gövdesi bulunmaz. Bir dosyanın boyutunu kontrol etmek, bir HTTP isteğinin önbelleğe alınmış olup olmadığını kontrol etmek, bir sunucunun desteklediği HTTP methodlarını kontrol etmek gibi kullanım alanları vardır. Örneğin, bir dosyayı indirmeden önce, dosyanın boyutunu kontrol etmek için sunucuya bir HEAD isteği gönderebilirsiniz.

```
@HEAD("/3/Cloud")
Call<CloudV3> head();
```

PATCH

Bir kaynağın bir kısmını güncellemek için kullanılır. Bir kullanıcının profil resmini güncellemek, bir ürünün açıklamasını güncellemek, bir blog yazısına yorum eklemek gibi kullanım alanları vardır. Örneğin, bir kullanıcının profil resmini güncellemek istediğinizde, tüm kullanıcı bilgilerini güncellemek yerine sadece profil resmini güncellemek için bir PATCH isteği kullanabilirsiniz.

```
@PATCH("posts/{id}")
suspend fun patchPost(
    @Path("id") id : Int,
    @Body user: User
):Response<User>
```

Retrofit İçerisinde En Çok Kullanılan Annotations'lar:

@Body

Java nesnelerini istek gövdesi olarak gönderir. POST ile iletilecek olan body verisinde kullanılır. Örnekte, **@Body** Annotation'ı ile **Article** nesnesi, isteğin gövdesindeki veriyi temsil eder.

```
public interface ApiService {
   @POST("articles")
   Call<Void> createArticle(@Body Article article);
}
```

@Path

Endpoint linkinde iletilmesi gereken herhangi bir parametre olduğunda kullanılır Örneğin, bir URL'de dinamik olarak değişen bir parametre bulunuyorsa, bu parametreyi belirtmek için kullanılır ./article/{id} ile request atılmak istenildiğinde @Path kullanılarak id verisi setlenebilir. **@Path** Annotation'ı ile **{id}** placeholder'ının **articleId** değişkeni ile doldurulması sağlanır.

```
public interface ApiService {
    @GET("articles/{id}")
    Call<Article> getArticle(@Path("id") int articleId);
}
```

@Query

@Query ile basitçe bir yöntem parametresi ve türü açıklayan bir sorgu parametresi adı ekleyebiliriz. Endpointe bağlı olarak iletilecek olan query parametrelerinde kullanılır. Query parametreleri, URL'de **?** işaretiyle başlar ve anahtar-değer çiftleri şeklinde gelir. Örneğin; /articles?order=id&isfeatured=true tarzı requestleri sağlar. **@Query** Annotation'ı ile **order** ve **isfeatured** parametreleri belirtilmiştir.

```
public interface ApiService {
    @GET("articles")
    Call<List<Article>> getArticles(@Query("order") String order, @Query("isfeatured")
boolean isFeatured);
}
```

@Header

Atılacak olan requestde headerde iletilmesi gereken herhangi bir bilgi varsa kullanılır. Örneğin, kimlik doğrulama bilgileri veya API anahtarları gibi bilgileri header'da taşımak için kullanılır. **@Header** Annotation'ı ile **Authorization** başlığına **authToken** değerinin eklenmesi sağlanır.

```
public interface ApiService {
    @GET("articles")
    Call<List<Article>> getArticles(@Header("Authorization") String authToken);
}
```

En Çok Kullanılan HttpStatus Kodlar Nedir ve Ne İşe Yarar?

HttpStatus kodları, istremci ve sunucu arasındaki iletişimi sağlar, istemcinin yaptığı bir HTTP isteğine sunucunun nasıl yanıt verdiğini gösterir. Bu kodlar, isteğin durumunu ve işlemin başarı durumunu gösterir. İşte en çok kullanılan HttpStatus kodları ve anlamları:

200 OK: İstek başarılı bir şekilde gerçekleştirildi. Sunucu, istenen içeriği başarıyla döndürdü. Bu kod, başarılı GET isteklerinin yanıtında yaygın olarak kullanılır. En ideal durum kodu budur ve kullanıcı, sitenizi olması gerektiği gibi görüntüler.

201 Created: Kaynak başarıyla oluşturuldu. Genellikle POST isteklerinin başarılı olması durumunda kullanılır. Yeni bir kaynak oluşturulduğunda veya bir kaynak güncellendiğinde kullanılabilir.

301 Moved Permanently: 301, en önemli durum kodlarından biridir. Bir URL adresindeki sayfanın, kalıcı olarak başka bir sayfaya yönlendirilmesi durumunda, kullanıcının da otomatik olarak yeni sayfayı görebilmesini sağlayan bir koddur. 301 ile yönlendirilen yeni sayfanın içeriği, eski URL'deki ile ilgili ya da benzer olur ve bu sayede ziyaretçi kaybı minimuma iner.

400 Bad Request: İstek hatalı veya geçersiz. Sunucu, isteği anlayamadı veya işleyemedi. Bu genellikle kullanıcı tarafından gönderilen verilerin geçersiz olduğu durumlarda görülür.

401 Unauthorized: İstemci kimlik doğrulaması gerekiyor. Kullanıcı kimlik doğrulaması yapmadan önce işlem yapamaz. Kullanıcı girişi yapmadığı veya yetki eksikliği olduğunda bu kod döndürülür.

403 Forbidden: İstemci, kaynağa erişim yetkisine sahip değil. Kimlik doğrulaması yapılsa bile erişim izni yok. Sunucu, istemciyi reddediyor ve kaynağa erişim izni vermiyor.

404 Not Found: Kullanıcıların en sık karşılaştığı ve hata bildiren HTTP durum kodlarından biridir. İstek yapılan kaynak bulunamadı. Sunucu, istenilen kaynağı bulamadı. Özellikle geçersiz URL'ler veya silinmiş kaynaklar için bu kod döndürülür.

405 Method Not Allowed: İstek yapılan method, sunucu tarafından desteklenmiyor. Örneğin, bir GET isteği yapılması gereken bir endpoint'e POST isteği yapıldığında bu kod döndürülür.

500 Internal Server Error: Sunucu tarafında bir hata oluştu. Sunucu, isteği işlerken bir hata oluştu ve işlem tamamlanamadı. Genellikle sunucu tarafında yazılım hataları veya yapılandırma sorunları nedeniyle oluşur. Sayfanın görüntülenmesine engel olan sunucu hatası çözülene kadar ilgili sayfaya erişebilmek söz konusu değildir.

503 Service Unavailable: Sunucu geçici olarak hizmet veremiyor. Genellikle sunucunun aşırı yüklenmesi veya bakım nedeniyle oluşur. Sunucunun geçici olarak erişilemez olduğunu belirtir. Ayrıca bant genişliğine yönelik siber saldırılar da bu duruma sebep olabilir.

Bengisu Şahin

Referanslar

https://square.github.io/retrofit/?source=post_page-----e10827a1f556----------

https://www.linkedin.com/learning/android-development-retrofit-with-kotlin/http-methods-and-how-retrofit-uses-them-in-your-android-app?resume=false

https://en.wikipedia.org/wiki/HTTP

https://www.digitalocean.com/community/tutorials/retrofit-android-example-tutorial

https://yazilimdersi.info/makaleler/detay/113/android-ile-retrofit-http-kutuphanesi-kullanimi.

https://johncodeos.com/how-to-make-post-get-put-and-delete-requests-with-retrofit-using-kotlin/

https://www.youtube.com/watch?v=OXdronPF8sw

https://www.ideasoft.com.tr/http-durum-kodlari-nedir-anlamlari-nelerdir/