RETROFİT'TE KULLANILAN HTTP METOD ÇEŞİTLERİ

@GET:

Tanım: Belirtilen bir URI'ye; HTTP, **@GET** isteği gönderir ve sunucudan belirtilen URI'den veri veya verileri (Json verileri, Xml verileri, Html verileri, vs.) alır ve URI parametrelerindeki endpointlere göre sorgu yapılabilir.

Kullanım Örnekleri: Sunucuya yeni veri göndermek için kullanılır. Web formu verileri, dosya yükleme işlemleri gibi veri gönderme işlemlerinde, veri oluşturma ve ekleme işlemleri için kullanılır.

@POST:

Tanım: @POST isteği gönderildiğinde, istemci sunucuya bir kaynak oluşturmak için gereken veriyi gönderir. Sunucu, bu veriyi alır, işler ve yeni bir kaynak oluşturur. Sonuç olarak, sunucu bir yanıt döndürür.

Kullanım Örnekleri: Örneğin, yeni bir kullanıcı kaydı oluşturmak, ürün siparişi vermek, yorum göndermek ve bir sunucuya veri göndermek için kullanılabilir.

@PUT:

Tanım: Belirtilen bir URI'deki var olan kaynağı değiştirmek veya güncellemek için HTTP **@PUT** isteği gönderir.

Kullanım Örnekleri: Veri tabanındaki bir kaydı güncellemek, bir sunucudaki bir dosyayı değiştirmek, bir nesnenin özelliklerini güncellemek, bir kaynağın durumunu değiştirmek için kullanılabilir.

@DELETE:

Tanım: Belirtilen bir URI'deki var olan kaynağı silmek için HTTP **@DELETE** isteği gönderir.

Kullanım Örnekleri: Bir veri tabanındaki kaydı silmek, bir sunucudaki dosyayı silmek bir nesneyi silmek için kullanılabilir.

@GET ve @POST ARASINDAKİ FARKLAR:

	GET	POST
Geri Butonu, Sayfa Yenileme Butonu	Varılan sayfa kaybolmaz.	Varılan sayfa için Re-Submit popup'ı görüntülenir.
Yer İmleme	Mümkün	Mümkün değil
Önbelleğe Kaydedilme	Mümkün	Mümkün değil
Kodlama Tipi(Encoding Type)	application/x-www-form-urlencoded[1]	application/x-www-form-urlencoded[1] ya da multipart/form-data[2]
Tarayıcı Geçmişi	Değişkenler tarayıcı geçmişinde kalır	Değişkenler tarayıcı geçmişine kaydedilmez
Karakter Limiti	2048 karakter	Limit yok
Veri Tipi Kısıtlaması	Sadece ASCII karakterleri izinlidir	Sınırlama yoktur(Binary tipli veri izinlidir)
Güvenlik	GET POST'a göre daha az güvenlidir, çünkü gönderilen veri URL'nin bir parçasını teşkil eder.	POST GET'e göre biraz daha güvenlidir, çünkü gönderilen değişken ve değerleri tarayıcı geçmişinde ya da sunucu log'larında depolanmaz.
Görünürlük	URL'deki veri herkes tarafından görünürdür.	Veri URL'de gösterilmez.

RETROFİT İÇERİSİNDE EN ÇOK KULLANILAN ANNOTATİONS'LAR

@BODY ANNOTATİON'U:

Tanım: @BODY annotation'u HTTP isteği gövdesine (request body) bir nesne eklemek için kullanılır. Bu anotation, isteğin gövdesine dönüştürülmesi gereken bir veri nesnesini belirtmek için kullanılır.

```
interface UserService {
    @POST("/users")
    fun createUser(@Body user: User): Call<UserResponse>

}
```

Bu örnekte, createUser fonksiyonu, POST metodu kullanılarak /users endpoint'ine bir User nesnesini gönderir. @Body anotasyonu, user parametresinin isteğin gövdesine yerleştirileceğini belirtir. Bu, Retrofit kütüphanesi tarafından otomatik olarak JSON formatına dönüştürülür ve HTTP isteği gönderilir.

@HEADER ANNOTATİON'U:

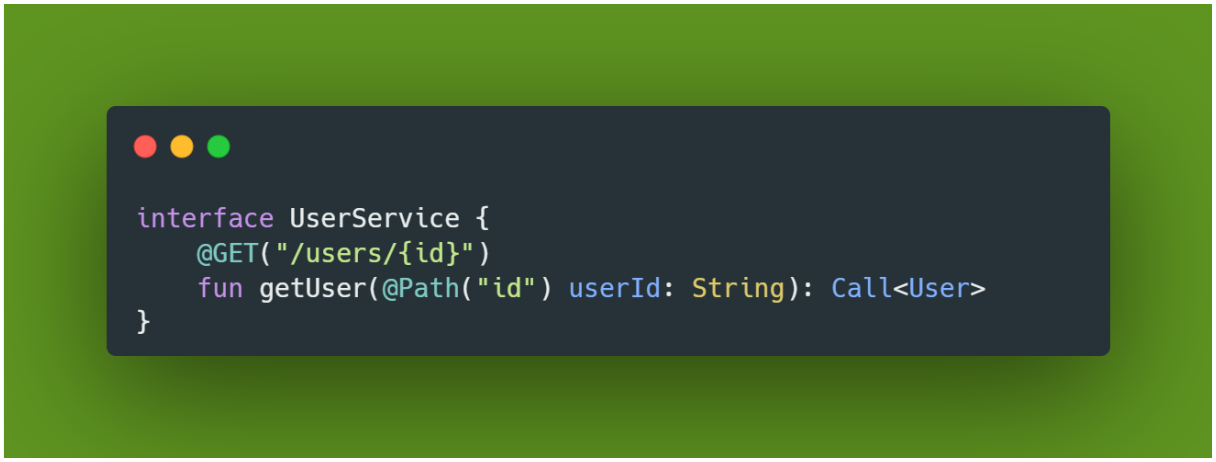
Tanım: **@ HEADER** anotation'u , HTTP isteğinin başlıklarını (headers) belirlemek için kullanılan bir anotation dur. Başlıklar, isteğin yanında sunucuya iletilen ek bilgilerdir ve genellikle isteğin içeriğini, kullanıcı kimlik doğrulama bilgilerini veya diğer özel gereksinimleri belirtmek için kullanılır.

```
interface UserService {
    @GET("/user")
    fun getUser(@Header("Authorization") token: String):
Call<User>
}
```

Bu örnekte, @Header anotasyonu, getUser fonksiyonuna bir Authorization başlığı ekler. Bu başlık, isteği gönderen kullanıcının kimlik doğrulama token'ını içerir. Bu şekilde, sunucu bu token'u kullanarak kullanıcının kimliğini doğrulayabilir ve gerekli işlemleri gerçekleştirebilir.

@PATH ANNOTATİON'U:

Tanım: **@ PATH** anotation'u, Retrofit kütüphanesinde kullanılan ve dinamik olarak değişen URL parçalarını belirlemek için kullanılan bir anotationdur. Bu anotation, bir URL şablonundaki değişkenleri belirtmek için kullanılır ve bu değişkenler, ilgili fonksiyonun parametrelerine bağlanır.



```
interface UserService {
    @GET("/users/{id}")
    fun getUser(@Path("id") userId: String): Call<User>
}
```

Bu örnekte, @Path anotasyonu, /users/{id} URL'sindeki {id} kısmını belirlemek için kullanılır. getUser fonksiyonunun userId parametresine bağlanır. Bu sayede, getUser fonksiyonu çağrıldığında, belirtilen userId değeri, URL'nin {id} kısmına yerleştirilir ve istek gönderilir.

@QUERY ANNOTATİON'U:

Tanım: **@ QUERY** anotation'u , Retrofit kütüphanesinde kullanılan ve HTTP isteği için sorgu parametrelerini belirlemek için kullanılan bir anotationdur. Bu anotation, URL'ye sorgu parametrelerini eklemek için kullanılır ve genellikle GET isteklerinde kullanılır.

```
interface UserService {
    @GET("/users")
    fun getUsers(@Query("page") page: Int, @Query("per_page") perPage: Int): Call<List<User>>
}
```

Bu örnekte, @Query anotasyonu, /users URL'sine sorgu parametreleri eklemek için kullanılır. getUsers fonksiyonu, page ve perPage parametrelerine sahiptir. Bu parametreler, isteğin gönderildiği URL'ye ?page=value&per_page=value şeklinde eklenir, örneğin getUsers(1, 10) çağrıldığında, /users?page=1&per_page=10 şeklinde bir URL oluşturulur ve bu istek gönderilir.

@FormUrlEncoded ANNOTATİON'U:

Retrofit'te en çok kullanılan anotationlardan biri de **@FormUrlEncoded** anotationudur ve HTTP POST isteklerinde form verilerini göndermek için kullanılır. Bu anotation, isteğin içeriğini "application/x-www-form-urlencoded" medya türünde kodlar ve POST isteklerinin gövdesinde form verilerini taşımak için kullanılır.

```
@FormUrlEncoded
@POST("/register")
fun registerUser(
    @Field("username") username: String,
    @Field("password") password: String
): Call<ResponseBody>
```

Bu örnekte, /register URL'sine username ve password adlı iki form gönderilir. @FormUrlEncoded anotationu, isteğin form verileri içerdiğini belirtir. @Field anotationlar, her bir form alanının adını ve değerini belirtmek için kullanılır. Bu veriler, POST isteğinin gövdesine "application/x-www-form-urlencoded" formatında eklenir.

EN ÇOK KULLANILAN HTTP DURUM KODLARI

200 Ok:

İstemci isteği başarıyla işlendi ve istenen kaynak sunuldu.

500 Internal Server Error:

Sunucu, istemcinin isteğini işlerken beklenmedik bir hata ile karşılaştı. Bu kod sunucu tarafında bir kodlama hatası veya teknik sorun olduğunu gösterir.

503 Service Unavailale:

Sunucu şu anda istemcinin isteğini işleyemiyor. Bu kod, sunucu aşırı yüklenmiş olduğunda veya bakımda olduğunda kullanılır.

404 Not Found:

İstenen kaynak sunucuda bulunamadı. Bu kod, kullanıcılar yanlış bir URL girdiğinde veya sayfa silindiğinde kullanılır.

403 Forbidden:

İstemciye, istenen kaynağa erişim izni verilmemiştir. Bu kod, belirli kullanıcılara veya gruplara erişimi kısıtlamak için kullanılır.

401 Unauthorized:

İstemci, istenen kaynağa erişmek için gerekli yetkilere sahip değildir. Bu kod, kullanıcıların giriş yapması gereken korumalı alanlarda kullanılır.

301 Moved Permanently:

İstenen kaynak yeni bir konuma taşınmıştır ve istemci otomatik olarak yeni konuma yönlendirilir.