

L'Internet



HTTP Protocol Tells the Server What to Do

Example Operations:

- GET - Get data from server
- POST/PUT - Put new data in the server
- ◉ DELETE - Delete data from the server

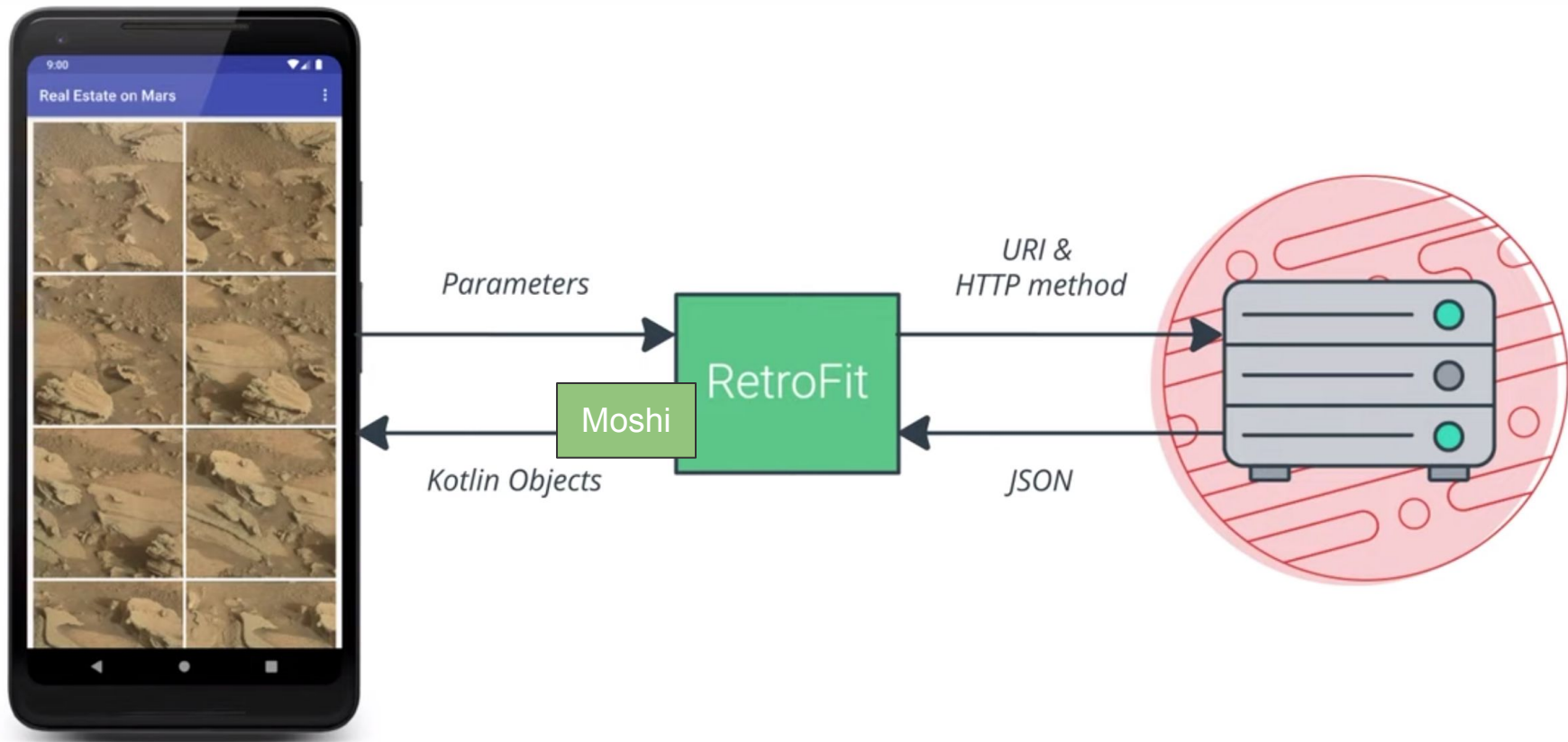
Permissions in AndroidManifest

```
// Necessary to make HTTP requests
<uses-permission android:name="android.permission.INTERNET"/>

// Necessary to get wifi, ethernet or mobile data status
<uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Check if network is available

```
val connectionManager =  
    getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager  
  
val networkInfo = connectionManager.activeNetworkInfo  
  
if (networkInfo != null && networkInfo.isConnected) doNetworkStuff()  
else textView.setText("No network connection available.")  
  
val isWifiConnected =  
    connectionManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).isConnected  
val mobileConnected =  
    connectionManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).isConnected
```



Build a URI for the request


```
val BASE_URL = "https://www.googleapis.com/books/v1/volumes?"
val QUERY_PARAM = "q"
val MAX_RESULTS = "maxResults"
val PRINT_TYPE = "printType"
val uri = Uri.parse(BASE_URL).buildUpon()
    .appendQueryParameter(QUERY_PARAM, "pride+prejudice")
    .appendQueryParameter(MAX_RESULTS, "10")
    .appendQueryParameter(PRINT_TYPE, "books")
    .build()
val requestURL = URL(uri.toString())
```

Moshi: JSON parser

```
val movieJson = "{\"id\": 19404, \"title\": \"Example Movie\", \"image_path\":  
\"/example-movie-image.jpg\" }"
```

```
val moshi: Moshi = Moshi.Builder().build()
```

```
val adapter: JsonAdapter<Movie> = moshi.adapter(Movie::class.java)
```

```
// Annotation to use Codegen instead of this 
```

```
@JsonClass(generateAdapter = true)  
data class Movie (  
    val id: Int,  
    val title: String,  
    @Json(name = "image_path")  
    val imagePath: String? = null,  
)
```

```
private val okHttpClient by lazy {  
    OkHttpClient.Builder()  
        .addInterceptor { chain ->  
            val newRequest = chain.request().newBuilder()  
                .addHeader("Authorization", "Bearer $TOKEN")  
                .build()  
            chain.proceed(newRequest)  
        }  
        .build()  
}
```


Retrofit

```
object MovieApi {  
    private const val BASE_URL = "https://movies.com/API/"  
  
    private val okHttpClient by lazy {...}  
  
    private val moshi = Moshi.Builder().build()  
  
    private val retrofit = Retrofit.Builder()  
        .client(okHttpClient)  
        .baseUrl(BASE_URL)  
        .addConverterFactory(MoshiConverterFactory.create(moshi))  
        .build()  
  
    val movieService: MovieService by lazy { retrofit.create(MovieService::class.java) }  
}  
  
interface MovieService {  
    @GET("movies/{user_id}")  
    suspend fun getMovies(@Path("user_id") userId: String): Response<List<Movie>>  
}
```