**유니티 네트워크 공부**

1. HTTP(Hyper Text Transfer Protocol) 통신
   1. Http란?
      * + Text 기반 통신 프로토콜
        + 실시간 여러 유저가 정보 전송 x
        + 클라가 서버에 요청하고 응답을 보내는 비동기적 서버 통신
2. HTTP in Unity
   1. Http 사용 방법
      * + UnityWebRequest 사용
        + Get, Post, Put, Delete 등 Http 메소드 존재
        + 코루틴으로 통신(대기 시간이 긴 비동기 작업 처리에 유용)
   2. Http Method
      1. Get
         * 요청 데이터의 정보를 주소에 담아 전송(유니티는 주소창이 없어 URL이 직접 노출 x)
         * GetCall(string path), IEnumerator GET(string Path)를 세트로 묶어 관리
           1. GetCall(string Path)

- Get 요청을 시작하기 위해 사용되는 Public Method

- 이 메소드 내의 GET coroutine을 실행하여 비동기적으로 HTTP GET 요청 수행

- 주로 외부에서 쉽게 접근하고 사용할 수 있도록 인터페이스 제공

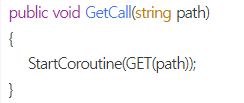


Figure 1. GetCall 함수

* + - * 1. GET(string Path)

- UnityWebRequest.GET을 사용해 API 주소로 get 통신 보냄

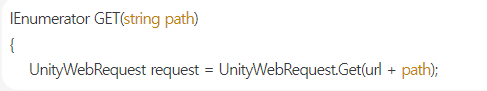


Figure 1.1 GET 함수

- HTTP 요청 헤더에 “Authorization” 헤더를 설정

- 보안을 위해 사용자의 Access Token을 사용해 서버와 통신



Figure 1.2. Authorization + Access Token 설정

- if(request.result != UnityWebRequest.Result.Success를 통해 성공 실패 로직 정리

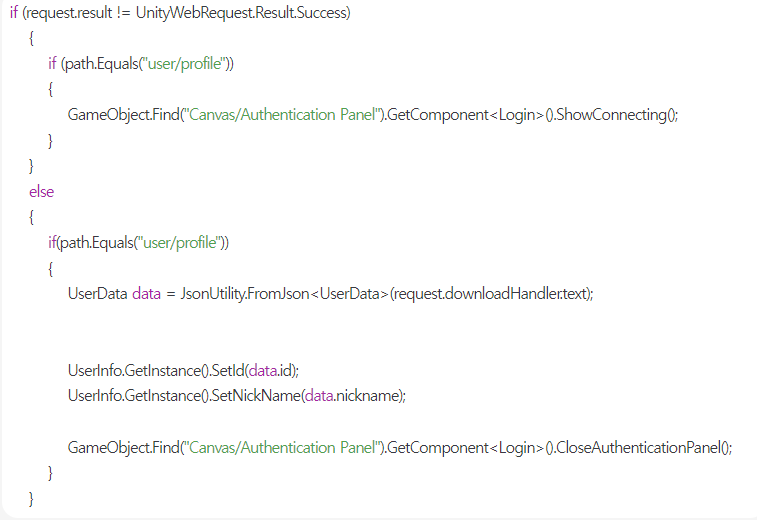


Figure 1.3. GET 함수 내부 예시

* + 1. Post
       - 요청 데이터의 정보를 HTTP 헤더에 담아서 전달하기 때문에 GET에 비해 보안성이 좋음
       - POSTCall(string Path, Dictionary<string, string> postParam)와 IEnumerator POST(string Path, Dictionary<string, string> postParam)을 세트로 묶어 관리
       - Post 처리를 위해 Json 문자열을 만들어야하는데 Parameter를 Dictionary로 받았으므로 이를 Json 형태로 변환해야함

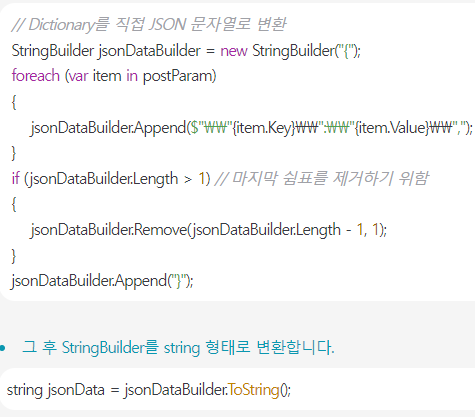


Figure 2.1. Json 변환

* + - * 변환된 Json -> Byte[]로 변환하여 전송



Figure 2.2. Json to Byte

* + - * POST 요청을 보내기 위해 UnityWebRequest 생성 -> 경로와 사용할 메소드 전달해 틀정 주소로 POST 방식으로 전송
      * UploadHandlerRaw = 서버로 보낼 데이터 처리
      * JsonData를 UploadHandlerRaw의 파라미터로 삽입하고 이를 UnityWebRequest.uploadHandler에 추가
      * 응답을 받기 위한 DownloadHandlerBuffer 사용해 속성 설정
      * HTTP 요청 헤더 설정(content-type , application/json)

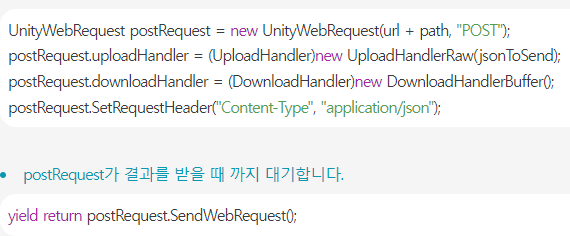


Figure 2.3. UnityWebRequest Upload 방법

* + - * Response 받는데 실패/성공을 나누어 작업

