

QuickSheet 가이드

QuickSheet를 활용하여 데이터 가져오기

작성자 : 안중재

본 문서는 QuickSheet를 활용하여 데이터를 다루는 방법에 대하여 설명합니다.
 기획자를 대상으로 작성된 문서로 QuickSheet의 내부 구조에 대해서는 다루지 않습니다.

[1] Data Parsing

- 작성한 데이터 테이블의 데이터를 사용하기 위하여, 특정한 규칙에 따라 추출하여 가공하는 것입니다.

데이터 테이블 (ex. Data.xlsx)

ID	Name	Hp	AttackPower
10001	Slime	300	0
10002	Goblin	500	10



데이터

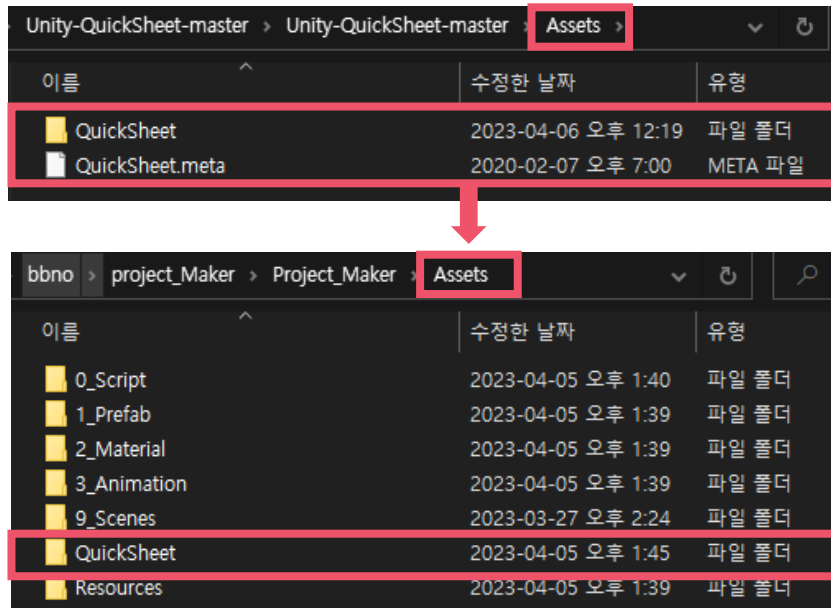
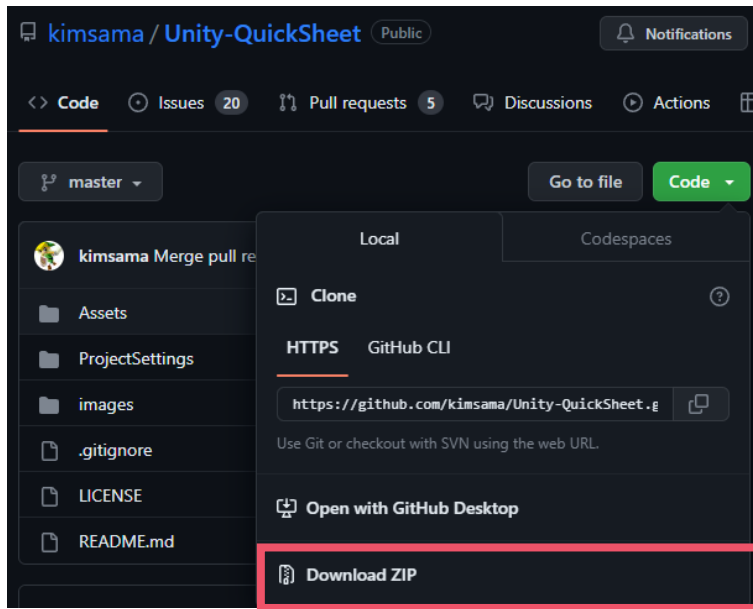
10001	Slime	300	0	10002	Goblin	500	10
-------	-------	-----	---	-------	--------	-----	----

[2] QuickSheet

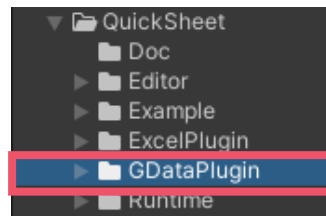
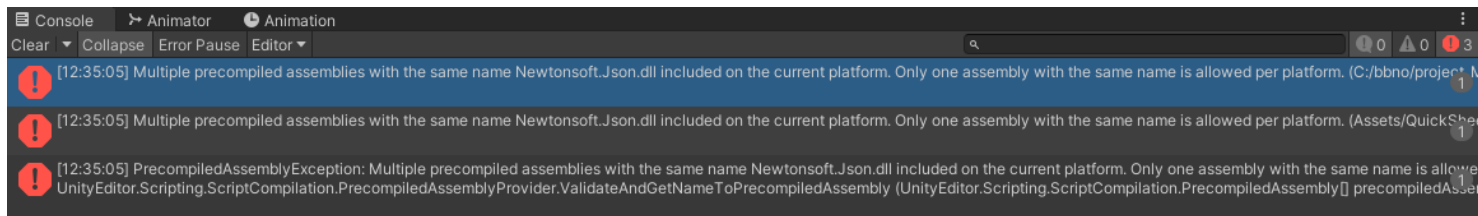
- Data Parsing을 도와주는 프로그램입니다. (<https://github.com/kimsama/Unity-QuickSheet>)
- 데이터를 Unity3D의 ScriptableObject로 직렬화 합니다.

[1] QuickSheet 세팅

- QuickSheet github에 접속하여 파일을 다운로드합니다. . (<https://github.com/kimsama/Unity-QuickSheet>)
- 다운로드 받은 파일의 압축을 풀고 **Assets 폴더의 파일**을 유니티 프로젝트의 Assets 폴더에 복사합니다.

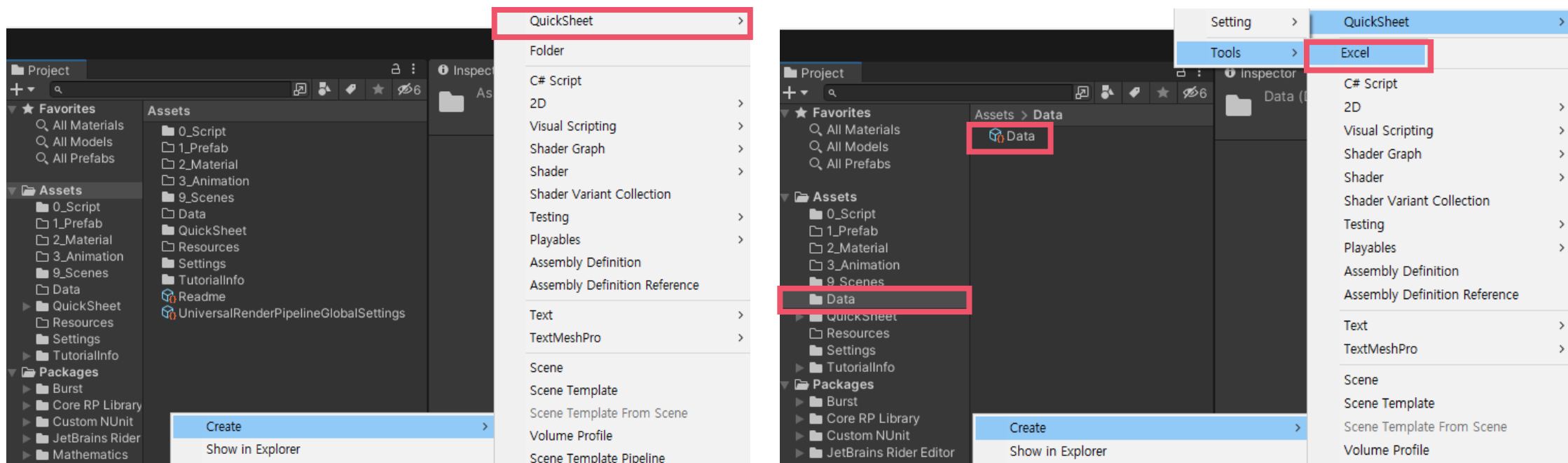


- 만약 아래와 같은 오류가 발생한다면, 우리는 Google을 활용하지 않을 것이기 때문에 **GDataPlugin** 폴더를 삭제하셔도 괜찮습니다.



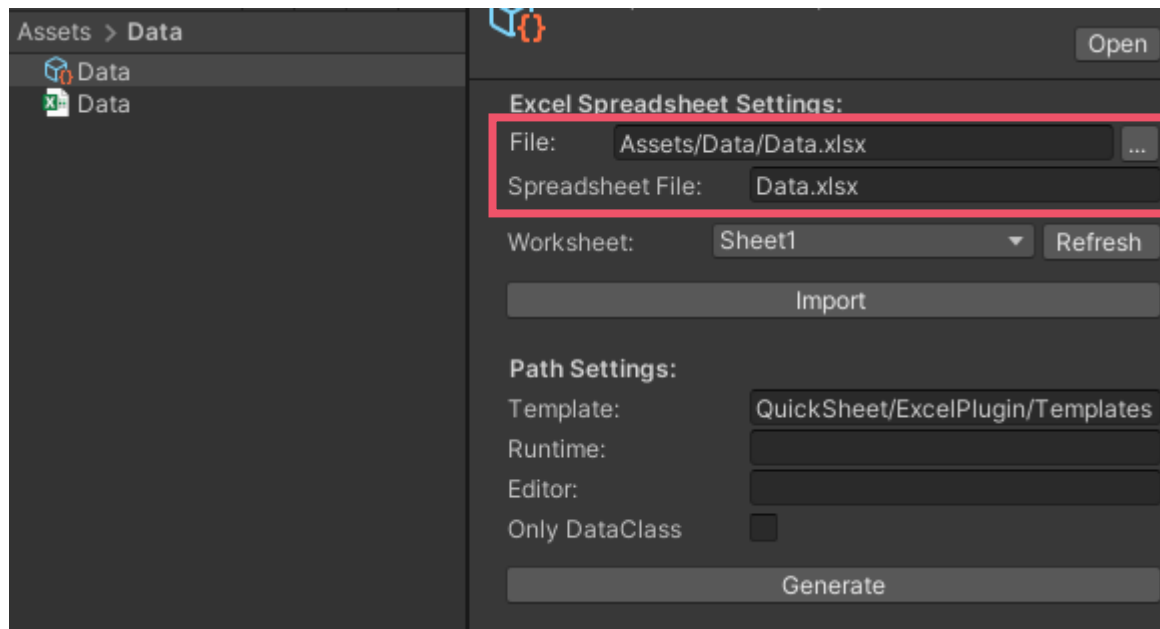
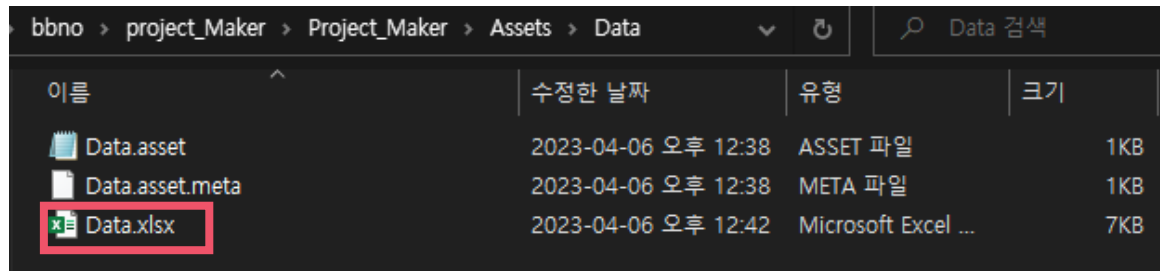
[1] QuickSheet 세팅

- Project에서 마우스 우클릭 > Create > QuickSheet가 보인다면 정상적으로 적용된 상태입니다.
- Data를 다룰 폴더를 하나 생성하고, 폴더에 **QuickSheet > Tools > Excel** 을 생성해줍니다.
- 폴더와 파일 이름은 자유롭게 작성해도 무관하나 예제에서는 Data로 작성하겠습니다.



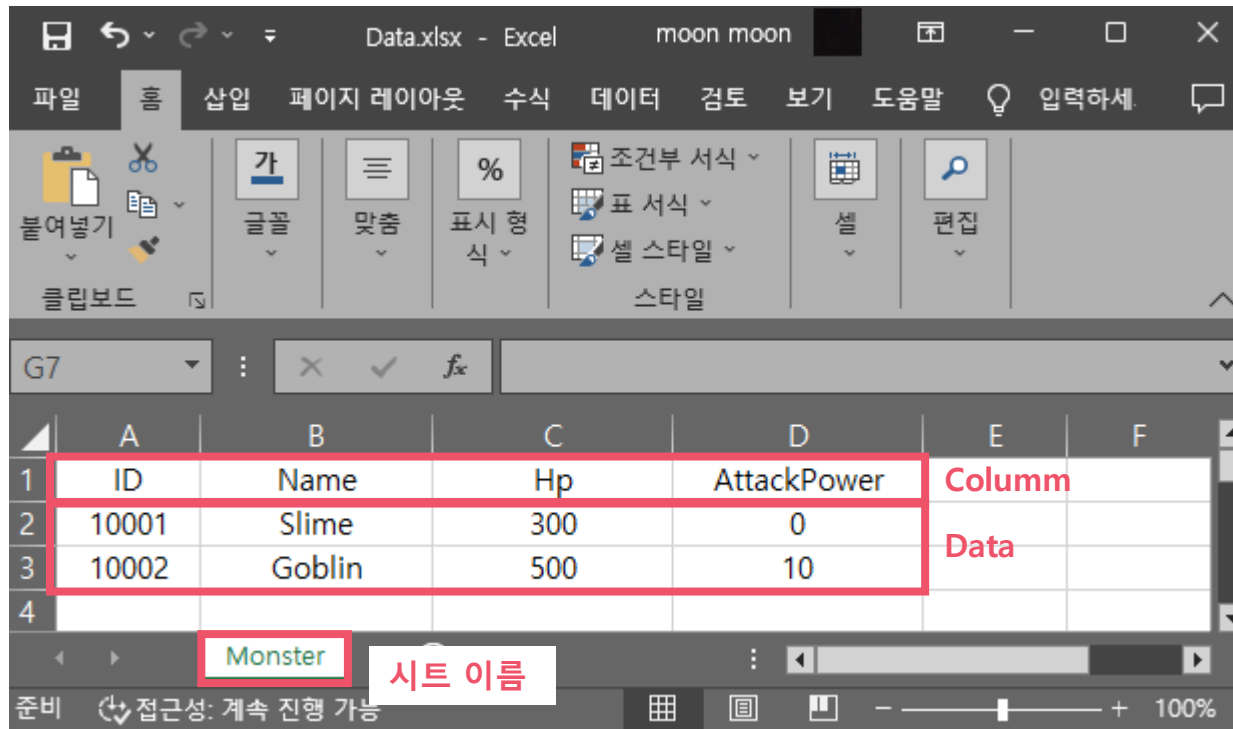
[1] QuickSheet 세팅

- Data를 다룰 **엑셀 파일을 생성**해줍니다. (예제에는 Data.xlsx 파일을 생성하였습니다)
- 생성이 완료되었으면 이전에 유니티에서 만든 Data 파일에 **엑셀 파일의 경로를 설정**해줍니다.



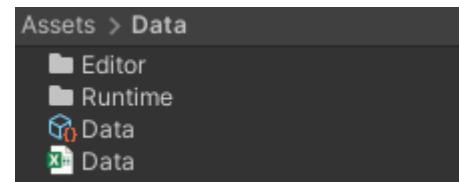
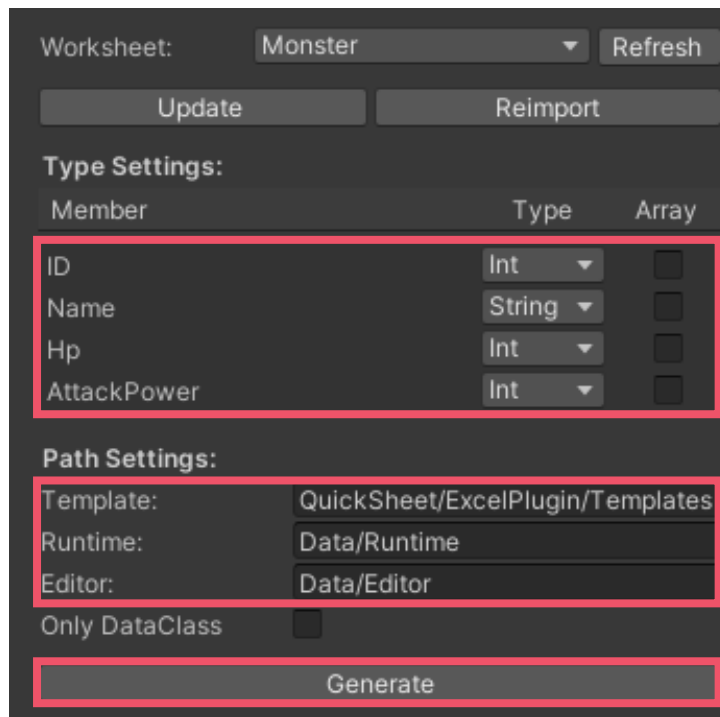
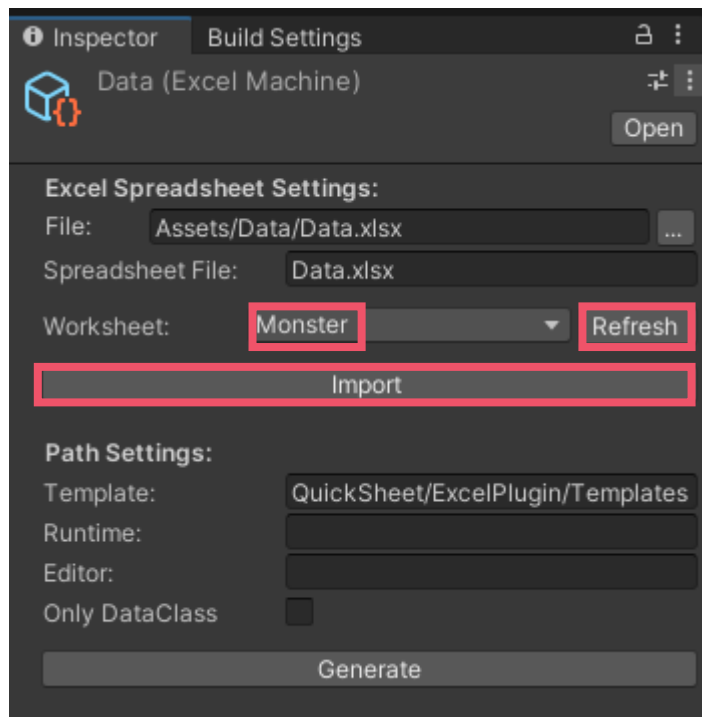
[2] QuickSheet 활용하기

- 생성한 엑셀파일에 데이터를 입력합니다.
- 데이터는 시트별로 구분할 수 있으니 용도에 맞게 구조를 설계합니다. (여러 개 추가 가능)
- 첫번째 행에는 데이터를 구분할 컬럼(Column)이 들어갑니다.



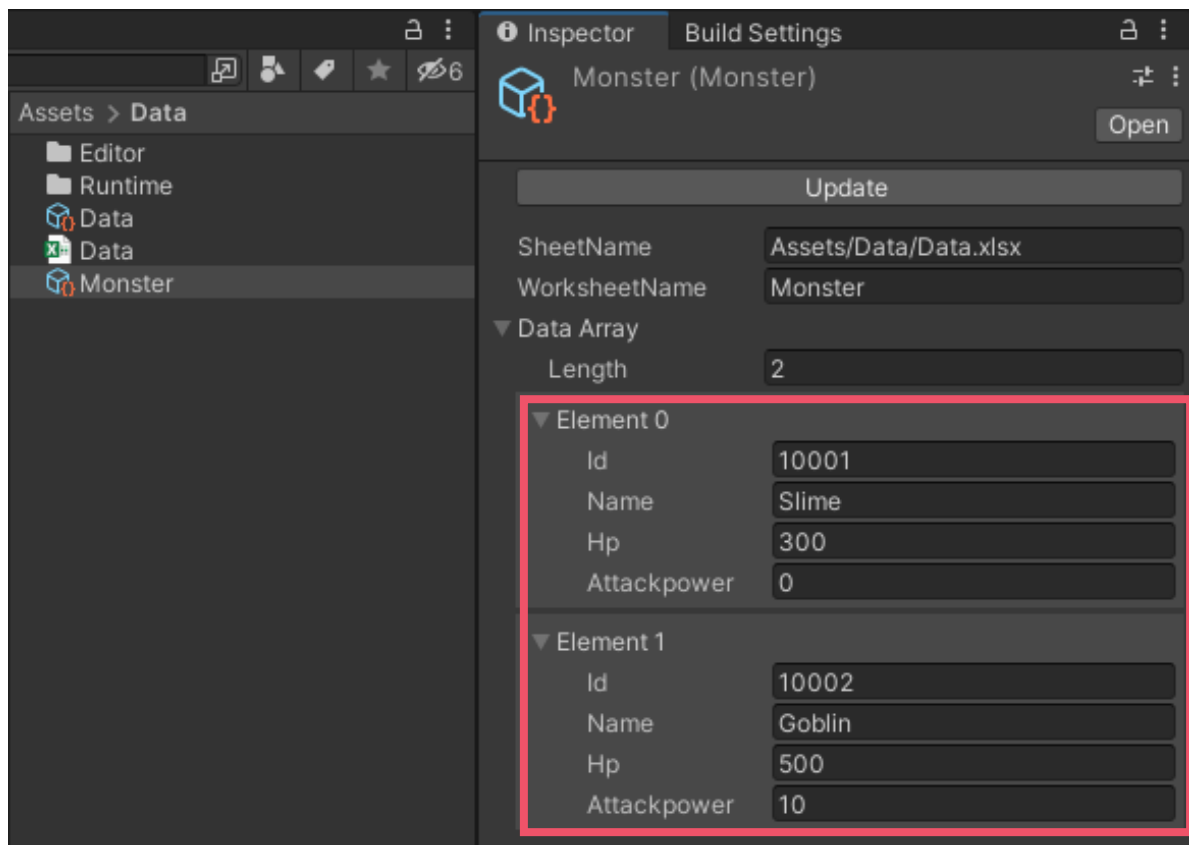
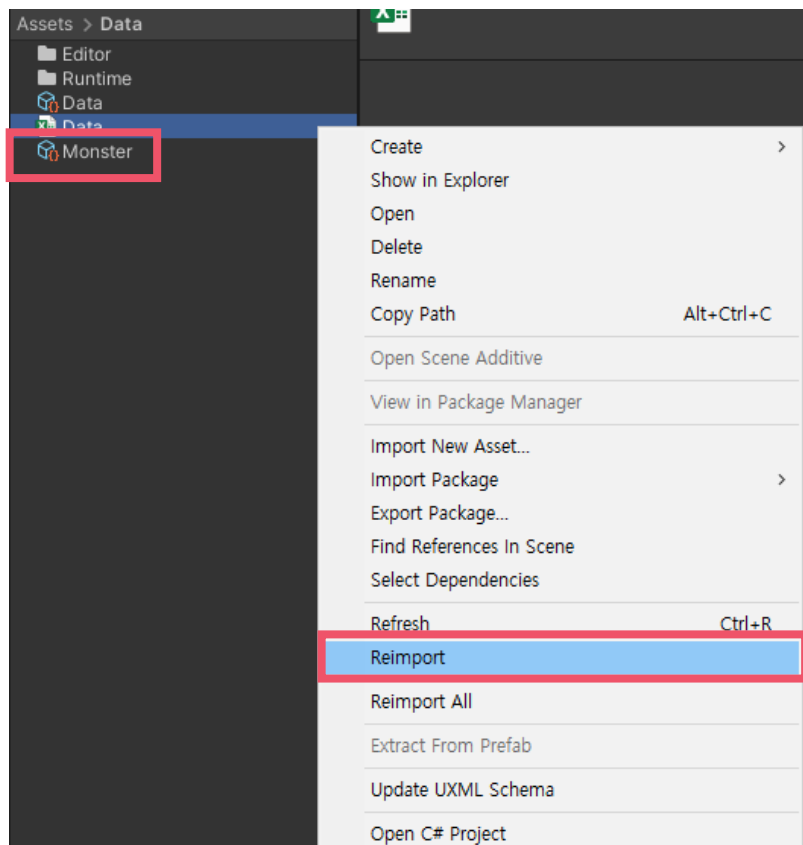
[2] QuickSheet 활용하기

- 유니티로 돌아와 Data 파일의 **Refresh** 버튼을 누르면 생성했던 Monster 시트가 나타나는 것을 확인할 수 있다.
- **Import**를 누르면 데이터 테이블에서 작성했던 컬럼이 나타난다.
- **Type Setting**에서 컬럼의 데이터 타입을 설정하고, **Path Setting**에서 경로를 입력해준다.
- 설정을 완료했으면 **Generate**를 눌러 생성을 진행한다.



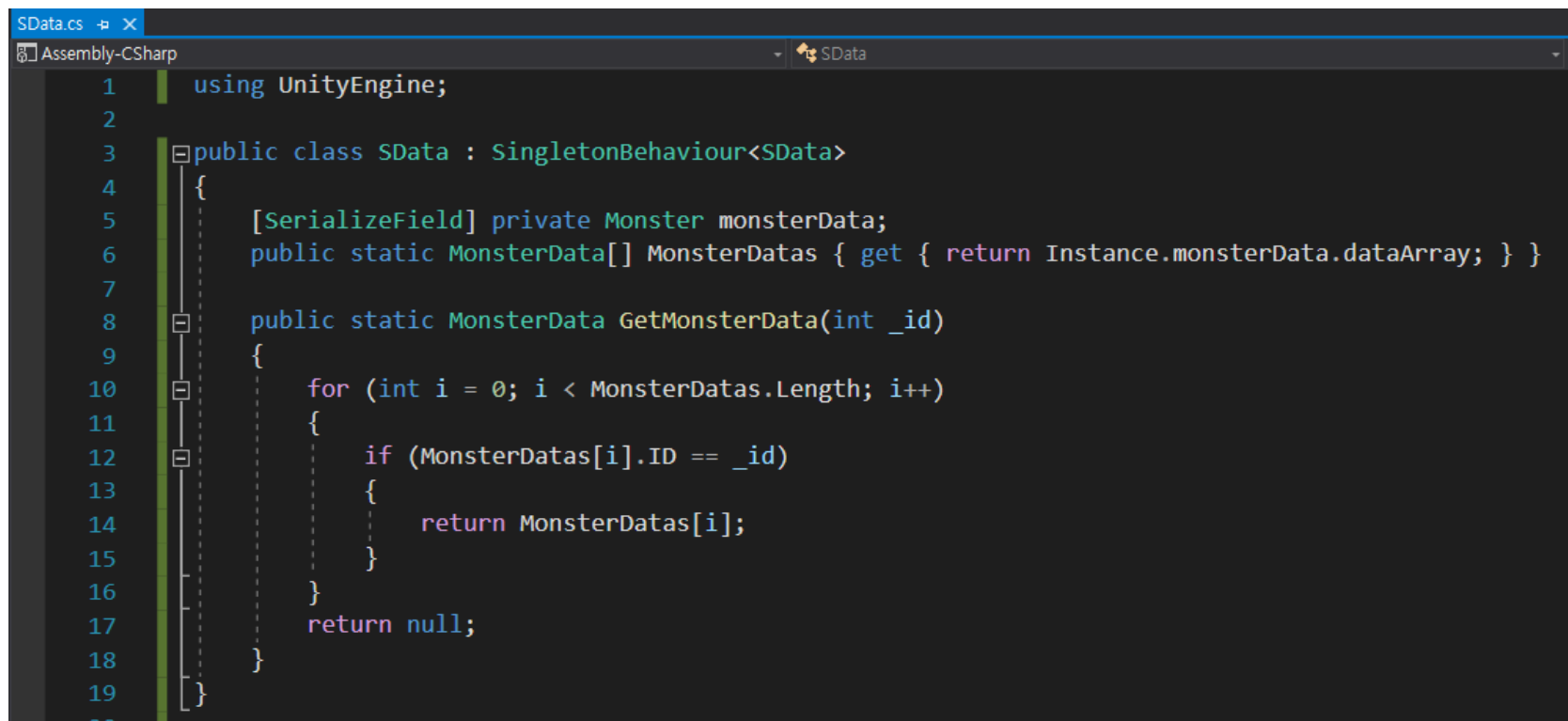
[2] QuickSheet 활용하기

- 엑셀 파일을 우클릭하여 Reimport를 하면 작성했던 Monster 시트의 파일이 추가되는 것을 확인할 수 있다.
- Monster 파일에는 엑셀에서 작성했던 데이터가 들어가 있는 것을 확인할 수 있다.



[3] 데이터 사용하기

- 데이터를 사용할 스크립트 파일을 생성하고 아래와 같이 작성한다.
- 아래 생성된 클래스(SData)는 **Singleton패턴을 상속**받아 사용한다.
- Monster와 MonsterData 클래스는 QuickSheet에서 생성해준 클래스를 사용하고 변수와 함수명은 자유롭게 작성한다.

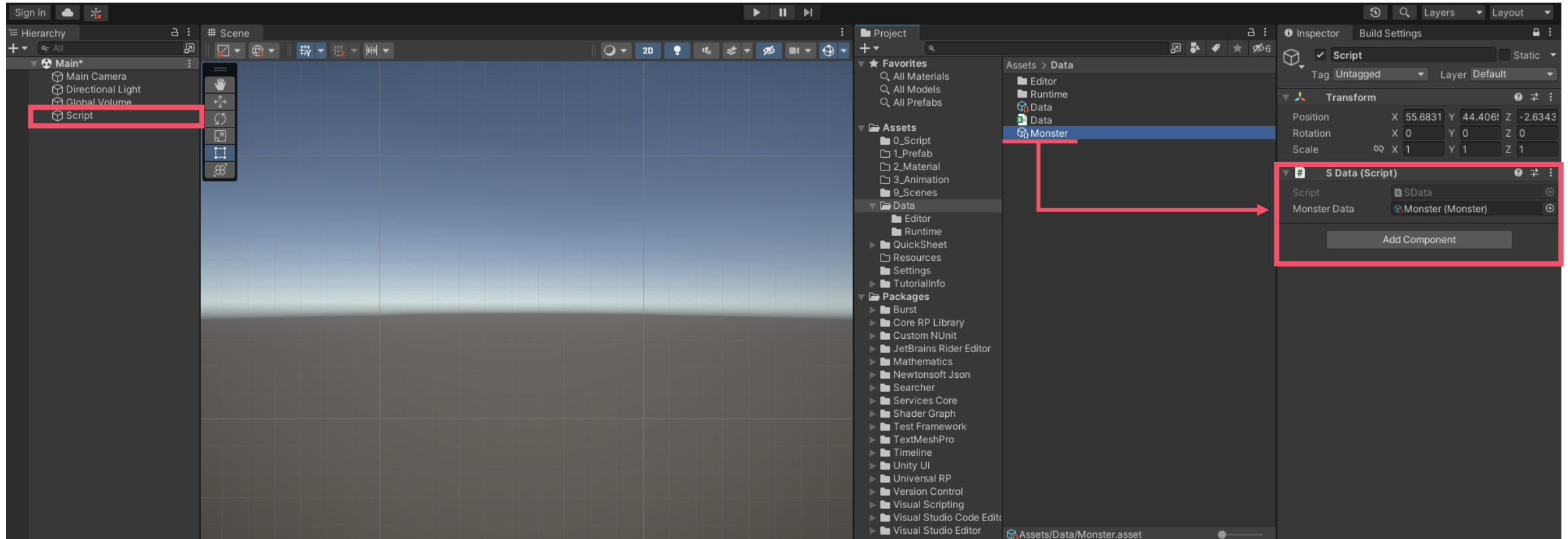


```
1 using UnityEngine;
2
3 public class SData : SingletonBehaviour<SData>
4 {
5     [SerializeField] private MonsterData monsterData;
6     public static MonsterData[] MonsterDatas { get { return Instance.monsterData.dataArray; } }
7
8     public static MonsterData GetMonsterData(int _id)
9     {
10         for (int i = 0; i < MonsterDatas.Length; i++)
11         {
12             if (MonsterDatas[i].ID == _id)
13             {
14                 return MonsterDatas[i];
15             }
16         }
17         return null;
18     }
19 }
```

- 코드는 데이터를 배열에 넣어 ID 값을 기준으로 탐색하여 사용하는 방식으로 구현되어 있다.
- 프로그래머라면 최적화에 신경쓸 부분이지만 기획자는 크게 상관하지 않아도 될 것이라 생각됨

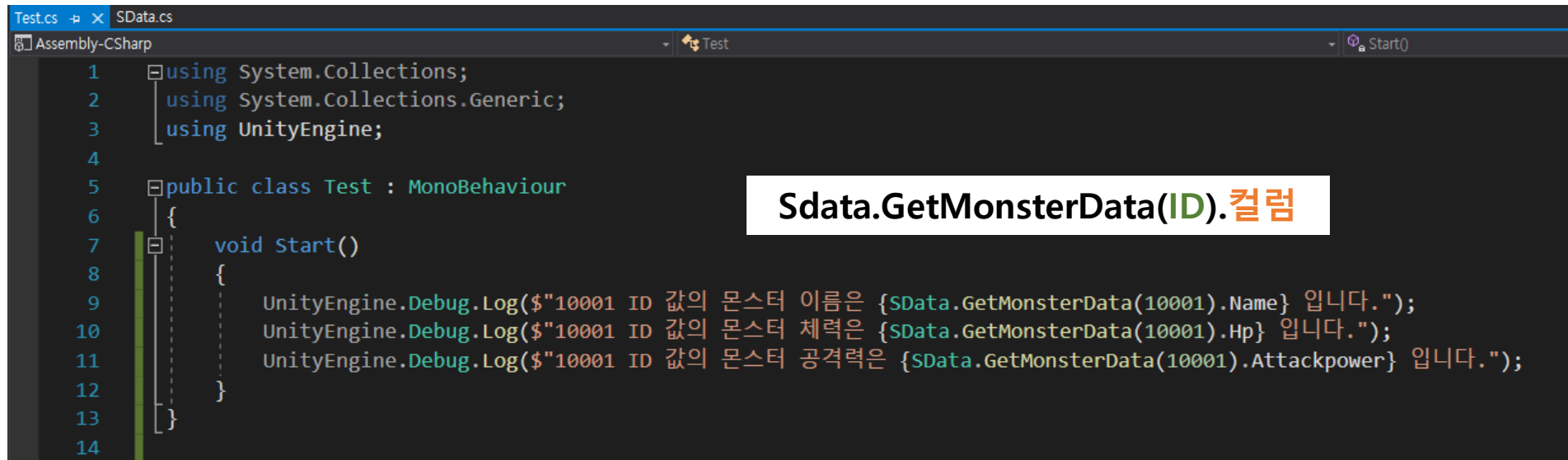
[3] 데이터 사용하기

- 생성한 스크립트를 적용시킬 GameObject를 추가하고 SData 스크립트를 추가한다.
- Inspector창에서 Sdata 스크립트의 MonsterData에 Monster 파일을 적용시킨다.



[3] 데이터 사용하기

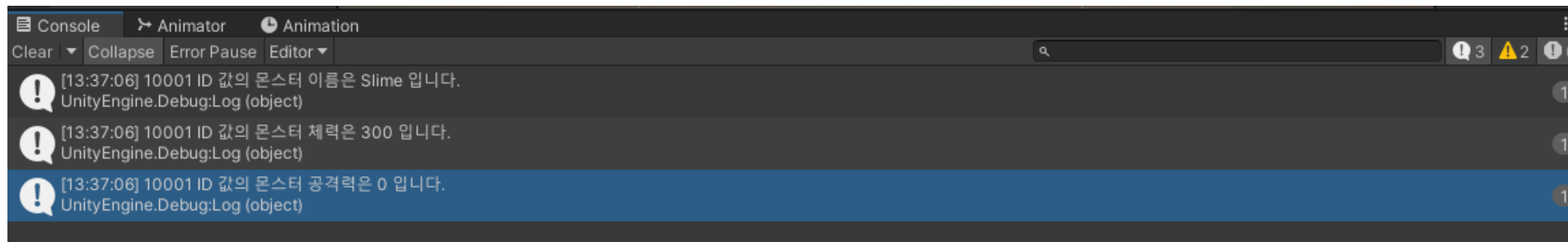
- 데이터 사용이 필요한 구간에서 아래와 같이 코드를 작성하여 사용합니다.



```
Test.cs SData.cs
Assembly-CSharp Test Start()

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Test : MonoBehaviour
6 {
7     void Start()
8     {
9         UnityEngine.Debug.Log($"10001 ID 값의 몬스터 이름은 {SData.GetMonsterData(10001).Name} 입니다.");
10        UnityEngine.Debug.Log($"10001 ID 값의 몬스터 체력은 {SData.GetMonsterData(10001).Hp} 입니다.");
11        UnityEngine.Debug.Log($"10001 ID 값의 몬스터 공격력은 {SData.GetMonsterData(10001).Attackpower} 입니다.");
12    }
13 }
14
```

Sdata.GetMonsterData(ID).컬럼



```
Console Animator Animation
Clear Collapse Error Pause Editor
[13:37:06] 10001 ID 값의 몬스터 이름은 Slime 입니다.
UnityEngine.Debug:Log (object) 1
[13:37:06] 10001 ID 값의 몬스터 체력은 300 입니다.
UnityEngine.Debug:Log (object) 1
[13:37:06] 10001 ID 값의 몬스터 공격력은 0 입니다.
UnityEngine.Debug:Log (object) 1
```

감사합니다