게임 데이터 변환 툴 사용 설명서 (Excel to JSON)

Contents

1. 개요

1.1. 시스템의 목적

이 문서는 우리 팀의 게임 데이터 관리 프로세스를 통일하고 자동화하기 위해 제작된 "XLSX to JSON 데이터 변환 툴"의 사용법을 설명합니다.

이 시스템의 핵심 목표는 다음과 같습니다.

- **기획자**: 복잡한 툴 없이, 익숙한 **Excel(엑셀)** 프로그램을 사용하여 게임의 모든 데이터(캐릭터 스탯, 몬스터 정보, 아이템 목록 등)를 쉽고 빠르게 관리합니다.
- **프로그래머**: 기획자가 수정한 Excel 데이터를 Unity 에디터 내에서 버튼 두 번 클릭으로 **C# 클래스 및 JSON 파일로 자동 변환**하여, 즉시 게임에 안정적으로 적용합니다.

이를 통해 팀원 간의 데이터 전달 과정을 최소화하고, 휴먼 에러를 줄이며, 개발 속도를 크게 향상시킬 수 있습니다.

1.2. 전체 워크플로우

데이터는 아래와 같은 흐름으로 게임에 적용됩니다.

기획자 (Excel 파일 작성) → 프로그래머 (변환 툴 실행) → 자동화 시스템 (C# 클래스 & JSON 파일 생성) → 게임 실행 (DataManager 가 JSON 로드) → 게임에 데이터 적용 완료

---

2. 기획자를 위한 가이드: Excel 데이터 작성 규칙

기획자 여러분은 아래 규칙에 따라 Excel 파일만 작성해주시면 됩니다. 이 규칙들은 시스템이 데이터를 정확하게 인식하기 위한 약속입니다.

2.1. 파일 및 시트 규칙

1. **파일 위치**: 모든 데이터 Excel 파일은 프로젝트 폴더 내의 **Assets/Data/Xlsx** 에 저장합니다.
2. **파일 형식**: 반드시 .xlsx 형식으로 저장해야 합니다.

3. **시트 이름(Sheet Name)**:
    - **가장 중요합니다.** 시트 이름이 곧 데이터의 이름(C# 클래스 이름)이 됩니다. (예: PlayerData, MonsterData)
    - **반드시 영문**으로 작성해야 하며, 공백이나 특수문자는 포함하지 않는 것을 권장합니다.
    - 하나의 Excel 파일 안에 여러 개의 시트를 만들어 각각 다른 데이터를 관리할 수 있습니다. (예: Item.xlsx 파일 안에 Equipment 시트, Consumable 시트)

2.2. 데이터 작성 규칙

1. **첫 번째 행 (헤더)**:
    - 시트의 1 행은 데이터의 각 항목(변수명)을 정의하는 헤더입니다.
    - **반드시 영문**으로, 공백 없이 작성해야 합니다. (공백 대신 언더바 _ 사용 권장. 예: attack_power)
    - 이 헤더 이름이 프로그래머가 코드에서 사용하는 변수 이름이 됩니다.
2. **데이터 타입**:
    - **정수**: 100, 50 과 같이 숫자만 입력합니다.
    - **실수:** 5f, 7.5f 와 같이 숫자 뒤에 접미사 f 를 붙입니다.
    - **문자**: Goblin, HP Potion 과 같이 텍스트를 입력합니다.
    - **참/거짓**: TRUE, FALSE 로 입력합니다.

2.3. 데이터 종류 구분 (매우 중요!)

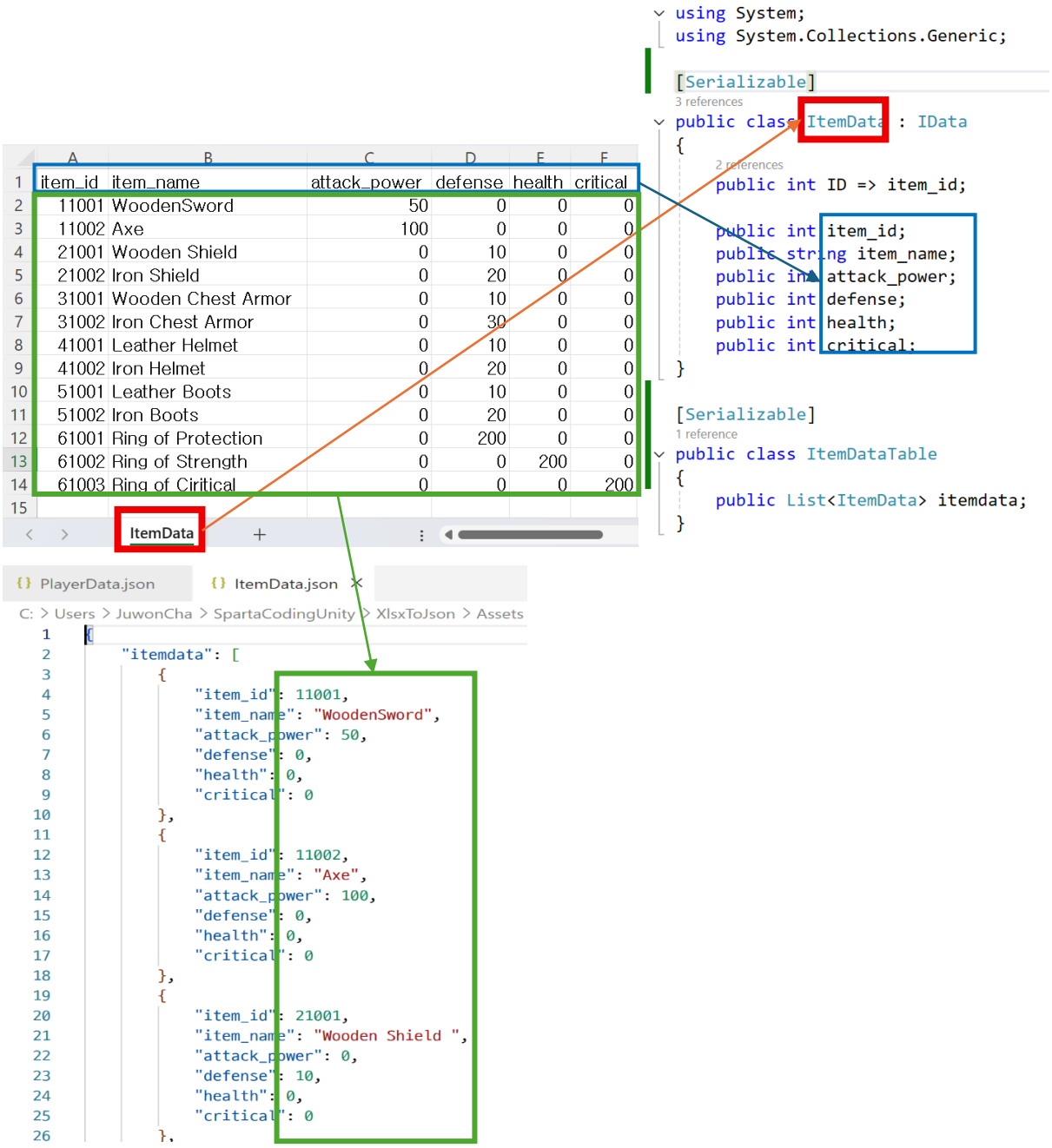
우리 시스템은 데이터의 종류를 첫 번째 열(A 열)의 헤더 이름으로 구분합니다.

A. 목록형 데이터 (Collection Data)

- **설명**: 몬스터, 아이템, 스킬처럼 여러 개의 데이터가 목록으로 존재하는 경우입니다.

- **규칙**: 첫 번째 열(A 열)의 헤더 이름에 **반드시 "id" 또는 "key" 라는 단어가 포함**되어야 합니다. (대소문자 무관)
- **예시**: monster_id, itemID, questKey 등
- 데이터는 2 행부터 원하는 만큼 추가할 수 있습니다. 각 id 는 고유해야 합니다.

예시

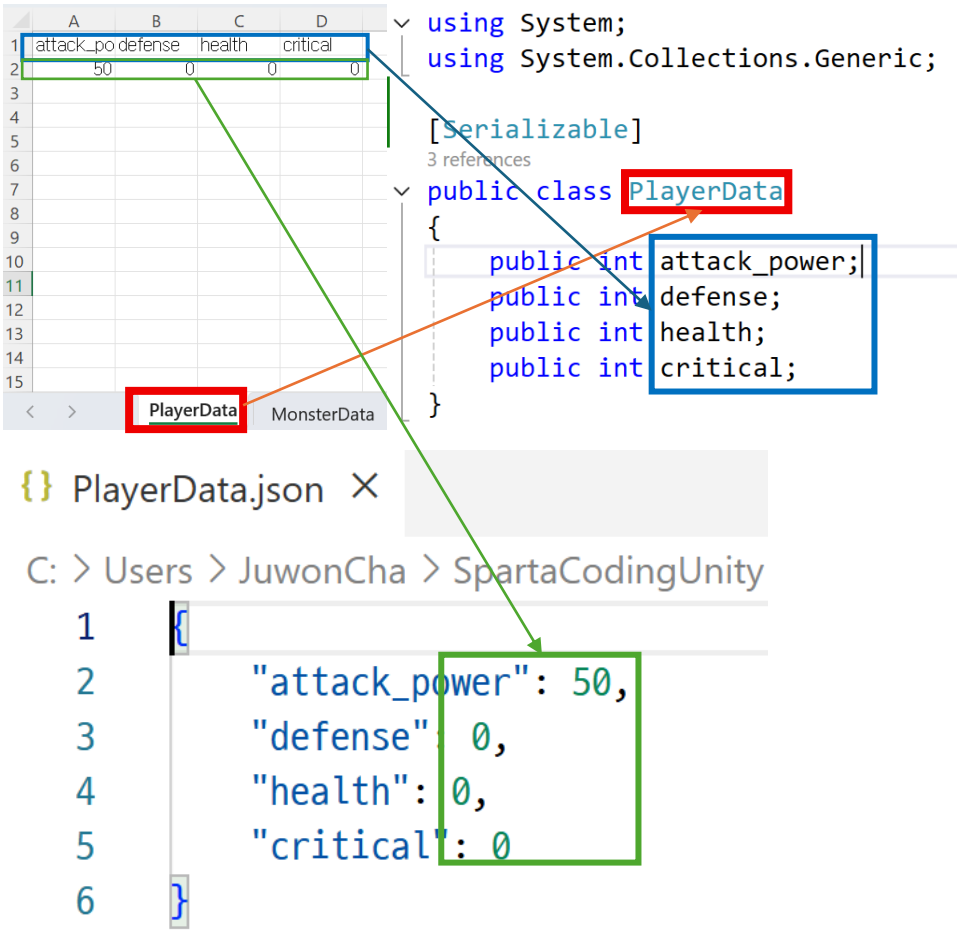


item_id	item_name	attack_power	defense	health	critical
11001	WoodenSword	50	0	0	0
11002	Axe	100	0	0	0
21001	Wooden Shield	0	10	0	0
21002	Iron Shield	0	20	0	0
31001	Wooden Chest Armor	0	10	0	0
31002	Iron Chest Armor	0	30	0	0
41001	Leather Helmet	0	10	0	0
41002	Iron Helmet	0	20	0	0
51001	Leather Boots	0	10	0	0
51002	Iron Boots	0	20	0	0
61001	Ring of Protection	0	200	0	0
61002	Ring of Strength	0	0	200	0
61003	Ring of Ciritical	0	0	0	200


```
{
    "itemdata": [
        {
            "item_id": 11001,
            "item_name": "WoodenSword",
            "attack_power": 50,
            "defense": 0,
            "health": 0,
            "critical": 0
        },
        {
            "item_id": 11002,
            "item_name": "Axe",
            "attack_power": 100,
            "defense": 0,
            "health": 0,
            "critical": 0
        },
        {
            "item_id": 21001,
            "item_name": "Wooden Shield ",
            "attack_power": 0,
            "defense": 10,
            "health": 0,
            "critical": 0
        },
```

B. 단일 객체 데이터 (Single Data)

- **설명**: 플레이어의 스탯처럼 게임 내에서 유일하게 하나만 존재하는 데이터입니다.
- **규칙**:
  - 첫 번째 열(A 열)의 헤더 이름에 **"id" 또는 "key"가 포함되지 않아야 합니다.**
  - 헤더 행(1 행) 아래에 **데이터는 반드시 단 한 줄(2 행)**만 존재해야 합니다.
- **예시**



[요약] 기획자 여러분은 이 규칙에 맞춰 Assets/Data/Xlsx 폴더에 엑셀 파일만 저장해주시면 나머지는 프로그래머가 툴을 통해 게임에 자동으로 반영할 것입니다.

---

3. 프로그래머를 위한 가이드: 툴 사용 및 데이터 활용

3.1. 데이터 변환 워크플로우

기획자가 Excel 작업을 완료하면, 프로그래머는 Unity 에디터에서 아래 두 단계를 순서대로 진행합니다.

툴 실행: Unity 상단 메뉴 Tools > Data Converter > XLSX to JSON Converter 클릭

1 단계: C# 클래스 생성

- **언제 사용하나요?**
    1. 새로운 종류의 Excel 파일이 추가되었을 때
    2. 기존 Excel 파일의 구조가 변경(열이 추가/삭제)되었을 때
- **사용법**: 에디터 창에서 **Step 1: Generate C# Classes** 버튼을 클릭합니다.
- **결과**:
    - Assets/Scripts/GeneratedData 폴더에 Excel 시트와 동일한 이름의 C# 클래스 파일(.cs)이 생성됩니다.
    - 완료 후 Unity 가 자동으로 스크립트 재컴파일을 시작합니다. **재컴파일이 끝날 때까지 잠시 기다려야 합니다.**

2 단계: JSON 으로 변환

- **언제 사용하나요?**
    1. 1 단계 실행 및 재컴파일이 완료된 후
    2. Excel 파일의 구조 변경 없이, 데이터 **값**만 수정되었을 때
- **사용법**: 에디터 창에서 **Step 2: Convert Excel to JSON** 버튼을 클릭합니다.
- **결과**:

- Assets/Resources/GeneratedJson 폴더에 C# 클래스와 동일한 이름의 JSON 파일(.json)이 생성됩니다.
- 이 JSON 파일들은 게임에서 직접 사용됩니다.

3.2. 런타임에 데이터 사용하기 (DataTableManager)

DataTableManager 는 자동 생성된 모든 데이터를 로드하고 게임 내 어디서든 쉽게 접근할 수 있도록 제공하는 싱글톤(Singleton) 클래스입니다.

A. 데이터 로딩

게임 시작 시점(예: GameManager 의 Start 메서드)에서 필요한 데이터를 명시적으로 로드해야 합니다.

- **목록형 데이터 로딩**: LoadCollectionData<[클래스이름]Table>() 를 사용합니다.
- **단일 객체 데이터 로딩**: LoadSingleData<[클래스이름]>() 을 사용합니다.

```
// GameManager.cs 예시
void Start()
{
    // 단일 객체 데이터 "PlayerData" 로드
    DataTableManager.Instance.LoadSingleData<PlayerData>();

    // 목록형 데이터 "MonsterData" 로드
    DataTableManager.Instance.LoadCollectionData<MonsterDataTable>();
}
```

B. 데이터 접근

- **목록형 데이터 가져오기**: GetCollectionData<[클래스이름]>(id) 를 사용합니다.
- **단일 객체 데이터 가져오기**: GetSingleData<[클래스이름]>() 을 사용합니다.

```
// GameManager.cs 예시

// 플레이어 데이터 가져오기
PlayerData player = DataTableManager.Instance.GetSingleData<PlayerData>();
if (player != null)
{
    Debug.Log($"Player Attack Power: {player.attack_power}");
}

// ID가 100인 몬스터 데이터 가져오기
MonsterData goblin = DataTableManager.Instance.GetCollectionData<MonsterData>(100);
if (goblin != null)
{
    Debug.Log($"Monster Name: {goblin.monster_name}");
}
```

3.3. IData 인터페이스

- IData 는 모든 목록형 데이터 클래스가 자동으로 구현하는 인터페이스입니다. (public interface IData { int ID { get; } })
- 이 인터페이스 덕분에 DataTableManager 는 어떤 종류의 목록형 데이터든 ID 를 기준으로 제네릭하게 관리할 수 있습니다.

---

4. 전체 워크플로우 예시: 'SkillData' 추가하기

1. **기획자**:
    - Assets/Data/Xlsx 폴더에 SkillData.xlsx 파일을 생성합니다.
    - 파일 안에 ActiveSkill 이라는 이름의 시트를 만듭니다.
    - A 열 헤더를 skill_id 로 지정하고, B 열부터 skill_name, mana_cost 등의 헤더를 추가합니다.
    - 2 행부터 스킬 데이터를 여러 줄 입력하고 저장합니다.
2. **프로그래머**:
    - Unity 에디터에서 XLSX to JSON Converter 툴을 엽니다.
    - **Step 1: Generate C# Classes** 버튼을 클릭합니다.
    - Unity 재컴파일이 완료되면 Assets/Scripts/GeneratedData 폴더에 ActiveSkill.cs 와 ActiveSkillTable.cs 파일이 생성된 것을 확인합니다.
    - **Step 2: Convert Excel to JSON** 버튼을 클릭합니다.
    - Assets/Resources/GeneratedJson 폴더에 ActiveSkill.json 파일이 생성된 것을 확인합니다.
    - GameManager.cs 의 Start 메서드에 DataTableManager.Instance.LoadCollectionData<ActiveSkillTable>(); 코드를 추가하여 새 데이터를 로드합니다.
    - 이제 게임 내 어디서든 DataTableManager.Instance.GetCollectionData<ActiveSkill>(스킬 ID); 코드로 스킬 데이터에 접근할 수 있습니다.

---

5. Q&A 및 문제 해결

- **Q: Step 2 를 눌렀는데 'Class not found' 에러가 발생합니다.**
    - **A:** Step 1 을 실행한 후 Unity 의 스크립트 재컴파일이 완전히 끝났는지 확인해주세요. 툴을 닫았다가 다시 열어보는 것도 좋습니다.
- **Q: C# 클래스는 생성되었는데, 변수(Field)가 몇 개 빠져있습니다.**
    - **A:** Excel 파일의 헤더(1 행)에 빈 칸이 있거나, 변수명으로 사용할 수 없는 공백/특수문자가 포함되어 있는지 확인해주세요.
- **Q: 분명 숫자만 입력했는데, 데이터 타입이 string 으로 생성됩니다.**
    - **A:** Excel 셀 서식이 '텍스트'로 되어 있는지 확인해주세요. 숫자 데이터는 '일반' 또는 '숫자' 서식을 사용하는 것이 가장 안전합니다.