SerializeField, Header và một số attributes trong Unity3D

Attribute trong Unity là tính năng được sử dụng rất nhiều trong scripts, hầu như class nào mình cũng xài tới nó. Đồng thời phần cuối series mình cũng sẽ nói cách tạo một custom attribute.

1. SerializeField attribute, chỉnh sửa các biến private trên inspector
Trong lập trình OOP, chúng ta đã biết về tính đóng gói, cơ bản như sau “Một class chỉ nên “trưng” ra các biến, các hàm nào bên ngoài cần sử dụng như là API, còn lại các hàm và biến (state) xử lý nội bộ trong class chúng ta đều đặt private.”

Tuy nhiên các biến private thì không được Unity serialize lên inspector để cho chúng ta hoặc các game designer chỉnh sửa và cân bằng game, đó là lý do chúng ta cần SerializeField:

```
Vector3 targetPosition;

[SerializeField] float moveDuration = 0.1f;
```

Một số lưu ý về khả năng [SerializeField] của Unity:

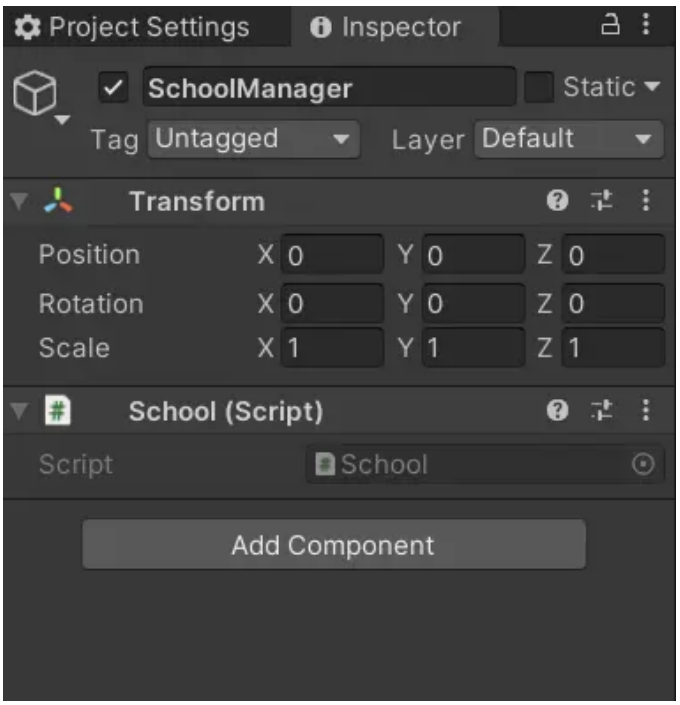
Không thể serialize biến static
Không thể serialize các field (get set của biến)
Không serialize được kiểu Dictionary
Chỉ có thể serialize các kiểu dữ liệu được đánh dấu [Serializable] (đọc phần dưới)
2. System.Serializable attribute, vì sao field không hiện lên inspector?
Ở attribute này mình sẽ không nói quá sâu bởi thuộc về package System của C#, tuy nhiên nó liên quan tới cách mà Unity serialize lên inspector nên mình đề cập vào.

```
public class Person {
    public string fullname;
    [SerializeField] string firstName;
    private string lastName;
}
```

Ở trên mình có 1 class Person non-MonoBehaviour đơn giản gồm 3 biến: public fullname, [SerializeField] firstName và private lastName.

```
public class School : MonoBehaviour
{
    public Person principal;
}
```

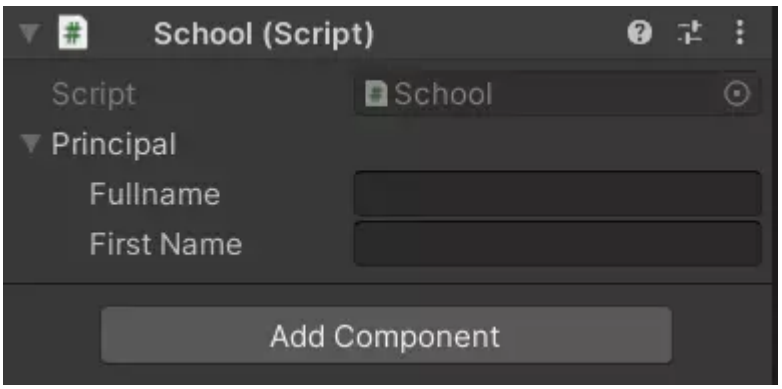
Các bạn cũng có thể đoán được là nó không xuất hiện trên Inspector như tiêu đề.




School Component

```
[System.Serializable]
public class Person {
    public string fullname;
    [SerializeField] string firstName;
    private string lastName;
}
```

Sau khi mình thêm [System.Serializable] hoặc [Serializable] nếu các bạn using System;, thì Unity đã có thể hiểu để serialize.




School Component sau khi sử dụng Serializable cho Person
Ngoài ra để đảm bảo một thực thể (instance) của custom class có thể xuất hiện trên Inspector, thì class đó cần phải tuân theo các quy định về serialization:

Có attribute [Serializable]
Không phải static

Không được là abstract
Không phải generic
Để hiểu rõ hơn về serialization là gì và làm gì thì các bạn có thể click vào link ở mục 3 (mục dưới)

3. HideInInspector attribute, giấu biến public trên inspector
Cái này trái ngược với [SerializeField], attribute này có khả năng ẩn các biến public VÀ các biến được đánh dấu [SerializeField] trên inspector, tức là độ ưu tiên cao hơn [SerializeField]

```
public class School : MonoBehaviour
{
    [HideInInspector] public Person principal;
}
```

Tuy nhiên thuộc tính này không làm mất khả năng serialization của biến.

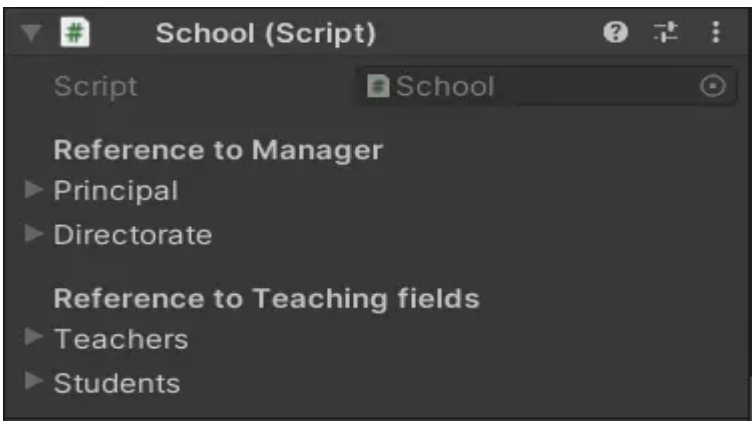
“Khả năng serialization của biến là gì?”

Tham khảo bài viết: Script Serialization trong Unity3D

4. Sử dụng attributes Header, Space, Range, TextArea… với inspector
Các attributes ở dưới giúp chúng ta tổ chức lại các biến xuất hiện trên inspector một cách có tổ chức, tăng khả năng đọc hiểu (readability).

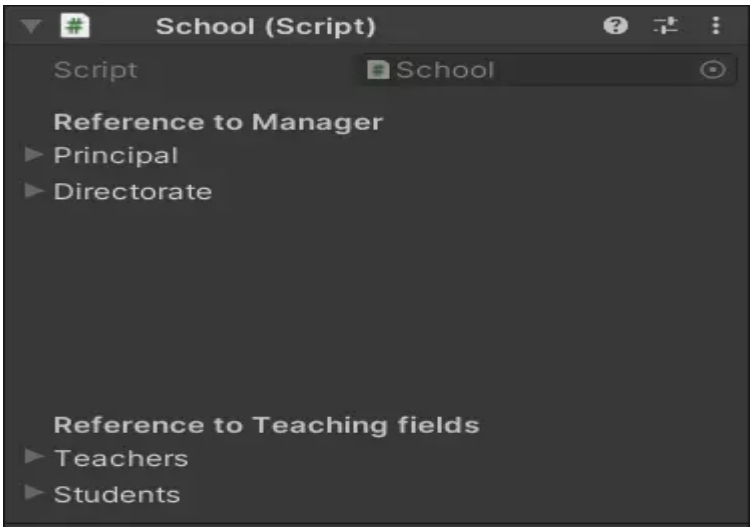
Header attribute: Tạo tiêu đề cho các biến

```
public class School : MonoBehaviour
{
    [Header("Reference to Manager")]
    public Person principal;
    public Person directorate;

    [Header("Reference to Teaching fields")]
    public Person[] teachers;
    public Person[] students;
}
```



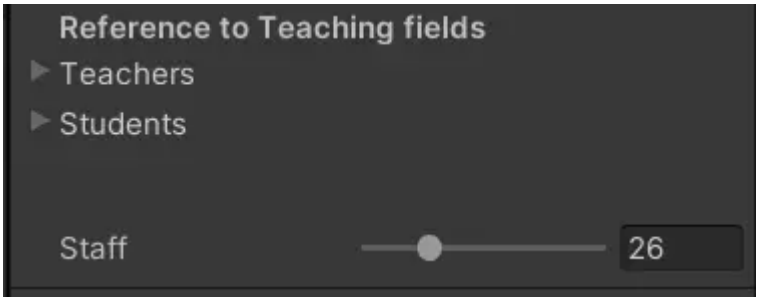
Sử dụng attribute \[Header\]
Space: tạo khoảng trống trên inspector

```
[Header("Reference to Manager")]
public Person principal;
public Person directorate;

[Header("Reference to Teaching fields")]
[Space(100)]
public Person[] teachers;
public Person[] students;
```




Sử dụng Space attribute
Range: tạo slider cho giá trị của biến, phù hợp với các biến có giá trị thuộc [min, max]

```
[Header("Reference to Teaching fields")]
public Person[] teachers;
public Person[] students;

[Space(30)]
[Range(1, 100)]
public int staff;
```

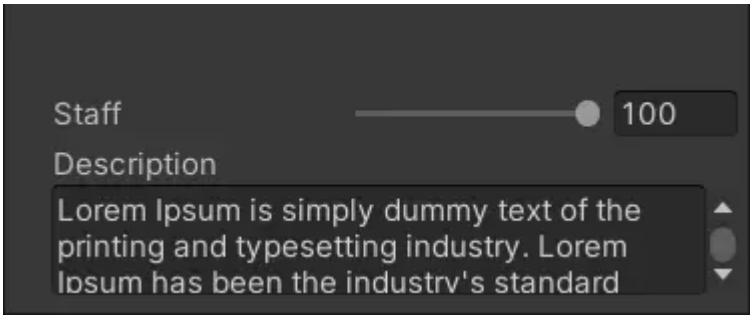



Tạo thanh slider cho staff giá trị từ 1 – 100
TextArea: tạo một container cho string, có chiều cao linh hoạt và khả năng cuộn (scrollable)

```
[Range(1, 100)]
```

public int staff;

[TextArea]
public string description;




TextArea cho biến string Description
Tooltip: cung cấp mô tả cho biến trên inspector
[TextArea]
[Tooltip("This is description for this school")]
public string description;

Khi đưa chuột hover (trỏ) vào biến Description này trên inspector, xuất hiện tooltip: