Ajouter de nouvelles methods à un objet

kazuma

Salut à tous, en ce jour de l'an 2021 , j'ai décidé de vous faire part ma dernière découverte du jour.

Aujourd'hui je vais vous montrer une façon de carrément rajouter une méthode à l'objet player en LUA grâce à une ligne de code

Comment ça une méthode ?

Petite introduction, depuis le début j'ai parlé de "méthode" car cette fois-ci nous nous attaquons à un objet et au niveau des termes techniques on dit qu'un objet est composé de méthode mais si on résume grossièrement :

Une fonction est une fonction, une méthode est une fonction associée à un objet.
Voilà, merci, aurevoir.

Utilisation

Pour l'exemple on va créer une fonction qui retire tout l'équipement actuellement sur le joueur.
On va voir la version 1 : Avec fonction
On va voir la version 2 : Avec méthode

Version 1 : Utiliser une fonction

```
-- VERSION 1
-- Fonction custom pour retirer tout l'équipement du joueur1
local function RemoveEquipment(player)
    for index = 0, 18 do
        local equipment = player:GetEquippedItemBySlot(index)
        if (equipment ~= nil) then
            player:SetData(index, equipment:GetEntry())
            player:RemoveItem(equipment, 1)
        end
    end
end

-- Appel de la fonction custom pour l'objet player
local function trigger(event, player, command)
    if (command == "toto") then
     RemoveEquipment(player)
    end
    return false
end
RegisterPlayerEvent(42, trigger)
```

Version 2 : Utiliser une méthode

```
-- VERSION 2
-- Ajout d'une methode à l'objet Player
rawset(_G['Player'],'RemoveEquipment', function(player) for index = 0, 18 do local equipment = player:GetEquippedItemBySlot(index) if (equipment ~

-- Appel de la méthode de l'objet Player
local function trigger2(event, player, command)
    if (command == "toto") then
```

```
        player:RemoveEquipment()
    end
    return false
end

RegisterPlayerEvent(42, trigger2)
```

Dans les deux version :
On a deux gros blocs dans notre script :

- La fonction / La méthode
- L'appel

Comment ça fonctionne

La nouveauté est qu'on donne une méthode à un objet.
la ligne qui s'en charge est :

```
rawset(_G['Player'],'RemoveEquipment', function(player) for index = 0, 18 do local equipment = player:GetEquippedItemBySlot(index) if (equipment ~
```

Concrètement, on insère une fonction dans un table.
L'objet Player est un objet mais en LUA beaucoup de choses sont gérées par des tables. Enfaite l'objet Player un table.

Quand on print : _G , on obtient TOUTE les variables globales de LUA.
On y retrouve ainsi toute les classes qui compose l'API Eluna.
Donc _G['Player'] , _G['Creature'], _G['Spell'], _G['Map'] etc... existent.
Et _G['Player'] est composé à son tour de toutes les méthodes de la classe Player, comme ceci :

- _G['Player']['AddComboPoints'] = function
- _G['Player']['AddItem'] = function

Tout cela est un table géant qui au final contient des fonctions. Donc on peut très bien créer nos propres méthodes pour l'objet Player, tout comme l'objet Spell, Map etc...

Et on fait tout cela grâce à une affectation du table : _G['Player']

De la manière que vous le voulez !
J'ai fais un rawset pour changer mais vous pouvez très bien faire ceci :

```
_G['Player']['RemoveEquipment'] = function(player)
    for index = 0, 18 do
        local equipment = player:GetEquippedItemBySlot(index)
        if (equipment ~= nil) then
            player:SetData(index, equipment:GetEntry())
            player:RemoveItem(equipment, 1)
        end
    end

end
```

Cela fonctionnera aussi.

Avantage / Inconvénient

Avantage :
D'après moi je pense que l'avantage est que ça nous permet de garder une même mise en forme où on appelle que des méthodes pour les objets, de plus je trouve ça plus pertinent d'en faire une méthode si vous réalisez une fonction destinée à UN objet. Je dirais que le code est un peu plus "beau".

Inconvénient :
Vous devrez tout de même garder une ligne qui insère la fonction comme étant une méthode pour l'objet. Cela dit, comme dans mon exemple vous pouvez bien synthétiser la ligne pour n'en former qu'une.

Voici le lien de téléchargement
J'espère que cela vous sera utile ! □

---

iThorgrim

Super tutoriel bien que très complexe à comprendre si on débute □
D'ailleurs je ne sais pas pourquoi tu utilises les global alors que tu pourrais gagner du temps en faisant ceci :

```
function Player:sendNotif()
  self:SendNotification('Coucou')
end

local function onCommand(event, player, command)
  if command == 'test' then
    player:sendNotif()
    return false
  end
end
RegisterPlayerEvent(42, onCommand)
```

Cela fonctionne tout aussi bien et sans devoir utiliser les Globals ;)

---

kazuma

> iThorgrim
Oui c'est vrai que pour un débutant c'est difficile à comprendre
Bah je ne savais même pas qu'on pouvais s'en servir comme ça, bah nice du coup !

---

noc

Bravo à vous, un tutoriel et trois méthodes différentes, merci beaucoup

---