Une erreur est survenue lors du chargement de la version complète de ce site. Veuillez vider le cache de votre navigateur et rafraîchir cette page pour corriger cette erreur.

Modificateur de rate d'EXP

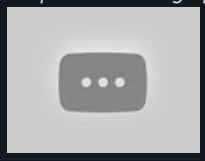
iThorgrim

Créer un modificateur de rates exp

Ma nouvelle participation au défi des 5 tutoriels !

Aujourd'hui nous allons apprendre à créer un modifcateur de rates exp et comme j'en ai dédiés une vidéo sur ma chaîne Youtube je me permet de vous mettre un petit lien si les vidéos c'est votre truc :)

(Oubliez pas le petit abonnement, la cloche toosa toosa)
Cliquez sur l'image pour voir la vidéo



D'ailleurs c'est un tutoriel en Lua (sisi), du coup si vous ne savez rien du Lua je vous invites à suivre les guides que j'ai fait pour vous initier au Lua.

Pour commencer il va nous falloir un éditeur de texte, un core qui prend en charge Eluna et un petit peu de patience :)

On va commencer par ce faire une déclaration de variable sous forme de tableau, pour rendre nos fonction local plus rapidement : local mod_exp = {}

```
local mod_exp = {}
```

Nous allons ensuite mettre en place la création de table pour stocker les informations du joueurs, là en l'occurence sa rate d'experience.

```
local mod_exp = {
    dbname = 'R1_Eluna',
}


CharDBExecute('CREATE DATABASE IF NOT EXISTS '..mod_exp.dbname)
CharDBExecute('CREATE TABLE IF NOT EXISTS `'..mod_exp.dbname..'`.`characters_exp` ( `guid` INT(10) NOT NULL, `rate` INT(10
```

Là ce qu'il se passe c'est qu'a chaque allumage du serveur le code suivant seras lu et il nous créera la base de donnée que nous avons choisis dans `mod_exp.dbname` ainsi que la table `character_exp` .

Maintenant nous allons utiliser l'event 'Lorsque le joueur se connecte' pour récupérer les informations relative à notre joueurs, pour ce faire nous allons utiliser les RegisterPlayerEvent et plus précisément le numéro 3.

```
-- Nous ajoutons 'mod_exp' devant onLogin pour que notre variable soit local
-- étant donnés que notre tableau 'mod_exp' est lui déjà local cela se "propage"
-- maintenant notre fonction 'onLogin' est une fonction contenu dans le tableau 'mod_exp'
function mod_exp.onLogin(event, player)
    -- Nous stockons le guid du joueur dans une variable
    local pGuid = player:GetGUIDLow()
    -- Nous stockons le résultat de notre requête de recherche des informations
    local getRate = CharDBQuery('SELECT rate FROM `'..mod_exp.dbname..'`.`characters_exp` WHERE guid = '..pGuid)
```

```
    -- Si nous n'avons pas de résultant alors
    if not getRate then
        -- Nous insérons une ligne par défaut pour que le joueur ai accés à ses informations
        CharDBExecute('INSERT INTO `'..mod_exp.dbname..'`.`characters_exp` VALUE ('..pGuid..', 1);')
        -- Puis nous allons ajouter 'attachés' une variable à notre objet player qui est le joueur
        -- nous donnons une valeur a cette variable qui seras le multiplicateur d'experience
        player:SetData('exp_modifier', 1)

    -- Sinon (si un résultat ressort)
    else
        -- Nous assignons notre résultant à notre variable
        player:SetData('exp_modifier', getRate:GetUInt32(0))
    end
end
-- Ici nous demandons à Eluna "d'écouter" l'event de connection d'un joueur (c'est le numéro 3)
-- Et lorsque que l'event est "entendu" on lui demande de "joueur" notre fonction
RegisterPlayerEvent(3, mod_exp.onLogin)
```

Alors, maintenant que cela est fait nous allons tout de suite faire une fonction au cas nous un développeur fait la commande `.reload eluna`, cette fonction va tout simplement appelés notre fonction `mod_exp.onLogin` et va remettr en place les valeurs des variable aux joueurs.

En gros quand quelqu'un fait la commande `.reload eluna` cela va vider tout les tableau et notre fonction `player:SetData` attache une colonne au tableau / objet player. C'est assez compliqués d'expliquer ça sans rentrer dans le détail mais en gros :

Player est un tableau, quand nous faisons `player:SetData` on ajoute une propriété à notre tableau qui porteras le nom du premier argument, dans notre cas `exp_modifer` et on donne donc une valeur à notre propriété qui est donc le second argument `1`.

Et lorsqu'un maître du jeu ou un développeur fait la commande de reload cela vide totalement le talbeau player des propriétés que nous avons plus lui attacher.

```
function mod_exp.getPlayersInformations(event)
    -- Nous faisons une boucle "for" sur le résultat de la fonction "GetPlayersInWorld" celle-ci nous retourne un tableau
    for _, player in pairs(GetPlayersInWorld()) do
        -- Pour chaque résultat nous envoyons l'objet player à notre fonction "onLogin" afin de remettre en place les infc
        mod_exp.onLogin(event, player)
    end
    -- Et on envoie un petit message pour dire que le système est allumés
    io.write('Eluna :: Mod_EXP System start \n')
end
RegisterServerEvent(33, mod_exp.getPlayersInformations)
```

Maintenant il nous faut mettre en place un GossipMenu pour choisir notre multiplicateur d'experience de la manière la plus propre.

Pour ce faire nous allons ajouter une propriété à notre tableau `mod_exp` qui seras donc la rate maximum possible à mettre en place.

```
local mod_exp = {
    dbname = 'R1_Eluna',
    max_rate = 5
}
```

Et ensuite l'on met en place le GossipMenu sur une créature déjà créés à cet effet (je ne vous apprendrais pas à créer une créature, tout ce que je peu vous dire c'est qu'il faut qu'elle est au moins le flag 1).

```
function mod_exp.onGossipHello(event, player, object)
    -- Nous mettons en place une boucle "for i" qui suivras donc les chiffre de 1 à mod_exp.max_rate (5)
    for i=1, mod_exp.max_rate, 1 do
        -- On affiche donc la sélection sur le menu du PNJ
        player:GossipMenuAddItem(0, "Multiplicateur d'experience x"..i, 1, i)
```

```
    end
    -- Cela envoie à notre joueur les informations du gossip.
    -- En l'occurence les sélection possible
    player:GossipSendMenu(1, object)
end
-- Ici le "197" c'est l'entry de votre PNJ
RegisterCreatureGossipEvent(197, 1, mod_exp.onGossipHello)
```

Comment on peu le voir sur mon PNJ de test, tout vas bien pour le moment, mais ce n'est pas finis.

Il nous faut désormais effectuer la seconde partie du GossipMenu, le traitement des informations et des réponses. Pour faire simple lorsque vous cliquez sur un des choix, une action doit se passer pour mettre en place le multiplicateur demandés.
C'est que nous allons faire.

```
function mod_exp.onGossipSelect(event, player, object, sender, intid, code, menuid)
    -- Ici notre traitement est très spéciale, comme nous n'avons que 5 choix qui vont de 1 à 5 nous allons simplement dir
    -- En tant normal il aurais fallut comparer "intid" avec une valeur par exemple "intid == 2" etc.
    -- Mais pour notre modificateur de rates pas besoin
    if intid then
        -- Nous allons attribués la nouvelle valeur de 'exp_modifier' au choix du joueur donc à la valeur de l'intid
        player:SetData('exp_modifier', intid)
        -- Là nous envoyons une notification en jeu pour dire au joueur que sont choix est appliqués
        player:SendNotification('Votre multiplicateur est bien modifiés')
        -- Et on finis par fermer le gossipMenu
        player:GossipComplete()
    end
end
-- Ici le "197" c'est l'entry de votre PNJ
RegisterCreatureGossipEvent(197, 2, mod_exp.onGossipSelect)
```

Maintenant nous allons "écouter" l'event qui est déclenché lorsque que le joueur gagne de l'experience pour modifier sa valeur et en envoyant la nouvelle valeur.
Notre calcul va être simple : `amount * player:GetData('exp_modifier')`

```
function mod_exp.onGiveExp(event, player, amount, victim)
    local pExp = player:GetData('exp_modifier')

    amount = amount * pExp
    return amount
end
RegisterPlayerEvent(12, mod_exp.onGiveExp)
```

Là je ne pense pas avoir besoin de commenter, la ligne reste très simple, on renvoie simplement le nouveau montant d'experience.

Et pour finis il nous faut sauvegarder les informations dans la base de donnée quand le joueur se deconnecte.

```
function mod_exp.onLogout(event, player)
    local pGuid = player:GetGUIDLow()
    local pExp = player:GetData('exp_modifier')

    CharDBExecute('UPDATE `'..mod_exp.dbname..'`.`characters_exp` SET rate = '..pExp..' WHERE guid = '..pGuid)
end
RegisterPlayerEvent(2, mod_exp.onLogout)
```

N'oublions notre fonction de sauvegarde global avant le reload d'eluna.
Ce qui est bien avec Eluna c'est qu'ils ont mis en place un Event avant son reload lorsqu'il est demandés, ainsi nous pouvons l'utiliser pour sauvegarder les informations de tout le monde juste avant un reload.

```
function mod_exp.setPlayersInformations(event)
    for _, player in pairs(GetPlayersInWorld()) do
        mod_exp.onLogout(event, player)
    end
end
RegisterServerEvent(16, mod_exp.setPlayersInformations)
```

Je ne pense pas avoir besoin de commenter plus longtemps les lignes de code étant donnés que (je pense) que vous avez compris le principe.

Le code complet

Pour ceux qui demande le code complete.

```
local mod_exp = {
    dbname = 'R1_Eluna',
    max_rate = 5
}


CharDBExecute('CREATE DATABASE IF NOT EXISTS '..mod_exp.dbname)
CharDBExecute('CREATE TABLE IF NOT EXISTS `'..mod_exp.dbname..'`.`characters_exp` ( `guid` INT(10) NOT NULL, `rate` INT(10

-- Nous ajoutons 'mod_exp' devant onLogin pour que notre variable soit local
-- étant donnés que notre tableau 'mod_exp' est lui déjà local cela se "propage"
-- maintenant notre fonction 'onLogin' est une fonction contenu dans le tableau 'mod_exp'
function mod_exp.onLogin(event, player)
    -- Nous stockons le guid du joueur dans une variable
    local pGuid = player:GetGUIDLow()
    -- Nous stockons le résultat de notre requête de recherche des informations
    local getRate = CharDBQuery('SELECT rate FROM `'..mod_exp.dbname..'`.`characters_exp` WHERE guid = '..pGuid)

    -- Si nous n'avons pas de résultant alors
    if not getRate then
```

```
        -- Nous insérons une ligne par défaut pour que le joueur ai accés à ses informations
        CharDBExecute('INSERT INTO `'..mod_exp.dbname..'`.`characters_exp` VALUE ('..pGuid..', 1);')
        -- Puis nous allons ajouter 'attachés' une variable à notre objet player qui est le joueur
        -- nous donnons une valeur a cette variable qui seras le multiplicateur d'experience
        player:SetData('exp_modifier', 1)

        -- Sinon (si un résultat ressort)
    else
        -- Nous assignons notre résultant à notre variable
        player:SetData('exp_modifier', getRate:GetUInt32(0))
    end
end
-- Ici nous demandons à Eluna "d'écouter" l'event de connection d'un joueur (c'est le numéro 3)
-- Et lorsque que l'event est "entendu" on lui demande de "joueur" notre fonction
RegisterPlayerEvent(3, mod_exp.onLogin)

function mod_exp.getPlayersInformations(event)
    -- Nous faisons une boucle "for" sur le résultat de la fonction "GetPlayersInWorld" celle-ci nous retourne un tableau
    for _, player in pairs(GetPlayersInWorld()) do
        -- Pour chaque résultat nous envoyons l'objet player à notre fonction "onLogin" afin de remettre en place les infc
        mod_exp.onLogin(event, player)
    end
    -- Et on envoie un petit message pour dire que le système est allumés
    io.write('Eluna :: Mod_EXP System start \n')
end
RegisterServerEvent(33, mod_exp.getPlayersInformations)

function mod_exp.OnGossipHello(event, player, object)
    -- Nous mettons en place une boucle "for i" qui suivras donc les chiffre de 1 à mod_exp.max_rate (5)
    for i=1, mod_exp.max_rate, 1 do
        -- On affiche donc la sélection sur le menu du PNJ
        player:GossipMenuAddItem(0, "Multiplicateur d'experience x"..i, 1, i)
    end
    -- Cela envoie à notre joueur les informations du gossip.
    -- En l'occurence les sélection possible
    player:GossipSendMenu(1, object)
end
-- Ici le "197" c'est l'entry de votre PNJ
RegisterCreatureGossipEvent(197, 1, mod_exp.OnGossipHello)

function mod_exp.onGossipSelect(event, player, object, sender, intid, code, menuid)
    -- Ici notre traitement est très spéciale, comme nous n'avons que 5 choix qui vont de 1 à 5 nous allons simplement dir
    -- En tant normal il aurais fallut comparer "intid" avec une valeur par exemple "intid == 2" etc.
    -- Mais pour notre modificateur de rates pas besoin
    if intid then
        -- Nous allons attribués la nouvelle valeur de 'exp_modifier' au choix du joueur donc à la valeur de l'intid
        player:SetData('exp_modifier', intid)
        -- Là nous envoyons une notification en jeu pour dire au joueur que sont choix est appliqués
        player:SendNotification('Votre multiplicateur est bien modifiés')
        -- Et on finis par fermer le gossipMenu
        player:GossipComplete()
    end
end
-- Ici le "197" c'est l'entry de votre PNJ
RegisterCreatureGossipEvent(197, 2, mod_exp.onGossipSelect)

function mod_exp.onGiveExp(event, player, amount, victim)
    local pExp = player:GetData('exp_modifier')
```

```
    amount = amount * pExp
    return amount
end
RegisterPlayerEvent(12, mod_exp.onGiveExp)

function mod_exp.onLogout(event, player)
    local pGuid = player:GetGUIDLow()
    local pExp = player:GetData('exp_modifier')

    CharDBExecute('UPDATE `'..mod_exp.dbname..'`.`characters_exp` SET rate = '..pExp..' WHERE guid = '..pGuid)
end
RegisterPlayerEvent(2, mod_exp.onLogout)

function mod_exp.setPlayersInformations(event)
    for _, player in pairs(GetPlayersInWorld()) do
        mod_exp.onLogout(event, player)
    end
end
RegisterServerEvent(16, mod_exp.setPlayersInformations)
```

Bonus pour les confirmés

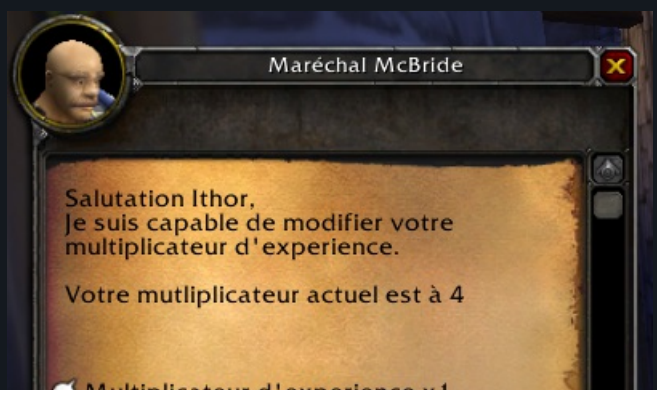
Si vous souhaitez ajouter un texte au gossip Menu je vous conseil d'utiliser l'extensions GossipSetText disponible en [cliquant ici] (**Scripts/GossipTextExtension.lua at master · ElunaLuaEngine/Scripts · GitHub**).

Ainsi dans votre fonction `onGossipHello` vous pourrez avoir quelque chose dans ce style :

```
function mod_exp.OnGossipHello(event, player, object)
    -- Nous ajouton un texte customisés, celui ci ne fonctionne qu'avec l'extension GossipTextExtension
    local pName = player:GetName()
    local pExp = player:GetData('exp_modifier')

    player:GossipSetText("Salutation "..pName..",\nJe suis capable de modifier votre multiplicateur d'experience.\n\nVotre

    -- Nous mettons en place une boucle "for i" qui suivras donc les chiffre de 1 à mod_exp.max_rate (5)
    for i=1, mod_exp.max_rate, 1 do
        -- On affiche donc la sélection sur le menu du PNJ
        player:GossipMenuAddItem(0, "Multiplicateur d'experience x"..i, 1, i)
    end
    -- Cela envoie à notre joueur les informations du gossip.
    -- En l'occurence les sélection possible
    player:GossipSendMenu(0x7FFFFFFF, object)
end
-- Ici le "197" c'est l'entry de votre PNJ
RegisterCreatureGossipEvent(197, 1, mod_exp.OnGossipHello)
```



Voilà j'espère que ce tutoriel vous seras utile ! Je vous dis à la prochaine pour un prochain tutoriel !