

Creating A Custom Fragment Generator Lua

It is possible to use Lua to develop a simple or new fragment generator for additional events.

First create a new folder in `/Isets/` named fragment-generators should be like `/Isets/fragment-generators/`. Once that is done, you should create a new file. You can name it whatever you want. For this guide, I will name it `example-generator.lua`.

Creating the generator file

We will create a generator for the `BlockBreakEvent`. It will be similar to the default generator provided by the plugin.

```
package.path = pluginFolder .. "\\fragment-generators\\listeners\\?.lua"

namespace = "example"
source = "blockBreakEvent"
require "BlockBreakEvent"

function handleGeneration(player, settingsMap)
    if settingsMap:getBoolean("Enabled") and math.random() <= settingsMap:getDouble("Chance")
    then
        local amount = (settingsMap:getDouble("Amount_To_Give") <= 0) and 1 or
settingsMap:getDouble("Amount_To_Give")
        local fragment = utils.findFragment(player)
        if
        if fragment ~= nil then
            if settingsMap:getBoolean("Physical") then
                givePhys(player, amount, fragment)
            else
                giveVirtual(player, amount, fragment)
            end
        end
    end
end
```

```

end
end

end

function givePhys(player, amount, fragment)
    fragment: giveOrUpdateFragment(player, math.floor(amount), false)
    utils.tell(player, "&2You been given " .. amount .. " &6fragments &2for &6" ..
fragment: getArmorSetName())
end

function giveVirtual(player, amount, fragment)
    local cache = utils.getCache(player)
    local currentBalance = cache: getFragmentAmount( fragment: getArmorSetName())
    cache: updateFragmentAmount( fragment. armorSetName, currentVirtualBalance +
math.floor(amount))
    utils.tell(player, "&2You been given " .. amount .. " &6fragments &2for &6" ..
fragment: getArmorSetName())
end

```

Registering an event listener

To actually create a listener for the `BlockBreakEvent`, you should first create a folder inside `/Isets/`. In this example, we are name the folder 'listeners'; `/Isets/fragment-generators/listeners/`. Create a new file and in this case, we name it `BlockBreak.lua`.

```

utils.subscribeToEvent("org. bukkit. event. block. BlockBreakEvent", function(event)
    local player = event:getPlayer()
    local armorSet =
utils.findArmorSet(utils.getCache(player): getFragmentDataManager(): getArmorSetFragmentGen())

    if armorSet ~= nil then
        local type = armorSet: getFragmentGeneration(): getString("Type")
local source = armorSet: getFragmentGeneration(): getString("Source")
        local fragmentGenerator = utils.findFragmentGenerator(type, source)

        if fragmentGenerator == nil then
            return
        end
    end
end

```

```

if type == namespace then
  local disabledWorlds = armorSet:getFragmentGeneration():getStringList("Disabled_Worlds")
  local worldName = player:getWorld():getName()
  for i = 0, disabledWorlds:size() - 1 do
    if disabledWorlds:get(i) == worldName then
      return
    end
  end
  fragmentGenerator:handleGeneration(player, armorSet:getFragmentGeneration())
end
end)

```

Utils package

The utils package has a few utility functions.

Util to register an event listener.

```

subscribeToEvent("path to event class", function(event)
  -- Code here
end)

```

Util to retrieve an armor set. If none was located, the function will return nil.

```

findArmorSet("name of the armor set")

```

Util to retrieve a fragment generator. If no fragment generator was discovered, the function will return nil.

```

findFragmentGenerator("type/namespace", "source")

```

Util to retrieve a player's cache. It will return nil if the cache cannot be located; if it can, it will return the cache.

```
getCache(player)
```

Use this to determine which fragment generator a player currently has enabled. If "none" is returned, then they did not enable one. The name of the armor set for which they activated the generator for will be returned.

```
findFragment(player)
```

Util to send a message to a player. It will auto translate colors.

```
tell(player, "message goes here color codes and hex is automaticly translated")
```

Revision #7

Created 2 November 2023 17:03:04 by Dragon

Updated 18 March 2024 06:52:26 by Dragon