B0 - HUB

B-INN-000

Workshop Plugin Minecraft

Créer son premier plugin Minecraft

Workshop Plugin Minec

language: java
build tool: gradle

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Workshop plugin Minecraft

Pré-requis

Si vous n'avez pas déjà installé les outils nécessaires à ce workshop, suivez les instructions situées ici :

https://github.com/alwyn974/workshop-mc-plugin/blob/master/REQUIREMENTS.md

Configuration

Création d'une task gradle

Pour pouvoir tester votre plugin, vous devez créer une task gradle, pour déplacer le plugin dans le dossier `server/plugins/`
Cette task aura pour nom `buildAndMove` devra dépendre de `build` et respecter les prérequis suivants :

- Etre exécuté en dernier
- Utiliser la task copy pour copier le plugin généré par le build dans le dossier `./server/plugins/`

Task avec gradle Script basique, Doc de la classe Task

Utilisez `String.format` pour récupérer le nom du jar plus facilement, avec `archivesBaseName` et `version`

Création de la classe principale

Créez votre propre package, moi, ça sera `re.alwyn974.plugin.workshop`, dans ce package créer une classe `PluginNamePlugin` (ex: WorkshopPlugin).
Cette classe devra hériter de `JavaPlugin` de `org.bukkit.plugin.java` et surcharger les méthodes `onEnable()` et `onDisable()`
Dans chacune des méthodes ajouter simplement un message dans la console.

Création du fichier `plugin.yml`

Dans le dossier `src/main/resources` créez le fichier `plugin.yml` avec le contenu suivant :

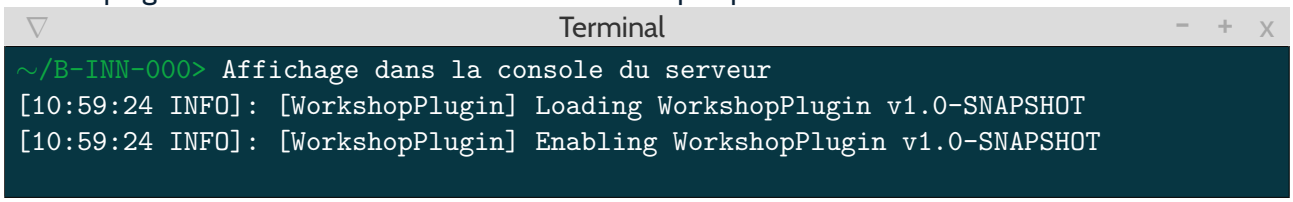
```
name: Plugin Name
version: 1.0
author: Your Name
main: com.github.pluginname.Main
```

Remplacez le `main` par votre classe qui hérite de `JavaPlugin` et le `name` par le nom de votre plugin.

Plus d'informations sur le plugin.yml

Build

Pour tester votre plugin, vous devez exécuter la task gradle `buildAndMove` et ensuite lancer le docker-compose.
(Lancez `buildAndMove` depuis l'onglet gradle à droite, ou en exécutant `./gradlew buildAndMove` dans le dossier du projet)
Si votre plugin est reconnu dans la console vous aurez quelque chose comme ceci :



Utilisation d'events

Comme avec Forge, les plugins utilisent aussi des events.
Créez une classe `Events` qui implementera `Listener` de `org.bukkit.event`.
Pour chaque méthode qui recevra un event il faudra ajouter une annotation spécifique.

Comment utiliser les events spigot Documentation

Modifier le message de connexion d'un joueur

En utilisant l'event `PlayerJoinEvent` créez une méthode `onPlayerJoin` qui recevra un event `PlayerJoinEvent`. Lorsqu'un joueur rejoindra le serveur il faudra afficher un nouveau message que `PlayerName joined the game`.

Exemple: `[+] alwyn974 - 1/20`

Si vous voulez ajouter des couleurs au message vous pouvez utiliser la classe `ChatColor` de `org.bukkit.ChatColor` ou utiliser les colors codes.

Color Codes.
Javadoc de ChatColor

Attention, pour utiliser les colors code, il faudra quand même utiliser la méthode qui permet de les traduires et qui provient de `ChatColor`. Sinon, il faudra utiliser le symbol §

Modifier le message de déconnexion d'un joueur

En utilisant l'event `PlayerQuitEvent` créez une méthode `onPlayerQuit` qui recevra un event `PlayerQuitEvent`. Lorsqu'un joueur quittera le serveur il faudra afficher un nouveau que `PlayerName left the game`.

Exemple: `[-] alwyn974 - 0/20`

Avec de la couleur, c'est toujours mieux !

Création d'une commande simple

Pour créer une commande, il faut créer une classe qui implémente `CommandExecutor` de `org.bukkit.command` et surcharger la méthode `boolean onCommand`.

Créez une classe `EpitechCommand` dans le package `command` qui hérite de `CommandExecutor` et surcharge la méthode.
La commande devra :

- Lorsqu'elle est exécutée répondre à l'utilisateur avec le message `Marvin -42` en cyan
- Elle devra être utilisable par tout le monde
- Elle devra avoir aucune permission
- Lorsque l'on fait /help epitech, il faudra que la description soit: `Vous donne -42 de la part de Marvin :)`

Exemple depuis la console :

```
[12:21:39 INFO]: --------- Help: /epitech ----------------------
[12:21:39 INFO]: Description: Vous donne -42 de la part de Marvin
[12:21:39 INFO]: Usage: /epitech
```

Documentation spigot pour créer une commande : lien

Modification des drops avec les Events

Si on faisait un miniplugin uhc ? Pour cela il faudra utiliser un event qui est appelé lorsque :

- Un block est cassé
- Un mob est tué (passif/hostile)
- Un craft est réalisé

Liste des events spigots: lien

Drop des blocks

Pour chaque block de minerais :

- Changer le drop du charbon pour donner des torches (4) et un de charbon
- Changer les drops de chaque minerai pour donner son item cuit
- Donner une quantité aléatoire d'xp

Pour les blocks de gravier et de feuille :

- Si un block de gravier est cassé, dropper un `Flint` avec un pourcentage de chance de 50%
- Si un block de feuile d'arbre est cassé, dropper une `Golden Apple` avec un pourcentage de chance de 1%

Exemple:

```
Minerais de charbon => 4 torches + un charbon
Minerais de fer => 1 fer en lingot
Minerais d'or => 1 or en lingot

Gravier => 50% de chances de drop un `Flint`
Feuille d'arbre => 1% de chances de drop une `Golden Apple`
```

Ne pas oublier d'annuler l'event, sinon vous aurez encore l'ancien drop

Drop des mobs

Pour chaque mob passif tué (vache, poulet, cochon, lapin) :

- Changer le drop de chaque mob passif pour donner son item cuit directement

Ne pas oublier de clear la liste des drops

Modification des crafts

Vous allez devoir changer le résultat d'un craft lorsqu'on essayera de craft un item.
Ceux-ci devront être transformés, pour y ajouter des enchantements
Exemple:

```
Armure de cuir => Armure de cuir avec un enchantement de protection de niveau 4
Armure de fer => Armure de fer avec un enchantement de protection de niveau 3
Armure en or => Armure en or avec un enchantement de protection de niveau 2
Armure de diamant => Armure de diamant avec un enchantement de protection de niveau 1

Outils en pierre => Outils en pierre avec un enchantement d'`efficiency` de niveau 5
Outils en or => Outils en or avec un enchantement d'`efficiency` de niveau 4
Outils en fer => Outils en fer avec un enchantement d'`efficiency` de niveau 3
Outils en diamant => Outils en diamant avec un enchantement d'`efficiency` de niveau
    2
```

Création d'un fichier de config

Pour créer un fichier de config, il suffit de créer notre fichier `config.yml` qu'on mettra dans le dossier `src/main/resources`.
Dans le fichier `config.yml` ajouter une ligne `motd` qui contiendra le message de bienvenue.

Dans votre classe qui hérite de `JavaPlugin` il faudra dans le `onEnable` ajouter un appel à la méthode `saveDefaultConfig` pour que notre configuration soit créer au premier lancement du plugin.

Maintenant, envoyez ce `message of the day (motd)` au joueur lorsqu'il se connecte.

Pour récupérer plus facilement la configuration du plugin, créez un singleton de votre classe qui hérite de `JavaPlugin`.

Qu'est-ce qu'un singleton ? Lien

Pour récupérer la configuration en dehors de la classe qui hérite de `JavaPlugin`, il faudra utiliser la méthode `getConfig` de `org.bukkit.plugin.Plugin`

Documentation spigot sur les fichiers de configuration : lien

Commande /spawn et /setspawn

Commande /spawn

On va ajouter une catégorie à notre fichier `config.yml`

```
spawn:
  world: world # le monde dans lequel est le spawn
  x: 42 # la position x du spawn
  y: 84 # la position y du spawn
  z: 42 # la position z du spawn
```

Vous allez devoir créer une commande `/spawn` qui va permettre de téléporter le joueur au spawn en fonction de la configuration.
Créez une classe `SpawnCommand` qui implémentera `CommandExecutor` et qui surcharge la méthode `onCommand`
Pour cette commande, il faudra :

- Vérifier que c'est bien un joueur qui execute la commande (`Player` de `org.bukkit.entity`)
- Récupérer les coordonnées du spawn dans la configuration
- Récupérer le monde dans la configuration
- Téléporter le joueur au spawn
- Afficher un message au joueur

Vérifiez aussi si la configuration est correcte, est qu'un spawn existe bien

Noubliez pas d'ajouter la commande au `plugin.yml`

Command /setspawn

Créez une classe `SetSpawnComamnd` qui implémentera `CommandExecutor` et qui surcharge la méthode `onCommand`
Pour cette commande, il faudra :

- Vérifier que c'est bien un joueur qui execute la commande (`Player` de `org.bukkit.entity`)
- Modifier la configuration avec les coordonnées du joueur
- Afficher un message au joueur

Si vous avez bien fait les deux commandes vous pouvez maintenant faire un `/spawn` et `/setspawn`

Noubliez pas d'ajouter la commande au `plugin.yml`

Pensez aussi à appeler la méthode `saveConfig` de `JavaPlugin` sinon la configuration sera inchangée

Pour approfondir vos connaissances

- Spigot API
- Spigot Wiki

Merci

Merci d'avoir suivi ce workshop, n'hésitez pas à me contacter si vous avez des questions ou des remarques.