



# BO - 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. Si votre plugin est reconnu dans la console vous aurez quelque chose comme ceci :

```
Terminal
~/B-INN-000> Affichage dans la console du serveur
[10:59:24 INFO]: [WorkshopPlugin] Loading WorkshopPlugin v1.0-SNAPSHOT
[10:59:24 INFO]: [WorkshopPlugin] Enabling WorkshopPlugin v1.0-SNAPSHOT
```

## 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'événement `PlayerJoinEvent` créez une méthode `onPlayerJoin` qui recevra un événement `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 codes de couleur.



Color Codes.  
Javadoc de `ChatColor`



Attention, pour utiliser les codes de couleur, il faudra quand même utiliser la méthode qui permet de les traduire et qui provient de `ChatColor`. Sinon il faudra utiliser le symbole §

## MODIFIER LE MESSAGE DE DÉCONNEXION D'UN JOUEUR

En utilisant l'événement `PlayerQuitEvent` créez une méthode `onPlayerQuit` qui recevra un événement `PlayerQuitEvent`. Lorsqu'un joueur quittera le serveur il faudra afficher un nouveau message 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 feuille 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 créée 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 exécute 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 exécute 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