eBook Gratuit

APPRENEZ minecraft

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#minecraft

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: minecraft

It is an unofficial and free minecraft ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official minecraft.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec minecraft

Remarques

Cette section fournit une vue d'ensemble de ce qu'est minecraft et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de minecraft, et établir un lien avec les sujets connexes. La documentation de minecraft étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
1.0.0	2011-11-18
1.2.5	2012-04-04
1.4.7	2013-01-09
1.6.4	2013-09-19
1.7.10	2014-06-26
1.8.9	2015-12-09
1.9.4	2016-05-10
1.10.2	2016-06-23
1.11	2016-11-14

Examples

Installation de Minecraft

1. Acheter Minecraft d' ici
2. Créez un compte Mojang ou connectez-vous si vous en avez déjà un.
3. Si vous avez créé un nouveau compte, vérifiez votre courrier électronique.
4. Remplissez vos détails de paiement. Assurez-vous que vous êtes sur minecraft.net et que vous êtes sur une connexion sécurisée (HTTPS)

5. Télécharger et exécuter Minecraft
6. Ouvrez le lanceur Minecraft
7. Connectez-vous avec votre adresse e-mail si vous avez acheté Minecraft après novembre 2012 ou si vous avez migré votre nom d'utilisateur vers un compte Mojang.

   Connectez-vous avec votre nom d'utilisateur si vous avez un compte Minecraft plus ancien et que vous n'avez pas encore migré vers le nouveau format de compte. Vous pouvez migrer vers un compte Mojang à partir de la page des comptes Mojang.
8. Cliquez sur "Play" si vous voulez jouer sur la nouvelle version.
9. Apprécier le jeu.

Lire Démarrer avec minecraft en ligne: https://riptutorial.com/fr/minecraft/topic/7952/demarrer-avec-minecraft

Chapitre 2: Auditeurs d'événement à Bukkit

Examples

Créer un écouteur d'événement

Pour enregistrer vos méthodes, la classe contenant le ou les gestionnaires d'événements doit implémenter l'interface Listener.

```
import org.bukkit.event.Listener;

public final class ListenerClass implements Listener {
}
```

Vous devez enregistrer l'écouteur d'événement en ajoutant l'appel suivant à votre méthode onEnable dans la classe qui étend JavaPlugin:

```
getServer().getPluginManager().registerEvents(new ListenerClass(), this);
```

Pour écouter un événement donné dans votre classe d'écouteur, vous devez créer une méthode avec l'annotation @EventHandler sur la méthode. Le type d'événement est spécifié par le Type dans le seul argument de la méthode. La méthode peut être nommée comme vous le souhaitez.

```
import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;
import org.bukkit.event.player.PlayerLoginEvent;

public class ListenerClass implements Listener {
    @EventHandler
    public void onPlayerLogin(PlayerLoginEvent event) {
        event.getPlayer().sendMessage("Welcome to the server!");
    }
}
```

Paramètres EventHandler

L'annotation `org.bukkit.event.EventHandler` accepte quelques paramètres.

priority - Indique la priorité de votre écouteur. Il y a six priorités différentes, dans l'ordre d'exécution: LOWEST, LOW, NORMAL [défaut], HIGH, HIGHEST, MONITOR. Ces constantes font référence à l' `org.bukkit.event.EventPriority` .

Si vous voulez changer le résultat d'un événement, choisissez très soigneusement de LOWEST à HIGHEST. Suggestions de plug-ins de protection généralisés sur LOWEST, plug-ins plus spécifiques sur NORMAL, et remplacement des plug-ins sur HIGH. Si vous souhaitez agir lorsqu'un événement se produit, mais ne changez pas le résultat, utilisez MONITOR.

Remarque: la priorité MONITOR ne doit être utilisée que pour la lecture. Cette priorité est

utile pour consigner les plug-ins afin de voir les résultats d'un événement et la modification des valeurs peut interférer avec ces types de plug-ins.

ignoreCancelled - Un booléen qui indique si votre auditeur doit ou non tirer si l'événement a été annulé avant que ce soit le tour de l'auditeur de gérer l'événement. Faux par défaut.

```
import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;
import org.bukkit.event.EventPriority;
import org.bukkit.event.player.PlayerLoginEvent;

public final class LoginListener implements Listener {
    @EventHandler
    public void normalLogin(PlayerLoginEvent event) {
        // Some code here
    }

    @EventHandler(priority = EventPriority.HIGH)
    public void highLogin(PlayerLoginEvent event) {
        // Some code here
    }
}
```

Création d'événements personnalisés

Parfois, vous devez créer votre propre événement, un autre que les autres plug-ins peuvent écouter (Vault, parmi d'autres, fait cela) et même annuler. L'API d'événement de Bukkit permet cela. Tout ce que vous devez faire est de faire une nouvelle classe, faites prolonger l' `Event` , ajoutez les gestionnaires et les attributs nécessaires à votre événement (comme joueur ou un message).

```
import org.bukkit.event.Event;
import org.bukkit.event.HandlerList;

public final class CustomEvent extends Event {
    private static final HandlerList handlers = new HandlerList();
    private String message;

    public CustomEvent(String example) {
        message = example;
    }

    public String getMessage() {
        return message;
    }

    public HandlerList getHandlers() {
        return handlers;
    }

    public static HandlerList getHandlerList() {
        return handlers;
    }
}
```

Appel de votre événement personnalisé

Vous contrôlez la création et l'appel de vos événements, à vous de choisir. Voici un exemple

```
// Create the event here
CustomEvent event = new CustomEvent("Sample Message");
// Call the event
Bukkit.getServer().getPluginManager().callEvent(event);
Bukkit.getServer().broadcastMessage(event.getMessage());
```

Rappelez-vous: vous contrôlez vos événements. Si vous ne l'appelez pas et agissez en conséquence, cela n'arrive pas!

Écouter un événement personnalisé

L'écoute d'un événement personnalisé est identique à l'écoute d'un événement normal.

```
import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;

public final class CustomListener implements Listener {

    @EventHandler
    public void onCustomEvent(CustomEvent event) {
    // Some code here
    }
}
```

Rendre votre CustomEvent annulable

Si vous souhaitez que votre événement soit annulable, ajoutez simplement des `implements Cancellable` , `boolean cancelled` et un getter et un setter:

```
import org.bukkit.event.Event;
import org.bukkit.event.HandlerList;
import org.bukkit.event.Cancellable;

public final class CustomEvent extends Event implements Cancellable {
    private static final HandlerList handlers = new HandlerList();
    private String message;
    private boolean cancelled;

    public CustomEvent(String example) {
        message = example;
    }

    public String getMessage() {
        return message;
    }
```

```
    public boolean isCancelled() {
        return cancelled;
    }

    public void setCancelled(boolean cancel) {
        cancelled = cancel;
    }

    public HandlerList getHandlers() {
        return handlers;
    }

    public static HandlerList getHandlerList() {
        return handlers;
    }
}
```

Ensuite, vous devriez vérifier si un plug-in a annulé l'événement personnalisé.

```
// Create the event here
CustomEvent event = new CustomEvent("Sample Message");
// Call the event
Bukkit.getServer().getPluginManager().callEvent(event);
// Check if the event is not cancelled
if (!event.isCancelled()) {
    Bukkit.getServer().broadcastMessage(event.getMessage());
}
```

Désinscription d'événements ou d'auditeurs

Vous pouvez désinscrire des événements individuels, des classes entières d'écoute ou tous les événements enregistrés par votre plugin ou même par d'autres plugins!

Annuler l'enregistrement d'un événement spécifique

Chaque classe d'événement a la méthode statique getHandlerList (), appelez cela et vous pouvez ensuite utiliser la méthode .unregister ().

```
PlayerInteractEvent.getHandlerList().unregister(plugin);
// this will unregister all PlayerInteractEvent instances from the plugin
// you can also specify a listener class instead of plugin.
```

Vous savez maintenant pourquoi vous aurez besoin de getHandlerList () dans vos événements personnalisés.

Annuler l'enregistrement de tous les événements

En utilisant la classe HandlerList et sa méthode statique unregisterAll (), vous pouvez facilement désinscrire les événements des classes d'écoute ou des plug-ins.

```
HandlerList.unregisterAll(plugin);
// this will unregister all events from the specified plugin
// you can also specify a listener class instead of plugin.
```

Lire Auditeurs d'événement à Bukkit en ligne:
https://riptutorial.com/fr/minecraft/topic/9739/auditeurs-d-evenement-a-bukkit

Chapitre 3: Auditeurs d'événement à Forge

Examples

Création d'un écouteur d'événement dans Forge

La création d'un écouteur d'événement dans Forge est très similaire à la création d'un écouteur dans Bukket.

Créer la classe d'écoute nécessite beaucoup moins. Il n'y a pas d'interface à mettre en œuvre ou d'autres importations.

```
public class ListenerClass { } //perfectly valid event listener class
```

L'enregistrer nécessite de passer l'instance au bus d'événements Forge:

```
MinecraftForge.EVENT_BUS.register(new ListenerClass());
```

Il existe deux bus d'événements différents en fonction de l'événement. Par exemple, les événements de génération de minerai sont déclenchés sur `ORE_GEN_BUS` . Vous pouvez appeler cet enregistrement depuis n'importe où, mais il est conseillé de l'appeler à partir de votre classe principale (avec l'annotation @Mod) ou d'une classe proxy (certains événements sont uniquement côté client et un gestionnaire d'événements côté client doit être appelé uniquement). à partir du proxy client, sinon le serveur dédié tombera en panne!)

Pour écouter un événement donné dans votre classe d'écouteur, vous devez créer une méthode avec l'annotation @SubscribeEvent sur la méthode. Le type d'événement est spécifié par le Type dans le seul argument de la méthode. La méthode peut être nommée comme vous le souhaitez.

Notez que certains types d'événement sont des sous-types (qui doivent être référencés par leur type englobant, par exemple `CropGrowEvent.Pre`) et que certains événements peuvent avoir une phase lorsqu'elle est déclenchée à plusieurs endroits (comme tous les `TickEvent` qui sont déclenchés avant et après tout code vanille). En tant que modérateur, vous devez toujours vérifier ces deux éléments et ne lancer votre code que lorsque cela est nécessaire.

```
public class ListenerClass {
    @SubscribeEvent
    public void onPlayerLogin(PlayerLoggedInEvent event) {
        event.player.addChatMessage(new TextComponentString("Welcome to the server!"));
    }
}
```

Comme les mods de Forge interagissent directement avec les composants internes de Minecraft, le modder a beaucoup de pouvoir pour affecter les choses, mais de la même manière, le code doit suivre le framework vanilla: il n'y a pas de raccourci pour envoyer des messages. ITextComponents manuellement, mais la possibilité de manipuler ces objets (comme appliquer

une mise en forme de couleur) est beaucoup plus facile. Par exemple:

```
TextComponentString txt = new TextComponentString(
    TextFormatting.LIGHT_PURPLE + "Welcome to the server!");
txt.appendSibling(new TextComponentString(
    TextFormatting.AQUA + "Server has been online for " + x + " days"));
event.player.addChatMessage(txt);
```

Ce qui donne le résultat suivant:

Lire Auditeurs d'événement à Forge en ligne:
https://riptutorial.com/fr/minecraft/topic/9744/auditeurs-d-evenement-a-forge

Chapitre 4: Commandes de plugin

Examples

Bonjour commande

Dans le code ci-dessous, vous pouvez voir comment ajouter une commande à votre plugin.

MainClass.java

```
package yourpackage;

import org.bukkit.command.Command;
import org.bukkit.command.CommandSender;
import org.bukkit.plugin.java.JavaPlugin;

public class MainClass extends JavaPlugin {

@Override
public boolean onCommand(CommandSender sender, Command command, String label, String[] args) {
    if (command.getName().equalsIgnoreCase("hello")) {
        sender.sendMessage("Hey!");
    }
    return false;
}
}
```

Plugin.yml

```
name: HelloCommand
main: yourpackage.MainClass
version: 1.0
commands:
  hello:
    description: Hello
```

Lire Commandes de plugin en ligne: https://riptutorial.com/fr/minecraft/topic/9708/commandes-de-plugin

Chapitre 5: Création d'un élément de base avec Forge

Introduction

Créer un objet simple avec Forge est l'une des premières tâches qu'un aspirant modder devra apprendre. Comment faire cela a changé sur les différentes versions de Minecraft et est probablement à une difficulté "modérée" après 1.7.10 en raison du grand nombre d'erreurs faciles à faire, en particulier avec le rendu correct.

Remarques

Si quelque chose ne va pas et que votre élément personnalisé a une texture manquante (noir / violet) ou un modèle (cube par défaut trop grand lorsqu'il est maintenu), vérifiez le journal. Les problèmes de ce type apparaîtront presque *toujours* dans le journal, à condition que vous ayez enregistré les éléments correctement.

La première chose à afficher est la ligne `"Exception loading model for variant..."` ou similaire, vous indiquant quel bloc ou élément n'a pas pu être chargé correctement. Après une douzaine de lignes commençant par `at...` sera une autre ligne commençant `Caused by...`

Cette ligne "causée par" est la plus importante, elle vous indiquera quel fichier n'a pas pu être chargé correctement et peut être l'une des erreurs suivantes:

- JSON mal formé (votre fichier JSON n'est pas valide et a une faute de frappe)
- Fichier introuvable (le fichier recherché par Minecraft n'est pas correctement nommé ou au bon endroit)
- Exception de variante manquante (votre JSON Blockstate est incomplet)

Vous pouvez même obtenir plus d'une erreur d'échec pour une seule erreur! Si vous ne savez pas quel élément est le problème, commentez chaque bloc et chaque élément que vous connaissez fonctionne, réduisez la liste des erreurs au niveau du ou des blocs en question. Vous devrez peut-être le faire plusieurs fois.

De plus, une texture de mixage montre comme une liste simple toutes les ressources manquantes pour un domaine donné (ID mod).

Examples

Classe d'objets

Cette partie n'a pas beaucoup changé sur les versions de Minecraft, bien qu'il y ait eu des mutations dans les signatures de méthodes exactes ainsi que dans la hiérarchie des classes. Un élément de base (celui qui n'a pas de fonctionnalité, comme un bâton ou un lingot: c'est vrai, les

deux sont des objets à ne rien faire!)

```
public class CustomItem extends Item {

    public CustomItem () {
        super();
        this.setMaxDamage(0);
        this.setCreativeTab(CreativeTabs.MISC);
    }
}
```

Peu de place pour la personnalisation à ce stade, contrairement aux blocs. Tout ce que nous pouvons faire, c'est de changer si l'élément peut subir des dégâts (les outils l'utilisent) et dans quel onglet créatif il existera. Nom et texture que nous traiterons lorsque nous enregistrons l'élément avec GameRegistry.

Cependant, c'est tout ce qui est nécessaire pour tenir, transporter, larguer, fabriquer, fondre et utiliser autrement l'objet. Certains éléments de Minecraft (tels que les bâtons) n'ont même pas une classe unique et utilisent simplement un `new Item()` . Nous pourrions le faire ici, mais tout élément avec des fonctionnalités supplémentaires nécessitera une classe.

Modèle d'article

Comme pour les blocs, les éléments ont également besoin de modèles.

```
{
    "parent": "item/generated",
    "textures": {
        "layer0": "example:items/basic"
    }
}
```

C'est à peu près tout ce dont vous avez besoin pour fonctionner une fois que l'article est enregistré. La seule chose importante est que le nom de fichier corresponde au nom de registre utilisé pour enregistrer le bloc et doit être en minuscule (les noms de fichiers 1.11+ doivent être en minuscules, avant cela, il est juste sensible à la casse).

Notez que "layer0" est la seule texture nécessaire et il est hautement improbable qu'une autre texture soit spécifiée (bien que certains éléments tels que les potions et les armures de cuir aient un "layer1"). Tous les noms sont définis par `item/builtin` (le modèle parent le plus haut interne pour les éléments), contrairement aux blocs.

Nommez le fichier JSON modèle `my_item.json` (correspondant au nom du registre que nous allons lui donner ultérieurement) et enregistrez-le dans `src\main\resources\assets\example\models\item\` (où `example` est l'identifiant du mod spécifié dans le @Motage de votre classe de mod principal.

Créez en outre une texture pour votre élément, nommez-le `basic.png` et enregistrez-le dans `src\main\resources\assets\example\textures\items\`

Le modèle d'élément utilise ici un parent d'élément / généré, ce qui signifie que la texture fournie unique sera utilisée (comme avec la plupart des éléments non bloquants) et sera conservée dans

la main du joueur dans une orientation par défaut. Il existe également un élément / ordinateur de poche qui spécifie différentes orientations d'affichage (pour les outils). Les articles peuvent également fournir leur propre attribut "display", remplaçant ceux du parent, mais ne sont pas nécessaires dans 99,9% des cas.

Enregistrement de l'article

L'enregistrement d'éléments est effectué à partir de votre classe de mod principale ou d'une méthode de classe ModItems appelée à partir de la classe mod principale lors de préInit.

```
Item item = new CustomItem();
string registryname = "my_item";
item.setRegistryName(registryname);
item.setUnlocalizedName(item.getRegistryName().toString());
GameRegistry.register(item);
```

Il existe une raison importante d'utiliser `item.setUnlocalizedName(item.getRegistryName().toString());` ainsi que! Il s'assure que le nom non localisé de votre article contient l'ID de mod pour éviter les conflits de fichiers de langue entre les mods.

Maintenant, l'élément a également besoin d'un modèle, et c'est là que les choses ont été difficiles après le 1.7.10, ce qui impliquait simplement de dire à Minecraft le nom de la texture à charger et pourrait être spécifié dans le constructeur de l'élément.

```
final ModelResourceLocation fullModelLocation = new
ModelResourceLocation(item.getRegistryName().toString(), "inventory");
ModelBakery.registerItemVariants(item, fullModelLocation);
ModelLoader.setCustomMeshDefinition(item, new ItemMeshDefinition()
    {
        public ModelResourceLocation getModelLocation(ItemStack stack)
        {
            return fullModelLocation;
        }
    });
```

Notez que cette section ne *doit* être située que côté client (c'est-à-dire le proxy client), car la plupart des classes référencées n'existent pas sur le serveur dédié.

Enregistrement d' articles avec des *variantes* par exemple Plantons, doit être fait d' une manière différente, en utilisant `ModelLoader.setCustomModelResourceLocation(item, metadata, resourceLocation)` les `ModelLoader.setCustomModelResourceLocation(item, metadata, resourceLocation)` bien qu'il nous permet d' utiliser un fichier Blockstate pour indiquer nos variantes (qui est *beaucoup* plus préféré à l'alternative). Notre article n'utilise pas de variantes, nous avons donc terminé.

Lire Création d'un élément de base avec Forge en ligne:
https://riptutorial.com/fr/minecraft/topic/9850/creation-d-un-element-de-base-avec-forge

Chapitre 6: Création du premier plugin Spigot

Examples

Premier plugin dans Eclipse

Prérequis

Ce guide suppose que vous avez déjà utilisé BuildTools et exécuté le serveur Spigot au moins une fois. Il suppose également que vous avez le fichier JAR Spigot-API que nous allons utiliser.

1) Lancez Eclipse ; vous pouvez changer l'emplacement de l'espace de travail si vous le souhaitez.

2) Créer un nouveau projet

1. Définissez le nom du projet sur ce que vous souhaitez. Ici, nous avons choisi MyFirstPlugin.
2. Cliquez sur Suivant.
3. Sélectionnez Ajouter des fichiers JAR externes sous l'onglet Bibliothèques. Dans la boîte de dialogue Sélection JAR, sélectionnez le fichier JAR spigot-api-shaded, qui se trouve dans Spigot / Spigot-API / target / dans votre dossier BuildTools.
4. Sélectionnez Terminer

3) Ajouter un nouveau paquet

Cliquez avec le bouton droit sur **src** et cliquez sur **Nouveau> Package** . Vous pouvez utiliser n'importe quelle convention d'espace de noms que vous souhaitez, juste être cohérent. (ex: com.google.android).

4) Créer une nouvelle classe

1. Cliquez avec le bouton droit sur le package nouvellement créé et sélectionnez **Nouveau> Classe** .

2. Donnez-lui un nom; souvent le même nom que le projet. Dans l'éditeur, la classe Java nouvellement créée s'ouvrira. Le code devrait ressembler à ceci:

```
package yourpackage;
public class MyFirstPlugin {
}
```

5) Modifier la déclaration de classe

1. Votre classe doit s'étendre depuis JavaPlugin. Eclipse produira une erreur car il ne sait pas ce qu'est JavaPlugin. Si vous avez importé avec succès l'API Spigot, vous pourrez importer JavaPlugin en ajoutant l'instruction import. Vous n'avez pas besoin de taper manuellement

cette ligne, cliquez simplement sur l'erreur et sélectionnez l'action appropriée. Votre code devrait maintenant ressembler à:

```
package yourpackage;
import org.bukkit.plugin.java.JavaPlugin;

public class MyFirstPlugin extends JavaPlugin {

}
```

6) Mettre en œuvre les méthodes nécessaires

La classe JavaPlugin a des méthodes abstraites qui doivent être implémentées par votre plugin. Par conséquent, ajoutez les fonctions onEnable et onDisable qui seront déclenchées lorsque le plug-in est désactivé ou activé dans la console. Vous pouvez laisser ces blancs pour le moment. Vous devez également écrire `@Override` au-dessus de la méthode.

Note: Vous n'avez pas besoin d'ajouter un getLogger lorsque votre plugin est activé ou désactivé, Bukkit le fait déjà pour vous.

```
package com.meeku.tutorialPlugin;
import org.bukkit.plugin.java.JavaPlugin;

public class MyFirstPlugin extends JavaPlugin {
    // Fired when plugin is enabled
    @Override
    public void onEnable() {
    }
    // Fired when plugin is disabled
    @Override
    public void onDisable() {

    }
}
```

7) Créer un fichier plugin.yml

Faites un clic droit sur le projet et créez un fichier **Nouveau> Fichier** . Nommez-le **plugin.yml** . Coller dans ce qui suit:

```
name: MyFirstPlugin
main: yourpackage.MyFirstPlugin
version: 1.0
commands:
```

8) exportation

Comme il n'y a pas d'erreurs, nous pouvons exporter ce projet en tant que JAR. Cliquez avec le bouton droit sur le nom du projet, sélectionnez Exporter. Dans la boîte de dialogue consécutive, sélectionnez le fichier JAR. Cliquez sur Suivant. Vous pouvez décocher le classpath et le projet include et changer la destination d'exportation dans votre dossier plugins

9) en cours d'exécution

Démarrez le serveur et vous devriez voir que votre plugin a été activé.

Lire Création du premier plugin Spigot en ligne:
https://riptutorial.com/fr/minecraft/topic/9766/creation-du-premier-plugin-spigot

Chapitre 7: Créer un bloc de base avec Forge

Introduction

Créer un bloc simple et décoratif avec Forge est l'une des premières tâches qu'un aspirant modder devra apprendre. Comment faire cela a changé sur les différentes versions de Minecraft et est probablement à un niveau de difficulté "modéré" après le 1.7.10 en raison du grand nombre d'erreurs faciles à faire.

Remarques

Si quelque chose ne va pas et que votre bloc personnalisé (placé ou en attente) a une texture manquante (noir / violet) ou un modèle (cube par défaut trop grand lorsqu'il est maintenu), vérifiez le journal. Les problèmes de ce type apparaîtront presque *toujours* dans le journal, à condition que vous ayez enregistré les éléments correctement.

La première chose à afficher est la ligne `"Exception loading model for variant..."` ou similaire, vous indiquant quel bloc ou élément n'a pas pu être chargé correctement. Après une douzaine de lignes commençant par `at...` sera une autre ligne commençant `Caused by...`

Cette ligne "causée par" est la plus importante, elle vous indiquera quel fichier n'a pas pu être chargé correctement et peut être l'une des erreurs suivantes:

- JSON mal formé (votre fichier JSON n'est pas valide et a une faute de frappe)
- Fichier introuvable (le fichier recherché par Minecraft n'est pas correctement nommé ou au bon endroit)
- Exception de variante manquante (votre JSON Blockstate est incomplet)

Vous pouvez même obtenir plus d'une erreur d'échec pour une seule erreur! Si vous ne savez pas quel est le problème, commentez chaque bloc et élément que vous connaissez, réduisez la liste des erreurs jusqu'au bloc (ou aux blocs) en question. Vous devrez peut-être le faire plusieurs fois.

De plus, une texture de mixage montre comme une liste simple toutes les ressources manquantes pour un domaine donné (ID mod).

Examples

La classe de bloc

Nous avons d'abord besoin d'une classe qui représente le bloc

```
public class CustomBlock extends Block {

    public CustomBlock () {
        super(Material.ROCK);
        setHardness(1.0f);
```

```
        setHarvestLevel("pickaxe", 0);
        setResistance(1.0f);
        setCreativeTab(CreativeTabs.DECORATIONS);
        this.setSoundType(SoundType.STONE);
    }
}
```

Même ici, plusieurs modifications sont possibles: matériau (qui régit certaines propriétés telles que pouvoir être poussé par des pistons et s'il peut être cassé à la main), dureté (combien de temps il faut pour casser), niveau de récolte (outil et outil appropriés) matériel: dans ce cas la pioche en bois), la résistance (vs explosions), l'onglet qui apparaît dans le menu créatif, et quelle étape sonne-t-il.

C'est là que les blocs de fonctionnalités fantaisistes devront fonctionner, mais pour le moment nous faisons un bloc qui a l'air sympa, donc nous avons fini.

Le modèle de bloc JSON

Ensuite, nous devons dire à Minecraft ce que nous voulons que notre bloc ressemble.

```
{
    "parent": "block/cube_all",
    "textures": {
        "all": "example:blocks/decorative"
    }
}
```

C'est à peu près tout ce qui est nécessaire pour que le bloc fonctionne une fois enregistré. La seule chose importante est que le **nom de fichier** corresponde **au nom de registre** utilisé pour enregistrer le bloc et doit être en minuscule (les noms de fichiers 1.11+ *doivent* être en minuscules, avant cela, il est juste sensible à la casse).

Nommez le fichier JSON modèle `my_block.json` (correspondant au nom du registre que nous allons lui donner ultérieurement) et enregistrez-le dans `src\main\resources\assets\example\models\block\` (où `example` est l'ID de mod spécifié dans le @Motage de votre classe de mod principal.

Le modèle de bloc utilise ici un parent de block / cube_all, ce qui signifie que la texture fournie unique sera utilisée sur toutes les faces. Il existe également d'autres modèles par défaut, tels que:

- bloc / cube (les six faces sont affectées indépendamment)
- block / cube_bottom_top (faces supérieure et inférieure indépendantes des côtés)
- bloc / orientable (bloc de revêtement directionnel, par exemple four)
- bloc / croix (fleurs, herbes hautes)
- bloc / culture (blé, carottes)

Notez que chaque modèle spécifie les textures qu'il utilise par un identifiant de nom (par exemple `"all"` ou `"top"`). Regardez le modèle parent pour déterminer quels sont ces noms si vous êtes incertain. Des textures mal spécifiées peuvent entraîner des problèmes de texture manquants dans les *rapports* de *non-erreurs* .

Il est également possible de créer un modèle entièrement personnalisé ou de créer un modèle parent personnalisé. Mais pour l'instant, cela suffira.

N'oubliez pas de créer une texture, nommez-la `decorative.png` (comme le spécifie le fichier JSON) et enregistrez-le dans `src\main\resources\assets\example\textures\blocks\`

Enregistrement de bloc

L'enregistrement de blocs est effectué à partir de votre classe de mod principale ou d'une méthode de classe ModBlocks appelée à partir de la classe de mod principale au cours de preInit.

```
Block myBlock = new CustomBlock();
string registryname = "my_block";
block.setRegistryName(registryname);
block.setUnlocalizedName(block.getRegistryName().toString());
GameRegistry.register(block);
```

Il existe une raison importante d'utiliser `block.setUnlocalizedName(block.getRegistryName().toString());` ainsi que! Il s'assure que vos noms non localisés de bloc (et d'élément) contiennent l'ID de mod pour éviter les conflits de fichiers de langue entre les mods.

Oh, vous voulez aussi une version d'article pour qu'elle puisse aussi exister dans votre inventaire? C'est créé séparément après 1.7.10 et fait comme ça:

```
ItemBlock ib = new ItemBlock(block);
ib.setRegistryName(registryname);
GameRegistry.register(ib);
```

Notez que nous définissons le nom du registre de ItemBlock sur la même chaîne que notre bloc. Voici comment forger et associer les blocs à leur homologue ItemBlock et vice versa.

Mais attendez, il y a plus!

Votre bloc peut avoir une *forme d'élément* , mais cet objet n'a pas encore de modèle ou de texture! Les modèles sont automatiquement enregistrés pour les blocs, mais pas pour les éléments. Ceci *ne peut être appelé qu'à partir du proxy client* et ne couvre pas les blocs avec des variantes (comme la laine ou les feuilles).

```
ModelLoader.setCustomModelResourceLocation(
    ib , 0, new ModelResourceLocation(ib.getRegistryName(),"normal"));
```

En règle générale, vous n'aurez pas non plus besoin d'un modèle d'objet JSON, car Forge et vanilla se rabattront sur le modèle du bloc, mais ce n'est pas toujours le cas. Si vous trouvez que vous avez besoin d'un modèle d'objet JSON, placez-le simplement par parent dans votre bloc JSON et enregistrez-le dans `src\main\resources\assets\example\models\item\` avec le même nom de fichier que le nom de registre du bloc.

```
{
```

```
    "parent": "example:block/my_block"
}
```

Lire Créer un bloc de base avec Forge en ligne:
https://riptutorial.com/fr/minecraft/topic/9748/creer-un-bloc-de-base-avec-forge

Chapitre 8: Installation d'un serveur Spigot

Examples

BuildTools

Qu'Est-ce que c'est?

BuildTools.jar est une solution pour construire Bukkit, CraftBukkit, Spigot et l'API Spigot. Tout cela se fait sur votre ordinateur! Quelques programmes préalables sont nécessaires, mais les instructions ci-dessous vous guideront à travers tout ce que vous devez faire.

Conditions préalables

Il existe deux applications nécessaires pour utiliser BuildTools: Git et Java.

les fenêtres

Git

Pour que BuildTools fonctionne sous Windows, vous devez installer Git. Pour Windows, il est distribué via git-scm, qui peut être téléchargé ici . Installez-le où vous voulez, il fournira git bash, qui sera utilisé pour exécuter le jar BuildTools. Continuez simplement à frapper lors de l'exécution du programme d'installation.

Java

Téléchargez JRE 8 à partir d' ici et installez-le. Continuez simplement à frapper lors de l'exécution du programme d'installation.

Linux

Les commandes git et Java, ainsi que les commandes util, peuvent être installées à l'aide d'une seule commande via votre gestionnaire de paquets.

Debian / Ubuntu: `sudo apt-get install git openjdk-7-jre-headless tar`

CentOS / RHEL: `sudo dnf install git java-1.7.0-openjdk-devel tar`

Arch: `pacman -S jdk8-openjdk git`

Mac

Git peut être téléchargé à partir de: http://sourceforge.net/projects/git-osx-installer/files/

Il se peut que Java doive être mis à jour à partir de la version distribuée d'Apple, et même s'il a déjà été mis à jour, il peut être nécessaire de le relier pour une utilisation en shell. Veuillez suivre les étapes trouvées ici: https://gist.github.com/johan/10590467

Lancer BuildTools

1. Télécharger BuildTools.jar depuis https://hub.spigotmc.org/jenkins/job/BuildTools/lastSuccessfulBuild/artifact/target/BuildTools.jar .

2. Ouvrez votre terminal si vous êtes sous Linux ou git bash sous Windows.

    1. Git bash se trouve sur le bureau ou dans le menu Démarrer sous le nom "git bash". Il est également possible de l'ouvrir en cliquant avec le bouton droit de la souris sur n'importe quoi, car il s'agit désormais d'un élément de votre menu contextuel.

3. Accédez à l'emplacement où vous avez téléchargé BuildTools.jar ou utilisez la ligne de commande pour télécharger le fichier jar dans votre répertoire actuel.

    1. Sous Windows, vous pouvez soit utiliser la commande cd pour modifier les répertoires, soit cliquer avec le bouton droit sur l'espace vide du dossier où se trouve BuildTools.jar (NE cliquez PAS sur BuildTools.jar lui-même) et cliquez sur "git bash" pour l'ouvrir. dans votre répertoire actuel.

4. Exécutez BuildTools.jar à partir du terminal (ne double-cliquez pas sur BuildTools.jar) en procédant comme suit:

    1. Sur Linux, exécutez git config --global --unset core.autocrlf, puis lancez java -jar BuildTools.jar dans bash ou un autre shell approprié.
    2. Sous Windows, exécutez la commande ci-dessous dans la fenêtre git bash qui s'ouvre: java -jar BuildTools.jar Veuillez noter que vous devez disposer de BuildTools # 35 ou version ultérieure, les anciennes versions ne fonctionneront pas.
    3. Sur Mac, exécutez les commandes ci-dessous, exportez MAVEN_OPTS = "- Xmx2G" java -Xmx2G -jar BuildTools.jar

5. Attendez comme il construit vos pots. Dans quelques minutes, vous devriez avoir des bocaux fraîchement compilés!

6. Vous pouvez trouver CraftBukkit et Spigot dans le même répertoire où vous avez exécuté le BuildTools.jar (craftbukkit-1.10.jar et spigot-1.10.jar). Vous pouvez trouver l'API Spigot dans \ Spigot \ Spigot-API \ target \ (spigot-api-1.10-R0.1-SNAPSHOT.jar).

les fenêtres

1. Obtenez spigot.jar en utilisant BuildTools ou d' ici .
2. Collez le texte suivant dans un document texte. Enregistrez-le sous le nom start.bat dans le même répertoire que spigot.jar: Vous devrez renommer votre fichier jar en fichier spigot.jar ou modifier le fichier dans le fichier bat pour qu'il pointe vers le fichier approprié. nb: Windows (par défaut) masque l'extension .jar du fichier.

```
@echo off
java -Xmx1G -jar spigot.jar
pause
```

3. Double-cliquez sur le fichier de commandes.

Linux

1. Obtenez spigot.jar en utilisant BuildTools ou d' ici .
2. Créez un nouveau script de démarrage (start.sh) dans le répertoire pour lancer le fichier JAR: #! / Bin / sh

```
java -Xmx1G -jar spigot.jar
```

3. Ouvrez votre terminal et exécutez les éléments suivants dans le répertoire: chmod + x start.sh
4. Exécutez votre script de démarrage:

```
./start.sh
```

Mac

1. Obtenez spigot.jar en utilisant BuildTools ou d' ici .
2. Créez un nouveau script de démarrage (start.command) pour lancer le fichier JAR: #! / Bin / sh

```
cd "$( dirname "$0" )"
java -Xmx1G -jar spigot.jar
```

3. Ouvrez Terminal et tapez-le: (Ne pas appuyer sur Entrée!)

```
chmod a+x
```

4. Faites glisser votre fichier de script de démarrage dans la fenêtre du terminal. (Assurez-vous de mettre un espace entre chmod a + x et votre script de démarrage!)

5. Double-cliquez sur votre script de démarrage.

Lire Installation d'un serveur Spigot en ligne:
https://riptutorial.com/fr/minecraft/topic/9702/installation-d-un-serveur-spigot

Chapitre 9: Modding avec Forge

Syntaxe

- MODID = représente l'identifiant du MOD
- MODPath = représente le chemin complet du répertoire qualifié vers votre dossier mod

Remarques

Ce sujet devrait contenir la plupart des modèles / exemples utilisés et du code bien testé pour modifier l'application Minecraft avec forge. Peut-être que cela peut remplacer la documentation officielle un jour.

Examples

Modèle d'implémentation pour les proxys d'initialisation

Cet exemple vous montre comment implémenter des classes proxy pour votre application Minecraft Mod, qui sont utilisées pour initialiser votre mod.

Tout d'abord, vous devrez implémenter la classe CommonProxy.java de base qui contient la méthode 3 principalement utilisée:

```
public class CommonProxy {
    public void preInit(FMLPreInitializationEvent e) {}
    public void init(FMLInitializationEvent e) {}
    public void postInit(FMLPostInitializationEvent e) {}
}
```

Normalement, votre mod dispose de deux packages différents pour le code client et serveur, vous aurez donc besoin dans chaque package d'une classe enfant CommonProxy.java comme:

```
public class ClientProxy extends CommonProxy {
    @Override
    public void preInit(FMLPreInitializationEvent e) {
        super.preInit(e);
    }

    @Override
    public void init(FMLInitializationEvent e) {
        super.init(e);
    }

    @Override
    public void postInit(FMLPostInitializationEvent e) {
        super.postInit(e);
    }
}
```

et pour le serveur:

```
public class ServerProxy extends CommonProxy {
    @Override
    public void preInit(FMLPreInitializationEvent e) {
        super.preInit(e);
    }

    @Override
    public void init(FMLInitializationEvent e) {
        super.init(e);
    }

    @Override
    public void postInit(FMLPostInitializationEvent e) {
        super.postInit(e);
    }
}
```

Une fois que vous avez créé ces classes, vous pouvez les étendre par des méthodes qui ne doivent s'exécuter que du côté client ou serveur, mais vous pouvez également les associer aux deux si vous appelez les méthodes dans la classe 'base'.

Enfin, vous devez définir quel proxy est pris à l'exécution. Vous devez étendre votre classe de mod principal avec l'annotation `@Mod` , en:

```
private static final String CLIENTPROXY = "com.yourpackage.client.ClientProxy";
private static final String SERVERPROXY = "com.yourpackage.client.ServerProxy";

@SidedProxy(clientSide = CLIENTPROXY, serverSide = SERVERPROXY)
public static CommonProxy PROXY;
```

Cela permettra à Forge de détecter quelle classe doit être prise à l'exécution. Dans les méthodes d'initialisation de votre Mod, vous pouvez maintenant utiliser cette propriété statique PROXY.

```
@EventHandler
public void init(FMLInitializationEvent event) {
    PROXY.init(event);
}

@Mod.EventHandler
public void preInit(FMLPreInitializationEvent event) {
    PROXY.preInit(event);
}

@EventHandler
public void postInit(FMLPostInitializationEvent event) {
    PROXY.postInit(event);
}
```

Ajout de sons personnalisés à votre MOD

Cet exemple vous montre comment ajouter de nouveaux sons à votre MOD et les lire. Tout d'abord, vous avez besoin d'un fichier audio au format `*.ogg` . Tout autre format n'est pas autorisé

par l'application Minecraft et sera rejeté.

Le fichier son a le nom: sound1.ogg

Placez le fichier son sous le chemin suivant:

    /YourPath/src/main/resources/assets/MODID/sounds/sound1.ogg

Remplacez 'MODID' par l'identifiant que vous avez défini pour votre MOD

Ensuite, vous devez créer un `sounds.json` en UTF-8 (Standard) qui définit le nom, la ressource, ... et d'autres éléments pour votre son personnalisé. Ce fichier ressemblera à:

```
{
  "sound1": {
    "category" : "player",
    "sounds": [{
        "name": "MODID:sound1",
        "stream": false
      }
    ]
  },
  "sound2": {
    "category" : "ambient",
    "sounds": [{
        "name": "MODID:subfolder/sound2",
        "stream": true
      }
    ]
  }
}
```

Comme explication pour ce sounds.json.

Deux sons définis sont définis, et j'ai ajouté un exemple sur la manière d'ajouter des sons multiples. `sound1` a la catégorie `player` le second est de catégorie `ambient` ce qui signifie que le volume du son est affecté par les réglages de volume que l'utilisateur a définis pour les sons du lecteur / ambiant. `name` est la propriété la plus importante car elle pointe vers la ressource du son. Le `MODID` est l'identifiant de votre MOD et est obligatoire car l'Application va chercher dans les ressources de votre mod pour le fichier, sinon il cherchera dans les ressources Minecraft et ne trouvera rien. La propriété `stream` signifie que le son sera diffusé depuis le système de fichiers, ce qui n'est nécessaire que pour les sons de plus de 4 secondes.

Votre fichier `sounds.json` personnalisé doit se `sounds.json` sous le chemin suivant:

    /YourPath/src/main/resources/assets/MODID/sounds.json

Vous pourrez maintenant charger les sons dans le registre du jeu. Vous devez donc créer une classe qui initialise `SoundEvent` et qui gère l'enregistrement.

```
public class SoundRegistrator {
    public static final SoundEvent SOUND_1;
    public static final SoundEvent SOUND_2;
```

```
    static {
        SOUND_1 = addSoundsToRegistry("sound1");
        SOUND_2 = addSoundsToRegistry("sound2");
    }

    private static SoundEvent addSoundsToRegistry(String soundId) {
        ResourceLocation shotSoundLocation = new ResourceLocation("MODID", soundId);
        SoundEvent soundEvent = new SoundEvent(shotSoundLocation);
        soundEvent.setRegistryName(shotSoundLocation);
        return soundEvent;
    }
}
```

Ensuite, vous devez créer un `SoundRegisterListener` :

```
public class SoundRegisterListener {
        @SubscribeEvent(priority = EventPriority.NORMAL, receiveCanceled = true)
        public void registerSoundEvents(RegistryEvent.Register<SoundEvent> event) {
           event.getRegistry().registerAll(SoundRegistrator.SOUND_1,SoundRegistrator.SOUND_2);
        }
}
```

et l'enregistrer sur le `MinecraftForge.EVENT_BUS` comme:

```
MinecraftForge.EVENT_BUS.register(new SoundRegisterListener());
```

Enfin, vous pourrez jouer vos sons:

```
void playSound(SoundEvent sound) {
    try {
        if (Minecraft.getMinecraft().world.isRemote) {
            EntityPlayerSP player = Minecraft.getMinecraft().player;
            Minecraft.getMinecraft().world.playSound(player, player.getPosition(), sound,
SoundCategory.PLAYERS, RandomGenerator.getNextRandomVolumeLoud(), 1.0F);
        }
    } catch (Exception ex) {
       //Error happened
    }
}
```

Envoi d'une commande

Cet exemple vous montre différentes manières d'exécuter des «commandes» pour Minecraft à partir du code:

```
EntityPlayerSP player = Minecraft.getMinecraft().player;
player.sendChatMessage("/Command here");
```

envoyer une commande dans SinglePlayer

Lire Modding avec Forge en ligne: https://riptutorial.com/fr/minecraft/topic/9956/modding-avec-forge

Chapitre 10: Rédaction de Mods Minecraft

Introduction

Comprendre l'écriture de vos propres mods pour Minecraft.

La première chose à savoir est qu'il existe principalement deux plates-formes sur lesquelles les mods sont construits. Forge et Bukkit.

Chaque plate-forme a ses avantages et ses inconvénients et n'est généralement pas compatible les unes avec les autres. Les mods de Forge vont dans la gamme et sont généralement orientés jeu. Les mods de Bukkit sont entièrement basés sur des serveurs et généralement orientés vers l'administration. Les mods de Bukkit sont appelés plugins.

Examples

Plugins de base de Bukkit

- Ecrire une commande Hello
- Écrire un écouteur d'événement

Mods Forge de base

- Créer un bloc de base
- Créer un élément de base
- Ecrire une commande Hello
- Ecrire un gestionnaire d'événement
- Démarrer avec les fonctionnalités [1.8+]

Lire Rédaction de Mods Minecraft en ligne: https://riptutorial.com/fr/minecraft/topic/9740/redaction-de-mods-minecraft

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec minecraft	Community, Martin W, SoniEx2
2	Auditeurs d'événement à Bukkit	Draco18s, Martin W
3	Auditeurs d'événement à Forge	Draco18s, Martin W
4	Commandes de plugin	Martin W
5	Création d'un élément de base avec Forge	Draco18s
6	Création du premier plugin Spigot	Martin W
7	Créer un bloc de base avec Forge	Draco18s
8	Installation d'un serveur Spigot	Martin W
9	Modding avec Forge	Code.IT
10	Rédaction de Mods Minecraft	Draco18s, Martin W