EBook Gratis

APRENDIZAJE bukkit

Free unaffiliated eBook created from
Stack Overflow contributors.

#bukkit

Tabla de contenido

Acerca de

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

It is an unofficial and free bukkit 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 bukkit.

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

Capítulo 1: Empezando con bukkit

Observaciones

Bukkit es una API simple que permite modificar la experiencia normal de multijugador de Minecraft mediante el uso de complementos.

Bukkit ahora está descontinuado y ya no está disponible para las versiones más recientes de Minecraft. Spigot, una versión de Bukkit que presume de mejorar el rendimiento del servidor está disponible. La API para Spigot es esencialmente la misma que Bukkit.

Versiones

Versión de Minecraft	Spigot Enlace de descarga	Fecha de lanzamiento
1.10.2	Enlazar	2016-11-03
1.10	Enlazar	2016-06-26
1.9.4	Enlazar	2016-06-09
1.9.2	Enlazar	2016-03-30
1.9	Enlazar	2016-02-29
1.8.8	Enlazar	2015-07-28
1.8.7	Enlazar	2015-06-05
1.8.6	Enlazar	2015-05-25
1.8.5	Enlazar	2015-05-22
1.8.4	Enlazar	2015-04-17
1.8.3	Enlazar	2015-02-20
1.8	Enlazar	2014-09-02
1.7.10	Enlazar	2014-06-26
1.7.9	Enlazar	2014-04-14
1.7.8	-	2014-04-11
1.7.5	Enlazar	2014-02-26
1.7.2	Enlazar	2013-10-25

Versión de Minecraft	Spigot Enlace de descarga	Fecha de lanzamiento
1.6.4	Enlazar	2013-09-19
1.6.2	Enlazar	2013-07-08
1.5.2	Enlazar	2013-05-02
1.5.1	Enlazar	2013-03-21
1.4.7	Enlazar	2013-01-09
1.4.6	-	2012-12-20

Examples

Creando un Plugin

Prerrequisitos

- JDK 7 o superior (recomendado: JDK 8+)

Añadiendo Bukkit como Dependencia

El método más sencillo para agregar la API de Bukkit a su proyecto es descargar el Bukkit.jar directamente desde el Repositorio Spigot y agregarlo a la ruta de clase de su proyecto. Las versiones heredadas de Bukkit se pueden encontrar en el Repositorio de Bukkit .

El otro es agregarlo como una dependencia de Maven, agregando las siguientes líneas a su `pom.xml` :

```
<repositories>
    <repository>
        <id>spigot-repo</id>
        <url>https://hub.spigotmc.org/nexus/content/repositories/snapshots/</url>
    </repository>
</repositories>
<dependencies>
    <!--Bukkit API-->
    <dependency>
        <groupId>org.bukkit</groupId>
        <artifactId>bukkit</artifactId>
        <version>{VERSION}</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Clase principal

La clase principal del complemento es el punto de entrada para que Bukkit cargue e interactúe con su complemento. Es una clase que extiende `JavaPlugin` y solo una instancia de ella debe `JavaPlugin` su complemento. Por convención, es bueno darle a esta clase el mismo nombre que su complemento.

Este es un ejemplo de una clase de complemento principal para el complemento "MyPlugin":

```
package com.example.myplugin; //{$TopLevelDomain}.{$Domain}.{$PluginName};

import org.bukkit.plugin.java.JavaPlugin;

public final class MyPlugin extends JavaPlugin {

    @Override
    public void onEnable() {
        //Called when the plugin is enabled
        getLogger().info("onEnable has been invoked!");
    }

    @Override
    public void onDisable() {
        //Called when the plugin is disabled
        getLogger().info("onDisable has been invoked!");
    }

}
```

Para acceder a su instancia de complemento desde otra clase, deberá almacenar la instancia de su clase MyPlugin creada por Bukkit para que sea accesible desde fuera de la clase.

```
public class MyPlugin extends JavaPlugin {

    private static MyPlugin instance; //Effectively final variable containing your plugin's
instance

    public MyPlugin(){
        if(MyPlugin.instance != null) { //Unnecessary check but ensures your plugin is only
initialized once.
            throw new Error("Plugin already initialized!");
        }

        MyPlugin.instance = this; //A plugin's constructor should only be called once
    }

    public static MyPlugin getInstance(){ //Get's your plugin's instance
        return instance;
    }

    //your other code...
}
```

Luego, para acceder a su clase principal desde otra clase, simplemente use `MyPlugin.getInstance()`

```
public class MyOtherClass {

    public void doSomethingWithMainClass(){
        MyPlugin.getInstance().getLogger().info("We just used MyPlugin");
    }

}
```

Creando un plugin.yml

El archivo plugin.yml se encuentra en la raíz de su archivo jar final y proporciona información esencial a Bukkit para cargar su complemento. El plugin.yml más simple se ve así

```
name: {$PluginName}               //The name of the plugin
main: {$PackageName}.{$MainClass} //The fully qualified name of the main class.
version: {$Version}               //The plugin's version
```

Por ejemplo, con la clase MyPlugin anterior

```
name: MyPlugin
main: com.example.myplugin.MyPlugin
version: 1.0
```

Crear un servidor de prueba en Windows

Para poder crear un servidor, necesita tener el spigot o el archivo de bukkit jar. Refiérase al tema de las versiones para seleccionar su tarro

1. Primero, crea una nueva carpeta. En esa carpeta, ponga el archivo jar spigot / bukkit.
2. Haga clic derecho en la carpeta y elija Nuevo> Documento de texto.
3. Nombre el nuevo documento start.bat, haga clic derecho sobre él y haga clic en Editar.
4. Agregue el siguiente código:

```
@echo off
java -Xms512M -Xmx1G -XX:+UseConcMarkSweepGC -jar {YOUR_JAR.jar}
pause
```

No olvide cambiar {YOUR_JAR.jar} para el archivo jar que descargó antes de comenzar estos temas.

5. Puede editar `-Xms` para cambiar la RAM mínima permitida (Ej: -Xms1024M = 1024MB, -Xms1G = 1GB). También puede editar `-Xmx` para cambiar la RAM máxima permitida. Asegúrese de que el máximo sea mayor que el mínimo.
6. Guarde el archivo, cierre la ventana e inicie su archivo `start.bat` . Su servidor debería estar abierto. Para ejecutar el servidor, debe aceptar el EULA .
7. Si acepta el EULA , abra `eula.txt` change `eula=false` a `eula=true` Haga clic en "Guardar" y luego debería poder iniciar su servidor.

8. Para conectarse a su servidor, ejecute `start.bat` , abra Minecraft, agregue un servidor y coloque `localhost` como IP.

BuildTools

¿Qué es?

BuildTools.jar es una solución para crear Bukkit, CraftBukkit, Spigot y Spigot-API. Todo lo cual se hace en tu computadora! Son necesarios algunos programas de requisitos previos, pero las instrucciones a continuación lo guiarán a través de todo lo que necesita hacer.

Prerrequisitos

Hay dos aplicaciones necesarias para usar BuildTools: Git y Java.

Windows

Git

Para que BuildTools se ejecute en Windows, deberá instalar Git. Para Windows se distribuye a través de git-scm, que se puede descargar aquí . Instálalo donde quieras, proporcionará git bash, que se usará para ejecutar el jar de BuildTools. Solo sigue presionando siguiente cuando ejecutes el instalador.

Java

Descarga JRE 8 desde aquí e instala. Solo sigue presionando siguiente cuando ejecutes el instalador.

Linux

Tanto git como Java, así como los comandos util, pueden instalarse usando un solo comando a través de su administrador de paquetes.

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`

Arco: `pacman -S jdk8-openjdk git`

Mac

Git se puede descargar desde: http://sourceforge.net/projects/git-osx-installer/files/

Es posible que Java deba actualizarse desde la versión distribuida de Apple, e incluso si se ha actualizado anteriormente, puede que deba estar vinculado para el uso de shell. Siga los pasos que se encuentran aquí: https://gist.github.com/johan/10590467

Ejecutando BuildTools

1. Descargue BuildTools.jar desde https://hub.spigotmc.org/jenkins/job/BuildTools/lastSuccessfulBuild/artifact/target/BuildTools.jar .

2. Abra su terminal si está en Linux, o git bash en Windows.

    1. Git bash se puede encontrar en el escritorio o en el menú Inicio bajo el nombre "git bash". También es posible abrirlo haciendo clic con el botón derecho en cualquier cosa, ya que ahora es un elemento en su menú contextual.

3. Navegue hasta donde descargó BuildTools.jar, o use la forma de la línea de comandos para descargar el jar en su directorio actual.

    1. En Windows, puede usar el comando cd para cambiar directorios, o puede hacer clic derecho en el espacio en blanco de la carpeta donde está BuildTools.jar (NO haga clic en BuildTools.jar) y haga clic en "git bash", que lo abrirá en su directorio actual.

4. Ejecute BuildTools.jar desde el terminal (no haga doble clic en BuildTools.jar) haciendo lo siguiente:

    1. En Linux, ejecute git config --global --unset core.autocrlf, luego ejecute java -jar BuildTools.jar en bash u otro shell apropiado.
    2. En Windows, ejecute el siguiente comando dentro de la ventana de git bash que se abrió: java -jar BuildTools.jar Tenga en cuenta que es necesario que tenga BuildTools # 35 o posterior, las versiones anteriores no funcionarán.
    3. En Mac, ejecute los siguientes comandos, exporte MAVEN_OPTS = "- Xmx2G" java -Xmx2G -jar BuildTools.jar
    4. Si necesita una versión anterior, puede especificar la versión usando el argumento --rev para BuildTools, por ejemplo para 1.8.8: java -jar BuildTools.jar --rev 1.8.8

5. Espera ya que construye tus jarras. ¡En unos minutos deberías tener frascos recién compilados!

6. Puede encontrar CraftBukkit y Spigot en el mismo directorio donde ejecutó BuildTools.jar (para la versión 1.10 de Minecraft, serían craftbukkit-1.10.jar y spigot-1.10.jar). Puede encontrar Spigot-API en \ Spigot \ Spigot-API \ target \ (para la versión 1.10 de Minecraft, sería spigot-api-1.10-R0.1-SNAPSHOT.jar).

Lea Empezando con bukkit en línea: https://riptutorial.com/es/bukkit/topic/5400/empezando-con-bukkit

Capítulo 2: Archivos de configuración

Sintaxis

- `String s = config.getString("path.to.string");`
- `int i = config.getInt("path.to.int");`
- `double d = config.getDouble("path.to.double");`
- `List<String> sl = config.getStringList("path.to.stringlist");`
- `List<Double> dl = config.getDoubleList("path.to.doublelist");`
- `List<Integer> il = config.getIntegerList("path.to.integerlist");`

Observaciones

Los archivos de configuración de Bukkit son archivos YAML (otro lenguaje de marcado) sencillos, y se implementan como tales.

Examples

Plugin Config.yml

Puede tener un archivo config.yml que se carga directamente desde su archivo jar. Debe agregarse a la carpeta de su proyecto, de la misma manera que el archivo plugin.yml.

En este archivo tiene los valores por defecto para su configuración.

Ejemplo de configuración:

```
# This is an YML comment
adminName: "Kerooker"
moderators: ["Romario", "Pelé", "Cafú"]
```

El archivo de configuración de ejemplo debe agregarse a la carpeta del proyecto.

Para cargar el archivo de configuración predeterminado en la carpeta de su complemento, debe agregar el siguiente código a su onEnable ():

```
saveDefaultConfig();
```

Esto hará que su archivo config.yml del proyecto sea el archivo de configuración de su complemento y lo agregará a la carpeta de su complemento.

Desde allí, puede acceder a su archivo de configuración desde cualquier lugar, utilizando su instancia de complemento:

```
JavaPlugin plugin; // Your plugin instance
FileConfiguration config = plugin.getConfig();  //Accessing the config file
```

Desde allí, podemos acceder a cualquier cosa que se haya configurado en la configuración del complemento.

Nota: El archivo de configuración predeterminado puede cambiar sus valores, si el usuario desea editar el archivo config.yml generado en la carpeta.

```
String adminName = config.getString("adminName");
List<String> moderators =  config.getStringList("moderators");
```

Sección de rutas múltiples

Lo que puede suceder en su archivo de configuración es tener una ruta a una variable que atraviesa varias secciones.

Ejemplo de configuración

```
admins:
 first-tier: "Kerooker"
 second-tier: "Mordekaiser"
 third-tier: "Yesh4"
```

El nombre "Kerooker" es de la sección "primer nivel", que es de la sección "administradores". Para acceder a las rutas internas de nuestro archivo, usamos un simple '.' Como una manera de decir que queremos la siguiente sección. Entonces, para que podamos acceder a "Kerooker", vamos:

```
config.getString("admins.first-tier");
```

Lea Archivos de configuración en línea: https://riptutorial.com/es/bukkit/topic/6824/archivos-de-configuracion

Capítulo 3: Comandos

Sintaxis

- `@Override public boolean onCommand(CommandSender sender, Command cmd, String label, String[] args)`

Examples

Manejando un Comando

Para manejar un comando, debe tener una clase que implemente la interfaz CommandExecutor. La clase JavaPlugin (la clase principal de su plugin) ya implementa esto.

Al implementar la interfaz CommandExecutor, se debe implementar el siguiente método:

```
public boolean onCommand(CommandSender sender, Command cmd, String label, String[] args) {
//Handle your command in here
return true;      ///Should return false if you want to show the usage

}
```

Remitente es el que envió el comando. Puede ser un jugador o la consola.

CMD es el comando que está escuchando, como se declara en plugin.yml. No debe confundirse con la etiqueta.

label es el alias utilizado para ejecutar este comando, es lo que el remitente escribe después de la barra diagonal.

y finalmente, args son los argumentos que el remitente puede haber usado para enviar su comando.

Un comando posible podría ir como

    / Dile a Kerooker ¡Hola, Kerooker!

Tell sería su etiqueta, y también puede definirse como su comando si lo dice en plugin.yml;

'Kerooker', 'Hola', 'Kerooker!' son sus argumentos 0, 1 y 2, respectivamente

Como retorno, probablemente siempre querrá volver verdadero cuando esperaba que todo sucediera de esa manera. Debe devolver falso si desea mostrar al remitente el uso del comando definido en su plugin.yml

Un comando simple set gameMode (/ gm)

Este ejemplo muestra un ejemplo muy básico de cómo utilizar onCommand. No sugiero procesar

sus comandos directamente en onCommand, pero esto sirve para este simple caso.

En este ejemplo intentamos configurar el modo de juego del jugador.

Lo primero que debemos hacer es asegurarnos de que el remitente no sea un ConsoleCommandSender, porque no podemos configurar el modo de juego de una consola. Esto se hace con (remitente de jugador).

A continuación, queremos que el jugador escriba / gm CREATIVE (o cualquier otro modo de modo de juego), así que tenemos que verificar 2 cosas:

1. Asegúrate de que pasan en 1 argumento (CREATIVO)
2. asegúrese de que su comando era "gm"

Logramos estas comprobaciones con: args.length == 1 && label.equalsIgnoreCase ("gm")

Ahora sabemos con seguridad que el jugador escribió "/ gm x".

Lo siguiente que debemos hacer es convertir args [0] en un objeto GameMode para que podamos aplicarlo al jugador. Esto se puede hacer con GameMode.valueOf (String) Sin embargo, de acuerdo con la documentación de enumeración de Java, si una cadena se pasa a valueOf () que no coincide con una enumeración, lanzará una IllegalArgumentException, por lo que debemos asegurarnos de atrapá eso.

Una vez que tengamos el modo de juego, podemos seguir adelante y simplemente usar p.setGameMode (gm) y el modo de juego del jugador cambiará. En el caso de que detectáramos una excepción, simplemente imprimimos una declaración y devolvemos el valor falso.

```
@Override
public boolean onCommand(CommandSender sender, Command command, String label, String[] args) {
    if (sender instanceof Player) {
        final Player p = (Player) sender;

        if (args.length == 1 && label.equalsIgnoreCase("gm")) {
            try {
                GameMode gm = GameMode.valueOf(args[0]);
                p.setGameMode(gm);
                p.sendMessage(ChatColor.GREEN + "Your gamemode has been set to: " +
gm.toString());
                return true;
            } catch (IllegalArgumentException e) {
                p.sendMessage(ChatColor.RED + "Invalid gamemode option!");
                return false;
            }

        }
    }
    return false;
}
```

Comando no en la clase principal

Si tiene muchos comandos, no debe ponerlos todos en la clase principal.

1. Haz una nueva clase y haz que implemente `CommandExecutor`

2. Agregue lo siguiente a la clase:

```
 @Override
 public boolean onCommand(CommandSender sender, Command cmd, String label, String[] args)
{

 }
```

3. En su clase principal agregue en onEnable (reemplace commandName con el nombre del comando y CommandExecutor con el nombre de la clase):

```
 getCommand("commandName").setExecutor(new CommandExecutor());
```

Lea Comandos en línea: https://riptutorial.com/es/bukkit/topic/6880/comandos

Capítulo 4: Entidades

Examples

Teletransportar una entidad a otra entidad

```
Entity entity;     //The entity you want to teleport
Entity teleportTo; //The entity where you want <entity> to be teleported to

boolean success = entity.teleport(teleportTo);   //Attempting to teleport.

if(success) {
//Teleport was successful
}else {
//Teleport wasn't successful
}
```

También puede agregar una causa a su teletransportador, para que pueda personalizar la forma en que su plugin o los demás tratarán su causa:

```
Entity entity;     //The entity you want to teleport
Entity teleportTo; //The entity where you want <entity> to be teleported to
PlayerTeleportEvent.TeleportCause cause;    //The cause you want the teleport to be of

boolean success = entity.teleport(teleportTo, cause);   //Attempting to teleport.

if(success) {
//Teleport was successful
}else {
//Teleport wasn't successful
}
```

Teletransportar una entidad a una ubicación

```
Entity toBeTeleported;    //The entity you want to teleport
Location teleportTo = new Location(world,x,y,z,yaw,pitch);    //The location to teleport to


boolean success = toBeTeleported.teleport(teleportTo);

if(success) {
    //Teleport was successful
}else {
    //Teleport wasn't successful
}
```

También puede agregar una causa a su teletransportador, para que pueda personalizar la forma en que su plugin o los demás tratarán su causa:

```
Entity toBeTeleported;    //The entity you want to teleport
Location teleportTo = new Location(world,x,y,z,yaw,pitch);    //The location to teleport to
```

```
PlayerTeleportEvent.TeleportCause cause;    //The cause you want the teleport to be of


boolean success = toBeTeleported.teleport(teleportTo, cause);

if(success) {
    //Teleport was successful
}else {
    //Teleport wasn't successful
}
```

Tipo de entidad

La enumeración EntityType representa todas las entidades de Bukkit / Spigot.

Todos sus valores se pueden encontrar a continuación.

Valor	Descripción
AREA_EFFECT_CLOUD	N / A
ARMOR_STAND	Entidad mecánica con un inventario para colocar armas / armaduras en.
FLECHA	Un proyectil de flecha; Puede quedar atrapado en el suelo.
MURCIÉLAGO	N / A
RESPLANDOR	N / A
BOTE	Un barco lugar
ARAÑA DE LAS CUEVAS	N / A
POLLO	N / A
COMPLEX_PART	N / A
VACA	N / A
ENREDADERA	N / A
DRAGON_FIREBALL	Como el FIREBALL, pero con efectos extra.
DROPPED_ITEM	Un objeto que descansa en el suelo.
HUEVO	Un huevo de gallina volando.
ENDER_CRYSTAL	N / A
ENDER_DRAGON	N / A

Valor	Descripción
PERLA DEL FIN	Una perla ender voladora.
ENDER_SIGNAL	Una señal ocular ender.
ENDERMAN	N / A
Endermita	N / A
EXPERIENCIA_ORB	Un orbe de experiencia.
FALLING_BLOCK	Un bloque que va a caer o está a punto de caer.
FIREBALL	Una bola de fuego grande voladora, lanzada por un Ghast, por ejemplo.
FUEGOS ARTIFICIALES	Representación interna de un fuego artificial una vez que ha sido lanzado.
ANZUELO DE PESCA	Una línea de pesca y bobber.
GHAST	N / A
GIGANTE	N / A
GUARDIÁN	N / A
CABALLO	N / A
GOLEM DE HIERRO	N / A
CUADRO DE ARTÍCULO	Un marco de elemento en una pared.
LEASH_HITCH	Una correa atada a un poste.
RELÁMPAGO	Un rayo.
LINGERING_POTION	Una poción voladora persistente
CUBO DE MAGMA	N / A
MINECART	N / A
MINECART_CHEST	N / A
MINECART_COMMAND	N / A
MINECART_FURNACE	N / A
MINECART_HOPPER	N / A

Valor	Descripción
MINECART_MOB_SPAWNER	N / A
MINECART_TNT	N / A
MUSHROOM_COW	N / A
OCELOTE	N / A
PINTURA	Una pintura en una pared.
CERDO	N / A
PIG_ZOMBIE	N / A
JUGADOR	N / A
OSO POLAR	N / A
PRIMED_TNT	Cebado TNT que está a punto de explotar.
CONEJO	N / A
OVEJA	N / A
SHULKER	N / A
SHULKER_BULLET	Bala disparada por SHULKER.
LEPISMA	N / A
ESQUELETO	N / A
LIMO	N / A
SMALL_FIREBALL	Una bola de fuego pequeña voladora, como lanzada por un Blaze o jugador.
BOLA DE NIEVE	Una bola de nieve volando.
MONIGOTE DE NIEVE	N / A
SPECTRAL_ARROW	Como TIPPED_ARROW pero causa el efecto PotionEffectType.GLOWING en todos los miembros del equipo.
ARAÑA	N / A
SPLASH_POTION	Una poción voladora.
CALAMAR	N / A

Valor	Descripción
THROWN_EXP_BOTTLE	Una experiencia de vuelo en botella.
TIPPED_ARROW	Como FLECHA pero con una poción específica que se aplica al contacto.
DESCONOCIDO	Una entidad desconocida sin una clase de entidad
ALDEANO	N / A
CLIMA	N / A
BRUJA	N / A
MARCHITAR	N / A
WITHER_SKULL	Un proyectil de calavera marchita.
LOBO	N / A
ZOMBI	N / A

Pasajero

Las entidades pueden tener pasajeros. Un buen ejemplo de un pasajero es un jugador que monta un cerdo ensillado o un zombi dentro de una minecart.

Aunque hay vehículos específicos, cualquier entidad puede ser un vehículo para cualquier otra entidad con el método SetPassenger.

```
Entity vehicle;
Entity passenger;
boolean result = vehicle.setPassenger(passenger);   //False if couldn't be done for whatever
reason
```

El pasajero ahora debe estar unido al vehículo.

Puede verificar si una entidad tiene un pasajero usando

```
boolean hasPassenger =  entity.isEmpty()
```

Si la entidad tiene un pasajero, puede recuperar la entidad de pasajeros con

```
Entity passenger = entity.getPassenger();
```

Solo devolverá el pasajero principal si el vehículo puede tener múltiplos.

Finalmente, puedes expulsar al pasajero de una entidad con

```
boolean b = entity.eject();   //Eject all passengers - returns true if there was a passenger
to be ejected
```

Entidades cercanas

Para recuperar una lista de entidades cercanas de una entidad, se puede usar

```
List<Entity> nearby = entity.getNearbyEntities(double x, double y, double z);
```

Bukkit calculará un cuadro delimitador centrado alrededor de la entidad, teniendo como parámetros:

- x: 1/2 del tamaño de la caja a lo largo del eje x
- y: 1/2 del tamaño de la caja a lo largo del eje y
- z: 1/2 del tamaño del cuadro a lo largo del eje z

La lista puede estar vacía, lo que significa que no hay entidades cercanas con los parámetros.

Este enfoque se puede usar para detectar entidades cerca de proyectiles personalizados, por ejemplo, lanzando un Itemstack y detectando cuando colisiona con un jugador

Lea Entidades en línea: https://riptutorial.com/es/bukkit/topic/7886/entidades

Capítulo 5: Eventos de la entidad

Introducción

Todos los eventos de entidad extienden EntityEvent, la superclase para EntityEvents.

Todos los EntityEvents conocidos se pueden encontrar a continuación, y se tratarán en esta documentación.

Examples

Evento EntityDamage

El evento EntityDamage se lanza cuando una Entidad está dañada.

EntityDamageEvent

```
@EventHandler
public void onEntityDamage(EntityDamageEvent e) {
    DamageCause cause = e.getCause();    //Get the event DamageCause
    double rawDamage = e.getDamage();    //Returns the damage before any calculation
    double damageModified = e.getDamage(DamageModifier.X);     //Returns the damage that would
be caused with the specified modifier
    double finalDamage = e.getFinalDamage();    //Gets the final damage of this event, with
all the calculations included

    e.setCancelled(boolean x);    //If for any reasons you want the event to not happen, you
can cancel it
    e.setDamage(double damage);    //You can change the full damage the event will cause
    e.setDamage(DamageModifier modifier, double damage);    //Changes the damage considering
any possible modifier
}
```

La mayoría de las veces, el EntityDamageEvent no se utilizará. En su lugar, se utilizará una de sus subclases, como **EntityDamageByEntityEvent** o **EntityDamageByBlockEvent** . Ambos se pueden ver a continuación.

EntityDamageByEntityEvent

```
@EventHandler
public void onEntityDamageByEntity(EntityDamageByEntityEvent e) {
    //Retrieving the Entity that dealt damage
    Entity damageDealer = e.getDamager();

    //Retrieving the Entity that took damage
    Entity damageTaker = e.getEntity();
```

```
    //Retrieving the cause of the damage
    DamageCause cause = e.getDamageCause();

    //damage is the double value of the damage before all the resistances and modifiers have
been applied
    double damage = e.getDamage();

    //FinalDamage is the double value of the damage after all the resistances and modifiers
have been applied
    double finalDamage = e.getFinalDamage();

   //You can also set the raw damage (before modifiers) for the event to a different value
   e.setDamage(20.0);
}
```

EntityDamageByBlockEvent

Una extensión simple para EntityDamageEvent, pero con un método diferente:

```
Block b = event.getDamager();    //Returns the block that dealt damage to the entity
```

EntityEvent (Superclase)

Las subclases conocidas para eventos de entidad son:

Subclases	Subclases	Subclases
CreatureSpawnEvent	CreeperPowerEvent	EntityChangeBlockEvent
EntityCombustEvent	EntityCreatePortalEvent	EntityDamageEvent
EntityDeathEvent	EntityExplodeEvent	EntityInteractEvent
EntityPortalEnterEvent	EntityRegainHealthEvent	EntityShootBowEvent
EntityTameEvent	EntityTargetEvent	EntityTeleportEvent
EntidadUnleashEvent	ExplosionPrimeEvent	ComidaNivelCambiarEvento
HorseJumpEvent	ArtículoDespawnEvent	ItemSpawnEvent
PigZapEvent	ProjectileHitEvent	ProjectileLaunchEvent
SheepDyeWoolEvent	SheepRegrowWoolEvent	SlimeSplitEvent

Además de esto, todas las subclases heredan los siguientes métodos:

```
Entity getEntity();            //Entity who is involved in this event
EntityType getEntityType();    //EntityType of the Entity involved in this event
```

Lea Eventos de la entidad en línea: https://riptutorial.com/es/bukkit/topic/6486/eventos-de-la-entidad

Capítulo 6: Eventos del Jugador

Introducción

Esta es una lista de eventos de jugadores y un ejemplo sobre cómo usarlos.

Examples

PlayerJoinEvent

```
public class PlayerJoinListener implements Listener {
    @EventHandler
    public void onPlayerJoin(PlayerJoinEvent evt) {
        Player joined = evt.getPlayer();
        String joinedName = joined.getName();

       //RETRIEVING THE JOIN MESSAGE ALREADY SET
       String joinMessage = evt.getJoinMessage();

       //SETTING THE JOIN MESSAGE
       evt.setJoinMessage(joinedName + " has joined the game");

       //CLEARING THE JOIN MESSAGE
       evt.setJoinMessage(null);
    }
}
```

PlayerMoveListener

```
public class PlayerMoveListener implements Listener {
    @EventHandler
    public void onPlayerMove(PlayerMoveEvent evt) {
        Location from = evt.getFrom();
        Location to = evt.getTo();
        double xFrom = from.getX();
        double yFrom = from.getY();
        double zFrom = from.getZ();
        double xTo = to.getX();
        double yTo = to.getY();
        double zTo = to.getZ();

        Bukkit.getLogger().info("Player " + evt.getPlayer().getName()
                        +  " has moved from x: " + xFrom + " y: " + yFrom + " z: "
                        + zFrom + " to x: " + xTo + " y: " + yTo + " z: " + zTo);
    }
}
```

PlayerLoginEvent

Detalles de tiendas de eventos para jugadores que intentan iniciar sesión

```
@EventHandler
public void onPlayerLogin(PlayerLoginEvent e) {
    Player tryingToLogin = e.getPlayer();

    //Disallowing a player login
    e.disallow(PlayerLoginEvent.Result.KICK_FULL , "The server is reserved and is full for
you!");

    //Allowing a player login
    if (e.getResult() != PlayerLoginEvent.Result.ALLOW) {
        if (isVip(tryingToLogin) ){
            e.allow();
        }
    }

    //Getting player IP
    String ip = e.getAddress();

    //Get the hostname player used to login to the server
    String ipJoined = e.getHostname();

    //Get current result from the login attempt
    PlayerLoginEvent.Result result = e.getResult();

    //Set kick message if Result wasn't ALLOW
    e.setKickMessage("You were kicked!");

    //Retrieve the kick message
    String s = e.getKickMessage();

}
```

PlayerLoginEvent.Result ENUM:

- PERMITIDO - El jugador puede iniciar sesión
- KICK_BANNED - El jugador no tiene permiso para iniciar sesión, debido a que están prohibidos
- KICK_FULL: el reproductor no tiene permiso para iniciar sesión debido a que el servidor está lleno
- KICK_OTHER - El jugador no tiene permitido iniciar sesión, por razones no definidas
- KICK_WHITELIST - El jugador no tiene permiso para iniciar sesión, debido a que no están en la lista blanca

Eventos de la cama del jugador

Evento disparado cuando el jugador entra en la cama: **PlayerBedEnterEvent**

PlayerBedEnterEvent (Player who, Block bed)

```
@EventHandler
public void onPlayerBedEnter(PlayerBedEnterEvent e) {
    Player entered = e.getPlayer();

   Block bedEntered = e.getBed();
```

```
}
```

Evento **activado** cuando el jugador deja una cama: **PlayerBedLeaveEvent**

PlayerBedLeaveEvent (Player who, Block bed)

```
@EventHandler
public void onPlayerBedEnter(PlayerBedEnterEvent e) {
    Player entered = e.getPlayer();

   Block bedEntered = e.getBed();
}
```

Lea Eventos del Jugador en línea: https://riptutorial.com/es/bukkit/topic/8905/eventos-del-jugador

Capítulo 7: Explotación florestal

Examples

Usando el Bukkit Logger

```
public class MyClass {

public void foo() {
    Logger logger = Bukkit.getLogger();

    logger.info("A log message");
    logger.log(Level.INFO, "Another message");
    logger.fine("A fine message");

        // logging an exception
        try {
            // code might throw an exception
        } catch (SomeException ex) {
            // log a warning printing "Something went wrong"
            // together with the exception message and stacktrace
            logger.log(Level.WARNING, "Something went wrong", ex);
        }

        String s = "Hello World!";

        // logging an object
        LOG.log(Level.FINER, "String s: {0}", s);

        // logging several objects
        LOG.log(Level.FINEST, "String s: {0} has length {1}", new Object[]{s, s.length()});
    }

}
```

Niveles de registro

Java Logging Api tiene 7 niveles . Los niveles en orden descendente son:

- `SEVERE` (valor más alto)
- `WARNING`
- `INFO`
- `CONFIG`
- `FINE`
- `FINER`
- `FINEST` (valor más bajo)

El nivel predeterminado es `INFO` (pero esto depende del sistema y utiliza una máquina virtual).

Nota : También hay niveles `OFF` (se pueden usar para desactivar el registro) y `ALL` (el opuesto de `OFF`).

Código de ejemplo para esto:

```
import java.util.logging.Logger;

public class Levels {
    private static final Logger logger = Bukkit.getLogger();

    public static void main(String[] args) {

        logger.severe("Message logged by SEVERE");
        logger.warning("Message logged by WARNING");
        logger.info("Message logged by INFO");
        logger.config("Message logged by CONFIG");
        logger.fine("Message logged by FINE");
        logger.finer("Message logged by FINER");
        logger.finest("Message logged by FINEST");

        // All of above methods are really just shortcut for
        // public void log(Level level, String msg):
        logger.log(Level.FINEST, "Message logged by FINEST");
    }
}
```

Por defecto, ejecutar esta clase solo generará mensajes con un nivel más alto que `CONFIG` :

```
Jul 23, 2016 9:16:11 PM LevelsExample main
SEVERE: Message logged by SEVERE
Jul 23, 2016 9:16:11 PM LevelsExample main
WARNING: Message logged by WARNING
Jul 23, 2016 9:16:11 PM LevelsExample main
INFO: Message logged by INFO
```

Lea Explotación florestal en línea: https://riptutorial.com/es/bukkit/topic/7202/explotacion-florestal

Capítulo 8: Generacion mundial

Examples

Generador de vacío

La clase de generador de vacío:

```
public class VoidGenerator extends ChunkGenerator
{

    @SuppressWarnings("deprecation")
    public byte[] generate(World w, Random rand, int x, int z)
    {
        byte[] result = new byte[32768]; //chunksized array filled with 0 - Air
        //Build a platform with Bedrock where the player shall spawn later
        if(x == 0 && z == 0)
        {
            result[xyz(0, 64, 0)] = (byte)Material.BEDROCK.getId();
            result[xyz(1, 64, 0)] = (byte)Material.BEDROCK.getId();
            result[xyz(0, 64, 1)] = (byte)Material.BEDROCK.getId();
        }
        return result;
    }

    private Integer xyz(int x, int y, int z)
    {
        return (x * 16 + z)*128+y; //position inside the chunk
    }

}
```

Cualquier clase donde quieras generar un nuevo mundo:

```
public void generateWorld(String mapName)
{
    try
    {
        WorldCreator w = new WorldCreator(mapName);
        w.generateStructures(false); //no trees, etc.
        w.generator(new VoidGenerator()); //use the VoidGenerator
        w.environment(Environment.NORMAL); //no nether, etc.
        w.createWorld(); //create the world
    }
    catch(Exception ex)
    {
      ex.printStackTrace();
    }
}
```

Lea Generacion mundial en línea: https://riptutorial.com/es/bukkit/topic/6652/generacion-mundial

Capítulo 9: Huevos de engendro

Observaciones

Consulte la Documentación de Entidades para entender mejor EntityType

Examples

Creando un ItemStack de un SpawnEgg

Para cualquier cosa por debajo de 1.9

```
SpawnEgg egg = new SpawnEgg(EntityType.CREEPER);
ItemStack creeperEgg = egg.toItemStack(5);
```

Para 1.9 y superior

En las versiones 1.9 y posteriores, Spigot no tiene una implementación para crear huevos de desove sin usar NMS. Para lograr esto, puedes usar una pequeña clase / envoltorio personalizado para que esto suceda:

```
public ItemStack toItemStack(int amount, EntityType type) {
    ItemStack item = new ItemStack(Material.MONSTER_EGG, amount);
    net.minecraft.server.v1_9_R1.ItemStack stack = CraftItemStack.asNMSCopy(item);
    NBTTagCompound tagCompound = stack.getTag();
    if(tagCompound == null){
        tagCompound = new NBTTagCompound();
    }
    NBTTagCompound id = new NBTTagCompound();
    id.setString("id", type.getName());
    tagCompound.set("EntityTag", id);
    stack.setTag(tagCompound);
    return CraftItemStack.asBukkitCopy(stack);
}
```

Lea Huevos de engendro en línea: https://riptutorial.com/es/bukkit/topic/7737/huevos-de-engendro

Capítulo 10: Manejo de eventos

Introducción

Cuando sucede algo dentro de Bukkit, se llama un evento para que cada complemento pueda decidir qué hacer cada vez que sucede algo.

Se llama a un evento cuando un jugador intenta jugar un bloque, cuando una entidad desaparece, cuando alguien inicia sesión ... Los complementos pueden escuchar eventos específicos y tratarlos de muchas maneras diferentes, por ejemplo, enviando un mensaje a un administrador cuando un jugador inicia sesión, a través del PlayerLoginEvent.

Sintaxis

- Bukkit.getPluginManager (). RegisterEvents (Listener l, Plugin p);

Observaciones

¡Al registrar un evento, eche un vistazo si no lo está registrando dos veces! O su complemento actuará dos veces para el evento registrado.

Eche un vistazo adicional sobre cómo manejar eventos específicos:

- Eventos del Jugador
- Eventos de la entidad

Examples

Registrar eventos dentro de la clase de oyente

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

public class MyEventListener implements Listener {

    /**
     * Constructor
     */
    public MyEventListener(Main plugin){
        //register Events of this class
        //with methode: registerEvents(Listener, Plugin);
        plugin.getServer().getPluginManager().registerEvents(this, plugin);
    }

    /**
```

```
     * A Event with HIGH priority
     */
    @EventHandler(priority = EventPriority.HIGH) //An EventHandler annotation
    public void onPlayerLogin(PlayerLoginEvent event){  //A bukkit event
        event.getPlayer().sendMessage("Welcome.");
    }
    /**
     * A Event with NORMAL (default) priority
     */
    @EventHandler
    public void onPlayerQuit(PlayerQuitEvent event){
        Bukkit.broadcastMessage(event.getPlayer().getName() + " left the Server.");
    }


}

/**
 * Main class
 */
public class Main extends JavaPlugin {
    public void onEnable(){
        //Register Events
        new MyEventListener(this);
    }
}
```

Registro de eventos a tu clase principal

```
public class Main extends JavaPlugin {

    @Override
    public void onEnable() {
        Bukkit.getPluginManager().registerEvents(this, this);
    }

    @EventHandler
    public void yourEvent(Event e) {
    //...
    }
}
```

Escuchar eventos

Bukkit utiliza un sistema basado en eventos que permite a los desarrolladores de complementos interactuar con el servidor y modificarlo y las acciones específicas que ocurren en el mundo.

Creación de un controlador de eventos

Los controladores de eventos son métodos que se llaman cuando ocurre su evento. En general, son públicos y no `on{EventNameStem}` así como se nombran `on{EventNameStem}` por convención. Sin embargo, todos los manejadores deben tener la anotación `@EventHandler` , así como contener su

evento como el ÚNICO parámetro. Aquí hay un ejemplo de un controlador de eventos para `PlayerJoinEvent`

```
@EventHandler
public void onPlayerJoin(PlayerJoinEvent event){
    //Run when a player joins
}
```

Nota: El formato de denominación para los eventos de Bukkit es el `{Source}{Action}({Target})Event` . Algunos ejemplos de estos nombres de eventos son: `PlayerInteractEvent` o `BlockBreakEvent` o `PlayerJoinEvent` . Se puede encontrar una lista de todos los eventos en los Javadocs de Spigot

Registro de eventos

Simplemente la creación de un controlador de eventos no es suficiente para permitir que Bukkit comience a enviar llamadas de eventos a su método. También debe registrarlo a través de la interfaz de PluginManager.

La forma más común de registrar eventos es crear una clase que implemente la interfaz de Oyente y usarla para envolver sus controladores de eventos.

```
public class EventListener implements Listener { //Implements the Listener interface

    @EventHandler
    public void onPlayerJoin(PlayerJoinEvent event){
        //Run when a player joins
    }

}
```

Esta clase de oyente y todos sus eventos se pueden registrar en su clase de complemento principal de la siguiente manera:

```
@Override
public void onEnable(){
    Bukkit.getPluginManager().registerEvents(new EventListener(), this); //Register your
listener and its event handlers
}
```

Creación de eventos personalizados

A veces necesitas crear tu propio evento, uno que otros complementos puedan escuchar (Vault, entre otros complementos, hace esto) e incluso cancelar. La API de eventos de Bukkit permite que esto sea posible. Todo lo que necesita hacer es crear una nueva clase, extender el `Event` , agregar los controladores y los atributos que su evento necesita (como Jugador o mensaje).

```
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;
    }
}
```

Llamando a tu evento personalizado

Usted tiene el control de crear y llamar a sus eventos, donde lo llame, depende completamente de usted. Aquí un ejemplo

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

Recuerda: tienes el control de tus eventos. Si no lo llamas, y actúas sobre él, ¡no sucede!

Escuchando un evento personalizado

Escuchar un evento personalizado es lo mismo que escuchar un evento 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
    }
}
```

Haciendo su CustomEvent Cancellable

Si alguna vez desea que su evento sea `boolean cancelled` , solo agregue `implements Cancellable` , `boolean cancelled` y un captador y configurador:

```
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;
    }
}
```

Luego, verificará si un complemento canceló el evento personalizado.

```
// 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());
}
```

Manejo básico de eventos

Bukkit utiliza un sistema basado en eventos que permite a los desarrolladores de complementos interactuar con el servidor y modificarlo y las acciones específicas que ocurren en el mundo.

Creación de un controlador de eventos

Los controladores de eventos son métodos que se llaman cuando ocurre su evento. En general, son públicos y no `on{EventNameStem}` así como se nombran `on{EventNameStem}` por convención. Sin embargo, todos los manejadores deben tener la anotación `@EventHandler` , así como contener su evento como el ÚNICO parámetro. Aquí hay un ejemplo de un controlador de eventos para `PlayerJoinEvent`

```
@EventHandler
public void onPlayerJoin(PlayerJoinEvent event){
    //Run when a player joins
}
```

Nota: El formato de denominación para los eventos de Bukkit es el `{Source}{Action}({Target})Event` . Algunos ejemplos de estos nombres de eventos son: `PlayerInteractEvent` o `BlockBreakEvent` o `PlayerJoinEvent` . Se puede encontrar una lista de todos los eventos en los Javadocs de Spigot

Registro de eventos

Simplemente la creación de un controlador de eventos no es suficiente para permitir que Bukkit comience a enviar llamadas de eventos a su método. También debe registrarlo a través de la interfaz de PluginManager.

La forma más común de registrar eventos es crear una clase que implemente la interfaz de Oyente y usarla para envolver sus controladores de eventos.

```
public class EventListener implements Listener { //Implements the Listener interface

    @EventHandler
    public void onPlayerJoin(PlayerJoinEvent event){
        //Run when a player joins
    }

}
```

Esta clase de oyente y todos sus eventos se pueden registrar en su clase de complemento principal de la siguiente manera:

```
@Override
public void onEnable(){
    Bukkit.getPluginManager().registerEvents(new EventListener(), this); //Register your
listener and its event handlers
}
```

Prioridades de eventos

Bukkit tiene un sistema llamado **Prioridad de eventos** para ayudar a los complementos a manejar los eventos en la antigüedad correcta. Las siete prioridades son (en la más antigua, desde la primera hasta la última):

- Más bajo
- Bajo
- Normal (por defecto)
- Alto
- Más alto
- Monitor

Si planea cancelar una gran cantidad de eventos (por ejemplo, un complemento de protección) sería una buena idea usar la prioridad más baja (o la más baja) para evitar problemas.

Nunca debes modificar el resultado de un evento en MONITOR.

```
@EventHandler //same as @EventHandler(priority = EventPriority.NORMAL)
public void onLogin(PlayerLoginEvent event) {
    // normal login
}

@EventHandler(priority = EventPriority.HIGH)
public void onLogin(PlayerLoginEvent event) {
    // high login
}
```

Más información:

- EventPriority at spigot javadocs
- Prioridades del evento en BukkitWiki

Lea Manejo de eventos en línea: https://riptutorial.com/es/bukkit/topic/5743/manejo-de-eventos

Capítulo 11: Manipulación mundial

Observaciones

Consulte World Generation para temas de generación mundial.

Examples

Creando Explosiones

Para crear una explosión, se pueden usar las siguientes firmas de métodos:

```
boolean createExplosion(double x, double y, double z, float power);
boolean createExplosion(double x, double y, double z, float power, boolean setFire);
boolean createExplosion(double x, double y, double z, float power,
                        boolean setFire, boolean breakBlocks);
boolean createExplosion(Location loc, float power);
boolean createExplosion(Location loc, float power, boolean setFire);
```

- x, y, z y loc representan la ubicación donde desea que ocurra la explosión.
- la potencia representa la potencia de su explosión, la potencia TnT es 4F.
- setFire representa la capacidad de la explosión para incendiar bloques
- breakBlocks representa la capacidad de la explosión para destruir bloques a su alrededor.
- todos los métodos devuelven verdadero si la explosión sucedió, y devuelve falso si un complemento canceló el evento de explosión.

Simulando una explosión TnT que rompe bloques y prende fuego a x = 0, y = 0 y z = 0

```
createExplosion(0.0, 0.0, 0.0, 4F, true, true);
```

Dejar caer un artículo

Los siguientes métodos se pueden usar para colocar un artículo en algún lugar del mundo:

```
Item dropItem(Location loc, ItemStack is);
Item dropItemNaturally(Location loc, ItemStack is);
```

`dropItem` significa dejar caer un artículo exactamente en la ubicación, devolviendo un objeto de artículo.

`dropItemNaturally` significa dejar caer el elemento en la ubicación, pero con un desplazamiento aleatorio, lo que significa que no estará exactamente en la ubicación, pero muy cerca. Esto se hace para simular un elemento que está cayendo una entidad o un bloque, como un Dispensador.

Generando un arbol

Los siguientes métodos se pueden utilizar para generar un árbol de forma natural (como si fuera un árbol joven) en el mundo.

```
boolean generateTree(Location loc, TreeType type);
boolean generateTree(Location loc, TreeType type, BlockChangeDelegate delegate);
```

- La ubicación es donde quieres que el árbol se engendre.
- TreeType es el tipo de árbol que desea generar y puede ser uno de los siguientes

Enumeración TreeType

Tipo	Descripción
ACACIA	árbol de acacia
ÁRBOL GRANDE	Árbol regular, extra alto con ramas.
ABEDUL	abedul
BROWN_MUSHROOM	Seta marrón grande; alto y parecido a un paraguas
CHORUS_PLANT	Planta grande originaria de The End.
COCOA_TREE	Árbol de la selva con plantas de cacao; 1 bloque de ancho
ROBLE OSCURO	Roble oscuro.
SELVA	Árbol de la selva estándar; 4 cuadras de ancho y alto
JUNGLE_BUSH	Arbusto pequeño que crece en la selva.
MEGA_REDWOOD	Mega secoya; 4 cuadras de ancho y alto
RED_MUSHROOM	Seta roja grande; Bajo y gordo
SECOYA	Árbol de secoya, con forma de pino
SMALL_JUNGLE	Árbol de la selva más pequeño; 1 bloque de ancho
PANTANO	Árbol de pantano (regular con enredaderas en el lateral)
TALL_BIRCH	Abedul alto
TALL_REDWOD	Árbol alto de secuoya con unas pocas hojas en la parte superior.
ÁRBOL	Árbol regular, sin ramas.

- se puede usar un delegado si desea que una clase llame para cada bloque cambiado como resultado de este método

Ambas firmas devolverán verdadero si el árbol se generó con éxito, de lo contrario falso.

Reglas de desove

Hay algunas reglas de desove en los mundos en Bukkit. Son:

- Desove Animal
- Desove de criaturas
- Cantidad de lo anterior que puede ser engendrada.

Desove Animal

El desove animal se puede dividir en las siguientes categorías:

- Animales acuáticos
- Animales terrestres

Para obtener la cantidad de animales que se pueden generar dentro del Mundo en tiempo de ejecución, puede utilizar el método

```
int getAnimalSpawnLimit()
```

Para animales terrestres y

```
int getWaterAnimalSpawnLimit();
```

Para animales de agua.

Ambos límites se pueden establecer con los métodos.

```
void setAnimalSpawnLimit(int limit);
void setWaterAnimalSpawnLimit(int limit);
```

Nota: si se configura en números inferiores a 0, se utilizará la cantidad predeterminada del mundo.

Minecraft intenta engendrar animales cada 400 garrapatas (predeterminado). Eso se puede cambiar si lo desea, utilizando las siguientes firmas:

```
void setTicksPerAnimalSpawns(int ticks);
void setTicksPerWaterAnimalSpawns(int ticks);
```

- Un valor de 1 significará que el servidor intentará engendrar animales en este mundo con cada tic.
- Un valor de 400 significará que el servidor intentará engendrar animales en este mundo cada 400 tic.

- Un valor por debajo de 0 se restablecerá al valor predeterminado de Minecraft.

Nota : Si se establece en 0, el desove animal se deshabilitará para este mundo. Se recomienda usar setSpawnFlags (boolean, boolean) para controlar esto en su lugar.

Lea Manipulación mundial en línea: https://riptutorial.com/es/bukkit/topic/7926/manipulacion-mundial

Capítulo 12: Manipulando logros

Sintaxis

- player.awardAchievement (Achievement ach);

Examples

Premiación de logros

```
Player player;     //The player you want to award your achievement with
Achievement achievement;     //The achievement you want to award your player

player.awardAchievement(achievement);    //Awarding the achievement
```

Comprobando si el jugador tiene logro:

```
Player player;
Achievement achievement;
boolean hasAchievement = player.hasAchievement(achievement);
```

El código otorgará el logro y cualquier logro de los padres.

Los posibles logros son:

- ACQUIRE_IRON (Adquirir hardware)
- BAKE_CAKE (La mentira)
- BOOKCASE (bibliotecario)
- BREED_COW (Repoblación)
- BREW_POTION (Cervecería local)
- BUILD_BETTER_PICKAXE (Obteniendo una actualización)
- BUILD_FURNACE (Tema de actualidad)
- BUILD_HOE (¡Hora de cultivar!)
- BUILD_PICKAXE (Time to Mine!)
- BUILD_SWORD (Time to Strike!)
- BUILD_WORKBENCH (Benchmarking)
- COOK_FISH (Pescado Delicioso)
- DIAMONDS_TO_YOU (¡Diamantes para ti!)
- ENCUENTROS (ENCANTADOR)
- END_PORTAL (¿El fin?)
- EXPLORE_ALL_BIOMES (Tiempo de aventura)
- FLY_PIG (Cuando los cerdos vuelan)
- FULL_BEACON (Beaconator)
- GET_BLAZE_ROD (Into Fire)
- GET_DIAMONDS (DIAMANTES!)

- GHAST_RETURN (Regresar al remitente)
- KILL_COW (Volquete de vaca)
- KILL_ENEMY (Cazador de monstruos)
- KILL_WITHER (El Principio).
- MAKE_BREAD (Hornear Pan)
- MINE_WOOD (Obtención de madera)
- NETHER_PORTAL (Necesitamos ir más profundo)
- ON_A_RAIL (en un carril)
- OPEN_INVENTORY (Inventario)
- Overkill (Overkill)
- SUPERPODERADO (Vencido)
- SNIPE_SKELETON (Duelo de francotirador)
- SPAWN_WITHER (¿El principio?)
- THE_END (The End.)

Referencia

Lea Manipulando logros en línea: https://riptutorial.com/es/bukkit/topic/7690/manipulando-logros

Capítulo 13: NMS

Introducción

NMS, también conocido como **N** et. **M** inecraft. **S** erver es el paquete que contiene el código del servidor central de Minecraft. Las clases en este paquete fueron creadas por Mojang (no Bukkit) y, por lo tanto, están en su mayor parte confusas y no están destinadas a ser utilizadas o alteradas. Sin embargo, la interacción con el código del servidor de Minecraft en este nivel le permite modificar casi todos los aspectos del mismo. Esto es importante porque hay numerosas modificaciones que Bukkit no admite.

Observaciones

La API de Bukkit es un envoltorio o capa de abstracción para NMS que permite a los desarrolladores de complementos interactuar con el servidor sin preocuparse por los cambios realizados en la base de código interna.

El uso del código NMS no se recomienda, ya que se interrumpe a menudo entre los cambios de la versión de Minecraft y no puede ser soportado por Bukkit o Spigot, ya que no lo crean, poseen ni mantienen.

Examples

Accediendo a la versión actual de Minecraft

Una de las partes más críticas de tratar con el código NMS es poder admitir múltiples versiones de Minecraft. Hay muchas formas de hacer esto, pero una solución simple es usar este código para almacenar la versión como un campo estático público:

```
public static final String NMS_VERSION =
Bukkit.getServer().getClass().getPackage().getName().substring(23);
```

Este fragmento de código funciona tomando la clase CraftServer:

```
org.bukkit.craftbukkit.VERSION.CraftServer.class
```

Obteniendo su paquete:

```
org.bukkit.craftbukkit.VERSION
```

Y tomando la subcadena del nombre del paquete a partir del índice 23 que siempre será después de 'org.bukkit.craftbukkit'. (que tiene una longitud de 23 caracteres). Resultando en la cadena final de la versión:

```
VERSION
```

Hay varias razones por las que es tan importante poder acceder a la versión actual de Minecraft.

Sobre todo porque cualquier acceso a una clase en un servidor que ejecute una versión de Minecraft diferente a la que codificó el complemento generará un Error.

Aquí hay un ejemplo que demuestra cómo resolver ese problema utilizando el campo `NMS_VERSION` para recuperar una instancia de CraftPlayer (que es una clase NMS) en cualquier versión de Minecraft.

```
/**
 * Invokes the getHandle() method on the player's CraftPlayer instance to
 * retrieve the EntityPlayer representation of the player as an Object to
 * avoid package version change issues
 *
 * @param player
 *            the player to cast
 * @return the NMS EnityPlayer representation of the player
 */
public static Object getCraftPlayer(Player player) {
    try {
        return Class.forName("org.bukkit.craftbukkit." + NMS_VERSION + ".entity.CraftPlayer")
                .getMethod("getHandle")
                .invoke(player);
    } catch (IllegalAccessException | IllegalArgumentException | InvocationTargetException |
NoSuchMethodException | SecurityException | ClassNotFoundException e) {
        throw new Error(e);
    }
}
```

El objeto resultante puede manipularse utilizando la reflexión para realizar tareas basadas en NMS sin preocuparse por intentar acceder a la versión incorrecta de la clase.

Sin embargo, incluso este método no es infalible, ya que los campos y los nombres de los métodos de NMS cambian fácilmente, así que lo único que garantiza al hacer esto es que su código no se romperá definitivamente cada vez que se actualice Minecraft.

Conseguir el ping de un jugador

Una cosa muy simple que podría querer hacer con NMS que Bukkit no admite es hacer ping al jugador. Esto se puede hacer así:

```
/**
 * Gets the players ping by using NMS to access the internal 'ping' field in
 * EntityPlayer
 *
 * @param player
 *            the player whose ping to get
 * @return the player's ping
 */
public static int getPing(Player player) {
    EntityPlayer entityPlayer = ((CraftPlayer) player).getHandle();
    return entityPlayer.ping;
}
```

Si está utilizando un método como getCraftPlayer (Player) que devuelve una instancia de la instancia de CraftPlayer correspondiente del Jugador como un Objeto. Puede acceder a los datos

sin importar las clases dependientes de la versión utilizando una reflexión como esta:

```
/**
 * Gets the player's ping using reflection to avoid breaking on a Minecraft
 * update
 *
 * @param player
 *            the player whose ping to get
 * @return the player's ping
 */
public static int getPing(Player player) {
    try {
        Object craftPlayer = getCraftPlayer(player);
        return (int) craftPlayer.getClass().getField("ping").get(craftPlayer);
    } catch (IllegalArgumentException | IllegalAccessException | NoSuchFieldException |
SecurityException e) {
        throw new Error(e);
    }
}
```

Lea NMS en línea: https://riptutorial.com/es/bukkit/topic/9576/nms

Capítulo 14: Ocultar jugadores

Sintaxis

- ocultar el vacío (Jugador para ocultar);
- show nulo (Player toShow);
- boolean canSee (Player toBeSeen);

Observaciones

Los eventos se cubren mejor en la lista de eventos de StackOverflow.

Examples

Ocultar un jugador de otros jugadores

```
Player playerToHide;
Player playerToNotSee;


playerToNotSee.hide(playerToHide);
//playerToHide will no longer be seen by playerToNotSee.
```

Si el jugador ya está oculto, no pasa nada.

Mostrando un jugador a otro jugador

```
Player toUnhide;
Player toSeeAgain

toSeeAgain.show(toUnhide);
//Player toSeeAgain will now see Player toUnhide again.
```

Si el jugador ya es visible, no pasa nada.

Comprobando si el jugador puede ser visto

```
Player playerToCheck;
Player playerSeeing;

boolean isVisible = playerSeeing.canSee(playerToCheck);
//isVisible returns true if playerSeeing can see playerToCheck and false otherwhise
```

Ocultar jugador de una entidad

Esto se puede hacer usando el evento EntityTargetEvent

Las entidades no apuntarán al jugador si cancelas el evento:

```
@EventHandler
public void onEntityTarget(EntityTargetEvent e) {
    Entity target = e.getEntity();
    if(target instanceof Player) {
        Player playerTargetted = (Player) target;
        if (shouldBeInvisible(playerTargetted) {
            e.setCancelled(true);    //Cancel the target event
        }
    }
}
```

Lea Ocultar jugadores en línea: https://riptutorial.com/es/bukkit/topic/7697/ocultar-jugadores

Capítulo 15: Programación del programador

Sintaxis

- `Bukkit.getScheduler().scheduleSyncRepeatingTask(Plugin plugin, Runnable task, int initialDelay, int repeatingDelay)`
- `Bukkit.getScheduler().scheduleSyncDelayedTask(Plugin plugin, Runnable task, int initialDelay)`
- `Bukkit.getScheduler().runTaskAsynchronously(Plugin plugin, Runnable task)`
- `Bukkit.getScheduler().runTask(Plugin plugin, Runnable task)`
- `new BukkitRunnable() { @Override public void run() { /* CODE */ } }.runTaskLater(Plugin plugin, long delay);`
- `new BukkitRunnable() { @Override public void run() { /* CODE */ } }.runTaskTimer(Plugin plugin, long initialDelay, long repeatingDelay);`

Observaciones

Pocos métodos de la API de Bukkit son seguros para subprocesos y se pueden llamar de forma asíncrona. Por esta razón, los métodos de la API de Bukkit *solo* deben ejecutarse, con algunas excepciones, en el hilo principal.

El código ejecutado dentro de `scheduleSync` métodos `scheduleSync` , así como el método `runTask` se ejecutarán en el hilo principal.

El código ejecutado dentro de `runTaskAsynchronously` se ejecutará de forma asíncrona desde el hilo principal. Los métodos asíncronos son muy útiles para realizar grandes operaciones matemáticas o de base de datos sin retrasar el servidor, pero causarán un comportamiento indefinido si se usan para llamar a los métodos de la API de Bukkit. Por esta razón, los métodos de la API de Bukkit que deben ejecutarse después del código asíncrono siempre deben ponerse en un método `runTask` .

Examples

Programador repitiendo tarea

El tiempo para las tareas del programador se mide en ticks. En condiciones normales, hay 20 garrapatas por segundo.

Las tareas programadas con `.scheduleSyncRepeatingTask` se ejecutarán en el subproceso principal

```
Bukkit.getScheduler().scheduleSyncRepeatingTask(plugin, new Runnable() {
    @Override
    public void run() {
        Bukkit.broadcastMessage("This message is shown immediately and then repeated every
second");
    }
}, 0L, 20L); //0 Tick initial delay, 20 Tick (1 Second) between repeats
```

Programador de tareas retrasadas

El tiempo para las tareas del programador se mide en ticks. En condiciones normales, hay 20 garrapatas por segundo.

Las tareas programadas con `.scheduleSyncDelayedTask` se ejecutarán en el subproceso principal

```
Bukkit.getScheduler().scheduleSyncDelayedTask(plugin, new Runnable() {
    @Override
    public void run() {
        Bukkit.broadcastMessage("This message is shown after one second");
    }
}, 20L); //20 Tick (1 Second) delay before run() is called
```

Ejecutar tareas de forma asíncrona

Puede hacer que el código se ejecute de forma asíncrona desde el hilo principal usando `runTaskAsynchronously` . Esto es útil para realizar operaciones intensivas de matemáticas o bases de datos, ya que evitarán que el hilo principal se congele (y que el servidor se retrase).

Algunos métodos de la API de Bukkit son seguros para la ejecución de subprocesos, por lo que muchos causarán un comportamiento indefinido si se los llama de forma asíncrona desde el subproceso principal.

```
Bukkit.getScheduler().runTaskAsynchronously(plugin, new Runnable() {
    @Override
    public void run() {
        Bukkit.getLogger().info("This message was printed to the console asynchronously");
        //Bukkit.broadcastMessage is not thread-safe
    }
});
```

Ejecutando tareas en el hilo principal

También puede hacer que el código se ejecute de forma síncrona con el hilo principal utilizando `runTask` . Esto es útil cuando desea llamar a los métodos de la API de Bukkit después de ejecutar el código de forma asíncrona desde el hilo principal.

El código llamado dentro de este Runnable se ejecutará en el subproceso principal, haciendo que sea seguro llamar a los métodos API de Bukkit.

```
Bukkit.getScheduler().runTask(plugin, new Runnable() {
    @Override
    public void run() {
        Bukkit.broadcastMessage("This message is displayed to the server on the main thread");
        //Bukkit.broadcastMessage is thread-safe
    }
});
```

Ejecutando un BukkitRunnable

El BukkitRunnable es un Runnable que se encuentra en Bukkit. Es posible programar una tarea directamente desde un BukkitRunnable, y también cancelarla desde su interior.

Importante: El tiempo en las tareas se mide en Ticks. Un segundo tiene 20 garrapatas.

Tarea no repetitiva:

```
JavaPlugin plugin;    //Your plugin instance
Long timeInSeconds = 10;
Long timeInTicks = 20 * timeInSeconds;
new BukkitRunnable() {

    @Override
    public void run() {
        //The code inside will be executed in {timeInTicks} ticks.

    }
}.runTaskLater(plugin, timeInTicks);   // Your plugin instance, the time to be delayed.
```

Repitiendo la tarea:

```
JavaPlugin plugin;    //Your plugin instance
Long timeInSeconds = 10;
Long timeInTicks = 20 * timeInSeconds;
new BukkitRunnable() {

    @Override
    public void run() {
        //The code inside will be executed in {timeInTicks} ticks.
       //After that, it'll be re-executed every {timeInTicks} ticks;
      //Task can also cancel itself from running, if you want to.

       if (boolean) {
           this.cancel();
       }

    }
}.runTaskTimer(plugin, timeInTicks, timeInTicks);   //Your plugin instance,
                                                     //the time to wait until first execution,
                                                    //the time inbetween executions.
```

Ejecución de código seguro de subproceso desde una tarea asíncrona

A veces necesitará ejecutar código síncrono desde una tarea asíncrona. Para hacer esto, simplemente programe una tarea síncrona desde dentro del bloque asíncrono.

```
Bukkit.getScheduler().runTaskTimerAsynchronously(VoidFlame.getPlugin(), () -> {

    Bukkit.getScheduler().runTask(VoidFlame.getPlugin(), () -> {
        World world = Bukkit.getWorld("world");
        world.spawnEntity(new Location(world, 0, 100, 0), EntityType.PRIMED_TNT);
    });

}, 0L, 20L);
```

Lea Programación del programador en línea:
https://riptutorial.com/es/bukkit/topic/5436/programacion-del-programador

Capítulo 16: Que cae

Observaciones

Actualmente no hay una manera consistente de evitar que una entidad sufra de gravedad, incluso si cancela su movimiento, el lado del jugador del jugador intentará caer antes de que se cancele el evento.

Examples

Entidad que cae distancia

La distancia de caída de la entidad es la distancia que la entidad ha caído sin alcanzar un bloque.

Puede utilizarse para calcular diferentes daños causados por caídas o la activación de un efecto después de una gran caída.

Recuperando la distancia de caída

```
float distanceFell = entity.getFallingDistance();
```

Ajuste de la distancia de caída

Esto se puede usar para simular una distancia de caída diferente a la real. Bukkit calculará el daño usando la nueva distancia de caída.

```
entity.setFallingDistance(float distance);
```

Cancelando Daños

Puedes cancelar un daño de caída usando el `EntityDamageEvent`

```
@EventHandler
public void onEntityDamage(EntityDamageEvent e) {
Entity tookDamage = e.getEntity();

DamageCause cause = e.getCause();

if (cause == DamageCause.FALL){
   //Damage was caused by falling, cancel it
   e.setCancelled(true);
}
```

Lea Que cae en línea: https://riptutorial.com/es/bukkit/topic/7899/que-cae

Capítulo 17: Scala

Introducción

Cómo implementar complementos de Bukkit en el lenguaje de programación Scala

Examples

Configuración del proyecto (Scala Eclipse)

Crear un proyecto en Scala es muy similar a crear uno en Java. Aquí es cómo debe verse la clase de entrada:

```
package com.example.myplugin; //{$TopLevelDomain}.{$Domain}.{$PluginName}

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

class PluginName extends JavaPlugin {

  override def onEnable() {

  }

  override def onDisable() {

  }

  override def onCommand(sender: CommandSender, cmd: Command, label: String, args:
Array[String]): Boolean = {

    false
  }

}
```

Primero, asegúrese de haber instalado la última versión de Scala que se encuentra aquí: https://www.scala-lang.org/download/

A continuación, querrá descargar Scala Eclipse, disponible aquí: http://scala-ide.org/ y extraer la descarga en una carpeta de su elección.

Una vez que ambos estén instalados, simplemente abra Scala Eclipse.

Por último, para que su complemento funcione, necesita tener algún tipo de complemento de tiempo de ejecución para cargar la biblioteca de Scala para usted, yo uso este: https://dev.bukkit.org/projects/scala-loader (coloque este jar en la carpeta de complementos como cualquier otro complemento)

De aquí en adelante, el proceso es casi idéntico a Java:

1. Presione `Alt+Shift+N` -> haga clic en `Scala Project`
2. Haga clic derecho en su proyecto - haga clic en `Properties`
3. Haga clic en `Java Build Path` , luego haga clic en la pestaña `Libraries`
4. Haga clic en `Add External Jars` y seleccione su archivo spigot-api jar
5. Haga clic en `Apply` y luego en `OK`

Para la configuración del proyecto, querrá crear un paquete para:

Haga clic derecho en el proyecto -> `New` -> `Package`

`com.yourdomain.pluginname` como quieras, normalmente: `com.yourdomain.pluginname`

Dentro de este paquete, cree una Clase Scala y `PluginName` nombre que desee, normalmente: `PluginName`

Make the class `extends JavaPlugin` y anula las funciones proporcionadas para una configuración básica como se muestra arriba.

Por último, haga clic derecho en la carpeta llamada "src" y seleccione Nuevo archivo. Nombre el archivo plugin.yml (NO el nombre de su complemento, sino explícitamente plugin.yml) y ábralo.

Una implementación básica debería verse así:

```
name: PluginName
main: com.example.pluginname.PluginName
version: 0.1
```

¡Y ahí lo tienes! Una vez que haya terminado de escribir su complemento, haga clic en `File` -> `Export` -> `Java` -> `Jar file` -> Seleccione su proyecto y especifique la `Jar file` complementos de su servidor como destino -> haga clic en `Finish`

Normalmente, puede simplemente volver a cargar su servidor para ver los cambios después de la exportación, sin embargo, **algunos complementos se interrumpirán al volver a cargar, ¡así que tenga cuidado!** Aconsejo siempre **reiniciar** el servidor a menos que sepa que la recarga no interrumpirá otros complementos.

Lea Scala en línea: https://riptutorial.com/es/bukkit/topic/9259/scala

Capítulo 18: Versiones

Examples

Obtener versión en tiempo de ejecución

```
@Override
public void onEnable() {
    String version = Bukkit.getBukkitVersion();    //The string version of this bukkit server
}
```

Acceso a NMS a través de versiones de implementación

```
// gets the Class objects from the net.mminecraft.server package with the given name
public Class<?> getNmsClass(String name) throws ClassNotFoundException {
    // explode the Server interface implementation's package name into its components
    String[] packageArray = Bukkit.getServer().getClass().getPackage().getName().split("\\.");

    // pick out the component representing the package version if it's present
    String packageVersion = array.length == 4 ? array[3] + "." : "";

    // construct the qualified class name from the obtained package version
    String qualName = "net.minecraft.server." + packageVersion + name;

    // simple call to get the Class object
    return Class.forName(qualName);
}
```

Lea Versiones en línea: https://riptutorial.com/es/bukkit/topic/7730/versiones

Creditos

S. No	Capítulos	Contributors
1	Empezando con bukkit	Community, Drayke, ItzPam, Jojodmo, Keenan Thompson, Kerooker, kmecpp, Martin W, RamenChef, Tassu, Unihedron
2	Archivos de configuración	Kerooker
3	Comandos	annon, Kerooker, Martin W
4	Entidades	Kerooker
5	Eventos de la entidad	Alw7SHxD, Ferrybig, Kerooker, kmecpp
6	Eventos del Jugador	Ferrybig, Kerooker, Tassu
7	Explotación florestal	Kerooker
8	Generacion mundial	Drayke
9	Huevos de engendro	Infuzed guy, Kerooker, RamenChef
10	Manejo de eventos	Drayke, Jarrod Dixon, Kerooker, kmecpp, Martin W, Tassu
11	Manipulación mundial	Kerooker
12	Manipulando logros	Kerooker
13	NMS	Alw7SHxD, kmecpp
14	Ocultar jugadores	Kerooker
15	Programación del programador	Ferrybig, Jojodmo, Kerooker, kmecpp, Pokechu22
16	Que cae	Kerooker
17	Scala	annon
18	Versiones	caseif, Kerooker