EBook Gratuito

APPRENDIMENTO bukkit

Free unaffiliated eBook created from
Stack Overflow contributors.

#bukkit

Sommario

Di

Capitolo 1: Iniziare con bukkit

Osservazioni

Bukkit è una semplice API che consente di modificare la normale esperienza multiplayer di Minecraft usando i plugin.

Bukkit è ora fuori produzione e non è più disponibile per le versioni più recenti di Minecraft. Spigot, una versione di Bukkit che vanta il miglioramento delle prestazioni del server è disponibile. L'API per Spigot è essenzialmente la stessa di Bukkit.

Versioni

Versione Minecraft	Spigot Link per il download	Data di rilascio
1.10.2	collegamento	2016/11/03
1.10	collegamento	2016/06/26
1.9.4	collegamento	2016/06/09
1.9.2	collegamento	2016/03/30
1.9	collegamento	2016/02/29
1.8.8	collegamento	2015/07/28
1.8.7	collegamento	2015/06/05
1.8.6	collegamento	2015/05/25
1.8.5	collegamento	2015/05/22
1.8.4	collegamento	2015/04/17
1.8.3	collegamento	2015/02/20
1.8	collegamento	2014/09/02
1.7.10	collegamento	2014/06/26
1.7.9	collegamento	2014/04/14
1.7.8	-	2014/04/11
1.7.5	collegamento	2014/02/26
1.7.2	collegamento	2013/10/25

Versione Minecraft	Spigot Link per il download	Data di rilascio
1.6.4	collegamento	2013/09/19
1.6.2	collegamento	2013/07/08
1.5.2	collegamento	2013/05/02
1.5.1	collegamento	2013/03/21
1.4.7	collegamento	2013/01/09
1.4.6	-	2012/12/20

Examples

Creare un plugin

Prerequisiti

- JDK 7 o superiore (consigliato: JDK 8+)

Aggiunta di Bukkit come dipendenza

Il metodo più semplice per aggiungere l'API Bukkit al progetto è scaricare Bukkit.jar direttamente dal repository Spigot e aggiungerlo al classpath del progetto. Le versioni legacy di Bukkit possono essere trovate nel Bukkit Repository .

L'altro è aggiungerlo come dipendenza Maven, aggiungendo le seguenti linee al tuo `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>
```

Classe principale

La classe principale del plugin è il punto di ingresso per Bukkit per caricare un'interazione con il tuo plug-in. È una classe che estende `JavaPlugin` e solo una sua istanza dovrebbe essere creata dal tuo plugin. Per convenzione è bene dare a questa classe lo stesso nome del plugin.

Ecco un esempio di una classe plugin principale per il plugin "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!");
    }

}
```

Per accedere all'istanza del plug-in da un'altra classe, è necessario memorizzare l'istanza della classe MyPlugin creata da Bukkit in modo che sia accessibile dall'esterno della classe.

```
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...
}
```

Quindi, per accedere alla tua classe principale da un'altra classe, usa semplicemente `MyPlugin.getInstance()`

```
public class MyOtherClass {

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

}
```

Creare un plugin.yml

Il file plugin.yml viene inserito nella radice del file jar finale e fornisce a Bukkit informazioni essenziali per caricare il plug-in. Il plugin.yml più semplice ha questo aspetto

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

Ad esempio con la classe MyPlugin sopra

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

Creare un server di prova su Windows

Per essere in grado di creare un server, è necessario avere il rubinetto o il file jar di bukkit. Riesegui l'accesso all'argomento delle versioni per selezionare il tuo jar

1. Innanzitutto, crea una nuova cartella. In quella cartella, inserisci il file jar di spigot / bukkit.
2. Fare clic con il tasto destro del mouse nella cartella e selezionare Nuovo> Documento di testo.
3. Denominare il nuovo documento start.bat, fare clic con il pulsante destro del mouse e fare clic su Modifica.
4. Aggiungi il seguente codice:

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

Non dimenticare di cambiare {YOUR_JAR.jar} per il barattolo scaricato prima di iniziare questi argomenti.

5. È possibile modificare `-Xms` per modificare la RAM minima consentita (Es: -Xms1024M = 1024 MB, -Xms1G = 1 GB). È inoltre possibile modificare `-Xmx` per modificare la RAM massima consentita. Assicurati che il massimo sia più grande del minimo.
6. Salva il file, chiudi la finestra e avvia il file `start.bat` . Il tuo server dovrebbe ora aprirsi. Per eseguire il server, è necessario accettare l' EULA .

7. Se accetti l' EULA , apri `eula.txt` cambia `eula=false` in `eula=true` Fai clic su "Salva" e ora dovresti essere in grado di avviare il tuo server.
8. Per connettersi al server, eseguire `start.bat` , aprire Minecraft, aggiungere un server e inserire `localhost` come IP.

buildtools

Che cos'è?

BuildTools.jar è una soluzione per la creazione di Bukkit, CraftBukkit, Spigot e Spigot-API. Tutto ciò è fatto sul tuo computer! Sono necessari alcuni programmi prerequisiti, ma le istruzioni seguenti ti guideranno attraverso tutto ciò che devi fare.

Prerequisiti

Ci sono due applicazioni necessarie per usare BuildTools: Git e Java.

finestre

Idiota

Affinché BuildTools possa essere eseguito su Windows, è necessario installare Git. Per Windows è distribuito tramite git-scm, che può essere scaricato qui . Installalo dove preferisci, fornirà git bash, che verrà usato per eseguire il jar BuildTools. Continua a colpire quando esegui il programma di installazione.

Giava

Scarica JRE 8 da qui e installa. Continua a colpire quando esegui il programma di installazione.

Linux

Sia git che Java, così come i comandi util, possono essere installati utilizzando un singolo comando tramite il gestore pacchetti.

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 può essere scaricato da: http://sourceforge.net/projects/git-osx-installer/files/

Potrebbe essere necessario aggiornare Java dalla versione distribuita di Apple e, anche se precedentemente aggiornato, potrebbe essere necessario collegarsi per l'utilizzo della shell. Si prega di seguire i passaggi trovati qui: https://gist.github.com/johan/10590467

Esecuzione di BuildTools

1. Scarica BuildTools.jar da https://hub.spigotmc.org/jenkins/job/BuildTools/lastSuccessfulBuild/artifact/target/BuildTools.jar .

2. Apri il tuo terminale se sei su Linux o git bash su Windows.

    1. Git bash può essere trovato sul desktop o nel menu Start sotto il nome "git bash". È anche possibile aprirlo facendo clic con il pulsante destro del mouse su qualsiasi cosa, poiché ora è un elemento nel menu di scelta rapida.

3. Passare al punto in cui è stato scaricato BuildTools.jar oppure utilizzare il metodo della riga di comando per scaricare il jar nella directory corrente.

    1. Su Windows, puoi usare il comando cd per cambiare directory, oppure puoi fare clic destro sullo spazio vuoto della cartella dove è BuildTools.jar (NON fare clic su BuildTools.jar) e fare clic su "git bash", che lo aprirà nella tua directory attuale.

4. Esegui BuildTools.jar dal terminale (non fare doppio clic su BuildTools.jar) nel modo seguente:

    1. Su Linux esegui git config --global --unset core.autocrlf, quindi esegui java -jar BuildTools.jar in bash o in un'altra shell appropriata.
    2. Su Windows, esegui il seguente comando all'interno della finestra di git bash che si apre: java -jar BuildTools.jar Tieni presente che è necessario che tu abbia BuildTools # 35 o successivo, le versioni precedenti non funzioneranno.
    3. Su Mac esegui i seguenti comandi, esporta MAVEN_OPTS = "- Xmx2G" java -Xmx2G -jar BuildTools.jar
    4. Se hai bisogno di una versione precedente, puoi specificare la versione usando l'argomento --rev a BuildTools, ad esempio per 1.8.8: java -jar BuildTools.jar --rev 1.8.8

5. Aspetta mentre costruisce i tuoi vasi. In pochi minuti dovresti avere dei vasetti appena compilati!

6. Puoi trovare CraftBukkit e Spigot nella stessa directory in cui hai eseguito BuildTools.jar (per Minecraft versione 1.10, sarebbero craftbukkit-1.10.jar e spigot-1.10.jar). Puoi trovare Spigot-API in \ Spigot \ Spigot-API \ target \ (per minecraft versione 1.10, sarebbe spigot-api-

1.10-R0.1-SNAPSHOT.jar).

Leggi Iniziare con bukkit online: https://riptutorial.com/it/bukkit/topic/5400/iniziare-con-bukkit

Capitolo 2: Caduta

Osservazioni

Al momento non esiste un modo coerente per evitare sofferenza gravitazionale, anche se annulli il suo movimento, il lato client del giocatore cercherebbe ancora di cadere prima che l'evento venga annullato.

Examples

Entity Falling Distance

La distanza di caduta delle entità è la distanza in cui l'entità è caduta senza raggiungere un blocco.

Può essere usato per calcolare danni diversi dalla caduta, o attivare un effetto dopo una grande caduta.

Recupero della distanza di caduta

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

Impostazione della distanza di caduta

Questo può essere usato per simulare una distanza di caduta diversa da quella reale. Bukkit calcolerà il danno usando la nuova distanza di caduta.

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

Annullare il danno

È possibile annullare un danno da caduta utilizzando `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);
}
```

Leggi Caduta online: https://riptutorial.com/it/bukkit/topic/7899/caduta

Capitolo 3: comandi

Sintassi

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

Examples

Gestire un comando

Per gestire un comando, è necessario disporre di una classe che implementa l'interfaccia CommandExecutor. La classe JavaPlugin (la classe principale del tuo plugin) implementa già questo.

Quando si implementa l'interfaccia CommandExecutor, è necessario implementare il seguente metodo:

```
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

}
```

Il mittente è colui che ha inviato il comando. Può essere un giocatore o la console.

CMD è il comando che stai ascoltando, come dichiarato in plugin.yml. Da non confondere con l'etichetta.

label è l'alias utilizzato per eseguire questo comando, è ciò che il mittente digita dopo la barra.

e infine, args sono gli argomenti che il mittente potrebbe aver usato per inviare il tuo comando.

Un possibile comando potrebbe andare come

    / dì a Kerooker Ciao, Kerooker!

Tell sarebbe la tua etichetta e potrebbe anche essere definito come tuo comando se lo hai detto in plugin.yml;

"Kerooker", "Ciao", "Kerooker!" sono i tuoi args 0, 1 e 2, rispettivamente

Come ritorno, probabilmente vorrai sempre tornare vero quando ti aspettavi che tutto succedesse in quel modo. Dovresti restituire false se vuoi mostrare al mittente l'utilizzo del comando definito nel tuo plugin.yml

Un semplice comando GameMode (/ gm)

Questo esempio mostra un esempio molto semplice di come utilizzare onCommand. Non suggerisco di elaborare i tuoi comandi direttamente in onCommand, ma questo è il trucco per questo semplice caso.

In questo esempio, tentiamo di impostare la gamemode del giocatore.

La prima cosa che dobbiamo fare è assicurarsi che il mittente non sia un ConsoleCommandSender, perché non possiamo impostare la gamemode di una console. Questo è fatto con (sender instanceof Player).

Quindi, vogliamo che il giocatore digiti / gm CREATIVE (o qualsiasi altro gamemode), quindi dobbiamo controllare 2 cose:

1. assicurati che passino in 1 argomento (CREATIVO)
2. assicurati che il loro comando fosse "gm"

Abbiamo eseguito questi controlli con: args.length == 1 && label.equalsIgnoreCase ("gm")

Ora sappiamo per certo che il giocatore ha digitato "/ gm x".

La prossima cosa che dobbiamo fare è trasformare args [0] in un oggetto GameMode in modo che possiamo applicarlo al giocatore. Questo può essere fatto con GameMode.valueOf (String) Tuttavia, secondo la documentazione di enumerazione Java, se una stringa viene passata in valueOf () che non corrisponde ad un'enumerazione, genererà un IllegalArgumentException - quindi dobbiamo assicurarci di prendilo.

Una volta che abbiamo il gamemode, possiamo andare avanti e usare semplicemente p.setGameMode (gm) e la gamemode del giocatore cambierà. Nel caso in cui abbiamo rilevato un'eccezione, semplicemente stampiamo una dichiarazione e restituiamo false.

```
@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 non nella classe principale

Se hai molti comandi, non dovresti metterli tutti nella classe principale.

1. Crea una nuova classe e implementa `CommandExecutor`
2. Aggiungi il seguente alla classe:

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

 }
```

3. Nella tua classe principale aggiungi onEnable (sostituisci commandName con il nome del comando e CommandExecutor con il nome della classe):

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

Leggi comandi online: https://riptutorial.com/it/bukkit/topic/6880/comandi

Capitolo 4: Entità

Examples

Teletrasporto di un'entità in un'altra entità

```
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
}
```

Puoi anche aggiungere una causa al tuo teleporto, in modo da poter personalizzare il modo in cui la causa verrà trattata dal tuo plug-in o da altri:

```
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
}
```

Teletrasporto di un'entità in una posizione

```
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
}
```

Puoi anche aggiungere una causa al tuo teleporto, in modo da poter personalizzare il modo in cui la causa verrà trattata dal tuo plug-in o da altri:

```
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 di entità

L'enum EntityType rappresenta tutte le entità di Bukkit / Spigot.

Tutti i suoi valori possono essere trovati sotto.

Valore	Descrizione
AREA_EFFECT_CLOUD	N / A
ARMOR_STAND	Entità meccanica con un inventario per l'inserimento di armi / armature.
FRECCIA	Un proiettile a freccia; potrebbe rimanere incastrato nel terreno.
BAT	N / A
FIAMMATA	N / A
BARCA	Un posto in barca
CAVE_SPIDER	N / A
POLLO	N / A
COMPLEX_PART	N / A
MUCCA	N / A
CREEPER	N / A
DRAGON_FIREBALL	Come FIREBALL, ma con effetti extra
DROPPED_ITEM	Un oggetto appoggiato a terra.
UOVO	Un uovo di gallina volante.
ENDER_CRYSTAL	N / A
ENDER_DRAGON	N / A

Valore	Descrizione
ENDER_PEARL	Una perla volante.
ENDER_SIGNAL	Un segnale d'occhio finale.
Enderman	N / A
ENDERMITE	N / A
EXPERIENCE_ORB	Un'esperienza orb.
FALLING_BLOCK	Un blocco che sta per o sta per cadere.
BOLIDE	Una grande palla di fuoco volante, come lanciata da un Ghast per esempio.
ARTIFICIO	Rappresentazione interna di un fuoco d'artificio una volta che è stato lanciato.
AMO	Una lenza e un galleggiante.
Ghast	N / A
GIGANTE	N / A
CUSTODE	N / A
CAVALLO	N / A
GOLEM DI FERRO	N / A
ITEM_FRAME	Una cornice di un oggetto su un muro.
LEASH_HITCH	Un guinzaglio attaccato ad un recinto.
LIGHTNING	Un fulmine.
LINGERING_POTION	Una pozione volante persistente
CUBO DI MAGMA	N / A
Minecart	N / A
MINECART_CHEST	N / A
MINECART_COMMAND	N / A
MINECART_FURNACE	N / A
MINECART_HOPPER	N / A

Valore	Descrizione
MINECART_MOB_SPAWNER	N / A
MINECART_TNT	N / A
MUSHROOM_COW	N / A
OCELOT	N / A
PITTURA	Un dipinto su un muro.
MAIALE	N / A
PIG_ZOMBIE	N / A
GIOCATORE	N / A
ORSO POLARE	N / A
PRIMED_TNT	TNT innescato che sta per esplodere.
CONIGLIO	N / A
PECORA	N / A
SHULKER	N / A
SHULKER_BULLET	Proiettile sparato da SHULKER.
pesciolino d'argento	N / A
SCHELETRO	N / A
SLIME	N / A
SMALL_FIREBALL	Una piccola palla di fuoco volante, come lanciata da un Blaze o un giocatore.
PALLA DI NEVE	Una palla di neve volante.
PUPAZZO DI NEVE	N / A
SPECTRAL_ARROW	Come TIPPED_ARROW ma causa l'effetto PotionEffectType.GLOWING su tutti i membri del team.
RAGNO	N / A
SPLASH_POTION	Una pozione splash splash
CALAMARO	N / A

Valore	Descrizione
THROWN_EXP_BOTTLE	Una bottiglia di esperienza volante.
TIPPED_ARROW	Come ARROW ma con una pozione specifica che viene applicata al contatto.
SCONOSCIUTO	Un'entità sconosciuta senza una classe di entità
CONTADINO	N / A
TEMPO METEREOLOGICO	N / A
STREGA	N / A
APPASSIRE	N / A
WITHER_SKULL	Un proiettile di teschio volante wither.
LUPO	N / A
ZOMBIE	N / A

Passeggeri

Le entità possono avere passeggeri. Un buon esempio di passeggero è un giocatore che cavalca un maiale sellato o uno zombi all'interno di un carretto.

Sebbene esistano veicoli specifici, qualsiasi entità può essere un veicolo per qualsiasi altra entità con il metodo SetPassenger.

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

Il passeggero dovrebbe ora essere collegato al veicolo

Puoi verificare se un'entità ha un passeggero che usa

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

Se l'entità ha un passeggero, puoi recuperare l'entità passeggero con

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

Restituirà solo il passeggero principale se il veicolo può avere multipli.

Infine, puoi espellere il passeggero di un'entità con

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

Entità vicine

Per recuperare un elenco di entità vicine di un'entità, si può usare

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

Bukkit calcolerà quindi una casella di delimitazione centrata attorno all'entità, avente come parametri:

- x: 1/2 la dimensione della scatola lungo l'asse x
- y: 1/2 della dimensione della scatola lungo l'asse y
- z: 1/2 della dimensione della scatola lungo l'asse z

L'elenco potrebbe essere vuoto, il che significa che non ci sono entità vicine con i parametri.

Questo approccio può essere utilizzato per rilevare entità vicino a proiettili personalizzati, ad esempio lanciando un Itemstack e rilevando quando si scontra con un giocatore

Leggi Entità online: https://riptutorial.com/it/bukkit/topic/7886/entita

Capitolo 5: Eventi del giocatore

introduzione

Questa è una lista di eventi del giocatore e un esempio su come usarli.

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

Dettagli dei negozi di eventi per i giocatori che tentano di accedere

```
@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:

- AMMESSO - Il giocatore è autorizzato ad accedere
- KICK_BANNED - Al giocatore non è permesso accedere, a causa del fatto che sono vietati
- KICK_FULL - Il giocatore non è autorizzato ad accedere, a causa del fatto che il server è pieno
- KICK_OTHER - Al giocatore non è permesso accedere, per ragioni non definite
- KICK_WHITELIST - Al giocatore non è permesso accedere, poiché non sono nella lista bianca

Eventi del giocatore

Evento sparato quando il giocatore entra in un letto: **PlayerBedEnterEvent**

PlayerBedEnterEvent (Player who, Block bed)

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

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

Evento sparato quando il giocatore lascia un letto: **PlayerBedLeaveEvent**

PlayerBedLeaveEvent (Player who, Block bed)

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

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

Leggi Eventi del giocatore online: https://riptutorial.com/it/bukkit/topic/8905/eventi-del-giocatore

Capitolo 6: Eventi di entità

introduzione

Tutti gli eventi entità estende EntityEvent, la superclasse di EntityEvents.

Tutti gli EntityEvents noti sono disponibili di seguito e saranno trattati in questa documentazione.

Examples

Evento EntityDamage

L'evento EntityDamage viene generato quando un'entità è danneggiata.

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 maggior parte delle volte, EntityDamageEvent non verrà utilizzato. Verrà utilizzata una sottoclasse, ad esempio **EntityDamageByEntityEvent** o **EntityDamageByBlockEvent** . Entrambi possono essere visti sotto.

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 semplice estensione a EntityDamageEvent, ma con un metodo diverso:

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

EntityEvent (Superclasse)

Le sottoclasse conosciute per Eventi entità sono:

Sotto-Classi	Sotto-Classi	Sotto-Classi
CreatureSpawnEvent	CreeperPowerEvent	EntityChangeBlockEvent
EntityCombustEvent	EntityCreatePortalEvent	EntityDamageEvent
EntityDeathEvent	EntityExplodeEvent	EntityInteractEvent
EntityPortalEnterEvent	EntityRegainHealthEvent	EntityShootBowEvent
EntityTameEvent	EntityTargetEvent	EntityTeleportEvent
EntityUnleashEvent	ExplosionPrimeEvent	FoodLevelChangeEvent
HorseJumpEvent	ItemDespawnEvent	ItemSpawnEvent
PigZapEvent	ProjectileHitEvent	ProjectileLaunchEvent
SheepDyeWoolEvent	SheepRegrowWoolEvent	SlimeSplitEvent

Oltre a questo, tutte le sottoclassi ereditano i seguenti metodi:

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

Leggi Eventi di entità online: https://riptutorial.com/it/bukkit/topic/6486/eventi-di-entita

Capitolo 7: File di configurazione

Sintassi

- `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");`

Osservazioni

I file di configurazione di Bukkit sono semplici file YAML (Yet Another Markup Language) e sono implementati in questo modo.

Examples

Plugin Config.yml

Puoi avere un file config.yml che carica direttamente dal tuo file jar. Deve essere aggiunto alla cartella del tuo progetto, allo stesso modo del file plugin.yml.

In questo file hai i valori predefiniti per la tua configurazione.

Esempio di configurazione:

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

Il file di configurazione di esempio deve essere aggiunto alla cartella del progetto.

Per caricare il file di configurazione predefinito nella cartella del plugin, è necessario aggiungere il seguente codice al tuo onEnable ():

```
saveDefaultConfig();
```

Questo renderà il tuo file config.yml dal progetto come il file di configurazione del tuo plugin e lo aggiungerà alla cartella del tuo plugin.

Da lì, puoi accedere al tuo file di configurazione da qualsiasi luogo, utilizzando l'istanza del tuo plug-in:

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

Da lì, possiamo accedere a tutto ciò che è stato impostato sulla configurazione del plugin.

Nota: il file di configurazione predefinito potrebbe avere i suoi valori modificati, se l'utente desidera modificare il file config.yml generato nella cartella.

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

Sezione Percorsi multipli

Ciò che può accadere nel tuo file di configurazione è avere un percorso verso una variabile che attraversa più sezioni.

Esempio di configurazione

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

Il nome "Kerooker" proviene dalla sezione "first-tier", che proviene dalla sezione "admins". Per accedere ai percorsi interni del nostro file, usiamo un semplice '.' come un modo per dire che vogliamo la prossima sezione. Quindi, per noi di accedere a "Kerooker", andiamo:

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

Leggi File di configurazione online: https://riptutorial.com/it/bukkit/topic/6824/file-di-configurazione

Capitolo 8: Generazione del mondo

Examples

Void Generator

La classe del generatore di vuoto:

```
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
    }

}
```

Qualsiasi classe in cui desideri generare un nuovo mondo:

```
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();
    }
}
```

Leggi Generazione del mondo online: https://riptutorial.com/it/bukkit/topic/6652/generazione-del-mondo

Capitolo 9: Gestione degli eventi

introduzione

Quando qualcosa accade all'interno di Bukkit, viene chiamato un evento in modo che ogni plugin possa decidere cosa fare ogni volta che succede qualcosa.

Un evento viene chiamato quando un giocatore tenta di giocare un blocco, quando un'entità si disfa, quando qualcuno effettua l'accesso ... I plugin possono ascoltare eventi specifici e affrontarli in molti modi diversi, ad esempio inviando un messaggio a un amministratore quando un giocatore accede tramite PlayerLoginEvent.

Sintassi

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

Osservazioni

Quando registri un evento, dai un'occhiata se non lo stai registrando due volte! Oppure il tuo plugin agirà due volte per l'evento registrato.

Dai un'occhiata in più a come gestire eventi specifici:

- Eventi del giocatore
- Eventi di entità

Examples

Registrare eventi all'interno della classe Listener

```
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);
    }
}
```

Registrazione degli eventi nella tua classe principale

```
public class Main extends JavaPlugin {

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

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

Ascoltando gli eventi

Bukkit utilizza un sistema basato su eventi che consente agli sviluppatori di plugin di interagire e modificare il server e le azioni specifiche che si verificano nel mondo.

Creazione di un gestore di eventi

I gestori di eventi sono metodi che vengono richiamati quando si verifica il loro evento. Sono generalmente pubblici e non validi così come sono denominati `on{EventNameStem}` per convenzione. Tuttavia, tutti i gestori devono avere l'annotazione `@EventHandler` e contenere il suo evento come

parametro SOLO. Ecco un esempio di gestore di eventi per `PlayerJoinEvent`

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

Nota: il formato di denominazione per gli eventi Bukkit è l'evento `{Source}{Action}({Target})Event` . Alcuni esempi di questi nomi degli eventi sono: `PlayerInteractEvent` o `BlockBreakEvent` o `PlayerJoinEvent` . Un elenco di tutti gli eventi può essere trovato sul Spigot Javadocs

Registrazione degli eventi

La semplice creazione di un gestore di eventi non è sufficiente per consentire a Bukkit di iniziare a inviare chiamate di eventi al proprio metodo. È inoltre necessario registrarlo tramite l'interfaccia PluginManager.

Il modo più comune per registrare eventi è creare una classe che implementa l'interfaccia Listener e utilizzarla per avvolgere i gestori di eventi.

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

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

}
```

Questa classe di ascoltatori e tutti i suoi eventi possono quindi essere registrati nella tua classe di plug-in principale come questa:

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

Creazione di eventi personalizzati

A volte hai bisogno di creare il tuo Evento, quello che altri plugin possono ascoltare (Vault, tra gli altri plugin, lo fa) e persino annullare. L'API Event di Bukkit consente questo di essere possibile. Tutto quello che devi fare è creare una nuova classe, farla estendere `Event` , aggiungere i gestori e gli attributi di cui ha bisogno l'evento (come Player o messaggio).

```
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;
    }
}
```

Chiamando il tuo evento personalizzato

Hai il controllo della creazione e della chiamata dei tuoi eventi, dove la chiami dipende interamente da te. Ecco un esempio

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

Ricorda: hai il controllo dei tuoi eventi. Se non lo chiami e agisci su di esso, non succede!

Ascoltare un evento personalizzato

Ascoltare un evento personalizzato è come ascoltare un evento normale.

```
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
    }
}
```

Rendi il tuo CustomEvent cancellabile

Se vuoi rendere cancellabile il tuo evento, aggiungi solo `implements Cancellable` , `boolean cancelled` e un getter and 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;
    }
}
```

In seguito, verificherai se un plug-in ha annullato l'evento personalizzato.

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

Gestione degli eventi di base

Bukkit utilizza un sistema basato su eventi che consente agli sviluppatori di plugin di interagire e modificare il server e le azioni specifiche che si verificano nel mondo.

Creazione di un gestore di eventi

I gestori di eventi sono metodi che vengono richiamati quando si verifica il loro evento. Sono generalmente pubblici e non validi così come sono denominati `on{EventNameStem}` per convenzione. Tuttavia, tutti i gestori devono avere l'annotazione `@EventHandler` e contenere il suo evento come parametro SOLO. Ecco un esempio di gestore di eventi per `PlayerJoinEvent`

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

Nota: il formato di denominazione per gli eventi Bukkit è l'evento `{Source}{Action}({Target})Event` . Alcuni esempi di questi nomi degli eventi sono: `PlayerInteractEvent` o `BlockBreakEvent` o `PlayerJoinEvent` . Un elenco di tutti gli eventi può essere trovato sul Spigot Javadocs

Registrazione degli eventi

La semplice creazione di un gestore di eventi non è sufficiente per consentire a Bukkit di iniziare a inviare chiamate di eventi al proprio metodo. È inoltre necessario registrarlo tramite l'interfaccia PluginManager.

Il modo più comune per registrare eventi è creare una classe che implementa l'interfaccia Listener e utilizzarla per avvolgere i gestori di eventi.

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

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

}
```

Questa classe di ascoltatori e tutti i suoi eventi possono quindi essere registrati nella tua classe di plug-in principale come questa:

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

Priorità dell'evento

Bukkit ha un sistema chiamato **Event Priorities** per aiutare i plugin a gestire gli eventi nella

versione precedente corretta. Le sette priorità sono (in vecchio dal primo eseguito all'ultimo):

- minore
- Basso
- Normale (predefinito)
- alto
- Più alta
- Tenere sotto controllo

Se stai pianificando di cancellare molti eventi (es. Plugin di protezione) sarebbe una buona idea usare una priorità più bassa (o più bassa) per evitare problemi.

Non si dovrebbe mai modificare l'esito di un evento in 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
}
```

Ulteriori informazioni:

- EventPriority at spigot javadocs
- Priorità dell'evento su BukkitWiki

Leggi Gestione degli eventi online: https://riptutorial.com/it/bukkit/topic/5743/gestione-degli-eventi

Capitolo 10: Manipolare i risultati

Sintassi

- player.awardAchievement (Achievement ach);

Examples

Assegnazione di obiettivi

```
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
```

Controllare se il giocatore ha successo:

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

Il codice assegnerà il risultato e qualsiasi risultato genitore.

I risultati possibili sono:

- ACQUIRE_IRON (acquisizione hardware)
- BAKE_CAKE (La bugia)
- LIBRERIA (Bibliotecario)
- BREED_COW (Ripopolamento)
- BREW_POTION (birreria locale)
- BUILD_BETTER_PICKAXE (Ottenere un aggiornamento)
- BUILD_FURNACE (Hot Topic)
- BUILD_HOE (Time to Farm!)
- BUILD_PICKAXE (Time to Mine!)
- BUILD_SWORD (Time to Strike!)
- BUILD_WORKBENCH (Benchmarking)
- COOK_FISH (Delicious Fish)
- DIAMONDS_TO_YOU (Diamanti a te!)
- INCANTESIMI (Enchanter)
- END_PORTAL (The End?)
- EXPLORE_ALL_BIOMES (Adventuring Time)
- FLY_PIG (Quando i maiali volano)
- FULL_BEACON (Beaconator)
- GET_BLAZE_ROD (Into Fire)
- GET_DIAMONDS (DIAMONDS!)

- GHAST_RETURN (Torna al mittente)
- KILL_COW (Cow Tipper)
- KILL_ENEMY (Monster Hunter)
- KILL_WITHER (The Beginning.)
- MAKE_BREAD (cuoce il pane)
- MINE_WOOD (Ottenere legna)
- NETHER_PORTAL (Dobbiamo andare più in profondità)
- ON_A_RAIL (Su una ferrovia)
- OPEN_INVENTORY (Taking Inventory)
- OVERKILL (Overkill)
- OVERPOWERED (Overpowered)
- SNIPE_SKELETON (Sniper Duel)
- SPAWN_WITHER (L'inizio?)
- THE_END (The End.)

Riferimento

Leggi Manipolare i risultati online: https://riptutorial.com/it/bukkit/topic/7690/manipolare-i-risultati

Capitolo 11: Manipolazione del mondo

Osservazioni

Fare riferimento a World Generation per argomenti di generazione mondiale

Examples

Creazione di esplosioni

Per creare un'esplosione, è possibile utilizzare le seguenti firme del metodo:

```
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 e loc rappresentano la posizione in cui si desidera che l'esplosione avvenga.
- il potere rappresenta la potenza della tua esplosione, il potere TnT è 4F.
- setFire rappresenta la capacità dell'esplosione di dare fuoco ai blocchi
- breakBlocks rappresenta la capacità dell'esplosione di distruggere blocchi attorno ad esso.
- tutti i metodi restituiscono true se l'esplosione è avvenuta e restituiscono false se un plug-in annulla l'evento di esplosione.

Simulazione di un'esplosione TnT che interrompe i blocchi e dà fuoco a x = 0, y = 0 e z = 0

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

Eliminazione di un articolo

I seguenti metodi possono essere usati per rilasciare un oggetto da qualche parte nel mondo:

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

`dropItem` significa far cadere un oggetto esattamente nella posizione, restituendo un oggetto Item.

`dropItemNaturally` significa lasciar cadere l'oggetto nella posizione, ma con un offset casuale, il che significa che non sarà esattamente nella posizione, ma molto vicino nelle vicinanze. Questo è fatto per simulare un oggetto che viene rilasciato da un'entità o un blocco come un Dispenser.

Generazione di un albero

I seguenti metodi possono essere usati per generare un albero naturalmente (come se fosse cresciuto da un alberello) nel mondo.

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

- La posizione è dove si desidera generare l'albero
- TreeType è il tipo di albero che si desidera generare e può essere uno dei seguenti

TreeType enum

genere	Descrizione
ACACIA	Albero di acacia
GRANDE ALBERO	Albero regolare, extra alto con rami
BETULLA	betulla
BROWN_MUSHROOM	Grande fungo marrone; alto e simile ad un ombrello
CHORUS_PLANT	Grande pianta originaria di The End
COCOA_TREE	Albero della giungla con piante di cacao; 1 blocco di larghezza
DARK_OAK	Albero di quercia scura
GIUNGLA	Albero della giungla standard; 4 blocchi di larghezza e altezza
JUNGLE_BUSH	Piccolo cespuglio che cresce nella giungla
MEGA_REDWOOD	Mega sequoia; 4 blocchi di larghezza e altezza
RED_MUSHROOM	Grande fungo rosso; basso e grasso
REDWOOD	Albero di sequoia, a forma di pino
SMALL_JUNGLE	Più piccolo albero della giungla; 1 blocco di larghezza
PALUDE	Palude (regolare con le viti sul lato)
TALL_BIRCH	Albero di betulla alto
TALL_REDWOD	Albero di sequoia alto con poche foglie in cima
ALBERO	Albero normale, senza rami

- delegato può essere utilizzato se si desidera che una classe invochi il blocco di ogni blocco come risultato di questo metodo

Entrambe le firme restituiranno true se l'albero è stato generato correttamente, altrimenti falso.

Regole di deposizione delle uova

Ci sono alcune regole di generazione in Worlds in Bukkit. Loro sono:

- Spawning animale
- Creazione delle uova
- Quantità di quanto sopra che può essere generato

Spawning animale

La deposizione degli animali può essere suddivisa nelle seguenti categorie:

- Animali acquatici
- Animali terrestri

Per ottenere la quantità di animali che possono essere generati all'interno del mondo in fase di runtime, è possibile utilizzare il metodo

```
int getAnimalSpawnLimit()
```

Per animali terrestri e

```
int getWaterAnimalSpawnLimit();
```

Per gli animali acquatici.

Entrambi i limiti possono essere impostati con i metodi

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

Nota: se impostato su numeri inferiori a 0, verrà utilizzato l'importo predefinito del mondo.

Minecraft fa un tentativo di generare animali ogni 400 zecche (predefinito). Questo può essere cambiato se lo desideri, usando le seguenti firme:

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

- Un valore di 1 significherà che il server tenterà di generare gli animali in questo mondo ogni tick.
- Un valore di 400 significherà che il server tenterà di generare gli animali in questo mondo ogni 400 ° tick.
- Un valore inferiore a 0 verrà reimpostato sul valore predefinito di Minecraft.

Nota : se impostato su 0, la riproduzione degli animali sarà disabilitata per questo mondo. Si consiglia di utilizzare setSpawnFlags (boolean, boolean) per controllare questo invece.

Leggi Manipolazione del mondo online: https://riptutorial.com/it/bukkit/topic/7926/manipolazione-del-mondo

Capitolo 12: Nascondere i giocatori

Sintassi

- void hide (Player toHide);
- show vuoto (Player to Show);
- canSee booleano (Player toBeSeen);

Osservazioni

Gli eventi sono trattati meglio nella documentazione Elenco degli eventi di StackOverflow

Examples

Nascondere un giocatore da altri giocatori

```
Player playerToHide;
Player playerToNotSee;


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

Se il giocatore è già nascosto, non succede nulla

Mostrare un giocatore a un altro giocatore

```
Player toUnhide;
Player toSeeAgain

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

Se il giocatore è già visibile, non succede nulla.

Verifica se il giocatore può essere visto

```
Player playerToCheck;
Player playerSeeing;

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

Nascondere il giocatore da un'entità

Questo può essere fatto usando l'evento EntityTargetEvent

Le entità non bersagliano il giocatore se annulli l'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
        }
    }
}
```

Leggi Nascondere i giocatori online: https://riptutorial.com/it/bukkit/topic/7697/nascondere-i-giocatori

Capitolo 13: NMS

introduzione

NMS, noto anche come **N** et. **M** inecraft. **S** erver è il pacchetto che contiene il codice del server Minecraft di base. Le classi in questo pacchetto sono state fatte da Mojang (non Bukkit) e sono quindi per lo più offuscate e non destinate a essere utilizzate o modificate. Tuttavia, l'interazione con il codice del server Minecraft a questo livello ti consente di modificarne quasi ogni aspetto. Questo è significativo perché ci sono numerose modifiche che Bukkit non supporta.

Osservazioni

L'API Bukkit è un wrapper o un livello di astrazione per NMS che consente agli sviluppatori di plug-in di interagire con il server senza doversi preoccupare delle modifiche apportate alla base di codice interna.

L'uso del codice NMS è sconsigliato poiché si interrompe spesso tra le modifiche alle versioni di Minecraft e non può essere supportato da Bukkit o Spigot poiché non lo creano, non lo possiedono o lo mantengono.

Examples

Accesso alla versione corrente di Minecraft

Una delle parti più critiche nell'affrontare il codice NMS è essere in grado di supportare versioni multiple di Minecraft. Esistono numerosi modi per farlo, ma una soluzione semplice consiste nell'utilizzare questo codice per memorizzare la versione come campo statico pubblico:

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

Questo frammento di codice funziona prendendo la classe CraftServer:

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

Ottenere il suo pacchetto:

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

E prendendo la sottostringa del nome del pacchetto a partire dall'indice 23 che sarà sempre dopo 'org.bukkit.craftbukkit.' (che ha una lunghezza di 23 caratteri). Risultante nella stringa VERSION finale:

```
VERSION
```

Ci sono una serie di motivi per cui è così importante poter accedere alla versione corrente di Minecraft. Principalmente perché qualsiasi accesso a una classe su un server che esegue una

versione di Minecraft diversa da quella con cui il plugin stava codificando genererà un errore.

Ecco un esempio che dimostra come risolvere il problema utilizzando il campo `NMS_VERSION` per recuperare un'istanza di CraftPlayer (che è una classe NMS) su qualsiasi versione di 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);
    }
}
```

L'oggetto risultante può quindi essere manipolato utilizzando la reflection per eseguire attività basate su NMS senza preoccuparsi di provare ad accedere alla versione errata della classe.

Anche questo metodo non è infallibile, tuttavia, poiché i nomi dei campi e dei metodi NMS cambiano facilmente, quindi l'unica cosa che garantisci è che il tuo codice non si rompe ogni volta che si aggiorna Minecraft.

Ottenere un ping del giocatore

Una cosa molto semplice che potresti voler fare con NMS che Bukkit non supporta è ottenere il ping del giocatore. Questo può essere fatto in questo modo:

```
/**
 * 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;
}
```

Se stai usando un metodo come getCraftPlayer (Player) che restituisce un'istanza dell'istanza di CraftPlayer corrispondente del Player come Object. È possibile accedere ai dati senza importare le classi dipendenti dalla versione utilizzando la riflessione come questa:

```
/**
 * 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);
    }
}
```

Leggi NMS online: https://riptutorial.com/it/bukkit/topic/9576/nms

Capitolo 14: Programmazione dello Scheduler

Sintassi

- `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);`

Osservazioni

Alcuni metodi API Bukkit sono thread-safe e possono essere chiamati in modo asincrono. Per questo motivo, metodi API Bukkit devono *solo,* con poche eccezioni, essere eseguito sul thread principale.

Il codice eseguito all'interno dei metodi `scheduleSync` e il metodo `runTask` verranno eseguiti sul thread principale.

Il codice eseguito all'interno di `runTaskAsynchronously` verrà eseguito in modo asincrono dal thread principale. I metodi asincroni sono molto utili per eseguire operazioni matematiche o di database di grandi dimensioni senza rallentare il server, tuttavia causeranno un comportamento indefinito se utilizzati per chiamare i metodi API Bukkit. Per questo motivo, i metodi API Bukkit che devono essere eseguiti dopo il codice asincrono devono sempre essere inseriti in un metodo `runTask` .

Examples

Scheduler Attività ricorrente

Il tempo per le attività dell'Utilità di pianificazione viene misurato in zecche. In condizioni normali, ci sono 20 tick al secondo.

Le attività pianificate con `.scheduleSyncRepeatingTask` verranno eseguite nella discussione principale

```
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
```

Pianificazione attività ritardata

Il tempo per le attività dell'Utilità di pianificazione viene misurato in zecche. In condizioni normali, ci sono 20 tick al secondo.

Le attività pianificate con `.scheduleSyncDelayedTask` verranno eseguite nella discussione principale

```
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
```

Esecuzione di attività in modo asincrono

È possibile eseguire il codice in modo asincrono dal thread principale utilizzando `runTaskAsynchronously` . Ciò è utile per eseguire operazioni matematiche o di database intensive, in quanto impediscono il blocco del thread principale (e il server in ritardo).

Pochi metodi API Bukkit sono sicuri per i thread, quindi molti causeranno un comportamento indefinito se chiamati in modo asincrono dal thread principale.

```
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
    }
});
```

Esecuzione di attività sul thread principale

È inoltre possibile eseguire il codice in modo sincrono con il thread principale utilizzando `runTask` . Ciò è utile quando si desidera chiamare i metodi API Bukkit dopo aver eseguito il codice in modo asincrono dal thread principale.

Il codice richiamato all'interno di questo Runnable verrà eseguito sul thread principale, rendendo più sicuro chiamare i metodi API 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
    }
});
```

Esecuzione di un BukkitRunnable

BukkitRunnable è un Runnable trovato in Bukkit. È possibile pianificare un'attività direttamente da

un BukkitRunnable e anche cancellarla dal suo interno.

Importante: il tempo delle attività è misurato in zecche. Un secondo ha 20 tick.

Non-RepeatingTask:

```
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.
```

Ripetizione attività:

```
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.
```

Esecuzione del codice sicuro del thread da un'attività asincrona

A volte è necessario eseguire il codice sincrono da un'attività asincrona. Per fare ciò, è sufficiente pianificare un'attività sincrona all'interno del blocco asincrono.

```
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);
```

Leggi Programmazione dello Scheduler online:

https://riptutorial.com/it/bukkit/topic/5436/programmazione-dello-scheduler

Capitolo 15: Registrazione

Examples

Utilizzo di 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()});
    }

}
```

Livelli di registrazione

Java Logging Api ha 7 livelli . I livelli in ordine decrescente sono:

- `SEVERE` (valore più alto)
- `WARNING`
- `INFO`
- `CONFIG`
- `FINE`
- `FINER`
- `FINEST` (valore più basso)

Il livello predefinito è `INFO` (ma dipende dal sistema e utilizza una macchina virtuale).

Nota : ci sono anche dei livelli `OFF` (può essere usato per disattivare la registrazione) e `ALL` (la prospettiva di `OFF`).

Esempio di codice per questo:

```
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");
    }
}
```

Per impostazione predefinita, in esecuzione questa classe verranno visualizzati solo i messaggi con livello superiore a `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
```

Leggi Registrazione online: https://riptutorial.com/it/bukkit/topic/7202/registrazione

Capitolo 16: Scala

introduzione

Come implementare i plugin Bukkit nel linguaggio di programmazione Scala

Examples

Impostazione del progetto (Scala Eclipse)

Creare un progetto in scala è molto simile a crearne uno in java. Ecco come dovrebbe apparire la classe di iscrizione:

```
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
  }

}
```

Innanzitutto, assicurati di aver installato l'ultima versione di Scala che si trova qui: https://www.scala-lang.org/download/

Successivamente, ti consigliamo di scaricare Scala Eclipse, disponibile qui: http://scala-ide.org/ ed estrarre il download in una cartella a tua scelta.

Una volta installati entrambi, apri semplicemente Scala Eclipse.

Infine, per far funzionare il tuo plugin - devi avere una sorta di plugin di runtime per caricare la libreria di scala per te, io uso questo: https://dev.bukkit.org/projects/scala-loader (inserisci questo jar nella tua cartella dei plugin come qualsiasi altro plugin)

Da qui in avanti, il processo è quasi identico a java:

1. Premi `Alt+Shift+N` -> fai clic su `Scala Project`
2. Fare clic con il tasto destro del mouse sul progetto - fare clic su `Properties`
3. Fare clic su `Java Build Path` , quindi fare clic sulla scheda `Libraries`
4. Fai clic su `Add External Jars` e seleziona il tuo file jar spigot-api
5. Fare clic su `Apply` e quindi `OK`

Per la configurazione del progetto, ti consigliamo di creare un pacchetto così:

Fare clic con il tasto destro del mouse su progetto -> `New` -> `Package`

`com.yourdomain.pluginname` come preferisci, in genere: `com.yourdomain.pluginname`

All'interno di questo pacchetto, crea una classe Scala e `PluginName` come preferisci, in genere: `PluginName`

Fai in modo che la classe `extends JavaPlugin` e sovrascriva le funzioni fornite per una configurazione di base come mostrato sopra.

Infine, fai clic con il pulsante destro sulla cartella denominata "src" e seleziona Nuovo file. Dai un nome al file plugin.yml (NON il nome del tuo plugin, ma esplicitamente plugin.yml) e aprilo.

Un'implementazione di base dovrebbe assomigliare a questa:

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

E il gioco è fatto! Dopo aver finito di scrivere il tuo plug-in, fai clic su `File` -> `Export` -> `Java` -> `Jar file` -> Seleziona il tuo progetto e specifica il plug-in del server come destinazione -> fai clic su `Finish`

In genere puoi semplicemente ricaricare il tuo server per vedere le modifiche dopo l'esportazione, tuttavia **alcuni plug-in si interromperanno di nuovo, quindi fai attenzione!** Consiglio sempre di **riavviare** il server a meno che non si sappia che ricaricare non interromperà altri plug-in.

Leggi Scala online: https://riptutorial.com/it/bukkit/topic/9259/scala

Capitolo 17: Uova di Spawn

Osservazioni

Fare riferimento a Documentazione delle entità per comprendere meglio EntityType

Examples

Creare una ItemStack di SpawnEgg

Per qualsiasi cosa inferiore a 1.9

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

Per 1.9 e sopra

Nelle versioni 1.9 e successive, Spigot non ha un'implementazione per creare uova di spawn senza usare NMS. Per fare ciò, puoi usare una piccola classe personalizzata / wrapper per farlo accadere:

```
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);
}
```

Leggi Uova di Spawn online: https://riptutorial.com/it/bukkit/topic/7737/uova-di-spawn

Capitolo 18: versioni

Examples

Ottenere la versione su runtime

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

Accesso a NMS attraverso versioni di implementazione

```
// 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);
}
```

Leggi versioni online: https://riptutorial.com/it/bukkit/topic/7730/versioni

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con bukkit	Community, Drayke, ItzPam, Jojodmo, Keenan Thompson, Kerooker, kmecpp, Martin W, RamenChef, Tassu, Unihedron
2	Caduta	Kerooker
3	comandi	annon, Kerooker, Martin W
4	Entità	Kerooker
5	Eventi del giocatore	Ferrybig, Kerooker, Tassu
6	Eventi di entità	Alw7SHxD, Ferrybig, Kerooker, kmecpp
7	File di configurazione	Kerooker
8	Generazione del mondo	Drayke
9	Gestione degli eventi	Drayke, Jarrod Dixon, Kerooker, kmecpp, Martin W, Tassu
10	Manipolare i risultati	Kerooker
11	Manipolazione del mondo	Kerooker
12	Nascondere i giocatori	Kerooker
13	NMS	Alw7SHxD, kmecpp
14	Programmazione dello Scheduler	Ferrybig, Jojodmo, Kerooker, kmecpp, Pokechu22
15	Registrazione	Kerooker
16	Scala	annon
17	Uova di Spawn	Infuzed guy, Kerooker, RamenChef
18	versioni	caseif, Kerooker