EBook Gratis

APRENDIZAJE minecraft

Free unaffiliated eBook created from
Stack Overflow contributors.

#minecraft

Tabla de contenido

Acerca de

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

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

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

Capítulo 1: Empezando con Minecraft

Observaciones

Esta sección proporciona una descripción general de qué es Minecraft y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema importante dentro de Minecraft y vincular a los temas relacionados. Dado que la Documentación para minecraft es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Versiones

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

Examples

Instalando Minecraft

1. Compra Minecraft desde aquí
2. Crea una cuenta Mojang o inicia sesión si ya tienes una.
3. Si ha creado una nueva cuenta, verifique su correo electrónico.
4. Rellene sus datos de pago. Asegúrate de que estás en minecraft.net y estás en una conexión segura (HTTPS)

5. Descargar y ejecutar Minecraft
6. Abre el lanzador de Minecraft.
7. Inicie sesión con su dirección de correo electrónico si compró Minecraft después de noviembre de 2012, o si migró su nombre de usuario a una cuenta de Mojang.

   Inicie sesión con su nombre de usuario si tiene una cuenta de Minecraft más antigua y aún no ha migrado al nuevo formato de cuenta. Puede migrar a una cuenta Mojang desde la página de cuentas Mojang.
8. Haga clic en "Jugar" si desea jugar en la versión más reciente.
9. Disfruta el juego.

Lea Empezando con Minecraft en línea:
https://riptutorial.com/es/minecraft/topic/7952/empezando-con-minecraft

Capítulo 2: Comandos de Complementos

Examples

Hola Comando

En el código a continuación puede ver cómo agregar un comando a su complemento.

MainClass.java

```
package yourpackage;

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

public class MainClass extends JavaPlugin {

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

Plugin.yml

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

Lea Comandos de Complementos en línea:
https://riptutorial.com/es/minecraft/topic/9708/comandos-de-complementos

Capítulo 3: Creando el primer plugin Spigot

Examples

Primer Plugin en Eclipse

Requisito previo

Esta guía asume que ya ha utilizado BuildTools y ha ejecutado el servidor Spigot al menos una vez. También supone que tienes el archivo jar de Spigot-API que usaremos.

1) Iniciar Eclipse ; puede cambiar la ubicación del espacio de trabajo si lo desea.

2) Crear un nuevo proyecto

1. Establece el nombre del proyecto a lo que desees. Aquí, elegimos MyFirstPlugin.
2. Haga clic en Siguiente.
3. Seleccione Agregar JAR externos en la pestaña Bibliotecas. En el cuadro de diálogo Selección de JAR, seleccione el archivo jar spigot-api-shaded, que se puede encontrar en Spigot / Spigot-API / target / dentro de su carpeta de BuildTools.
4. Seleccione Finalizar

3) Añadir un nuevo paquete

Haga clic derecho en **src** y haga clic en **Nuevo> Paquete** . Puede usar cualquier convención de espacio de nombres que desee, solo sea consistente. (por ejemplo: com.google.android).

4) Crear una nueva clase

1. Haga clic con el botón derecho en el paquete recién creado y seleccione **Nuevo> Clase** .

2. Dale algún nombre; A menudo el mismo nombre que el proyecto. Dentro del editor, la clase de Java recién creada se abrirá. El código debería verse algo así:

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

5) Modificar declaración de clase.

1. Su clase debe extenderse desde JavaPlugin. Eclipse producirá un error ya que no sabe qué es JavaPlugin. Si ha importado correctamente la API de Spigot, podrá importar JavaPlugin agregando la declaración de importación. No necesita escribir manualmente esa línea, simplemente haga clic en el error y seleccione la acción apropiada. Tu código ahora debería verse como:

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

public class MyFirstPlugin extends JavaPlugin {

}
```

6) Implementar los métodos necesarios.

La clase JavaPlugin tiene algunos métodos abstractos que tu complemento debe implementar. Por lo tanto, agregue las funciones onEnable y onDisable que se activarán cuando el complemento esté deshabilitado o habilitado en la consola. Puedes dejarlos en blanco por ahora. También se requiere que escriba `@Override` sobre el método.

Nota: no es necesario que agregue un getLogger cuando su complemento esté habilitado o deshabilitado, Bukkit ya lo hace por usted.

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

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

    }
}
```

7) Crear archivo plugin.yml

Haga clic derecho en el proyecto y cree un archivo **Nuevo> Archivo** . **Nómbrelo plugin.yml** . Pega lo siguiente:

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

8) Exportación

Como no hay errores, podemos exportar este proyecto como un JAR. Haga clic derecho en el nombre del proyecto, seleccione Exportar. En el cuadro de diálogo consecuente, seleccione el archivo JAR. Haga clic en Siguiente. Puede desmarcar la ruta de clase y la inclusión del proyecto y cambiar el destino de exportación a su carpeta de complementos

9) Correr

Inicia el servidor y deberías ver que tu plugin estaba habilitado.

Lea Creando el primer plugin Spigot en línea:
https://riptutorial.com/es/minecraft/topic/9766/creando-el-primer-plugin-spigot

Capítulo 4: Creando un bloque básico con forja.

Introducción

Crear un bloque simple y decorativo con Forge es una de las primeras tareas que un aspirante a modder tendrá que aprender. La forma de hacerlo ha cambiado en las distintas versiones de Minecraft y probablemente se encuentra en una dificultad "moderada" de la versión 1.7.10 debido a la gran cantidad de errores fáciles de cometer.

Observaciones

Si algo sale mal y su bloque personalizado (ya sea cuando se coloca o mantiene) tiene una textura que falta (negro / púrpura) o un modelo (el cubo predeterminado que es demasiado grande cuando se mantiene), verifique el registro. Los problemas de este tipo casi *siempre* aparecerán en el registro, siempre que haya registrado correctamente las cosas.

Lo primero que aparecerá es la línea de `"Exception loading model for variant..."` o similar, que le indica qué bloque o elemento no se cargó correctamente. Después de una docena de líneas que comiencen con `at...` habrá otra línea comenzando `Caused by...`

Esta línea "Causada por" es la importante, le dirá qué archivo no se cargó correctamente y puede tener uno de varios errores, como:

- JSON con formato incorrecto (su archivo JSON no es válido y tiene un error tipográfico)
- Archivo no encontrado (el archivo que Minecraft está buscando no tiene el nombre correcto o no está en el lugar correcto)
- Falta la excepción de variante (su JSON Blockstate está incompleto)

¡Incluso puede obtener más de un error por un solo error! Si no sabe qué bloque es el problema, comente cada bloque y elemento que sepa que funciona, reduzca la lista de errores al bloque (o bloques) en cuestión. Tu debes hacer esto varias veces.

Además, una textura de mezcla se muestra de manera tan diferente como una simple lista de todos los recursos que faltan para un dominio determinado (ID de mod).

Examples

La clase de bloque

Primero necesitamos una clase que represente el bloque.

```
public class CustomBlock extends Block {
```

```
    public CustomBlock () {
        super(Material.ROCK);
        setHardness(1.0f);
        setHarvestLevel("pickaxe", 0);
        setResistance(1.0f);
        setCreativeTab(CreativeTabs.DECORATIONS);
        this.setSoundType(SoundType.STONE);
    }
}
```

Incluso aquí hay varias modificaciones disponibles: material (que gobierna algunas propiedades, como poder ser empujado por pistones y si se puede romper a mano), dureza (cuánto tarda en romperse), nivel de cosecha (herramienta y herramienta adecuadas) material: en este caso piqueta de madera), resistencia (frente a explosiones), la pestaña que se muestra en el menú creativo y qué sonido de paso tiene.

Aquí es donde deberá ir cualquier bloque de funcionalidad elegante, pero por ahora estamos haciendo un bloque que se ve bien, así que hemos terminado.

El modelo de bloques JSON

Luego debemos decirle a Minecraft cómo queremos que se vea nuestro bloque.

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

Eso es prácticamente todo lo que se necesita para que funcione una vez que se registra el bloqueo. Lo único importante es que el **nombre** del **archivo** coincide con el **nombre de registro** utilizado para registrar el bloque y debe estar en minúsculas (1.11+ nombres de los archivos *deben* estar en minúsculas, antes de que se acaba de mayúsculas y minúsculas).

Asigne un nombre al archivo JSON modelo `my_block.json` (que coincida con el nombre de registro que le daremos más adelante) y guárdelo en `src\main\resources\assets\example\models\block\` (donde `example` es el ID de mod especificado en el @Mod anotación de tu clase de mod principal).

El modelo de bloque aquí utiliza un padre de block / cube_all, lo que significa que la textura suministrada única se usará en todas las caras. También hay otros modelos predeterminados, como:

- Bloque / cubo (las seis caras asignadas independientemente)
- block / cube_bottom_top (caras superior e inferior independientes de los lados)
- bloque / orientable (bloque orientado direccional, por ejemplo, horno)
- bloque / cruz (flores, hierba alta)
- bloque / cosecha (trigo, zanahorias)

Tenga en cuenta que cada modelo especifica las texturas que utiliza mediante un ID de nombre (por ejemplo, `"all"` o `"top"`). Mire el modelo principal para determinar cuáles son esos nombres si

no está seguro. Texturas incorrectamente especificadas pueden llevar a problemas de textura faltantes *que* no *reportan errores* .

También es posible crear un modelo totalmente personalizado o crear un modelo primario personalizado. Pero por ahora, esto será suficiente.

No olvide crear una textura, asígnele un nombre `decorative.png` (ya que eso es lo que especificó el archivo JSON) y guárdelo en `src\main\resources\assets\example\textures\blocks\`

Bloquear registro

El registro de bloques se realiza desde su clase de mod principal, o un método de clase ModBlocks invocado desde la clase de mod principal durante preInit.

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

Hay una razón importante para usar `block.setUnlocalizedName(block.getRegistryName().toString());` ¡también! Asegura que sus nombres no localizados de bloque (y elemento) contengan el ID de mod para evitar conflictos de archivos de idioma entre los mods.

¿Oh, también quieres una versión del artículo para que pueda existir en tu inventario? Eso se crea por separado el post 1.7.10 y se hace así:

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

Observe que establecemos el nombre de registro del ItemBlock en la misma cadena que nuestro bloque. Así es como Forja y combina bloques con su contraparte de ItemBlock y viceversa.

¡Pero espera hay mas!

Su bloque puede tener un *formulario de elemento* , ¡pero ese elemento aún no tiene un modelo o textura! Los modelos se registran automáticamente para bloques, pero no artículos. Esto *solo se puede llamar desde el Proxy del cliente* y no cubre bloques con variantes (como lana u hojas).

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

En general, no necesitará un JSON de modelo de artículo, ya que Forge y vainilla volverán a caer en el modelo de bloque, sin embargo, este no es siempre el caso. Si encuentra que necesita un JSON modelo de artículo, simplemente páselo a su bloque JSON y guárdelo en `src\main\resources\assets\example\models\item\` con el mismo nombre de archivo que el nombre de registro del bloque.

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

Lea Creando un bloque básico con forja. en línea:
https://riptutorial.com/es/minecraft/topic/9748/creando-un-bloque-basico-con-forja-

Capítulo 5: Creando un item básico con forja

Introducción

Crear un elemento simple con Forge es una de las primeras tareas que un aspirante a modder tendrá que aprender. La forma de hacerlo ha cambiado en las distintas versiones de Minecraft y probablemente se encuentre en una dificultad "moderada" de la versión 1.7.10 debido a la gran cantidad de errores fáciles de cometer, especialmente al hacer que se reproduzca correctamente.

Observaciones

Si algo sale mal y su elemento personalizado tiene una textura faltante (negro / púrpura) o un modelo (el cubo predeterminado que es demasiado grande cuando se mantiene), verifique el registro. Los problemas de este tipo casi *siempre* aparecerán en el registro, siempre que haya registrado correctamente las cosas.

Lo primero que aparecerá es la línea de `"Exception loading model for variant..."` o similar, que le indica qué bloque o elemento no se cargó correctamente. Después de una docena de líneas que comiencen con `at...` habrá otra línea comenzando `Caused by...`

Esta línea "Causada por" es la importante, le dirá qué archivo no se cargó correctamente y puede tener uno de varios errores, como:

- JSON con formato incorrecto (su archivo JSON no es válido y tiene un error tipográfico)
- Archivo no encontrado (el archivo que Minecraft está buscando no tiene el nombre correcto o no está en el lugar correcto)
- Falta la excepción de variante (su JSON Blockstate está incompleto)

¡Incluso puede obtener más de un error por un solo error! Si no sabe qué elemento es el problema, comente cada bloque y elemento que sepa que funciona, reduzca la lista de errores al bloque (o bloques) en cuestión. Tu debes hacer esto varias veces.

Además, una textura de mezcla se muestra de manera tan diferente como una simple lista de todos los recursos que faltan para un dominio determinado (ID de mod).

Examples

Clase de artículo

Esta parte no ha cambiado mucho en las versiones de Minecraft, aunque ha habido algunas mutaciones en las firmas del método exacto, así como en la jerarquía de clases. Un elemento básico (uno que no tiene funcionalidad, como un palo o un lingote: así es, ¡ambos son elementos de no hacer nada!)

```
public class CustomItem extends Item {
```

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

No hay mucho espacio para la personalización en este punto, a diferencia de los bloques. Todo lo que podemos hacer es cambiar si el elemento puede recibir daño (las herramientas lo usan) y en qué pestaña de creatividad existirá. Nombre y textura que manejaremos cuando registre el elemento en GameRegistry.

Sin embargo, esto es todo lo que se necesita para sostener, transportar, soltar, fabricar, fundir y utilizar el objeto. Algunos artículos en Minecraft (como los palos) ni siquiera tienen una clase única y simplemente usan un `new Item()` . Podríamos hacer eso aquí, sin embargo, cualquier elemento con funcionalidad adicional necesitará una clase.

Modelo del artículo

Al igual que con los bloques, los artículos también necesitan modelos.

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

Eso es prácticamente todo lo que se necesita para que funcione una vez que se registra el artículo. Lo único importante es que el nombre del archivo coincida con el nombre de registro utilizado para registrar el bloque y debe estar en minúsculas (se requiere que los nombres de los archivos de 1.11+ sean minúsculas, antes de eso solo se distingue entre mayúsculas y minúsculas).

Tenga en cuenta que "layer0" es la única textura necesaria y es muy poco probable que se especifique alguna otra textura (aunque algunos elementos como pociones y armaduras de cuero tienen una "layer1"). Todos los nombres se definen por `item/builtin` (el modelo primario superior interno para los elementos) a diferencia de los bloques.

Asigne un nombre al archivo JSON modelo `my_item.json` (que coincida con el nombre de registro que le daremos más adelante) y guárdelo en `src\main\resources\assets\example\models\item\` (donde `example` es el ID de mod especificado en el @Mod anotación de tu clase de mod principal).

Además, cree una textura para su elemento, `basic.png` nombre `basic.png` y `basic.png` en `src\main\resources\assets\example\textures\items\`

El modelo del elemento aquí utiliza un elemento primario del elemento / generado, lo que significa que se usará la textura suministrada única (como con la mayoría de los elementos no bloqueados) y se mantendrá en la mano del jugador en una orientación predeterminada. También hay un elemento / dispositivo portátil que especifica diferentes orientaciones de visualización

(para herramientas). Los artículos también pueden proporcionar su propio atributo de "visualización", anulando los del padre, pero no es necesario en el 99.9% de los usos.

Registro de artículos

El registro de elementos se realiza desde su clase de mod principal, o un método de clase ModItems invocado desde la clase de mod principal durante preInit.

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

Hay una razón importante para usar `item.setUnlocalizedName(item.getRegistryName().toString());` ¡también! Asegura que el nombre no localizado de tu artículo contenga la ID del mod para evitar conflictos de archivos de idioma entre los mods.

Ahora el elemento también necesita un modelo, y aquí es donde las cosas se pusieron difíciles después de la versión 1.7.10, que solo implicaba decirle a Minecraft el nombre de la textura que se cargaría y podría especificarse en el constructor del elemento.

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

Tenga en cuenta que esta sección *debe* ubicarse solo en el lado del cliente (es decir, el proxy del cliente) ya que muchas de las clases a las que se hace referencia no existen en el servidor dedicado.

El registro de elementos con *variantes,* p. Ej. Saplings, se debe hacer de una manera diferente, utilizando `ModelLoader.setCustomModelResourceLocation(item, metadata, resourceLocation)` aunque nos permite usar un archivo Blockstate para especificar nuestras variantes (lo cual es *muy* preferido a la alternativa). Nuestro artículo no usa variantes, así que hemos terminado.

Lea Creando un item básico con forja en línea:
https://riptutorial.com/es/minecraft/topic/9850/creando-un-item-basico-con-forja

Capítulo 6: Escribiendo Mods de Minecraft

Introducción

Entendiendo escribir tus propios mods para Minecraft.

Lo primero que debe saber es que existen principalmente dos plataformas en las que se basan los mods. Forja y Bukkit.

Cada plataforma tiene sus ventajas y desventajas y generalmente no son compatibles entre sí. Los mods de Forge abarcan toda la gama y generalmente están orientados al juego. Los mods de Bukkit están completamente basados en el servidor y generalmente orientados a herramientas de administración. Los mods de Bukkit se llaman plugins.

Examples

Complementos básicos de Bukkit

- Escribiendo un comando de saludo
- Escribir un oyente de eventos

Mods básicos de forja

- Creando un bloque básico.
- Creando un artículo básico
- Escribiendo un comando de saludo
- Escribir un controlador de eventos
- Comenzando con Capacidades [1.8+]

Lea Escribiendo Mods de Minecraft en línea:
https://riptutorial.com/es/minecraft/topic/9740/escribiendo-mods-de-minecraft

Capítulo 7: Escuchas de eventos en Bukkit

Examples

Creando un detector de eventos

Para registrar sus métodos, la clase que contiene el EventHandler (s) debe implementar la interfaz de escucha.

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

public final class ListenerClass implements Listener {
}
```

Debe registrar el detector de eventos agregando la siguiente llamada a su método onEnable en la clase que extiende JavaPlugin:

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

Para escuchar cualquier evento dado en su clase de oyente, debe crear un método con la anotación @EventHandler en el método. El tipo de evento es especificado por el Tipo en el único argumento del método. El método puede ser nombrado como quieras.

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

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

Parámetros de EventHandler

La anotación `org.bukkit.event.EventHandler` acepta un par de parámetros.

prioridad : indica la prioridad de su oyente. Existen seis prioridades diferentes, en orden de ejecución: LOWEST, LOW, NORMAL [predeterminado], HIGH, HIGHEST, MONITOR. Estas constantes se refieren a la enumeración `org.bukkit.event.EventPriority` .

Si desea cambiar el resultado de un evento, elija con mucho cuidado desde el MÁS BAJO hasta el MÁS ALTO. Se sugieren complementos de protección generalizada en LOWEST, complementos más específicos en NORMAL y anulación de complementos en ALTO. Si desea actuar cuando ocurre un evento, pero no cambiar el resultado, use MONITOR.

Nota: La prioridad de MONITOR solo debe usarse para lectura. Esta prioridad es útil para el

registro de complementos para ver los resultados de un evento y la modificación de valores puede interferir con esos tipos de complementos.

ignoreCancelled : un valor booleano que indica si su oyente debe disparar si el evento se ha cancelado antes de que sea el turno del oyente para manejar el evento. Falso por defecto.

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

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

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

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());
}
```

Anular el registro de eventos u oyentes

Puede anular el registro de eventos individuales, clases de oyentes completas o todos los eventos registrados por su complemento o incluso por otros complementos.

Anular registro de evento específico

Cada clase de evento tiene el método estático getHandlerList (), llámelo y luego puede usar el método .unregister ().

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

Ahora sabes por qué necesitarás getHandlerList () en tus eventos personalizados.

Anular el registro de todos los eventos

Usando la clase HandlerList y su método estático unregisterAll (), puede cancelar fácilmente el registro de eventos de clases de escuchas o complementos.

```
HandlerList.unregisterAll(plugin);
```

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

Lea Escuchas de eventos en Bukkit en línea:
https://riptutorial.com/es/minecraft/topic/9739/escuchas-de-eventos-en-bukkit

Capítulo 8: Escuchas de eventos en Forge

Examples

Creando un detector de eventos en Forge

Crear un detector de eventos en Forge es muy similar a crear uno en Bukket.

Crear la clase de oyente requiere mucho menos. No hay interfaz para implementar u otras importaciones.

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

Para registrarlo es necesario pasar la instancia al bus de eventos de Forge:

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

Hay un par de diferentes autobuses de eventos dependiendo del evento. Por ejemplo, los eventos de generación de mineral se `ORE_GEN_BUS` en el `ORE_GEN_BUS` . Puede llamar a este registro desde cualquier lugar, pero se recomienda hacerlo desde su clase de mod principal (con la anotación @Mod) o desde una clase de proxy (algunos eventos son solo para el cliente y un controlador de eventos del lado del cliente solo debe llamarse desde el proxy del cliente, de lo contrario el servidor dedicado se bloqueará!)

Para escuchar cualquier evento dado en su clase de oyente, debe crear un método con la anotación @SubscribeEvent en el método. El tipo de evento es especificado por el Tipo en el único argumento del método. El método puede ser nombrado como quieras.

Tenga en cuenta que algunos tipos de eventos son subtipos (a los que se debe hacer referencia por su tipo adjunto, por ejemplo, `CropGrowEvent.Pre`) y que algunos eventos pueden tener una Fase ya que se dispara en más de un lugar (como todos los `TickEvent` s que se disparan antes) y después de todo el código de vainilla). Como moderador, siempre debe verificar estas dos cosas y solo ejecutar su código cuando sea necesario.

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

Como los mods de Forge interactúan directamente con los elementos internos de Minecraft, se le da mucha potencia al modder para poder afectar a las cosas, pero de manera similar, el código debe seguir el marco de vainilla: no hay atajos para enviar mensajes, el mensaje debe construirse desde ITextComponents manualmente, pero la capacidad de manipular estos objetos (como aplicar el formato de color) es mucho más fácil. Por ejemplo:

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

Lo que da el siguiente resultado:

Lea Escuchas de eventos en Forge en línea:
https://riptutorial.com/es/minecraft/topic/9744/escuchas-de-eventos-en-forge

Capítulo 9: Instalando un servidor Spigot

Examples

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

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 en el que ejecutó BuildTools.jar (craftbukkit-1.10.jar y spigot-1.10.jar). Puede encontrar Spigot-API en \ Spigot \ Spigot-API \ target \ (spigot-api-1.10-R0.1-SNAPSHOT.jar).

Instalación de espiga

Windows

1. Obtén spigot.jar usando BuildTools o desde aquí .
2. Pegue el siguiente texto en un documento de texto. Guárdelo como start.bat en el mismo directorio que spigot.jar: deberá cambiar el nombre de su archivo jar a spigot.jar, o modificar el archivo en el archivo bat para que apunte al archivo correcto. nb: Windows (de forma predeterminada) ocultará la extensión .jar del archivo.

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

3. Haga doble clic en el archivo por lotes.

Linux

1. Obtén spigot.jar usando BuildTools o desde aquí .
2. Cree un nuevo script de inicio (start.sh) en el directorio para iniciar el JAR: #! / Bin / sh

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

3. Abra su terminal y ejecute lo siguiente en el directorio: chmod + x start.sh
4. Ejecute su script de inicio:

```
./start.sh
```

Mac

1. Obtén spigot.jar usando BuildTools o desde aquí .
2. Cree un nuevo script de inicio (start.command) para iniciar el JAR: #! / Bin / sh

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

3. Abre la Terminal y escribe en ella: (¡No pulses entrar!)

```
chmod a+x
```

4. Arrastre su archivo de script de inicio en la ventana de la Terminal. (¡Asegúrese de poner un espacio entre chmod a + x y su script de inicio!)

5. Haga doble clic en su script de inicio.

Lea Instalando un servidor Spigot en línea:
https://riptutorial.com/es/minecraft/topic/9702/instalando-un-servidor-spigot

Capítulo 10: Modding con forja

Sintaxis

- MODID = representa el identificador de la MOD
- MODPath = representa la ruta completa del directorio a su carpeta mod

Observaciones

Este tema debe contener los patrones / ejemplos más utilizados y el código bien probado para modificar la aplicación de Minecraft con forge. Tal vez esto pueda reemplazar la documentación oficial algún día.

Examples

Patrón de implementación para proxies de inicialización

Este ejemplo le muestra cómo implementar clases de proxy para su aplicación de Mod de Minecraft, que se utilizan para inicializar su mod.

En primer lugar, deberá implementar la clase CommonProxy.java base que contiene el método 3 principalmente utilizado:

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

Normalmente su mod tiene 2 paquetes diferentes para Cliente y Código de Servidor, por lo que necesitará en cada paquete una clase secundaria de CommonProxy.java como:

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

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

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

y para el servidor:

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

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

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

Una vez que haya creado estas clases, podrá ampliarlas mediante métodos que solo deben ejecutarse en el lado del cliente o del servidor, pero también puede adjuntarlas a ambas si llama a los métodos en la clase "base".

Por último, debe definir qué proxy se toma en tiempo de ejecución. Tienes que extender tu clase de mod principal con la anotación `@Mod` , mediante:

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

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

Esto permitirá a Forge detectar qué clase debe tomarse en tiempo de ejecución. En los métodos de inicialización de su Mod, ahora puede usar esta propiedad PROXY estática.

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

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

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

Añadiendo sonidos personalizados a tu MOD

Este ejemplo te muestra cómo agregar nuevos sonidos a tu MOD y reproducirlos. En primer lugar, necesita un archivo de sonido que tenga el formato `*.ogg` . Cualquier otro formato no está

permitido por la aplicación de Minecraft y será rechazado.

El archivo de sonido tiene el nombre: sonido1.ogg

Ponga el archivo de sonido en la siguiente ruta:

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

Reemplace 'MODID' por el identificador que definió para su MOD

A continuación, debe crear un `sounds.json` en codificación UTF-8 (Estándar) que define el nombre, el recurso, ... y otras cosas para su sonido personalizado. Este archivo se verá como:

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

Como explicación de esto suena.json.

Hay definidos 2 sonidos definidos, ya que agregué un ejemplo en el que puedes investigar cómo agregar múltiples sonidos. `sound1` tiene la categoría de `player` la segunda es de categoría `ambient` que significa que el volumen del sonido se ve afectado por la configuración de volumen que el usuario ha establecido para los sonidos del reproductor / ambiente. `name` es la propiedad más importante ya que apunta al recurso del sonido. El `MODID` es el identificador de su MOD y es obligatorio ya que la Aplicación buscará en los recursos de su mod para el archivo, de lo contrario buscará en los recursos de Minecraft y no encontrará nada. La propiedad de `stream` significa que el sonido se transmitirá desde el sistema de archivos, que solo es necesario para sonidos de más de 4 segundos.

El archivo de `sounds.json` personalizado.json `sounds.json` ir por debajo de la siguiente ruta:

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

Ahora podrás cargar los sonidos en el registro del juego. Así que tienes que crear una clase que está inicializando `SoundEvent` s y manejando el registro.

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

```
    public static final SoundEvent SOUND_2;

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

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

Después tienes que crear un `SoundRegisterListener` :

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

y regístrelo en `MinecraftForge.EVENT_BUS` como:

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

Finalmente podrás reproducir tus sonidos:

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

Enviando un comando

Este ejemplo muestra diferentes formas de ejecutar 'comandos' para Minecraft desde el código:

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

enviar un comando en SinglePlayer

Lea Modding con forja en línea: https://riptutorial.com/es/minecraft/topic/9956/modding-con-forja

Creditos

S. No	Capítulos	Contributors
1	Empezando con Minecraft	Community, Martin W, SoniEx2
2	Comandos de Complementos	Martin W
3	Creando el primer plugin Spigot	Martin W
4	Creando un bloque básico con forja.	Draco18s
5	Creando un item básico con forja	Draco18s
6	Escribiendo Mods de Minecraft	Draco18s, Martin W
7	Escuchas de eventos en Bukkit	Draco18s, Martin W
8	Escuchas de eventos en Forge	Draco18s, Martin W
9	Instalando un servidor Spigot	Martin W
10	Modding con forja	Code.IT