Бесплатная электронная книга

УЧУСЬ

minecraft

Free unaffiliated eBook created from
Stack Overflow contributors.

#minecraft

Около

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

глава 1: Начало работы с minecraft

замечания

В этом разделе представлен обзор того, что такое minecraft, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках программы minecraft и ссылки на соответствующие темы. Поскольку документация для minecraft нова, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Дата выхода
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

Установка Minecraft

1. Купить Minecraft отсюда
2. Создайте учетную запись Mojang или войдите в нее, если у вас ее уже есть.
3. Если вы создали новую учетную запись, проверьте свою электронную почту.
4. Заполните платежные данные. Убедитесь, что вы на minecraft.net, и вы находитесь в

безопасном соединении (HTTPS)

5. Скачать и запустить Minecraft
6. Открыть пусковую установку Minecraft
7. Войдите в систему с помощью своего адреса электронной почты, если вы приобрели Minecraft после ноября 2012 года, или перенесли свое имя пользователя на учетную запись Mojang.

   Войти под своим именем пользователя , если у вас есть старый аккаунт Minecraft, и еще не перенесены в новый формат счета. Вы можете перейти на учетную запись Mojang на странице учетных записей Mojang.
8. Нажмите «Воспроизвести», если вы хотите играть в новейшей версии.
9. Наслаждайся игрой.

Прочитайте Начало работы с minecraft онлайн: https://riptutorial.com/ru/minecraft/topic/7952/начало-работы-с-minecraft

глава 2: Команды плагина

Examples

Команда Hello

В приведенном ниже коде вы можете увидеть, как добавить команду в свой плагин.

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
```

Прочитайте Команды плагина онлайн: https://riptutorial.com/ru/minecraft/topic/9708/команды-плагина

глава 3: Моддинг с кузницей

Синтаксис

- MODID = представляет Идентификатор MOD
- MODPath = обозначает полный путь к каталогу в папке мод

замечания

Этот раздел должен содержать большинство использованных шаблонов / примеров и хорошо протестированный код для модификации приложения Minecraft с помощью кузницы. Возможно, это может заменить официальную документацию в один прекрасный день.

Examples

Схема реализации для инициализационных прокси

В этом примере показано, как реализовать прокси-классы для приложения Minecraft Mod, которые используются для инициализации вашего мода.

Прежде всего вам понадобится реализовать базовый класс CommonProxy.java, который содержит 3 главным образом используемый метод:

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

Обычно ваш мод имеет 2 разных пакета для кода клиента и сервера, поэтому вам понадобится в каждом пакете дочерний класс CommonProxy.java, например:

```
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);
```

```
    }
}
```

и для сервера:

```
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);
    }
}
```

После того, как вы создали эти классы, вы можете расширять их методами, которые должны выполняться только на стороне клиента или сервера, но также могут присоединяться к ним, если вы вызываете методы в классе «base».

Наконец, вы должны определить, какой прокси-сервер принимается во время выполнения. Вы должны расширить свой основной класс mod с `@Mod` аннотации `@Mod` , путем:

```
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;
```

Это позволит Forge определить, какой класс следует выполнять во время выполнения. В методах инициализации вашего Mod теперь вы можете использовать это статическое свойство PROXY.

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

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

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

Добавление пользовательских звуков в ваш MOD

В этом примере показано, как вы добавляете новые звуки в ваш MOD и воспроизводите их. Прежде всего вам нужен звуковой файл, который имеет формат `*.ogg` . Любой другой формат не допускается приложением Minecraft и будет отклонен.

Звуковой файл имеет название: sound1.ogg

Поместите звуковой файл по следующему пути:

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

Замените «MODID» на идентификатор, определенный для вашего MOD

Затем вам нужно создать `sounds.json` в кодировке UTF-8 (Standard), которая определяет имя, ресурс и т. Д. Для вашего пользовательского звука. Этот файл будет выглядеть так:

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

Как объяснение для этого sounds.json.

Определены 2 звуковых сигнала, так как я добавил пример, в котором вы можете исследовать, как добавлять многократные звуки. `sound1` имеет `player` категории второй категории `ambient` что означает, что на громкость звука влияют настройки громкости, которые пользователь установил для звуков игрока / окружающего звучания. `name` является самым важным свойством, поскольку оно указывает на ресурс звука. `MODID` является идентификатором вашего MOD и является обязательным, так как приложение будет искать в ресурсах вашего мода для файла, иначе он будет искать в ресурсах Minecraft и ничего не найдет. Свойство `stream` означает, что звук будет передаваться из файловой системы, которая требуется только для звуков более 4 секунд.

Ваш пользовательский файл `sounds.json` должен пройти по следующему пути:

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

Теперь вы сможете загрузить звуки в реестр игры. Поэтому вам нужно создать класс, который инициализирует `SoundEvent` s и обрабатывает регистрацию.

```
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;
    }
}
```

После этого вы должны создать `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);
        }
}
```

и зарегистрируйте его в `MinecraftForge.EVENT_BUS` например:

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

Наконец, вы сможете воспроизводить свои звуки:

```
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
    }
}
```

Отправка команды

В этом примере показаны различные способы выполнения «команд» для Minecraft из кода:

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

отправить команду в SinglePlayer

Прочитайте Моддинг с кузницей онлайн: https://riptutorial.com/ru/minecraft/topic/9956/моддинг-с-кузницей

глава 4: Написание Minecraft Mods

Вступление

Понимание написания собственных модов для Minecraft.

Первое, что нужно знать, это то, что в основном есть две платформы, на которых основываются моды. Кузница и Буккит.

Каждая платформа имеет свои преимущества и недостатки и, как правило, несовместима между собой. Моды Forge управляют гаммой и, как правило, ориентированы на игры. Моды Bukkit полностью ориентированы на сервер и обычно ориентированы на инструмент. Моды Bukkit называются плагинами.

Examples

Основные плагины Bukkit

- Написание команды Hello
- Написание прослушивателя событий

Базовые кузницы

- Создание базового блока
- Создание базового элемента
- Написание команды Hello
- Запись обработчика событий
- Начало работы с возможностями [1.8+]

Прочитайте Написание Minecraft Mods онлайн: https://riptutorial.com/ru/minecraft/topic/9740/написание-minecraft-mods

глава 5: Слушатели событий в Forge

Examples

Создание прослушивателя событий в Forge

Создание прослушивателя событий в Forge очень похоже на создание в Bukket.

Для создания класса слушателя требуется намного меньше. Интерфейса для реализации или другого импорта нет.

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

Для его регистрации необходимо передать экземпляр на шину событий Forge:

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

В зависимости от события есть несколько различных автобусов событий. Например, события генерации руды `ORE_GEN_BUS` на `ORE_GEN_BUS` . Вы можете вызывать эту регистрацию в любом месте, но рекомендуется называть ее либо из вашего основного класса мод (с аннотацией @Mod), либо из прокси-класса (некоторые события только на стороне клиента, а обработчик событий на стороне клиента должен быть вызван из прокси-сервера клиента, иначе выделенный сервер сработает!)

Чтобы прослушать любое заданное событие в вашем классе слушателя, вы должны создать метод с аннотацией @SubscribeEvent метода. Тип события определяется типом в единственном аргументе метода. Метод может быть назван как угодно.

Обратите внимание, что некоторые типы событий являются подтипами (на которые должен ссылаться их `CropGrowEvent.Pre` тип, например `CropGrowEvent.Pre`), и что некоторые события могут иметь Фазу, поскольку она запускается в нескольких местах (например, все `TickEvent` s, которые `TickEvent` как перед и после всего ванильного кода). В качестве моддера вы всегда должны проверять обе эти вещи и только запускать свой код при необходимости.

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

Поскольку мосты Forge взаимодействуют напрямую с внутренними компонентами Minecraft, для моддера требуется много энергии, чтобы иметь возможность влиять на вещи, но аналогичным образом код должен соответствовать каркасу ванили: нет ярлыков для

отправки сообщений, сообщение должно быть построено из ITextComponents вручную, но способность управлять этими объектами (например, применение форматирования цветов) намного проще. Например:

```
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);
```

Это дает следующий результат:

Прочитайте Слушатели событий в Forge онлайн: https://riptutorial.com/ru/minecraft/topic/9744/слушатели-событий-в-forge

глава 6: Слушатели событий в Буките

Examples

Создание прослушивателя событий

Чтобы зарегистрировать свои методы, класс, содержащий EventHandler (s), должен реализовать интерфейс Listener.

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

public final class ListenerClass implements Listener {
}
```

Вам необходимо зарегистрировать прослушиватель событий, добавив следующий вызов к методу onEnable в классе, который расширяет JavaPlugin:

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

Чтобы прослушать любое заданное событие в вашем классе слушателя, вы должны создать метод с аннотацией @EventHandler в методе. Тип события определяется типом в единственном аргументе метода. Метод может быть назван как угодно.

```
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!");
    }
}
```

Параметры EventHandler

Аннотация `org.bukkit.event.EventHandler` принимает пару параметров.

priority - Указывает приоритет вашего слушателя. Есть шесть разных приоритетов в порядке исполнения: LOWEST, LOW, NORMAL [default], HIGH, HIGHEST, MONITOR. Эти константы относятся к перечислению `org.bukkit.event.EventPriority` .

Если вы хотите изменить исход события, выберите очень осторожно от LOWEST до HIGHEST. Предлагаемые обобщенные защитные плагины на LOWEST, более конкретные плагины на NORMAL и переопределение плагинов на HIGH. Если вы хотите действовать, когда происходит событие, но не измените результат, используйте MONITOR.

Примечание. Приоритет MONITOR должен использоваться только для чтения. Этот приоритет полезен для регистрации плагинов, чтобы увидеть результаты события, и изменение значений может помешать этим типам плагинов.

ignoreCancelled - логическое значение, указывающее, должен ли ваш слушатель срабатывать, если событие было отменено до того, как очередь слушателя обработает событие. Неверно по умолчанию.

```
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
    }
}
```

Создание пользовательских событий

Иногда вам нужно создать свое собственное Событие, которое другие плагины могут прослушать (Vault, среди других плагинов, делает это) и даже отменить. API событий Bukkit позволяет это сделать возможным. Все, что вам нужно сделать, это создать новый класс, расширить его `Event` , добавить обработчики и атрибуты, необходимые вашему событию (например, Player или message).

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

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

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

    public String getMessage() {
        return message;
    }

    public HandlerList getHandlers() {
        return handlers;
    }

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

```
    }
}
```

Вызов пользовательского события

Вы контролируете создание и вызывать свои события, где вы называете это полностью зависит от вас. Вот пример

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

Помните: вы контролируете свои события. Если вы не назовете это и не подействуйте на него, этого не произойдет!

Прослушивание пользовательского события

Прослушивание пользовательского события совпадает с прослушиванием обычного события.

```
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
    }
}
```

Внедрение CustomEvent Cancellable

Если вы хотите, чтобы ваше событие было `boolean cancelled` , просто добавьте `implements Cancellable` , `boolean cancelled` и getter и 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;
    }
}
```

После этого вы можете проверить, отменял ли плагин пользовательское событие.

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

Отмена регистрации событий или слушателей

Вы можете отменить регистрацию отдельных событий, всех классов слушателей или всех событий, зарегистрированных вашим плагином или даже другими плагинами!

Отменить регистрацию определенного события

Каждый класс событий имеет статический метод getHandlerList (), вызывает это, а затем вы можете использовать метод .unregister ().

```
PlayerInteractEvent.getHandlerList().unregister(plugin);
// this will unregister all PlayerInteractEvent instances from the plugin
```

```
// you can also specify a listener class instead of plugin.
```

Теперь вы знаете, зачем вам нужен getHandlerList () в ваших пользовательских событиях.

Отменить регистрацию всех событий

Используя класс HandlerList и его статический метод unregisterAll (), вы можете легко отменить регистрацию событий из классов или плагинов-слушателей.

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

Прочитайте Слушатели событий в Буките онлайн:
https://riptutorial.com/ru/minecraft/topic/9739/слушатели-событий-в-буките

глава 7: Создание базового блока с помощью Forge

Вступление

Создание простого, декоративного блока с помощью Forge - одна из первых задач, которую должен научиться начинающий модератор. Как это изменилось в разных версиях Minecraft и, вероятно, на «умеренной» позиции сложности 1.7.10 из-за большого количества ошибок.

замечания

Если что-то пойдет не так, и ваш пользовательский блок (либо при размещении, либо удерживается) имеет отсутствующую текстуру (черный / фиолетовый) или модель (куб по умолчанию, который слишком велик, когда удерживается) проверяет журнал. Проблемы такого рода будут почти *всегда* появляться в журнале, если вы зарегистрировали вещи правильно.

Первое, что будет отображаться, - это строка `"Exception loading model for variant..."` или аналогичная, сообщающая, какой блок или элемент не удалось загрузить должным образом. После дюжины строк, начинающихся с `at...` будет еще одна строка, начинающаяся `Caused by...`

Эта строка «Causeed By» является важной, она скажет вам, какой файл не удалось загрузить должным образом и может быть одной из нескольких ошибок, таких как:

- Malformed JSON (ваш JSON-файл недействителен и имеет опечатку)
- Файл не найден (файл, который Minecraft ищет, неправильно назван или находится в нужном месте)
- Отсутствие исключения варианта (ваш Blockstate JSON является неполным)

Вы можете даже получить более одной ошибки при ошибке! Если вы не знаете, какой блок является проблемой, закомментируйте каждый блок и элемент, который, как вы знаете, работаете, уменьшите список ошибок до блока (или блоков), о котором идет речь. Возможно, вам придется делать это несколько раз.

Кроме того, текстура смешивания отображается так же, как и простой список всех недостающих ресурсов для данного домена (идентификатор мод).

Examples

Класс блоков

Сначала нам нужен класс, представляющий блок

```
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);
    }
}
```

Даже здесь есть несколько доступных модификаций: материал (который управляет некоторыми свойствами, такими как возможность выталкивать поршни и может ли он быть разбит вручную), твердость (как долго это требуется для разрыва), уровень урожая (соответствующий инструмент и инструмент материал: в данном случае деревянная кирка), сопротивление (против взрывов), вкладка, отображаемая в творческом меню, и какой шаг звука у него есть.

Здесь вам понадобятся любые фантастические функциональные блоки, но пока мы делаем блок, который выглядит просто красиво, поэтому мы закончили.

Блок-модель JSON

Затем нам нужно сообщить Minecraft, что мы хотим, чтобы наш блок выглядел.

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

Это почти все, что нужно, чтобы он работал после регистрации блока. Важно только то, что **имя файла** соответствует имени **реестра,** используемому для регистрации блока, и должно быть во всех строчных (имена файлов 1.11+ *должны* быть строчными, до этого они чувствительны к регистру).

Назовите модель JSON-файла `my_block.json` (сопоставив имя реестра, которое мы собираемся дать позже) и сохраним его в `src\main\resources\assets\example\models\block\` (где, `example` это идентификатор мод, указанный в @Mod аннотация вашего основного класса мод).

Здесь в блочной модели используется родительский элемент block / cube_all, что означает, что единая поставляемая текстура будет использоваться на всех грани. Существуют и

другие модели по умолчанию, такие как:

- блок / куб (все шесть граней назначены независимо)
- block / cube_bottom_top (верхняя и нижняя грани независимы от сторон)
- блок / ориентируемый (направленный облицовочный блок, например печь)
- блок / крест (цветы, высокая трава)
- блок / культура (пшеница, морковь)

Обратите внимание, что каждая модель указывает текстуры, которые она использует с помощью идентификатора имени (например, `"all"` или `"top"`). Посмотрите на родительскую модель, чтобы определить, что эти имена, если вы не уверены. Некорректно определенные текстуры могут привести к *отсутствию ошибок при обработке* текстур.

Также возможно создать полностью настраиваемую модель или создать пользовательскую родительскую модель. Но пока этого будет достаточно.

Не забудьте создать текстуру, назовите ее `decorative.png` (как это указано в файле JSON) и сохраните его в `src\main\resources\assets\example\textures\blocks\`

Регистрация блока

Регистрация блоков выполняется из основного класса mod или метода класса ModBlocks, вызванного из основного класса mod во время preInit.

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

Существует важная причина использования `block.setUnlocalizedName(block.getRegistryName().toString());` также! Это гарантирует, что ваши неблокированные имена блоков (и элементов) содержат идентификатор мод, чтобы избежать конфликтов языковых файлов между модами.

О, вы хотите, чтобы версия продукта тоже могла существовать в вашем инвентаре? Это создается отдельно после 1.7.10 и выполняется следующим образом:

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

Обратите внимание, что мы установили имя реестра ItemBlock в ту же строку, что и наш блок. Это то, как Forge и match блокирует их экземпляр ItemBlock и наоборот.

Но подождите, есть еще!

У вашего блока может быть *форма элемента* , но у этого элемента еще нет модели или текстуры! Модели автоматически регистрируются для блоков, но не для элементов. Это *может быть вызвано только из прокси-сервера клиента* и не охватывает блоки с вариантами (например, шерсть или листья).

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

Вообще говоря, вам также не понадобится модель товара JSON, поскольку Forge и vanilla снова упадут на модель блока, но это не всегда так. Если вы обнаружите, что вам нужна модель товара JSON, просто откройте его на своем блоке JSON и сохраните его в `src\main\resources\assets\example\models\item\` с тем же именем файла, что и имя реестра блока.

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

Прочитайте Создание базового блока с помощью Forge онлайн:
https://riptutorial.com/ru/minecraft/topic/9748/создание-базового-блока-с-помощью-forge

глава 8: Создание базового элемента с кузницей

Вступление

Создание простого элемента с Forge - одна из первых задач, которую должен научиться начинающий модем. Как это изменилось в разных версиях Minecraft и, вероятно, на «умеренной» задаче с ошибкой 1.7.10 из-за большого количества ошибок, которые можно совершить, особенно с тем, чтобы сделать их правильно отображенными.

замечания

Если что-то пойдет не так, и ваш пользовательский элемент имеет отсутствующую текстуру (черный / фиолетовый) или модель (куб по умолчанию, который слишком велик, когда он удерживается), проверьте журнал. Проблемы такого рода будут почти *всегда* появляться в журнале, если вы зарегистрировали вещи правильно.

Первое, что будет отображаться, - это строка `"Exception loading model for variant..."` или аналогичная, сообщающая, какой блок или элемент не удалось загрузить должным образом. После дюжины строк, начинающихся с `at...` будет еще одна строка, начинающаяся `Caused by...`

Эта строка «Causeed By» является важной, она скажет вам, какой файл не удалось загрузить должным образом и может быть одной из нескольких ошибок, таких как:

- Malformed JSON (ваш JSON-файл недействителен и имеет опечатку)
- Файл не найден (файл, который Minecraft ищет, неправильно назван или находится в нужном месте)
- Отсутствие исключения варианта (ваш Blockstate JSON является неполным)

Вы можете даже получить более одной ошибки при ошибке! Если вы не знаете, какой элемент является проблемой, закомментируйте каждый блок и элемент, который, как вы знаете, работаете, уменьшите список ошибок до блока (или блоков), о котором идет речь. Возможно, вам придется делать это несколько раз.

Кроме того, текстура смешивания отображается так же, как и простой список всех недостающих ресурсов для данного домена (идентификатор мод).

Examples

Класс предмета

Эта часть не изменилась по версиям Minecraft очень много, хотя в точных сигнатурах метода также были некоторые мутации, а также иерархия классов. Основной предмет (тот, у которого нет функциональности, например, палки или слитка: это правильно, оба не делают ничего!)

```
public class CustomItem extends Item {

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

В отличие от блоков, на данный момент не так много места для настройки. Все, что мы можем сделать, это изменить, может ли элемент нанести урон (инструменты используют это) и какую таковую будет в нем. Название и текстура мы будем обрабатывать, когда мы регистрируем элемент с помощью GameRegistry.

Тем не менее, это все, что необходимо для удержания, переноса, опускания, изготовления, плавки и другого использования предмета. Некоторые элементы в Minecraft (например, палки) даже не имеют уникального класса и просто используют `new Item()` . Мы могли бы сделать это здесь, однако для любого элемента с дополнительной функциональностью потребуется класс.

Модель товара

Как и в случае с блоками, элементам также нужны модели.

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

Это почти все, что необходимо для его работы после регистрации элемента. Важно только то, что имя файла соответствует имени реестра, используемому для регистрации блока, и должно быть во всех строчных (имена файлов 1.11+ должны быть строчными, до этого они чувствительны к регистру).

Обратите внимание, что «layer0» - единственная текстура, и маловероятно, что любая другая текстура будет указана вообще (хотя некоторые предметы, такие как зелья и кожаные доспехи, имеют «layer1»). Все имена определяются `item/builtin` (внутренней самой верхней родительской моделью для элементов), в отличие от блоков.

Назовите модель JSON-файла `my_item.json` (сопоставив имя реестра, которое мы собираемся дать позже) и сохраните его в `src\main\resources\assets\example\models\item\` (где

`example` - это идентификатор мод, указанный в @Mod аннотация вашего основного класса мод).

Кроме того, создайте текстуру для своего элемента, назовите ее `basic.png` и сохраните ее в `src\main\resources\assets\example\textures\items\`

Модель элемента здесь использует родителя элемента / сгенерированного, что означает, что единая поставляемая текстура будет использоваться (как и для большинства неблокированных элементов) и будет удерживаться в руке игрока в ориентации по умолчанию. Существует также пункт / handheld, который определяет различные ориентации дисплея (для инструментов). Элементы могут также предоставлять свой собственный атрибут «display», переопределяя их от родителя, но не нужны в 99,9% использования.

Регистрация позиции

Регистрация элементов производится из основного класса mod или метода класса ModItems, вызванного из основного класса mod во время preInit.

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

Существует важная причина использовать `item.setUnlocalizedName(item.getRegistryName().toString());` также! Это гарантирует, что нелокализованное имя вашего объекта содержит идентификатор мод, чтобы избежать конфликтов языковых файлов между модами.

Теперь предмет тоже нуждается в модели, и в этом случае ситуация стала трудной после 1.7.10, которая просто включала указание Minecraft имя текстуры для загрузки и могло быть указано в конструкторе элемента.

```
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;
        }
    });
```

Обратите внимание, что этот раздел *должен* быть расположен только на стороне клиента (т. Е. Клиентский прокси), так как многие ссылочные классы не существуют на выделенном сервере.

Регистрация элементов с *вариантами,* например Saplings, должна выполняться по-другому, используя `ModelLoader.setCustomModelResourceLocation(item, metadata, resourceLocation)` хотя он позволяет нам использовать файл Blockstate, чтобы указать наши варианты (что *намного* предпочтительнее альтернативы). Наш товар не использует варианты, поэтому мы закончили.

Прочитайте Создание базового элемента с кузницей онлайн:
https://riptutorial.com/ru/minecraft/topic/9850/создание-базового-элемента-с-кузницей

глава 9: Создание первого плагина Spigot

Examples

Первый плагин в Eclipse

необходимое условие

В этом руководстве предполагается, что вы уже использовали BuildTools и запустили сервер Spigot хотя бы один раз. Он также предполагает, что у вас есть файл jar-файла Spigot-API, который мы будем использовать.

1) Запустите Eclipse ; при желании вы можете изменить местоположение рабочего пространства.

2) Создайте новый проект

1. Задайте имя проекта так, как хотите. Здесь мы выбрали MyFirstPlugin.
2. Нажмите кнопку "Далее.
3. Выберите «Добавить внешние JAR» на вкладке «Библиотеки». В диалоговом окне Выбор JAR выберите файл jar-файла с палитрой-api-shaded, который можно найти в Spigot / Spigot-API / target / внутри вашей папки BuildTools.
4. Выберите «Готово»

3) Добавить новый пакет

Щелкните правой кнопкой мыши на **src** и выберите « **Создать»> «Пакет»** . Вы можете использовать любое соглашение по пространству имен, которое вы хотите, просто быть последовательным. (например: com.google.android).

4) Создайте новый класс

1. Щелкните правой кнопкой мыши только что созданный пакет и выберите « **Создать»> «Класс»** .

2. Дайте ему имя; часто с тем же именем, что и проект. Внутри редактора откроется только что созданный Java-класс. Код должен выглядеть примерно так:

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

5) Изменить объявление класса

1. Ваш класс должен простираться от JavaPlugin. Eclipse выдает ошибку, так как не знает, что такое JavaPlugin. Если вы успешно импортировали Spigot-API, вы сможете импортировать JavaPlugin, добавив оператор import. Вам не нужно вручную вводить эту строку, просто щелкните по ошибке и выберите соответствующее действие. Теперь ваш код должен выглядеть так:

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

public class MyFirstPlugin extends JavaPlugin {

}
```

6) Внедрить необходимые методы

Класс JavaPlugin имеет некоторые абстрактные методы, которые должны быть реализованы вашим плагином. Следовательно, добавьте функции onEnable и onDisable, которые будут запускаться, когда плагин отключен или включен в консоли. Теперь вы можете оставить эти пустые поля. Вам также необходимо написать `@Override` над методом.

Примечание. Если ваш плагин включен или отключен, вам не нужно добавлять getLogger, Bukkit уже делает это за вас.

```
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) Создайте файл plugin.yml

Щелкните правой кнопкой мыши проект и создайте файл **New> File** . Назовите его **plugin.yml** . Вставьте следующее:

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

8) Экспорт

Поскольку ошибок нет, мы можем экспортировать этот проект как JAR. Щелкните правой

кнопкой мыши имя проекта и выберите «Экспорт». В следующем диалоговом окне выберите JAR-файл. Нажмите кнопку "Далее. Вы можете снять флажок в classpath и включить проект и изменить пункт назначения экспорта в папку плагинов

9) Запуск

Запустите сервер, и вы увидите, что ваш плагин включен.

Прочитайте Создание первого плагина Spigot онлайн:
https://riptutorial.com/ru/minecraft/topic/9766/создание-первого-плагина-spigot

глава 10: Установка сервера Spigot

Examples

инструмент сборки

Что это?

BuildTools.jar - это решение для создания Bukkit, CraftBukkit, Spigot и Spigot-API. Все это делается на вашем компьютере! Необходимо несколько предварительных программ, но приведенные ниже инструкции помогут вам в том, что вам нужно сделать.

Предпосылки

Для использования BuildTools необходимы два приложения: Git и Java.

Windows

Гит

Чтобы BuildTools запускался в Windows, вам нужно будет установить Git. Для Windows он распространяется через git-scm, который можно скачать здесь . Установите его там, где вам нравится, он предоставит git bash, который будет использоваться для запуска баннера BuildTools. Просто продолжайте работать, когда запускаете установщик.

Джава

Загрузите JRE 8 отсюда и установите. Просто продолжайте работать, когда запускаете установщик.

Linux

Как git, так и Java, а также команды утилиты могут быть установлены с помощью одной команды через диспетчер пакетов.

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

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

Arch: `pacman -S jdk8-openjdk git`

макинтош

Git можно загрузить с сайта: http://sourceforge.net/projects/git-osx-installer/files/

Возможно, Java необходимо обновить из распространенной версии Apple, и даже если она была ранее обновлена, возможно, потребуется связать ее с оболочкой. Следуйте приведенным ниже шагам: https://gist.github.com/johan/10590467

Запуск BuildTools

1. Загрузите BuildTools.jar из https://hub.spigotmc.org/jenkins/job/BuildTools/lastSuccessfulBuild/artifact/target/BuildTools.jar .
2. Откройте ваш терминал, если вы находитесь в Linux, или git bash на Windows.
    1. Git bash можно найти на рабочем столе или в меню «Пуск» под названием «git bash». Также можно открыть его, щелкнув правой кнопкой мыши на чем угодно, так как теперь это элемент в вашем контекстном меню.
3. Перейдите туда, где вы загрузили BuildTools.jar, или воспользуйтесь способом командной строки, чтобы загрузить банку в ваш текущий каталог.
    1. В Windows вы можете либо использовать команду cd для изменения каталогов, либо щелкнуть правой кнопкой мыши пустое место в папке, где находится BuildTools.jar (не нажимайте на CreateTools.jar) и нажмите «git bash», который откроет его в вашем текущем каталоге.
4. Запустите BuildTools.jar с терминала (не дважды щелкните BuildTools.jar), выполнив следующие действия:
    1. В Linux запустите git config --global --unset core.autocrlf, затем запустите java -jar BuildTools.jar в bash или другой соответствующей оболочке.
    2. В Windows запустите следующую команду в открывшемся окне git bash: java -jar BuildTools.jar. Имейте в виду, что требуется, чтобы у вас были BuildTools № 35 или более поздние версии, более старые версии не будут работать.
    3. На Mac выполните приведенные ниже команды, экспортируйте MAVEN_OPTS = "- Xmx2G" java -Xmx2G -jar BuildTools.jar
5. Подождите, как он строит ваши банки. Через несколько минут вы должны иметь свежую сборку банок!

6. Вы можете найти CraftBukkit и Spigot в том же каталоге, в котором вы запускали BuildTools.jar (craftbukkit-1.10.jar и spigot-1.10.jar). Вы можете найти Spigot-API в \ Spigot \ Spigot-API \ target \ (spigot-api-1.10-R0.1-SNAPSHOT.jar).

Установка штепсельной вилки

Windows

1. Получите spigot.jar, используя BuildTools или отсюда .
2. Вставьте следующий текст в текстовый документ. Сохраните его как start.bat в том же каталоге, что и spigot.jar: вам нужно будет переименовать свою банку в spigot.jar или изменить файл в файле bat, чтобы указать на правильный файл. nb: Windows (по умолчанию) скроет расширение .jar файла.

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

3. Дважды щелкните командный файл.

Linux

1. Получите spigot.jar, используя BuildTools или отсюда .
2. Создайте новый сценарий запуска (start.sh) в каталоге для запуска JAR: #! / Bin / sh

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

3. Откройте терминал и выполните следующие действия в каталоге: chmod + x start.sh
4. Запустите скрипт запуска:

```
./start.sh
```

макинтош

1. Получите spigot.jar, используя BuildTools или отсюда .
2. Создайте новый сценарий запуска (start.command), чтобы запустить JAR: #! / Bin / sh

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

3. Откройте терминал и введите в него: (Не нажимайте Enter!)

```
chmod a+x
```

4. Перетащите файл сценария запуска в окно терминала. (Не забудьте поставить пробел между chmod a + x и вашим сценарием запуска!)

5. Дважды щелкните сценарий запуска.

Прочитайте Установка сервера Spigot онлайн: https://riptutorial.com/ru/minecraft/topic/9702/установка-сервера-spigot

кредиты

S. No	Главы	Contributors
1	Начало работы с minecraft	Community, Martin W, SoniEx2
2	Команды плагина	Martin W
3	Моддинг с кузницей	Code.IT
4	Написание Minecraft Mods	Draco18s, Martin W
5	Слушатели событий в Forge	Draco18s, Martin W
6	Слушатели событий в Буките	Draco18s, Martin W
7	Создание базового блока с помощью Forge	Draco18s
8	Создание базового элемента с кузницей	Draco18s
9	Создание первого плагина Spigot	Martin W
10	Установка сервера Spigot	Martin W