

DO ZERO AO CRIATIVO



PLUGINS MINECRAFT COM JAVA

Diego D. Leitão

Antes de tudo

O que é um plugin?

Um plugin de Minecraft é como um 'aplicativo' que você instala em um servidor de Minecraft para adicionar ou modificar funcionalidades do jogo.

Imagine que o Minecraft é como uma casa: com os plugins, você pode adicionar novos móveis, mudar a pintura das paredes ou até criar novas regras para quem mora nela.

Se você quer personalizar seu servidor Minecraft, criar um plugin é o primeiro passo. Neste ebook, vamos listar os principais projetos, apresentar a estrutura básica de um plugin e ensinar como lidar com eventos e adicionar itens personalizados.

01

PRINCIPAIS PROJETOS

Principais Projetos

Bukkit/Spigot



O Bukkit é a base mais usada para desenvolvimento de plugins.

O Spigot, uma versão otimizada, é amplamente adotado pela comunidade.

- Público-alvo: Servidores de médio a grande porte.
- Recursos principais: Hooks para eventos, APIs para entidades, blocos e jogadores.

Link: [SpigotMC](https://www.spigotmc.org/)

Principais Projetos

Paper



O Paper é um fork do Spigot, com mais desempenho e funcionalidades extras.

- Público-alvo: Servidores que precisam de alta performance.
- Recursos principais: Suporte a otimizações e melhorias na API.

Link: [PaperMC](https://papermc.io/)



ESTRUTURA

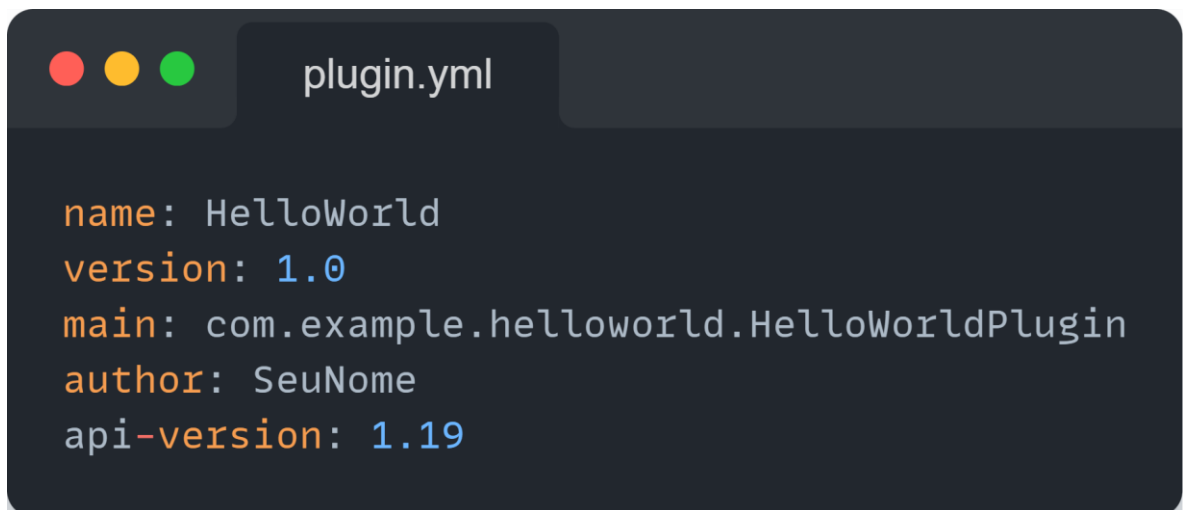
DE UM PLUGIN

Estrutura de um Plugin

Vamos criar um plugin simples que exibe uma mensagem ao jogador quando ele entra no servidor.

Arquivo plugin.yml

O plugin.yml é o arquivo de configuração principal do plugin.



```
name: HelloWorld
version: 1.0
main: com.example.helloworld.HelloWorldPlugin
author: SeuNome
api-version: 1.19
```

Estrutura de um Plugin

Classe Principal do Plugin

Essa classe é o ponto de entrada do plugin.

```
helloworldplugin.java

package com.example.helloworld;

import org.bukkit.plugin.java.JavaPlugin;
import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;
import org.bukkit.event.player.PlayerJoinEvent;

public class HelloWorldPlugin extends JavaPlugin implements Listener {

    @Override
    public void onEnable() {
        //Toda a lógica do plugin ao ser ativado
        getServer().getPluginManager().registerEvents(this, this);
        getLogger().info("HelloWorld Plugin Ativado!");
    }

    @Override
    public void onDisable() {
        //Toda a lógica do plugin ao ser desativado
        //Ou quando ocorre erro de carregamento
        getLogger().info("HelloWorld Plugin Desativado!");
    }

    //Evento quando um jogador entra no servidor
    @EventHandler
    public void onPlayerJoin(PlayerJoinEvent event) {
        //Envia uma mensagem de boas vindas ao jogador
        String nome = event.getPlayer().getName();
        event.getPlayer()
            .sendMessage("Bem-vindo, " + nome + "!");
    }
}
```




PRINCIPAIS EVENTOS

Principais Eventos

Eventos permitem interagir com o comportamento do jogo. Aqui estão alguns exemplos práticos:

Jogador Quebrando Bloco

Esse evento sempre é acionado quando um bloco é quebrado pelo jogador

```
eventdeathplayer.java

package com.example.helloworld;

import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;
import org.bukkit.event.block.BlockBreakEvent;

public class EventBlockBreak implements Listener {

    @EventHandler
    public void onBlockBreak(BlockBreakEvent event) {
        event.getPlayer().sendMessage("Você quebrou um bloco!");
    }
}
```

Principais Eventos

Jogador Morre

Esse evento sempre é acionado quando um jogador é derrotado

```
eventdeathplayer.java

package com.example.helloworld;

import org.bukkit.event.Listener;
import org.bukkit.event.EventHandler;
import org.bukkit.event.entity.PlayerDeathEvent;

public class EventDeathPlayer implements Listener {

    @EventHandler
    public void onPlayerDeath(PlayerDeathEvent event) {
        String nome = event.getEntity().getName();
        event.setDeathMessage(nome+" morreu! Aperte F por respeito");
    }
}
```

OH

ITEM

CUSTOMIZADO

Item customizado

Você pode adicionar itens customizados ao jogo usando a API do Bukkit.

Aqui está um exemplo de como criar uma espada especial:

Criando um Item Personalizado

```
import org.bukkit.Material;
import org.bukkit.inventory.ItemStack;
import org.bukkit.inventory.meta.ItemMeta;

public ItemStack createCustomSword() {
    ItemStack item = new ItemStack(Material.NETHERITE_SWORD,1);
    ItemMeta meta = item.getItemMeta();
    //Definindo um nome para o item
    meta.displayName(Component.text("$6Relíquia do Guerreiro"));
    //Adicionando uma descrição
    List<Component> loreitem = new ArrayList<>();
    loreitem.add(Component.text("$7Uma espada antiga"));
    loreitem.add(Component.text("$7forjada no nether"));
    meta.lore(loreitem);
    //Adicionando encantamentos
    meta.addEnchant(Enchantment.BANE_OF_ARTHROPODS,5,true);
    meta.addEnchant(Enchantment.FIRE_ASPECT,2,true);
    meta.addEnchant(Enchantment.LOOTING,3,true);
    meta.addEnchant(Enchantment.SHARPNESS,5,true);
    meta.addEnchant(Enchantment.SMITE,5,true);
    meta.addEnchant(Enchantment.SWEEPING_EDGE,5,true);
    meta.addItemFlags(ItemFlag.HIDE_ENCHANTS);
    //Adicionando atributo customizado
    meta.addAttributeModifier(ENTITY_INTERACTION_RANGE,
        new AttributeModifier(ENTITY_INTERACTION_RANGE.getKey(),+7,
            AttributeModifier.Operation.ADD_NUMBER));
    //Mudando a raridade do item
    meta.setRarity(ItemRarity.EPIC);
    //Definindo como inquebrável
    meta.setUnbreakable(true);
    //Para caso queira usar um textura personalizada
    meta.setCustomModelData(1);
    item.setItemMeta(meta);
    return item;
}
```

Plugins Minecraft Com Java - Diego D. Leitão

Item customizado

Dando o Item ao Jogador

```
import org.bukkit.event.player.PlayerJoinEvent;

@EventHandler
public void onPlayerJoin(PlayerJoinEvent event) {
    event.getPlayer().getInventory().addItem(createCustomSword());
    event.getPlayer().sendMessage("Receba sua Espada Lendária!");
}
```

05

EXPORTANDO O PLUGIN

Exportando o Plugin

Depois de escrever e testar seu plugin, é hora de exportá-lo para uso no servidor.

Passo a Passo para Exportar

Compile o Código:

- Use sua IDE (como IntelliJ IDEA ou Eclipse) para compilar o projeto.
- Certifique-se de que todas as dependências necessárias estão incluídas.

Crie o Arquivo JAR:

- Configure sua IDE para exportar o projeto como um arquivo JAR.
- Inclua o arquivo plugin.yml no JAR.



Exportando o Plugin

Passo a Passo para Exportar

Teste no Servidor:

- Copie o arquivo JAR para a pasta plugins do servidor Minecraft.
- Inicie o servidor e verifique os logs para garantir que o plugin foi carregado corretamente.

Depure Problemas:

- Caso o plugin não funcione, confira os logs do servidor para identificar erros.

Exportando o Plugin

Ferramenta Alternativa: Maven/Gradle

Se você utiliza Maven ou Gradle, adicione o seguinte trecho ao seu arquivo de configuração para empacotar o plugin:

Exemplo com Maven

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.2.0</version>
      <configuration>
        <archive>
          <manifestEntries>
            <Main-Class>com.example.helloworld.HelloWorldPlugin</Main-Class>
          </manifestEntries>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>;
```

06

DICAS



Dicas

Em caso de duvidas ou decida explorar mais conteúdo além do mostrado nesse ebook, recomendo que acesse a documentação ou converse em fóruns da comunidade do projeto que está utilizando para obter ajuda com erros ou tirar dúvidas que possa surgir

AGRADECIMENTOS



Obrigado por ter lido!

Agora você tem o conhecimento básico para criar, personalizar e exportar plugins para seu servidor Minecraft.

Divirta-se desenvolvendo!

Esse ebook foi feito utilizando IA e diagramado por um humano

Esse ebook foi feito com fins didáticos em construção, não realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados pela IA

Meu GitHub de um plugin: [ValentPlugin](#)