Co musi być w głównej klasie pluginu Bukkit/Spigot, żeby config działał?

# 1. Dziedziczenie po JavaPlugin

Twoja główna klasa pluginu MUSI dziedziczyć po JavaPlugin:

public class MyPlugin extends JavaPlugin {

# 2. Tworzenie i inicjalizacja ConfigManager

W klasie głównej tworzysz obiekt ConfigManager i go inicjalizujesz w onEnable():

private ConfigManager configManager;  
  
@Override  
public void onEnable() {  
 saveDefaultConfig();  
 this.configManager = new ConfigManager(this);  
  
 // Testowy odczyt wartości z configa  
 getLogger().info("Enabled: " + configManager.getBoolean("settings.enabled"));  
}

# 3. saveDefaultConfig()

Użyj saveDefaultConfig() w onEnable(), żeby utworzyć config.yml jeśli jeszcze nie istnieje.

# 4. Struktura plugin.yml

Upewnij się, że masz poprawną ścieżkę do klasy głównej w plugin.yml:

name: MojPlugin  
version: 1.0  
main: me.twojplugin.MyPlugin  
api-version: 1.20

# 5. Użycie ConfigManager w innych klasach (opcjonalne)

Możesz przekazywać ConfigManager do listenerów i innych klas, np.:

public class MojListener implements Listener {  
  
 private final ConfigManager config;  
  
 public MojListener(ConfigManager config) {  
 this.config = config;  
 }  
  
 @EventHandler  
 public void onJoin(PlayerJoinEvent event) {  
 String msg = config.getString("messages.welcome");  
 event.getPlayer().sendMessage(ChatColor.translateAlternateColorCodes('&', msg));  
 }  
}

Rejestracja w klasie głównej:

getServer().getPluginManager().registerEvents(new MojListener(configManager), this);

# 6. Komenda do przeładowania configa (opcjonalnie)

if (cmd.getName().equalsIgnoreCase("reloadconfig")) {  
 configManager.reload();  
 sender.sendMessage("Config przeładowany!");  
 return true;  
}

Dodatki do systemu configa: Cache, reload i zapis opcji

Jak mieć system, gdzie opcje w pluginie zależą od configa i mogą się dynamicznie zmieniać:

# 1. Cache wartości z configa

Nie czytaj configa za każdym razem z pliku, tylko wczytaj wartości do pól w klasie i korzystaj z nich:

private boolean featureEnabled;  
  
public void loadSettings() {  
 featureEnabled = configManager.getBoolean("features.coolFeature");  
}

Potem w kodzie odpalaj checki po polu featureEnabled.

# 2. Obsługa zapisu do configa

Przykład zapisu do configa (np. statystyk, zmiany ustawień) i zapisu do pliku:

configManager.getConfig().set("stats.kills", 10);  
configManager.getConfig().set("features.coolFeature", true);  
try {  
 configManager.getConfig().save(new File(plugin.getDataFolder(), "config.yml"));  
} catch (IOException e) {  
 e.printStackTrace();  
}

# 3. Metoda reload() z aktualizacją cache

W ConfigManager zrób reload configa i odśwież cache:

public void reload() {  
 plugin.reloadConfig();  
 loadSettings();  
}

W onEnable() też wołaj loadSettings() po inicjalizacji.

# 4. Event lub callback po reloadzie

Możesz zrobić listenerów, którzy dostaną info o reloadzie configa:

public interface ConfigReloadListener {  
 void onConfigReload();  
}  
  
private final List<ConfigReloadListener> listeners = new ArrayList<>();  
  
public void addListener(ConfigReloadListener listener) {  
 listeners.add(listener);  
}  
  
public void reload() {  
 plugin.reloadConfig();  
 loadSettings();  
 for (ConfigReloadListener listener : listeners) {  
 listener.onConfigReload();  
 }  
}

# 5. Sprawdzanie warunków dynamicznie

Trzymaj opcje w polach, udostępniaj getter i sprawdzaj stan zawsze po getterze:

public boolean isFeatureEnabled() {  
 return featureEnabled;  
}

W ten sposób masz pewność, że wszędzie masz aktualny stan opcji.