HomePlugin

Visão Geral

Este é um plugin para Minecraft desenvolvido usando a API Bukkit. O plugin permite que os jogadores salvem, teleportem e gerenciem locais específicos chamados "home". Ele utiliza um banco de dados MySQL para armazenar as informações de localização das homes.

Estrutura do Código

A classe principal `HomePlugin` estende `JavaPlugin`, que é a classe base para plugins Bukkit. Ela contém métodos para inicialização (`onEnable`), desativação (`onDisable`) e manipulação de comandos (`onCommand`).

Variáveis e Objetos

connection

- **Tipo:** Connection
- **Descrição:** Mantém a conexão com o banco de dados MySQL.

cooldowns

- **Tipo:** HashMap<UUID, Long>
- **Descrição:** Armazena os tempos de cooldown dos jogadores para usar o comando `/home`.

cooldownTime

- **Tipo:** long
- **Descrição:** Tempo de cooldown em milissegundos para o comando `/home`, lido do arquivo de configuração.

particleEffect

- **Tipo:** Particle
- **Descrição:** Efeito de partícula utilizado ao teleportar, lido do arquivo de configuração.

Métodos Principais

onEnable

- **Descrição:** Método chamado quando o plugin é ativado. Inicializa a configuração, estabelece a conexão com o banco de dados, e cria a tabela `homes` se ela não existir.
- **Detalhamento dos passos:**
    1. **Configuração:**
        - `saveDefaultConfig()`: Salva o arquivo de configuração padrão se ele não existir.
        - `cooldownTime = getConfig().getLong("cooldown", 60) * 1000;`: Lê o tempo de cooldown do arquivo de configuração, com um valor padrão de 60 segundos, e converte para milissegundos.
        - `particleEffectName = getConfig().getString("particle-effect", "FLAME");`: Lê o nome do efeito de partícula do arquivo de configuração, com um valor padrão de "FLAME".
        - Tenta definir `particleEffect` com o valor lido da configuração. Se o valor não for válido, usa o valor padrão `Particle.FLAME`.
    2. **Conexão com o Banco de Dados:**
        - Conecta ao banco de dados MySQL e cria o banco de dados `minecraft` se ele não existir.
        - Reestabelece a conexão ao banco de dados `minecraft`.
        - Loga mensagens de sucesso ou erro durante a conexão.
    3. **Criação da Tabela:**
        - Cria a tabela `homes` no banco de dados se ela não existir.

onDisable

- **Descrição:** Método chamado quando o plugin é desativado. Fecha a conexão com o banco de dados se ela estiver aberta.
- **Detalhamento dos passos:**
    - Verifica se a conexão com o banco de dados está aberta.
    - Fecha a conexão e loga uma mensagem de desconexão bem-sucedida.
    - Loga qualquer exceção que ocorra ao fechar a conexão.

onCommand

- **Descrição:** Método chamado quando um comando é executado. Lida com os comandos `/sethome`, `/home`, `/delhome` e `/homes`.
- **Parâmetros:**
    - `CommandSender sender`: Quem enviou o comando (pode ser um jogador ou console).
    - `Command command`: O comando enviado.
    - `String label`: O alias do comando.
    - `String[] args`: Argumentos do comando.
- **Retorno:** `boolean` indicando se o comando foi tratado com sucesso.

Comandos e sua Implementação

/sethome <nome>

- **Descrição:** Salva a localização atual do jogador com o nome fornecido.
- **Funcionamento:**
    1. Verifica se o comando foi enviado por um jogador.
    2. Verifica se o nome da home foi fornecido.
    3. Obtém a localização atual do jogador.
    4. Insere ou atualiza a localização da home no banco de dados.
    5. Envia uma mensagem ao jogador confirmando que a home foi salva.
- **Código relevante:**

```
if (command.getName().equalsIgnoreCase("sethome")) {
  if (args.length < 1) {
    player.sendMessage("Por favor especifique o nome da sua home entre
aspas");
    return true;
  }

  String homeName = args[0];
  Location location = player.getLocation();
  try (PreparedStatement statement = connection.prepareStatement(
      "REPLACE INTO homes (uuid, home_name, world, x, y, z) VALUES (?,
?, ?, ?, ?, ?)")) {
    statement.setString(1, playerUUID.toString());
    statement.setString(2, homeName);
    statement.setString(3, location.getWorld().getName());
    statement.setDouble(4, location.getX());
    statement.setDouble(5, location.getY());
    statement.setDouble(6, location.getZ());
    statement.execute();
    player.sendMessage("Home '" + homeName + "' salva com sucesso!");
  } catch (Exception e) {
    e.printStackTrace();
    player.sendMessage("Um erro ocorreu ao tentar salvar o home '" +
homeName + "'");
  }
  return true;
}
```

/home <nome>

- **Descrição:** Teleporta o jogador para a home salva com o nome fornecido.
- **Funcionamento:**

1. Verifica se o comando foi enviado por um jogador.
2. Verifica se o nome da home foi fornecido.
3. Verifica o cooldown do jogador.
4. Obtém a localização da home do banco de dados.
5. Teleporta o jogador para a home e exibe partículas.
6. Atualiza o cooldown do jogador.

- **Código relevante:**

```
if (command.getName().equalsIgnoreCase("home")) {
  if (args.length < 1) {
    player.sendMessage("Por favor especifique uma home para se
teleportar");
    return true;
  }

  String homeName = args[0];
  long currentTime = System.currentTimeMillis();
  if (cooldowns.containsKey(playerUUID) && (currentTime -
cooldowns.get(playerUUID)) < cooldownTime) {
    long remainingTime = (cooldownTime - (currentTime -
cooldowns.get(playerUUID))) / 1000;
    player.sendMessage("Você precisa aguardar " + remainingTime + "
segundos antes de usar esse comando novamente.");
    return true;
  }

  try (PreparedStatement statement = connection.prepareStatement(
      "SELECT world, x, y, z FROM homes WHERE uuid = ? AND
home_name = ?")) {
    statement.setString(1, playerUUID.toString());
    statement.setString(2, homeName);
    ResultSet results = statement.executeQuery();

    if (results.next()) {
      String world = results.getString("world");
      double x = results.getDouble("x");
      double y = results.getDouble("y");
      double z = results.getDouble("z");

      Location homeLocation = new Location(Bukkit.getWorld(world), x, y,
z);
      player.getWorld().spawnParticle(particleEffect, player.getLocation(),
100);
      player.teleport(homeLocation);
```

```
            player.getWorld().spawnParticle(particleEffect, homeLocation, 100);
            player.sendMessage("Teleported to home '" + homeName + "'.");

            cooldowns.put(playerUUID, currentTime);
        } else {
            player.sendMessage("Home '" + homeName + "' não existe use o
comando /sethome para salvar uma nova home.");
        }
    } catch (Exception e) {
        e.printStackTrace();
        player.sendMessage("um erro ocorreu ao teleportar para sua home.");
    }
    return true;
}
```

/delhome <nome>

- **Descrição:** Deleta a home com o nome fornecido.
- **Funcionamento:**
    1. Verifica se o comando foi enviado por um jogador.
    2. Verifica se o nome da home foi fornecido.
    3. Deleta a home do banco de dados.
    4. Informa ao jogador se a home foi deletada com sucesso ou se não existe.
- **Código relevante:**

```
if (command.getName().equalsIgnoreCase("delhome")) {
    if (args.length < 1) {
        player.sendMessage("Por favor especifique uma home para deletar");
        return true;
    }

    String homeName = args[0];
    try (PreparedStatement statement = connection.prepareStatement(
            "DELETE FROM homes WHERE uuid = ? AND home_name = ?")) {
        statement.setString(1, playerUUID.toString());
        statement.setString(2, homeName);
        int rowsAffected = statement.executeUpdate();
        if (rowsAffected > 0) {
            player.sendMessage("Home '" + homeName + "' deletada com
sucesso!");
        } else {
            player.sendMessage("Home '" + homeName + "' não existe.");
        }
    } catch (Exception e) {
        e.printStackTrace();
        player.sendMessage("Um erro ocorreu ao deletar sua home.");
    }
```

```
    return true;
}
```

/homes

- **Descrição:** Lista todas as homes salvas do jogador.
- **Funcionamento:**
    1. Verifica se o comando foi enviado por um jogador.
    2. Recupera os nomes das homes do banco de dados.
    3. Envia ao jogador uma mensagem listando suas homes.

- ***Código relevante:***

```
if (command.getName().equalsIgnoreCase("homes")) {
    try (PreparedStatement statement = connection.prepareStatement(
            "SELECT home_name FROM homes WHERE uuid = ?")) {
        statement.setString(1, playerUUID.toString());
        ResultSet results = statement.executeQuery();

        StringBuilder homesList = new StringBuilder("Suas homes salvas: ");
        boolean hasHomes = false;
        while (results.next()) {
            if (hasHomes) {
                homesList.append(", ");
            }
            homesList.append(results.getString("home_name"));
            hasHomes = true;
        }

        if (!hasHomes) {
            homesList.append(" none.");
        }

        player.sendMessage(homesList.toString());
    } catch (Exception e) {
        e.printStackTrace();
        player.sendMessage("Um erro ocorreu ao recuperar suas homes.");
    }
    return true;
}
```

Banco de Dados

Tabela *homes*

- **Descrição:** Armazena as informações das homes dos jogadores.
- **Estrutura:**
  - uuid VARCHAR(36): UUID do jogador.
  - home_name VARCHAR(255): Nome da home.
  - world VARCHAR(255): Nome do mundo.
  - x DOUBLE: Coordenada X.
  - y DOUBLE: Coordenada Y.
  - z DOUBLE: Coordenada Z.
  - **Primary Key:** (uuid, home_name).

Exceções e Tratamento de Erros

- **Conexão com o Banco de Dados:**
  - Se a conexão falhar durante a inicialização (onEnable), o plugin será desativado.
  - Exceções são logadas e mensagens de erro são enviadas ao console.
- **Execução de Comandos:**
  - Erros ao executar comandos são logados e uma mensagem de erro é enviada ao jogador.
  - Tratamento específico para SQLExceptions ao interagir com o banco de dados.