Developer Guide

Contains all the information relating to using the VirtualPlayers API.

Getting Started

VirtualPlayers comes with an API which can be used by developers. Follow the steps below to add the API as a dependency.

Adding the BattlePlugins Repository

Gradle (Kotlin DSL)

```
repositories {
    maven("https://repo.battleplugins.org/releases")
    maven("https://repo.battleplugins.org/snapshots")
}
```

Gradle (Groovy)

```
repositories {
    maven {
        url "https://repo.battleplugins.org/releases"
    }
    maven {
        url "https://repo.battleplugins.org/snapshots"
    }
}
```

Maven

```
<repositories>
  <repository>
    <id>battleplugins-releases</id>
    <name>BattlePlugins Releases</name>
    <url>https://repo.battleplugins.org/releases</url>
  </repository>
  <repository>
    <id>battleplugins-snapshots</id>
    <name>BattlePlugins Snapshots</name>
```

```
    <url>https://repo.battleplugins.org/snapshots</url>
  </repository>
</repositories>
```

Adding the Dependency

Gradle (Kotlin DSL)

```
dependencies {
    compileOnly("org.battleplugins.virtualplayers:api:3.0.0")
}
```

Gradle (Groovy)

```
dependencies {
    compileOnly "org.battleplugins.virtualplayers:api:3.0.0"
}
```

Maven

```
<dependencies>
  <dependency>
    <groupId>org.battleplugins.virtualplayers</groupId>
    <artifactId>api</artifactId>
    <version>3.0.0</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

VirtualPlayers API

The VirtualPlayers API can be accessed through the **VirtualPlayers** class, by calling **VirtualPlayers.api().**

A **VirtualPlayer** can be created using the **createVirtualPlayer(String)** method. A **VirtualPlayer** extends Bukkit's **Player** class, meaning all functions available to a normal **Player** will be available here. Keep in mind that some methods may not work as intended, since a **VirtualPlayer** is.. well... a virtual player at the end of the day, so functionality like sending block changes or adding them as a passenger to an entity will not work.

Within the **VirtualPlayer** class, you can also run **addObserver(CommandSender)** to observe a virtual player. You can also get a list of all observers by running **getObservers()**.

VirtualPlayers can be removed using the **removeVirtualPlayer(VirtualPlayer)** method in the **VirtualPlayers** API class. You can also retrieve a **VirtualPlayer** by calling **getVirtualPlayer(String)**, taking in the name of the **VirtualPlayer**.

For a full list of methods and classes offered by the API, see the VirtualPlayers **Javadocs**.