

The Event System

The event system is at the heart of BattleArena, and is how game logic and most features will be configured. Most classes you will extend (i.e. Arena, Competition, Map, etc.) do not have (m)any overridable methods, but instead use events.

The BattleArena event system is an extension of Bukkit's event system, which should be familiar to most people reading this documentation. However, it comes with a set of additional features that let you scope events for specific arenas.

Arena Events

When specifying an event for BattleArena, rather than using Bukkit's `@EventHandler` annotation, BattleArena's `@ArenaEventHandler` annotation is available, letting you listen for events that **only** happen in your arena:

```
public class MyArena extends Arena {

    @ArenaOption(name = "infection-time", description = "How long a player should be infected once hit.")
    private Duration infectionTime = Duration.ofSeconds(5);

    @ArenaEventHandler
    public void onInteract(PlayerInteractEvent event) {
        event.getPlayer().sendMessage("Interact while in Arena!");
    }
}
```

In this example, the `PlayerInteractEvent` is listened for, but what makes this different from using `@EventHandler` is this event will only be called for players **inside** a `MyArena`. This also means players in say a `Battlegrounds` arena would never see this message. If a random player playing survival were to click a block, nothing would happen, but if a player had ran `/myarena join` and started clicking blocks, you would see messages in console.

Custom Parameters

In addition to simply scoping out events per-Arena, BattleArena also lets you specify additional values in your event method that would otherwise not work in Bukkit. These two options are a **Competition** and an **ArenaPlayer**. For events that are per-player (i.e. `PlayerInteractEvent`), you

can specify both, however some events may not be per-player, but may only be called for an Arena (i.e. `ArenaPhaseStartEvent`). Here are two examples:

Player Event

```
@ArenaEventHandler
public void onInteract(PlayerInteractEvent event, ArenaPlayer player) {
    event.getPlayer().sendMessage(player.getPlayer().getName() + " interacted in arena: " +
    player.getArena().getName());
}
```

Arena Event

```
@ArenaEventHandler
public void onPhaseStart(ArenaPhaseStartEvent event, Competition<?> competition) {
    System.out.println("The phase " + event.getPhase().getType().getName() + " was started in competition " +
    competition.getMap().getName());
}
```

The second parameter is dynamic, meaning you can also specify the competition for your arena. So if you had a **SpleefCompetition** for instance, you could replace the **Competition<?>** reference with that.

Registering Listeners

By default, placing custom event listeners inside an **Arena** class will automatically register them for you without any additional code needed. However, if you wish to segregate this logic and instead have it in a separate class, a few things will need to be done.

Creating an ArenaListener

Rather than implementing Bukkit's listener, you will need to implement **ArenaListener**. Here is an example:

```
public class MyArenaListener implements ArenaListener {

    @ArenaEventHandler
    public void onInteract(PlayerInteractEvent event) {
        event.getPlayer().sendMessage("Interact while in Arena!");
    }
}
```

Registering Your Listener

In order to register your listener, you need to register it against the Arena in which you want to capture events for. In the example of MyArena, it is best done in the constructor like so:

```
public class MyArena extends Arena {

    @ArenaOption(name = "infection-time", description = "How long a player should be infected once hit.")
    private Duration infectionTime = Duration.ofSeconds(5);

    public MyArena() {
        super();

        this.getEventManager().registerEvents(new MyArenaListener());
    }
}
```

It is also important to note that if you are using standard `@EventHandler` annotations in your `MyArenaListener`, **they will not be registered using the above code**. You will need to run the standard **Bukkit#getPluginManager#registerEvents** method to do this. However, it is **not recommended** to mix these two together, and standard Bukkit events should be done in a separate listener, registered in the main class of your plugin.

Limitations

It is worth noting that not all events can be scoped by the `@ArenaEventHandler` annotation. This is mainly because the event system needs to have a pathway to extract an Arena player from an event. For instance, all events that are an instance of a **PlayerEvent** will be usable in BattleArena, since BattleArena can pull an ArenaPlayer from a standard Bukkit player. However, an event such as the **WeatherChangeEvent** cannot be referenced back to an Arena, because it is a global event not tied to a player. If you attempted to use the `@ArenaEventHandler`, nothing would happen.

Revision #3

Created 5 December 2024 18:02:19 by Redned

Updated 5 December 2024 18:33:31 by Redned