

Developer API

Use our robust API to put your unique ideas into practice.

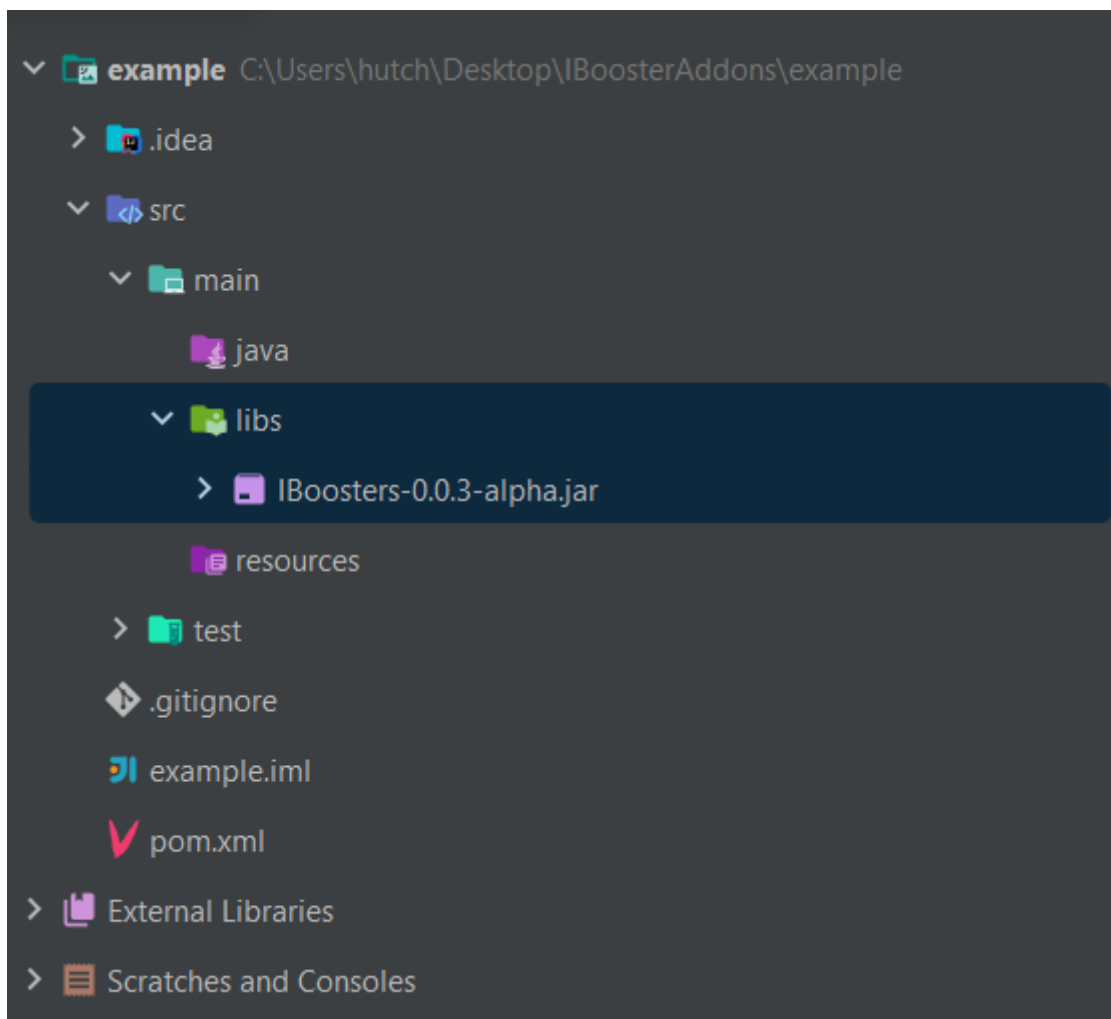
- [Creating an Addon](#)
- [Creating a Custom Booster \(Java\)](#)
- [Creating an Effect Booster \(Java\)](#)

Creating an Addon

Assuming you have already created your empty project, you will be required to create a new directory

called `libs` by right-clicking your `main` folder and clicking `New -> Directory`.

Cope and paste your IBoosters JAR file into that new folder.



Adding InsurgenceBoosters as a dependency

Add the below dependencies to your pom.xml file; replace `{JAR-NAME}` with the name of your IBoosters JAR file. A reference to the Spigot API is also included.

```

<dependencies>
  <dependency>
    <groupId>org. insurgencedev</groupId>
    <artifactId>IBoosters</artifactId>
    <version>LATEST</version>
    <scope>system</scope>
    <systemPath>${project.basedir}/src/main/libs/{JAR- NAME}</systemPath>
  </dependency>
  <dependency>
    <groupId>org. spigotmc</groupId>
    <artifactId>spigot- api</artifactId>
    <version>1. 20. 2- R0. 1- SNAPSHOT</version>
  </dependency>
</dependencies>

```

Setting up the main class

After creating your main class, it should extend `InsurgenceBoostersAddon` and override the necessary methods. The class should also be annotated with `@IboostersAddon`.

```

package com. insurgencedev. exampleaddon;

import org. insurgencedev. insurgenceboosters. api. addon. IBoostersAddon;
import org. insurgencedev. insurgenceboosters. api. addon. InsurgenceBoostersAddon;

@IBoostersAddon(name = "ExampleAddon", version = "1. 0. 0", author = "Insurgence Dev Team",
description = "Example Addon")
public class ExampleAddon extends InsurgenceBoostersAddon {

    @Override
    public void onAddonStart() {
    }

    @Override
    public void onAddonReloadablesStart() {
    }

    @Override

```

```

    public void onAddonReload() {
    }

    @Override
    public void onAddonStop() {
    }

}

```

Creating a custom config

Make a fresh class. It should extend the `AddonConfig` class and override the `onLoad()` function. To load and generate your configuration file based on the file in your internal resource folder, make sure the `loadAddonConfig()` method is used in the constructor.

```

package org.insurgencedev.exampleaddon;

import org.insurgencedev.insurgencesets.api.addon.AddonConfig;

public class CustomConfig extends AddonConfig {

    public static String MYSTRING = null;

    public CustomConfig() {
        loadAddonConfig("config.yml", "config.yml");
    }

    @Override
    protected void onLoad() {
        MYSTRING = getString("Test");
    }

}

```

Creating a Custom Booster (Java)

Below is an example of an Experience booster. When a player receives exp, the amount received will be multiplied based on the boosters that are active. We locate active personal and global boosters of the required `Type` and `Namespace`. If located, we apply the associated multipliers.

```
package com.insurgencedev.exampleaddon.listeners;

import org.bukkit.event.EventHandler;
import org.bukkit.event.Listener;
import org.bukkit.event.player.PlayerExpChangeEvent;
import org.insurgencedev.insurgencesboosters.api.IBoosterAPI;
import org.insurgencedev.insurgencesboosters.models.booster.GlobalBoosterManager;
import org.insurgencedev.insurgencesboosters.settings.IBoostersPlayerCache;

public final class ExperienceChangeListener implements Listener {

    @EventHandler
    private void onChange(PlayerExpChangeEvent event) {
        final String TYPE = "Experience";
        final String NAMESPACE = "CUSTOM";
        double totalMulti = 1;

        IBoostersPlayerCache.BoosterFindResult pResult =
        IBoosterAPI.getCache(event.getPlayer()).findActiveBooster(TYPE, NAMESPACE);
        if (pResult instanceof IBoostersPlayerCache.BoosterFindResult.Success boosterResult) {
            totalMulti += boosterResult.getBooster().getMultiplier();
        }

        GlobalBoosterManager.BoosterFindResult gResult =
        IBoosterAPI.getGlobalBoosterManager().findBooster(TYPE, NAMESPACE);
        if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success boosterResult) {
            totalMulti += boosterResult.getBooster().getMultiplier();
        }
    }
}
```

```
        event.setAmount((int) (event.getAmount() * totalMulti));
    }
}
```

Register your listener

For your listener class to take effect, it must be registered.

```
package com.insurgencedev.exampleaddon;

import com.insurgencedev.exampleaddon.listeners.ExperienceChangeListener;
import org.insurgencedev.insurgenceboosters.api.addon.IBoostersAddon;
import org.insurgencedev.insurgenceboosters.api.addon.InsurgenceBoostersAddon;

@IBoostersAddon(name = "ExampleAddon", version = "1.0.0", author = "Insurgence Dev Team",
description = "Example Addon")
public class ExampleAddon extends InsurgenceBoostersAddon {

    @Override
    public void onAddonReloadablesStart() {
        registerEvent(new ExperienceChangeListener());
    }

}
```

Creating an Effect Booster (Java)

Below is an example of how you can create a booster that applies effects to the player

Main Class

```
package com.insurgencedev.potioneffectsaddon;

import com.insurgencedev.potioneffectsaddon.listeners.BoosterListener;
import org.insurgencedev.insurgencesboosters.api.addon.IBoostersAddon;
import org.insurgencedev.insurgencesboosters.api.addon.InsurgencesBoostersAddon;

@IBoostersAddon(name = "PotionEffects", version = "1.0.0", author = "Insurgence Dev Team",
description = "Apply Potion Effects")
public class PotionEffectsAddon extends InsurgencesBoostersAddon {

    @Override
    public void onAddonStart() {
        new MyConfig();
    }

    @Override
    public void onAddonReloadablesStart() {
        registerEvent(new BoosterListener());
    }

}
```

Listener Class

```
package com.insurgencedev.potioneffectsaddon.listeners;
```

```

import com.insurgencedev.potioneffectsaddon.MyConfig;
import org.bukkit.entity.Player;
import org.bukkit.event.EventHandler;
import org.bukkit.event.EventPriority;
import org.bukkit.event.Listener;
import org.bukkit.event.player.PlayerItemConsumeEvent;
import org.bukkit.event.player.PlayerJoinEvent;
import org.bukkit.event.player.PlayerRespawnEvent;
import org.bukkit.potion.PotionEffectType;
import org.insurgencedev.insurgencesboosters.api.IBoosterAPI;
import org.insurgencedev.insurgencesboosters.events.IBoosterEndEvent;
import org.insurgencedev.insurgencesboosters.events.IBoosterStartEvent;
import org.insurgencedev.insurgencesboosters.libs.fo.Common;
import org.insurgencedev.insurgencesboosters.libs.fo.remain.CompMaterial;
import org.insurgencedev.insurgencesboosters.models.booster.GlobalBoosterManager;
import org.insurgencedev.insurgencesboosters.settings.IBoostersPlayerCache;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

public final class BoosterListener implements Listener {

    private final List<UUID> activeList = new ArrayList<>();
    private final String NAMESPACE = "CUSTOM_EFFECT";

    @EventHandler
    private void onStart(IBoosterStartEvent event) {
        if (event.getBoosterData().getType().equals(MyConfig.boosterType) &&
            event.getBoosterData().getNamespace().equals(NAMESPACE)) {

            Player player = event.getPlayer();

            if (MyConfig.disabledWorlds.contains(player.getWorld().getName())) {
                event.getBoosterData().setActive(false);
                return;
            }

            if (alreadyHasEffect(player)) {
                event.getBoosterData().setActive(false);
            }
        }
    }

```



```

        return;
    }

    activeList.add(player.getUniqueId());
    applyEffect(player);
}

}

@EventHandler
private void onEnd(IBoosterEndEvent event) {
    if (event.getBoosterData().getType().equals(MyConfig.boosterType) &&
        event.getBoosterData().getNamespace().equals(NAMESPACE)) {

        Player player = event.getPlayer();

        activeList.remove(player.getUniqueId());
        removeEffect(player);
    }
}

@EventHandler
private void onConsume(PlayerItemConsumeEvent event) {
    if (CompMaterial.fromItem(event.getItem()).equals(CompMaterial.MILK_BUCKET)) {
        Player player = event.getPlayer();

        IBoostersPlayerCache.BoosterFindResult pResult =
IBoosterAPI.getCache(player).findActiveBooster(MyConfig.boosterType, NAMESPACE);
        if (pResult instanceof IBoostersPlayerCache.BoosterFindResult.Success) {
            Common.runLater(2, () -> applyEffect(player));
            return;
        }

        GlobalBoosterManager.BoosterFindResult gResult =
IBoosterAPI.getGlobalBoosterManager().findBooster(MyConfig.boosterType, NAMESPACE);
        if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success) {
            Common.runLater(2, () -> applyEffect(player));
        }
    }
}
}

```

```

@EventHandler(priority = EventPriority.MONITOR)
private void onRespawn(PlayerRespawnEvent event) {
    Player player = event.getPlayer();

    IBoostersPlayerCache.BoosterFindResult pResult =
IBoosterAPI.getCache(player).findActiveBooster(MyConfig.boosterType, NAMESPACE);
    if (pResult instanceof IBoostersPlayerCache.BoosterFindResult.Success) {
        Common.runLater(2, () -> applyEffect(player));
        return;
    }

    GlobalBoosterManager.BoosterFindResult gResult =
IBoosterAPI.getGlobalBoosterManager().findBooster(MyConfig.boosterType, NAMESPACE);
    if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success) {
        Common.runLater(2, () -> applyEffect(player));
    }
}

@EventHandler(priority = EventPriority.MONITOR)
private void onJoin(PlayerJoinEvent event) {
    GlobalBoosterManager.BoosterFindResult gResult =
IBoosterAPI.getGlobalBoosterManager().findBooster(MyConfig.boosterType, NAMESPACE);
    if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success) {
        Player player = event.getPlayer();

        if (!activeList.contains(player.getUniqueId())) {
            activeList.add(player.getUniqueId());
            Common.runLater(2, () -> applyEffect(player));
        }
    }
}

@EventHandler(priority = EventPriority.MONITOR)
private void onLeave(PlayerQuitEvent event) {
    Player player = event.getPlayer();

    IBoostersPlayerCache.BoosterFindResult pResult =
IBoosterAPI.getCache(player).findActiveBooster(MyConfig.boosterType, NAMESPACE);
    if (pResult instanceof IBoostersPlayerCache.BoosterFindResult.Success) {
        removeEffect(player);
    }
}

```

```

        return;
    }

    GlobalBoosterManager.BoosterFindResult gResult =
    IBoosterAPI.getGlobalBoosterManager().findBooster(MyConfig.boosterType, NAMESPACE);
    if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success) {
        removeEffect(player);
    }
}

@EventHandler(priority = EventPriority.MONITOR)
private void onKick(PlayerKickEvent event) {
    Player player = event.getPlayer();

    IBoostersPlayerCache.BoosterFindResult pResult =
    IBoosterAPI.getCache(player).findActiveBooster(MyConfig.boosterType, NAMESPACE);
    if (pResult instanceof IBoostersPlayerCache.BoosterFindResult.Success) {
        removeEffect(player);
        return;
    }

    GlobalBoosterManager.BoosterFindResult gResult =
    IBoosterAPI.getGlobalBoosterManager().findBooster(MyConfig.boosterType, NAMESPACE);
    if (gResult instanceof GlobalBoosterManager.BoosterFindResult.Success) {
        removeEffect(player);
    }
}

private boolean alreadyHasEffect(Player player) {
    PotionEffectType type = PotionEffectType.getByName(MyConfig.effectNamespace);
    return type != null && player.hasPotionEffect(type);
}

private void applyEffect(Player player) {
    PotionEffectType type = PotionEffectType.getByName(MyConfig.effectNamespace);
    if (type != null) {
        player.addPotionEffect(type.createEffect(Integer.MAX_VALUE,
MyConfig.effectAmplifier - 1));
    }
}
}

```

```

private void removeEffect(Player player) {
    PotionEffectType type = PotionEffectType.getByName(MyConfig.effectNamespace);
    if (type != null) {
        player.removePotionEffect(type);
    }
}
}
}

```

Custom Config Class

```

package com.insurgencedev.potioneffectsaddon;

import lombok.Getter;
import org.insurgencedev.insurgencesboosters.api.addon.AddonConfig;

import java.util.List;

@Getter
public class MyConfig extends AddonConfig {

    public static String boosterType;
    public static String effectNamespace;
    public static int effectAmplifier;
    public static List<String> disabledWorlds;

    public MyConfig() {
        loadAddonConfig("config.yml", "config.yml");
    }

    @Override
    protected void onLoad() {
        boosterType = getString("Type");
        effectNamespace = getString("Effect");
        effectAmplifier = getInteger("Amplifier");
        disabledWorlds = getStringList("Disabled_Worlds");
    }
}

```

Internal Config file

Type: Strength

Effect: INCREASE_DAMAGE

Amplifier: 2

Disabled_Worlds:

- example