# Developers (API)

GWorld v2 offers a modern API for managing and creating worlds and changing settings (flags) programmatically.

Version v2.0.1


- 1. Setup & Integration
- 2. Create worlds (Builder)
- 3. Management & Control
- 4. Flags & Properties

# 1. Setup & Integration

To use GWorld v2 in your plugin, you must add it as a dependency.

# 1. Maven Dependency

Add the `core` module from GWorld to your `pom.xml`. Since the API is provided by the server at runtime, we use the `provided` scope.

```
<dependency>
  <groupId>de.gilljan</groupId>
  <artifactId>gworld-core</artifactId>
  <version>2.0.1-SNAPSHOT</version>
  <scope>provided</scope>
</dependency>
```

We use the GitHub Maven Package Registry. To use it, please refer to the GitHub documentation: https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-apache-maven-registry#installing-a-package

# 2. plugin.yml

To ensure that your plugin can access GWorld, it must be loaded *after* GWorld. To do this, add the entry to your `plugin.yml`:

```
depend: [GWorld]
# Alternatively, if GWorld is optional:
# softdepend: [GWorld]
```

# 3. Access to the API

The main entry point is the `GWorldAPI` interface. You can obtain the instance using the static method of the main class.

```
import de.gilljan.gworld.GWorld;
import de.gilljan.gworld.api.GWorldAPI;
import de.gilljan.gworld.api.IWorldManager;

public class MyPlugin extends JavaPlugin {

    private IWorldManager worldManager;

    @Override
    public void onEnable() {
        if (Bukkit.getPluginManager().isPluginEnabled("GWorld")) {
            // Retrieve API instance
            GWorldAPI api = GWorld.getInstance();

            // Load the manager for worlds
            this.worldManager = api.getWorldManager();
        }
    }
}
```

# 2. Create worlds (Builder)

In GWorld v2, new worlds are created using the **builder pattern**. This separates the **configuration** (registration) from the **generation** (load).

# Step 1: The Builder (Registration)

With the `WorldCreationBuilder` you define all the properties of the world. Calling `.build()` registers the world in the database, but does not yet physically create it.

```
import de.gilljan.gworld.api.IManageableWorld;
import de.gilljan.gworld.enums.WorldTypeMapping;

// Start the builder
IManageableWorld newWorld = worldManager.createBuilder("MeineEventWelt")
    .worldType(WorldTypeMapping.NORMAL) // Environent: Normal, Nether, End, Large Biomes,
Amplified, Flat
    .generator("PlotSquared")           // Optional: Custom Generator Name
    .seed(987654321L)                   // Optional: Set seed
    .build();                           // -> Saves the world to Config/DB
```

> **Note:** The `.build()` method internally calls `addWorldFromBuilder`. The world is then known ( `registered`), but still `unloaded`.

# Step 2: Load (generation)

```
// Creates the bukkit world and loads chunks
boolean success = newWorld.createMap();

if (success) {
  getLogger().info("World is ready!");
}
```

> The same applies to imports. Use the folder name of the world and use the `importMap()` method instead of `createMap()`.

# Why two steps?

1. **Performance:** You can register worlds when the server starts, but only load them when a player starts a minigame.
2. **Security:** You can set flags (e.g., PvP off) before the first player enters the world.

# 3. Management & Control

The `IWorldManager` manages all worlds, while the `IManageableWorld` object allows control over a specific world.

# Retrieve worlds

```
// Retrive a single world
Optional<IManageableWorld> worldOpt = worldManager.getWorld("Lobby");

// Retrive all worlds
List<IManageableWorld> allWorlds = worldManager.getWorlds();
```

# World actions

Each `IManageableWorld` object provides methods for control:

| Method | Description |
|---|---|
| `loadMap()` | Loads the world from the hard drive (Bukkit World Init). |
| `unloadMap()` | Unloads the world and teleports players to the main spawn. |
| `deleteMap()` | Irrevocably deletes the world (files & database entry). |
| `reCreate(boolean save)` | Deletes the world and regenerates it (reset). Optionally with backup (true) or without (false). |
| `clone(String name)` | Creates a copy of the world under a new name. |

# Example: Performing a world reset

```
worldManager.getWorld(“Farmworld”).ifPresent(world -> {
    // Reset world and keep copy of old world
    world.reCreate(true);
});
```

# Example: Remove world (from the system)

If you want to remove a world from GWorld (including deleting the files):

```
// Option A: Directly via the object (recommended)
world.deleteMap();

// Option B: Over the manager
worldManager.removeWorld(world);
```

# 4. Flags & Properties

GWorld allows you to change world settings (flags) programmatically. These settings are stored persistently.

# Set flags

You can change flags directly on the `IManageableWorld` object or set them during creation in the builder.

```
// Example: Disable PVP and set difficulty to hard
world.setAllowPvP(false);
world.setDifficulty(Difficulty.HARD);

// IMPORTANT: Save changes to disk. Ensures data integrity after a restart.
world.saveProperties();
```

# Available settings

Here is an overview of the most important methods in `IManageableWorld` interface:

| Category | Methods (Getter / Setter) | Description |
|---|---|---|
| **PVP** | `isAllowPvP`, `setAllowPvP` | Global PvP on/off. |
| **Spawning** | `isMonsterSpawning`, `setMonsterSpawning` | Spawning monsters. |
| | `isAnimalSpawning`, `setAnimalSpawning` | Spawning animals. |
| **Environment** | `isWeatherCycle`, `setWeatherCycle` | Whether the weather changes. |
| | `isTimeCycle`, `setTimeCycle` | Whether the time of day is advancing. |
| | `getTime`, `setTime` | Current time in ticks. |
| **Player** | `getGameMode`, `setGameMode` | Default game mode of the world. |
| **System** | `isLoadOnStartup`, `setLoadOnStartup` | Should the world be loaded when the server starts? |
| | `isAllowPvP`, `setAllowPvP` | Keep spawn chunks in RAM. |

# Use in the Builder

When creating a world, you can also set flags generically via `WorldProperty`:

```
import de.gilljan.gworld.data.properties.WorldProperty;


manager.createBuilder("Lobby")
    .property(WorldProperty.PVP, false)
    .property(WorldProperty.ANIMALS, false)
    .build();
```