Main Mod Class

Posted on June 19, 2018

In this tutorial, we'll be setting up the main mod class, the class that Forge loads your mod from. First, create a new package in `src/main/java`. This can be called anything you want, as long as it follows Java package naming conventions (https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html). Typically, the top-level package reflects a domain name that you own, but since `io.github.cubicoder.tutorialmod` is a bit long for me, I'll just call mine `cubicoder.tutorialmod`. **IMPORTANT: don't use a domain for your top-level package if you don't own it. You can use your username or the name of your mod, or really anything else.**

In `cubicoder.tutorialmod` (or your equivalent package), create a new class called `TutorialMod`. This will be our main mod class. Edit yours to look like this:

```
package cubicoder.tutorialmod;

import org.apache.logging.log4j.Logger;
import org.apache.logging.log4j.LogManager;

import net.minecraftforge.fml.common.Mod;
import net.minecraftforge.fml.common.Mod.EventHandler;
import net.minecraftforge.fml.common.event.FMLInitializationEvent;
import net.minecraftforge.fml.common.event.FMLPostInitializationEvent;
import net.minecraftforge.fml.common.event.FMLPreInitializationEvent;

@Mod(modid = TutorialMod.MODID, name = TutorialMod.NAME, version = Tutori
public class TutorialMod {

        public static final String MODID = "tutorialmod";
        public static final String NAME = "Tutorial Mod";
        public static final String VERSION = "0.0.1";
        public static final String MC_VERSION = "[1.12.2]";

        public static final Logger LOGGER = LogManager.getLogger(Tutorial

        @EventHandler
        public void preInit(FMLPreInitializationEvent event) {

        }

        @EventHandler
        public void init(FMLInitializationEvent event) {
                LOGGER.info(TutorialMod.NAME + "says hi!");
        }

        @EventHandler
        public void postInit(FMLPostInitializationEvent event) {

        }

}
```

There's a lot to go over here, so let's start with the `@Mod` line. `@Mod` is an annotation added by Forge to designate the main class of a mod.

`MODID` : The public, unique identifier of your mod. This can be up to 64 characters long, so don't make it too much of an abbreviation (e.g. tm instead of tutorialmod).

After you release your mod, you **cannot** change this, as Forge would consider it to be a completely separate mod.

`NAME` : The name of your mod.

`VERSION` : The version of your mod. Versioning systems vary between mods, but a popular one is semantic versioning (https://semver.org).

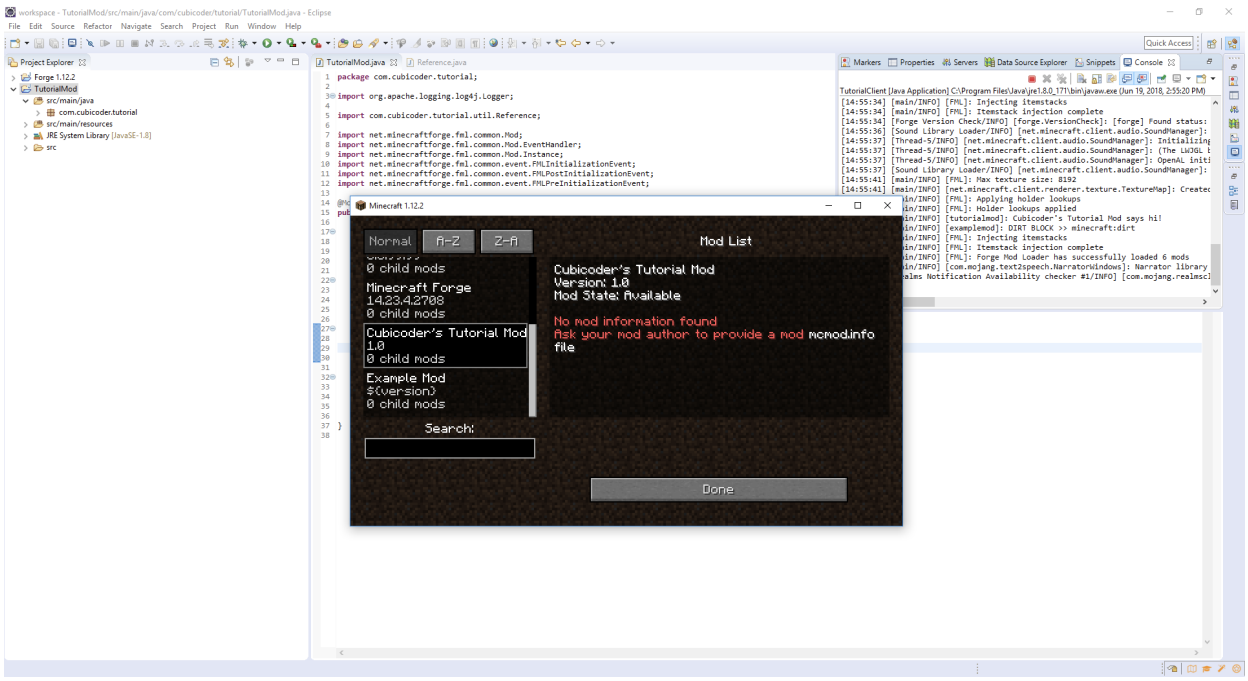
`MC_VERSION` : The Minecraft version(s) your mod is compatible with.

The `Logger` is used to log things that happen in your mod. Always use this instead of System.out.println(). This is mostly useful for debugging purposes.

The last things in our main mod class are the three Forge initialization methods: `preInit()`, `init()`, and `postInit()`. Forge initializes in three stages, and different parts of your mod must initialize at certain times to avoid conflict. For example, furnace recipes are registered in the `init()` method. If you tried to register them earlier, in `preInit()`, the game would probably crash, as the blocks/items you were trying to access for your recipes wouldn't have been created yet! For this and other reasons, it is very important to call everything in the right method for the right stage.

The `@EventHandler` annotation at the top of each initialization method simply lets Forge know that these methods are handling their corresponding event.

If you run this now, you should see a message appear in the console from your mod's logger and your mod should appear in the mods list. Read more about the main mod class in Forge's offical docs (https://mcforge.readthedocs.io/en/latest/gettingstarted/structuring/).



(/feed.xml) (https://github.com/cubicoder)

cubicoder • 2019 • Home (https://cubicoder.github.io)

Theme by beautiful-jekyll (https://deanattali.com/beautiful-jekyll/)