Guide contents

What tModLoader gives you

First mod

Adding more than one feature

When a build fails

Release checklist

A complete first project: the generated sword

Scope and sources

Sources / References

Documentation: In progress **Quick start: Complete**

Offline copy from ModDocs. Source links remain clickable.

Terraria modding

tModLoader is the starting point for making Terraria code and content mods. Use the tModLoader version that matches the game branch you intend to support.

Branch check (2026-10-06): the maintained basic guide describes a .NET 8 development setup, while the separate 1.4.5 migration draft describes .NET 10 and breaking API changes. Follow the guide, API reference and ExampleMod for the branch actually installed; do not combine signatures or build requirements across them.

> **Quick start:** generate a tModLoader mod skeleton, build and reload it unchanged, then add one sample item based on the current ExampleMod. Check it in game before adding NPCs, projectiles, or world systems.

What tModLoader gives you

tModLoader supplies the mod development workflow, a project skeleton, build/reload commands, and APIs for content and behavior. Start there instead of placing a random DLL in Terraria's directory. A content mod can include C# classes,

sprites, localization, and metadata; the generated example shows how those pieces fit together.

Goal	Where to begin
A new item or weapon	Generated sample item and ExampleMod
NPCs, projectiles, tiles, or recipes	The matching ExampleMod feature and API docs
A gameplay rule change	A small code mod with a narrowly scoped hook
Sharing a mod	Build, test, then package for the supported tModLoader branch

---

First mod

The tModLoader guide walks through generating a mod skeleton in its Develop Mods menu, editing a sample item, then using Build + Reload to test it. Begin with that generated project instead of guessing file layout or APIs. Keep a backup of your source as you expand it.

1. Install Terraria and a compatible tModLoader branch. Read the basic guide for the current development requirements.
2. In tModLoader, open **Workshop** → **Develop Mods** → **Create Mod**. Choose a unique internal name, display name, and author.
3. Open the generated source folder. Inspect the sample item, its image, the project file, `build.txt`, and localization file. These show what the build system expects.
4. Use **Build + Reload** before editing anything. Find the sample item in a test world. This proves the generated project and toolchain work.
5. Change one property of the sample item, rebuild, and verify the difference in-game. Then make a separate new item with its own code and sprite.

The current tModLoader wiki says a generated skeleton includes a central `Mod` class, a project file, build metadata, an item class and PNG, and localization data. Treat that list as version-specific; inspect the files your installed tool actually generated.

```text
ModSources/
  YourMod/
    build.txt
    YourMod.csproj
    Content/
      Items/
    Localization/
```

Adding more than one feature

Keep one feature type at a time. For an item, test its name, sprite, recipe, and behavior. For a projectile, test the item that creates it and its lifecycle. For an NPC, test spawn conditions and multiplayer behavior if the mod is meant for servers. The stable ExampleMod is the reference for the stable workflow; switch to another branch only when your installed tModLoader is on that branch. Copying code from a different branch can produce API errors that are hard to understand as a beginner.

Use a unique mod name and identifiers before releasing. Keep art and source files in the project, not only in a compiled output. Test a clean installation of the built mod so a file that exists only on your development PC does not go missing for players.

When a build fails

Read the first compiler error, including its file and line. If a type or method is missing, compare the example's branch to your installed tModLoader version. If the mod builds but an asset is invisible, check its path and naming against the generated sample. If it loads but behaves differently in a server, test the feature in a multiplayer session and consult the framework's networking guidance before adding more code.

- Basic tModLoader modding guide - setup and first mod.
- Stable ExampleMod source - examples for additional features; select the branch matching your tModLoader version.

Release checklist

- Record the tModLoader and Terraria versions used for testing.
- Build from the project source and install the resulting mod in a clean test environment.
- Check that sprites, localization, recipes, and any dependencies are present.
- Test single-player and multiplayer if you claim both are supported.
- Keep a source backup and explain known limitations to players.

A complete first project: the generated sword

The generated item is a real, buildable learning project. Create the skeleton through **Workshop** → **Develop Mods** → **Create Mod**, then inspect the sword class and sprite produced by your installed tModLoader. Use **Build + Reload** before editing. In a test world, obtain or craft the sword using the recipe from the generated example. The official basic guide walks through this same loop; the exact generated files may change by branch.

Now change **one** property in the sword's `SetDefaults` method, such as its damage, save, and Build + Reload again. Acquire a fresh sword and compare the value in game. If the change does not appear, confirm you edited the project tModLoader is building, not a copy of its output. Duplicate the item only after the original works: give the new class and its PNG unique matching names, add its localization, and verify it appears separately. Use ExampleMod for the matching branch's current API and asset conventions, not an old class signature copied from a forum post.

For release, build the mod once more, disable your source/workspace copy, install the packaged output as a player would, and enter a new test world. A sprite that only loads while source folders are present signals a missing or misnamed packaged asset. Try a multiplayer host-and-client run before claiming server compatibility.

Scope and sources

The generated sword, changed stat, Build + Reload test and clean-package check make a complete first tModLoader mod. This walkthrough is source-reviewed, not locally run in Terraria here. NPCs, projectiles and multiplayer systems need their own tests; the tModLoader basic guide and matching ExampleMod branch provide the exact APIs for those projects.

Continue with project and branch structure, content classes and hooks, or save and multiplayer state. Terraria's full documentation remains **In progress**.

Sources / References

External project and documentation links cited in this guide. Check the upstream release and version before following version-sensitive steps.

- basic guide
- 1.4.5 migration draft
- stable ExampleMod
- ExampleMod