

Softwareentwicklung

Interne Hinweise und Best Practices zur .NET-Softwareentwicklung.

- [Nuget Pakete](#)
- [Directory.Build.props / Directory.Build.targets](#)

Nuget Pakete

NuGet-Pakete

XML-Dokumentation / IntelliSense

Für Bibliotheken, die als NuGet-Paket veröffentlicht werden, sollten öffentliche APIs mit XML-Dokumentationskommentaren (`///`) beschrieben sein. Damit diese Kommentare in der IDE (IntelliSense) sichtbar sind, muss beim Build eine **XML-Dokumentationsdatei** erzeugt und mit dem Paket ausgeliefert werden.

Aktivierung im Build

In der Repository-`Directory.Build.props` (Root) ist das global gesetzt:

```
<GenerateDocumentationFile>true</GenerateDocumentationFile>
```

Der Compiler erzeugt dabei pro Assembly eine Datei `AssemblyName.xml` im Build-Output (neben der DLL).

NuGet-Verpackung

Beim `dotnet pack` nimmt das SDK die XML-Datei in der Regel **automatisch** in das Paket auf (gleicher Pfad wie die DLL unter `lib/<tfm>/`). Konsumenten erhalten damit die API-Dokumentation ohne zusätzliche Konfiguration.

Hinweise

Fehlende `///`-Kommentare

Mit aktivierter XML-Generierung meldet der Compiler Warnungen (z. B. CS1591). Für Bibliotheken sinnvoll; für Tests/Apps oft abschalten.

Tests, Apps, UI

In Unterordnern `GenerateDocumentationFile` auf `false` setzen, wenn keine API-Doku ausgeliefert wird (siehe [*Directory.Build.props*](#)).

Symbole

Unabhängig davon: `snupkg` /PDB für Debugging (z. B. `IncludeSymbols` in der Root-Props).

Directory.Build.props / Directory.Build.targets

Directory.Build.props

Automatischer Import durch MSBuild

MSBuild importiert beim Build **genau eine** `Directory.Build.props`: die **nächstgelegene** Datei, beginnend im Verzeichnis der `.csproj` und nach oben wandernd.

Wichtig: Liegt z. B. unter `src/Directory.Build.props` eine eigene Datei, wird die Root-`Directory.Build.props` **nicht** automatisch geladen. Einstellungen aus dem Repository-Root gelten dann nicht, solange sie nicht explizit verknüpft werden.

Verschachtelte Props: Parent importieren

In jeder untergeordneten `Directory.Build.props` den Parent **explizit** importieren:

```
<?xml version="1.0" encoding="utf-8"?>
<Project>
  <Import Project="$([MSBuild]::GetPathOfFileAbove('Directory.Build.props',
'$(MSBuildThisFileDirectory)../'))" />

  <!-- Lokale Ergänzungen oder Overrides -->
  <PropertyGroup>
```

```
<TargetFrameworks>net8.0;net10.0</TargetFrameworks>  
</PropertyGroup>  
</Project>
```

Directory.Build.targets

Dasselbe Prinzip gilt für **Directory.Build.targets**: nur die nächste Datei wird automatisch importiert. Bei verschachtelten **targets**-Dateien ebenfalls den Parent per **GetPathOfFileAbove** importieren.