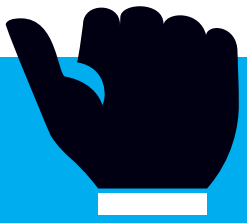
HIGH FIVE

TOOLS, TIPPS, SITES A2306HIGHFIVE



FÜNFMAL PROGRAMM- UND BIBLIOTHEKSANALYSE IN .NET

Sag mir, was du kannst

Per Reflection Informationen aus Assemblies erhalten und auf Controls zugreifen.

Quellcode und dokumentierte Programmierschnittstellen sind die Grundpfeiler der Softwareentwicklung. Was aber, wenn gewisse Funktionalitäten einer Assembly zur Entwicklungszeit noch nicht bekannt sind? Dann tritt .NET Reflection auf. Diese Technologie ermittelt interne Typinformationen und Metadaten einer Assembly. Sie nutzen Reflection zudem für Late Binding (spätes Binden) von Komponenten zur Laufzeit. Dabei müssen Sie allerdings in Kauf nehmen, dass anders als beim Early Binding (frühes Binden) der Compiler weder Optimierungen vornehmen kann noch Funktionen zur Autovervollständigung und für dynamische Hilfen bereitgestellt werden können.

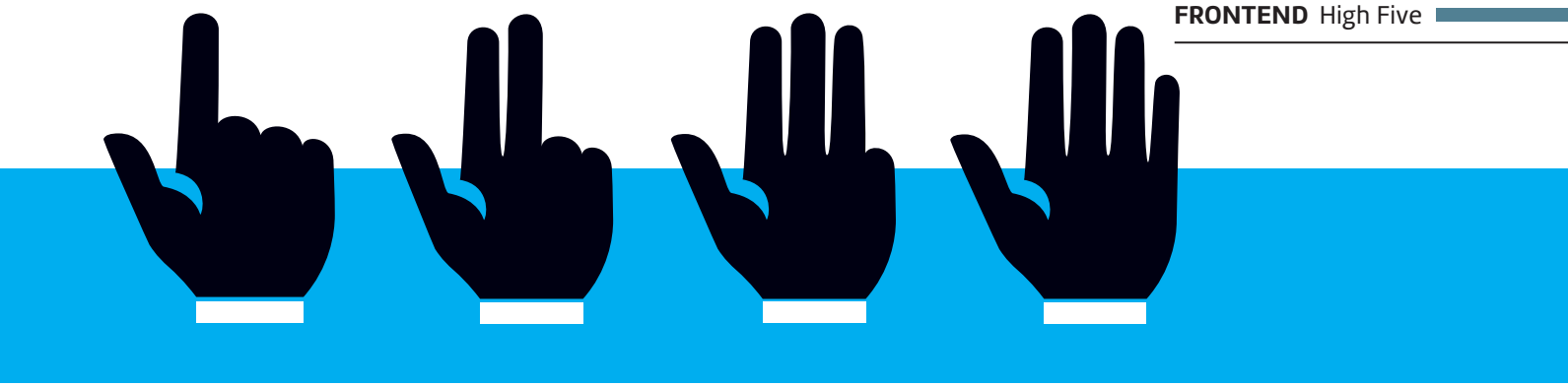
Beim Late Binding nutzen Sie für die Anbindung den allgemeinen Objektdatentyp *Object*. Dabei erhalten Sie lediglich Zugriff auf die öffentlich deklarierten Typmitglieder (Typen, Eigenschaften, Methoden und Ereignisse). Sie greifen per Reflection-API direkt auf Methoden und Eigenschaften zu. Die Funktionen zur Analyse und Nutzung der Assemblies stehen über den Namespace *System.Reflection* bereit. Spezielle Open-Source-Zusatzbibliotheken unterstützen Sie bei der Nutzung der Reflection-Funktionalität durch erweiterte Funktionalitäten oder Geschwindigkeitsoptimierungen.
www.dotnetpro.de/SL2306HighFive1

1 Cecil: Assembly-Analyse für Mono

Cecil beziehungsweise Mono.Cecil ist eine Bibliothek der Mono-Plattform und sowohl unter Mono als auch .NET nutzbar. Mono ist wiederum eine plattformübergreifende Open-Source-Variante zum .NET Framework für Windows, MacOS und Linux. Mit Cecil erzeugen und analysieren Sie Programme und Bibliotheken nach den ECMA-Standards (European Computer Manufacturers Association). Die Bibliothek bietet Reflection über ein leistungsfähiges Objektmodell. Die Analysefunktionen sind direkt nutzbar, ohne die Assemblies selbst laden zu müssen. Mithilfe von Cecil können Sie binäre .NET-Dateien ändern und Strukturen zu den Metadaten bearbeiten und erweitern. Auch der IL-Code (Intermediate Language) ist direkt bearbeitbar. Nach der Bearbeitung setzen Sie die geänderten und auf die Laufwerke zurückgeschriebenen Assemblies wie gewohnt ein. Auch Generics sowie Debug-Symbole werden bei der Verarbeitung unterstützt. Aktuell liegt die Bibliothek in der Version 0.11.4 vor und unterliegt der MIT-Lizenz. Weiterführende Informationen zur Bibliothek finden Sie im zugehörigen Wiki auf GitHub. Hier erhalten Sie Grundlagen zu den Codierungsrichtlinien und zu Programmübersetzungen. Diese erfordern als Grundlage .NET Core 3.1 oder höher und erlauben die Generierung von Anwendungen für .NET 4.0 und .NET Standard 2.0. Im Wiki finden Sie grundlegende Quelltextbeispiele zur Verwendung von Cecil in C. Viele Open-Source-Projekte, die sich der Implementierung von (De-)Compilern (zum Beispiel Visual Basic Compiler für Mono, ILSpy), Entwicklungsumgebungen wie SharpDevelop, der Anwendungsdokumentation, dem Testen, der Assembly-Manipulation sowie der Verfremdung von Programmen widmen, greifen auf die Cecil-Bibliothek zurück. Die Programmanbindung ist über NuGet-Pakete in Visual Studio möglich.
https://cecil.pe

FluentIL: Reflection mit fließendem Ansatz

FluentIL ist eine Open-Source-Bibliothek, die sich dem Thema Reflection mit einem Fluent-Ansatz widmet. Diese Bibliothek unterliegt der MIT-Lizenz und wird im Quelltext über GitHub bereitgestellt. Sie bietet Funktionen zum Definieren von Typen, Feldern, Eigenschaften, Methoden und Ereignissen sowie High-Level-Funktionen für die Anlage von Kontrollstrukturen (If, Else, For, Do und While). Die grundlegenden Funktionen werden anhand von C#-Quelltextbeispielen auf GitHub erläutert.
www.dotnetpro.de/SL2306HighFive6



2 ImmediateReflection: Programmanalyse

Die Open-Source-Bibliothek ImmediateReflection macht es sich zur Aufgabe, Reflection insbesondere in den Bereichen Konstruktoren und der Get- und Set-Zugriffsmethoden geschwindigkeitsoptimiert auszuführen. Dazu wird ein Object-Wrapper-Objekt eingeführt. Es gibt typisierte Delegaten sowie geschwindigkeitsoptimierte Typen, Member, Fields und Properties. Alle Quelltextbeispiele liegen für die Programmiersprache C vor. Die Bibliothek ist als NuGet-Paket in Visual-Studio-Entwicklungsprojekte einbindbar und unterliegt der MIT-Lizenz. Eine API-Referenz gibt es noch nicht, aber in der Online-Dokumentation finden Sie einen Schnelleinstieg, Codierungsrichtlinien sowie umfassende Benchmarks. Die Bibliothek hat keine Abhängigkeiten und adressiert .NET Standard 2.0 und höher, .NET Core 2.0 und höher sowie das .NET Framework 4.0 und höher.

www.dotnetpro.de/SL2306HighFive2

3 Namotion.Reflection: Erweitertes API

Namotion.Reflection ist eine erweiterte Reflection-Bibliothek aus dem Bereich Open Source, die mit C# umgesetzt wurde und in den Projekten NSwag und NJsonSchema zum Einsatz kommt. Die Bibliothek ist unter.NET Standard 1.0 und höher sowie dem .NET Framework 4.0 und höher einsetzbar und unterliegt der MIT-Lizenz. Die Erweiterungen der Bibliothek widmen sich dem Lesen von XML-Dokumentationen, der Unterstützung von Referenztypen, die den Nullwert annehmen können (Nullable Reference Types) in C# 8 und höher sowie den zeichenkettenbasierten Typüberprüfungen. Die Nutzung der Erweiterungen wird auf der GitHub-Projektseite jeweils praktisch in C-Syntax veranschaulicht.

www.dotnetpro.de/SL2306HighFive3

Faithlife.Reflection: Die optimierte .NET-Typanalyse

Die Open-Source-Bibliothek Faithlife.Reflection wurde mit C# realisiert, unterliegt der MIT-Lizenz und bietet Hilfsfunktionen für Reflection bei .NET-Typen an. Derzeit liegt der Fokus bei der Entwicklung in den Bereichen DTOs (Data Transfer Objects) und Tupeln. Die Online-Dokumentation bietet grundlegende Informationen, Quelltextbeispiele im C#-Format sowie eine kompakte Programmierreferenz.

www.dotnetpro.de/SL2306HighFive7

FasterReflect: Mehr Geschwindigkeit

Soll es bei Programm- und Bibliotheksanalysen per Reflection mal wieder schneller gehen, greifen Sie auf die Bibliothek FasterReflect zurück. FasterReflect bietet Erweiterungsmethoden zu Objekten, Typen und Metadaten und erweiterte Funktionalität gegenüber dem Namespace System.Reflection. Die Beschleunigung wird durch dynamische Code-Generierungen erreicht. Die Optimierungen liegen in den Bereichen Metadatenabfrage und Zugriffen auf Schnittstellenelemente sowie Funktionen. Erweiterte Funktionen werden für das Klonen von Objekten und die optimierte Objektanlage bereitgestellt. Die Bibliothek ist als NuGet-Paket verfügbar. Sie unterliegt der Apache-2.0-Lizenz.

www.dotnetpro.de/SL2306HighFive8

4 Liersch.Reflection: Der Spezialist

Liersch.Reflection ist eine Bibliothek, die sich der Geschwindigkeitsoptimierung bei Reflection-Operationen und den Typ- und Elementzugriffen zur Laufzeit widmet. Auch hier wird die Beschleunigung im Wesentlichen durch eine dynamische IL-Code-Generierung realisiert. Die Bibliothek konzentriert sich auf die (De-)Serialisierung von Objekten, beschleunigt das Instanzieren von Klassen, das Erzeugen von Wertetypen, das Aufrufen von Funktionen sowie den Zugriff auf Eigenschaften und Felder. .NET Standard 2.1 und höher, .NET Framework 4.0 und höher, .NET 5.0 und höher sowie .NET Core 2.0 und höher sind Voraussetzung. Die Bibliothek unterliegt der LGPL-2.1-Lizenz.

www.dotnetpro.de/SL2306HighFive4

5 Albedo: Abstraktionen und Hilfen

Die Open-Source-Bibliothek Albedo wurde ursprünglich von Mark Seeman entwickelt. Der Großteil der Umsetzung erfolgte mit C#, Teile mit F#. Mittlerweile hat die Organisation AlbedoOrg die Produktpflege übernommen. Die Bibliothek liegt derzeit in der Version 2.0.0 vor und hat sich das Ziel gesetzt, die Reflection-Programmierung insgesamt konsistenter und einfacher zu machen. Dazu werden eine Vielzahl an Abstraktionen und Utilities bereitgestellt. Die Quellen zur Bibliothek finden sich ebenso wie C#-Quelltextbeispiele auf GitHub.

Andreas Maslo

www.dotnetpro.de/SL2306HighFive5 ■