

ВИКОРИСТАННЯ РЕФЛЕКСІЇ В ПРОЦЕСІ РОЗРОБКИ ДОДАТКІВ НА ПЛАТФОРМІ .NET

ПОЗУР М.Ю., ВОЙТКО В.В.

mixalchik545@gmail.com, dekanfki@i.ua

Вінницький національний технічний університет

У статті розглянуто використання рефлексії в процесі розробки додатків на .NET. Розглянуто основні сценарії використання рефлексії. Описано загальні переваги та недоліки.

Сучасне програмне забезпечення є доволі складним. Як правило, воно складається з кількох модулів, кожен з яких має значний функціонал. Через це доволі гострим є питання часткової автоматизації розробки програмного забезпечення або більшого узагальнення його функціоналу.

Одним із варіантів вирішення таких задач є метапрограмування. Такий підхід дозволяє розробити додаток, який може зчитувати, аналізувати та модифікувати інші додатки та самого себе. Часто метапрограмування дозволяє мінімізувати кількість коду, необхідного для вирішення тієї чи іншої задачі, що, у свою чергу, зменшує час, необхідний для розробки програми [1].

Кожна мова програмування має свої інструменти для метапрограмування. У випадку з мовою програмування C# та платформою .NET основним засобом є рефлексія (Reflection).

Reflection – набір інструментів у платформі .NET для роботи з метаданими додатку. В .NET додатки компілюються в Microsoft Intermediate Language (MSIL) і разом з метаданими зберігаються в Portable Executable (PE) файлі. Таким чином механізм рефлексії має доступ до метаданих додатку та його коду під час виконання самого додатку. Це дозволяє отримувати інформацію про класи, їх методи та атрибути, зчитувати та змінювати значення полів класу, викликати методи динамічно та навіть додавати новий код у процесі виконання [2]. Приклад використання рефлексії зображено на рисунку 1.

```
var propertyName = nameof(SomeClass.Property1);  
  
var obj = new SomeClass();  
var typeData = obj.GetType();  
var property = typeData.GetProperty(propertyName);  
  
property.SetValue(obj, "new value");
```

Рисунок 1 – Приклад використання рефлексії на мові програмування C#

Існує чимало сценаріїв використання рефлексії в процесі розробки програмного забезпечення на .NET.

Найпоширенішим сценарієм є зчитування інформації про поля класу та зміна значень цих полів. Такий сценарій є найбільш актуальним у задачах серіалізації та десеріалізації даних та у випадках, коли необхідно перенести значення полів з одного об'єкта в інший (копіювання або мапінг). В обох випадках доцільним буде створити загальний функціонал, що може працювати з будь-якими типами даних.

Наступним доволі поширеним сценарієм використання рефлексії в .NET є доступ до приватних полів та методів класу. З точки зору концепції об'єктно-орієнтованого програмування це є порушенням, але трапляються випадки, коли це є необхідним. Частіше всього така необхідність виникає у випадку, коли використовуються сторонні бібліотеки. Для того, щоб змінити функціонал або виправити помилку в таких бібліотеках, як правило, необхідно мати доступ до «внутрішніх» елементів. На рисунку 2 зображено приклад використання рефлексії для доступу до приватних методів класу.

```

private static T CopyObject<T>(T obj) where T : class
{
    if (obj == null)
        return null;

    MethodInfo inst = obj.GetType()
        .GetMethod("MemberwiseClone",
            BindingFlags.Instance | BindingFlags.NonPublic);

    if (inst != null)
        return (T)inst.Invoke(obj, null);
    else
        return null;
}

```

Рисунок 2 – Використання рефлексії для доступу до приватного методу класу

Сфери застосування рефлексії в .NET можна значно розширити за рахунок використання атрибутів (Attribute) метаданих. Наприклад, це дозволить вибирати поля класу в залежності від того, чи мають вони певний атрибут, і в залежності від параметрів цього атрибута виконувати той чи інший код.

Не зважаючи на те, що рефлексія є доволі потужним механізмом платформи .NET, вона має свої недоліки.

Головним недоліком є те, що рефлексія працює під час виконання додатку, що робить неможливою її перевірку на етапі компіляції та може призвести до непередбачуваної поведінки додатку. Це потребує від розробника бути більш уважним під час написання коду, що використовує рефлексію.

Наступним значним недоліком є те, що рефлексія потребує більших витрат ресурсів системи в порівнянні з класичним підходом. Наприклад, зчитування та присвоювання значення через рефлексію може бути до 1000 разів повільнішим у порівнянні зі звичайним підходом [3].

Таким чином, рефлексія в .NET є ефективним механізмом, що дозволяє вирішувати більшість задач, які стоять перед розробником. Але перед її використанням завжди необхідно ретельно зважувати всі чинники та розглянути рішення, що не потребуватимуть її використання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Hazzard K. Metaprogramming in .NET / K. Hazzard, J. Brock., 2013. – 360 с.
2. Reflection in .NET [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/dotnet/framework/reflection-and-codedom/reflection>.
3. Why is reflection slow? [Електронний ресурс] – Режим доступу до ресурсу: <https://mattwarren.org/2016/12/14/Why-is-Reflection-slow>.