Reflection, bir programın çalışma zamanında kendisini incelemesine ve değiştirmesine olanak tanıyan bir mekanizmadır. Bu mekanizma, programların dinamik olarak türleri keşfetmesine, özelliklere ve metodlara erişmesine, örnekler oluşturmasına ve hatta kod çalıştırmasına olanak tanır. C# ve .NET platformunda reflection, genellikle `System.Reflection` ad alanı kullanılarak gerçekleştirilir.

### Reflection'ın Kullanım Alanları

1. \*\*Dinamik Tür Bilgisi Erişimi\*\*: Çalışma zamanında bir nesnenin tür bilgilerini almak ve bu türün özelliklerine, metodlarına ve alanlarına erişmek için kullanılır.

2. \*\*Dinamik Nesne Oluşturma\*\*: Çalışma zamanında tür adını bilmeden nesne oluşturma imkanı sağlar.

3. \*\*Özellik ve Metod Manipülasyonu\*\*: Nesnelerin özelliklerine ve metodlarına dinamik olarak erişme ve bunları değiştirme olanağı sunar.

4. \*\*Özellik ve Yöntem Keşfi\*\*: Bir sınıfın hangi özellikleri ve yöntemleri içerdiğini keşfetmek için kullanılabilir.

5. \*\*Özelleştirilmiş Object Serialization/Deserialization\*\*: Özel serileştirme ve deserileştirme işlemlerini gerçekleştirmek için kullanılabilir.

6. \*\*İşletim Zamanı Yüklenici Yönetimi\*\*: Derlenmiş bileşenleri veya eklentileri çalışma zamanında yüklemek ve yönetmek için kullanılır.

### Reflection Kullanımına Örnekler

#### 1. \*\*Assembly Yükleme ve Türleri Listeleme\*\*

```csharp

using System;

using System.Reflection;

class Program

{

static void Main()

{

// Assembly yükleme

Assembly assembly = Assembly.Load("mscorlib");

// Assembly içindeki tüm türleri listeleme

foreach (Type type in assembly.GetTypes())

{

Console.WriteLine(type.FullName);

}

}

}

```

#### 2. \*\*Bir Nesnenin Özelliklerine Erişme\*\*

```csharp

using System;

using System.Reflection;

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

}

class Program

{

static void Main()

{

Person person = new Person { Name = "John", Age = 30 };

// Person nesnesinin türünü elde etme

Type type = person.GetType();

// Tüm özellikleri listeleme ve değerlerini yazdırma

foreach (PropertyInfo property in type.GetProperties())

{

Console.WriteLine($"{property.Name} = {property.GetValue(person)}");

}

}

}

```

#### 3. \*\*Dinamik Olarak Metod Çalıştırma\*\*

```csharp

using System;

using System.Reflection;

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

}

class Program

{

static void Main()

{

Calculator calc = new Calculator();

// Calculator türünü elde etme

Type type = typeof(Calculator);

// Add metodunu elde etme

MethodInfo method = type.GetMethod("Add");

// Metodu dinamik olarak çağırma

object result = method.Invoke(calc, new object[] { 5, 3 });

Console.WriteLine(result); // Çıktı: 8

}

}

```

#### 4. \*\*Dinamik Nesne Oluşturma\*\*

```csharp

using System;

public class MyClass

{

public MyClass()

{

Console.WriteLine("MyClass instance created!");

}

}

class Program

{

static void Main()

{

// MyClass türünü elde etme

Type type = typeof(MyClass);

// MyClass türünden bir nesne oluşturma

object instance = Activator.CreateInstance(type);

// Çıktı: MyClass instance created!

}

}

```

### Reflection'ın Avantajları ve Dezavantajları

\*\*Avantajları:\*\*

- \*\*Esneklik ve Güç:\*\* Kodunuzu dinamik olarak inceleme ve değiştirme yeteneği sağlar.

- \*\*Yeniden Kullanılabilirlik:\*\* Genel ve esnek kod yazmanıza olanak tanır.

- \*\*Gelişmiş İşlevsellik:\*\* Özellikle framework geliştirme ve araç yaratma gibi ileri düzey senaryolarda çok kullanışlıdır.

\*\*Dezavantajları:\*\*

- \*\*Performans:\*\* Reflection, doğrudan kod çağrılarından daha yavaştır çünkü çalışma zamanında tür bilgilerini elde etmek ve metod çağrıları yapmak daha fazla işlem gerektirir.

- \*\*Güvenlik:\*\* Yanlış kullanılırsa güvenlik açıklarına neden olabilir.

- \*\*Bakım Zorluğu:\*\* Dinamik kod, statik koda göre daha zor anlaşılabilir ve bakım yapılabilir.

### Özet

Reflection, bir programın çalışma zamanında kendi yapısını incelemesine ve değiştirmesine olanak tanıyan güçlü bir araçtır. Ancak, performans ve güvenlik konularında dikkatli olunması gerektiği için yalnızca gerekli durumlarda ve dikkatle kullanılması önerilir.