Tehnici Avansate de Programare

Delegates, Attributes, Lambda Expressionis & Extension Methods

Petru Rebeja, Marius Apetrii

25 Martie 2021

Facultatea de Matematică
Universitatea Alexandru Ioan Cuza, Iași

Introducere

Recapitulare

- Principiile `SOLID`
- Modularizarea codului-sursă

Despre ce vom discuta azi

- Delegates
- Expresii lambda
- Atribute
- Extension methods

Delegates

Ce este un `Delegate`

- Un obiect care știe să apeleze o metodă[1]
- Diferențiem între:
  - Un `tip Delegate`
  - O `instanță Delegate`

[1] Joseph Albahari and Ben Albahari. 2012. C# 5.0 in a Nutshell: The Definitive Reference (5th. ed.). O'Reilly Media, Inc.

Tipuri `Delegate`

- Un tip `Delegate` definește semnătura metodelor pe care le poate apela o instanță a acestui tip.

Cum definim un tip Delegate:

```
delegate TResult Transformer<TItem, TResult>(TItem item);
```

Instanță `Delegate`

- Este slab echivalentă cu un `pointer la o funcție` din limbajul `C`.
- Este delegată de apelant cu invocarea propriu-zisă: apelantul invocă instanța tipului `Delegate` iar aceasta la rândul ei apelează metoda(ele) țintă[2].
- Decuplează apelantul de metoda-țintă.

Exemplu

```
var parse = new Transformer<string, int>(System.Convert.ToInt32);
var result = parse("2021");
```

[2] Joseph Albahari and Ben Albahari. 2012. C# 5.0 in a Nutshell: The Definitive Reference (5th. ed.). O'Reilly Media, Inc.

Instanță `Delegate`

Compilatorul ne oferă o sintaxă mai simplă pentru declararea instanțelor `Delegate`:

Exemplu

```
Transformer<string, int> parse = System.Convert.ToInt32;
var result = parse("2021");
```

`Delegate` vs interfață[3]

- Problemele pe care le rezolvă un `Delegate` pot fi rezolvate și cu o interfață.
- Alegem să folosim un `Delegate` atunci când:
  - Definirea și implementarea unei interfețe necesită să scriem mult *cod de umplutură*.
  - Trebuie să apelăm mai multe metode (`multicasting`).

[3] Joseph Albahari and Ben Albahari. 2012. C# 5.0 in a Nutshell: The Definitive Reference (5th. ed.). O'Reilly Media, Inc.

Multicast Delegates

- O instanță `Delegate` poate referenția mai multe metode.
- Acestea sunt apelate în ordinea în care au fost adăugate la instanță.
- Adăugarea unei metode la lista de invocare se face cu operatorul `+=`.
- Eliminarea din listă se face cu `-=`.

Exemplu

```
delegate void ReportProgress(int percentage, string status);

void ReportProgressToConsole(int percentage, string status){
    Console.WriteLine("Progress: {0}%", percentage);
    Console.WriteLine("Status: {0}.", status);
}

void ReportProgressToFile(int percentage, string status){
    System.IO.File.WriteAllText("status.txt",
        String.Format("{0}% Done. Status: {1}.", percentage, status));
}

static void Main(){
    var reportProgress = ReportProgressToConsole;
    reportProgress+=ReportProgressToFile;
    reportProgress(50, "Waiting...");
}
```

Expresii Lambda

Ce este o expresie lambda

- O metodă fără nume menită să înlocuiască o instanță `delegate`.
- O sintaxă mai simplă pentru declararea unei instanțe `delegate` ad-hoc.

Forma expresiei lambda

```
(parametri) => expresie | { bloc; }
```

Expresii lambda

Exemplu

```
delegate void ReportProgress(int percentage, string status);

static void Main(){
    ReportProgress p = (percentage, status)=>
        Console.WriteLine("{0}%. Status: {1}.", percentage, status)
}
```

Atribute

Ce sunt atributele

- Un mecanism pentru adăugarea de date suplimentare la elementele codului-sursă (metode, clase, parametri etc.).
- O clasă derivată din clasa abstractă `System.Attribute`.

Exemplu

Marcarea unui tip perimat

```
[Obsolete]
public class GodObject
{
}
```

Compilatorul va emite un mesaj de avertizare la întâlnirea unei referințe către tipul `GodObject`.

Alte exemple

- Serializare: atributele descriu relația de corespondență dintre un membru al clasei și un element XML,
- Securitate: atributele conțin cerințele necesare pentru ca apelantul să aibă acces la resursa decorată,
- Depanare: compilatorul ignoră metodele decorate cu `ConditionalAttribute`.

Parametrii atributelor

- Atributele pot avea două tipuri de parametri:
  1. Parametri de ordine (`positional`) — sunt parametrii constructorilor publici ai atributului,
  2. Parametri cu nume (`named`) — corespund câmpurilor și proprietăților publice ale atributului.

Parametrii atributelor — exemplu

```
public class XmlElementAttribute: Attribute
{
    public XmlElementAttribute(string elementName)
    {
        ElementName = elementName;
    }

    public string Namespace {get; set;}
}

[XmlElement(nameof(Student), Namespace="https://www.math.uaic.ro")]
public class Student
{
}
```

Aici `nameof(Student)` este parametru de ordine iar `Namespace="https://www.math.uaic.ro"` este parametru cu nume.

Metodele de extensiune

Ce sunt `extension methods`

- Metode care extind funcționalitatea unui tip existent fără să-l modifice.
- Metode statice definite într-o clasă statică dar care sunt apelate ca și metode ale unei instanțe.
- Primul parametru al metodei are modificatorul `this` și denotă tipul de date care va fi extins.

Exemplu

```
public static class DateTimeExtensions
{
    public static string ToIsoShortDate(this DateTime date)
    {
        return date.ToString("yyyy-MM-dd");
    }
}

static void Main()
{
    Console.WriteLine(DateTime.Today.ToIsoShortDate());
    // 2021-03-26
}
```

Avantajele metodelor de extensiune

- Îmbunătățesc lizibilitatea codului și cresc calitatea acestuia,
- Permit înlănțuirea apelurilor `Where().Select()`
- Decuplează codul-sursă.

Încheiere

Recapitulare

- Delegates
- Expresii lambda
- Atribute
- Extension methods

Vă mulțumesc!

Mulțumesc pentru atenție!