# Bài 10.3: C# delegate

- ✓ Tổng quan về delegate
- ✓ Các thao tác với delegate
- ✓ Multicast delegate
- ✓ Các delegate có sẵn
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

# Tổng quan về C# delegate

- ✓Phải làm thế nào nếu ta muốn truyền một phương thức vào làm đối số của một phương thức nào đó?
- ✓Cách mà C# xử lý các sự kiện, callback function?
- ✓Câu trả lời là sử dụng delegate.
- ✓Delegate là một kiểu dữ liệu đại diện cho các tham chiếu đến các phương thức có bộ tham số và kiểu trả về cụ thể.
- ✓Ta có thể định nghĩa các biến kiểu delegate giống như các kiểu dữ liệu khác và cho biến này tham chiếu tới một phương thức có dấu hiệu nhận biết tương tự như dấu hiệu của delegate ta đang sử dụng.
- ✓Các bước để làm việc với delegate gồm:
  - ✓ Khai báo một delegate.
  - ✓ Gán phương thức mục tiêu.
  - ✓ Gọi phương thức qua biến kiểu delegate.

# Thao tác khai báo delegate

- ✓Cú pháp khai báo một delegate:
  **[access modifier] delegate type DelegateName(params);**
- ✓Trong đó [access modifier] là phần cấp độ truy cập của delegate.
- ✓Keyword delegate là bắt buộc.
- ✓Phần type là kiểu trả về của delegate.
- ✓DelegateName là tên delegate. Đặt tên theo nhiều dạng khác nhau:
  - ✓ [method name]Delegate: CalculateDelegate;
  - ✓ [Task][State]Handler: UITaskFinishedHandler;
  - ✓ [Event]Handler: MouseClickHandler;
  - ✓ [Task]Callback: DataLoadedCallback;
- ✓Phần params là các tham số có thể có của delegate.

# Ví dụ

```
// delegate trả về kiểu void và nhận vào một tham số string
public delegate void MyDelegate(string msg);

// delegate nhận vào 2 số nguyên và trả về kết quả là một số nguyên
public delegate int CalculateDelegate(int a, int b);

// delegate nhận vào một list và trả về 1 giá trị
public delegate T FindMaxDelegate<T>(List<T> list) where T : IComparable<T>;
```

# Sử dụng delegate

- ✓Để sử dụng được delegate, gán tên phương thức cần gọi cùng dấu hiệu nhận biết cho delegate. Phương thức đích có thể là phương thức static hoặc instance method.
- ✓Ta có thể gán phương thức cụ thể, phương thức vô danh, local function, biểu thức lambda cho delegate.
- ✓Ta có thể gọi delegate qua cú pháp: delegateName(args); hoặc delegateName.Invoke(args);
- ✓Ví dụ:

```
// delegate nhận vào 2 số nguyên và trả về kết quả là một số nguyên
public delegate int CalculateDelegate(int a, int b);

1 reference
public static int Add(int a, int b)
{
    return a + b;
}

0 references
static void Main()
{
    int a = 200;
    int b = 300;
    CalculateDelegate calculateDelegate = Add;
    var sum = calculateDelegate(a, b);
    Console.WriteLine($"{a} + {b} = {sum}"); // 200 + 300 = 500
}
```

# Ví dụ

```
int a = 200;
int b = 300;
// sử dụng biểu thức lambda
CalculateDelegate addDelegate = (x, y) => x + y;
var sum = addDelegate.Invoke(a, b);
Console.WriteLine($"{a} + {b} = {sum}");
// gán phương thức vô danh cho delegate
CalculateDelegate substractDelegate = delegate (int x, int y)
{
    int s = x + y;
    return s;
};
var diff = substractDelegate.Invoke(a, b);
Console.WriteLine($"{a} - {b} = {diff}");
// sử dụng hàm cục bộ
int Mul(int m, int n)
{
    return m * n;
}
CalculateDelegate multiplyDelegate = Mul;
var product = multiplyDelegate(a, b);
Console.WriteLine($"{a} * {b} = {product}");
```

# Sử dụng delegate làm tham số

```
1 reference
public static int Add(int a, int b)
{
    return a + b;
}
1 reference
public static int Mul(int a, int b)
{
    return a * b;
}
2 references
public static void PrintResult(int a, int b, string op, CalculateDelegate cal)
{
    Console.WriteLine($"{a} {op} {b} = {cal(a, b)}");
}
0 references
public static void Main()
{
    int a = 200;
    int b = 300;
    // truyền delegate vào phương thức khác
    PrintResult(a, b, "+", Add);
    PrintResult(a, b, "*", Mul);
}
```

# Multicast delegate

- ✓Một biến delegate có thể tham chiếu đến nhiều phương thức có cùng dấu hiệu nhận biết với delegate đó tại 1 thời điểm.
- ✓Lúc này ta có multicast delegate.
- ✓Sử dụng toán tử +, += để thêm một phương thức vào danh sách các lời gọi mà delegate đang tham chiếu tới.
- ✓Sử dụng toán tử -, -= để loại bỏ một phương thức khỏi danh sách các lời gọi mà delegate đang tham chiếu.
- ✓Khi ta thực hiện lời gọi từ biến delegate, các phương thức trong danh sách lời gọi mà delegate này tham chiếu tới sẽ lần lượt được thực hiện theo đúng thứ tự mà nó được thêm vào tham chiếu của delegate.

# Ví dụ

```
public delegate void PrintMessageDelegate(string msg);

1 reference
public static void PrintMethodA(string msg)
{
    Console.WriteLine($"{msg} trong phương thức PrintMethodA");
}
1 reference
public static void PrintMethodB(string msg)
{
    Console.WriteLine($"{msg} trong phương thức PrintMethodB");
}
1 reference
public static void PrintMethodC(string msg)
{
    Console.WriteLine($"{msg} trong phương thức PrintMethodC");
}
0 references
public static void Main()
{
    Console.OutputEncoding = Encoding.UTF8;
    PrintMessageDelegate printMessage = PrintMethodA;
    printMessage += PrintMethodB;
    printMessage += PrintMethodC;
    printMessage("Learn C#");
}
```

```
C:\WINDOWS\system32\cmd.exe
Learn C# trong phương thức PrintMethodA
Learn C# trong phương thức PrintMethodB
Learn C# trong phương thức PrintMethodC
Press any key to continue . . .
```

# Ví dụ

```
1 reference
public static void PrintMethodB(string msg)
{
    Console.WriteLine($"{msg} trong phương thức PrintMethodB");
}
1 reference
public static void PrintMethodC(string msg)
{
    Console.WriteLine($"{msg} trong phương thức PrintMethodC");
}
0 references
public static void Main()
{
    Console.OutputEncoding = Encoding.UTF8;
    PrintMessageDelegate printMessage = PrintMethodA;
    printMessage += PrintMethodB;
    printMessage += PrintMethodC;
    printMessage -= PrintMethodA; // loại bỏ tham chiếu đến phương thức PrintMethodA
    printMessage("Learn C#");
}
```

```
C:\WINDOWS\system32\cmd.exe
Learn C# trong phương thức PrintMethodB
Learn C# trong phương thức PrintMethodC
Press any key to continue . . .
```

# Delegate generic

- ✓Delegate generic là delegate có kiểu là một tham số T nào đó chỉ được biết cụ thể khi ta cho nó tham chiếu đến tên một phương thức cụ thể.

```
1 reference
public static int Add(int a, int b)
{
    return a + b;
}
1 reference
public static int Mul(int a, int b)
{
    return a * b;
}
2 references
public static void PrintResult<T>(T a, T b, string op, SolveDelegate<T> cal)
{
    Console.WriteLine($"{a} {op} {b} = {cal(a, b)}");
}
0 references
public static void Main()
{
    int a = 200;
    int b = 300;
    // sử dụng delegate generic
    SolveDelegate<int> add = Add;
    SolveDelegate<int> mul = Mul;
    PrintResult(a, b, "+", add); // 200 + 300 = 500
    PrintResult(a, b, "*", mul); // 200 * 300 = 60000
}
```

# Các delegate có sẵn

- ✓Trong namespace System có sẵn 3 loại delegate generic để ta sử dụng:
  - ✓ Func: delegate cho phép tham chiếu đến các phương thức nhận vào 0 hoặc nhiều tham số và trả về một kết quả cụ thể khác void.
  - ✓ Action: tương tự Func ngoại trừ việc Action không trả về kết quả sau khi thực hiện lời gọi. Tức là nó dùng để tham chiếu tới các phương thức có kiểu trả về là void.
  - ✓ Predicate: là một loại delegate nhận vào 1 tham số và trả về kết quả true hoặc false. Sử dụng để kiểm tra một tính chất, điều kiện nào đó.
- ✓Ta thường kết hợp sử dụng biểu thức lambda với các delegate này.

# Ví dụ

```
0 references
public static void Main()
{
    // delegate đếm số từ trong câu
    Func<string, int> countWordFunc = (str) =>
    {
        var data = str.Split(' ');
        return data.Length;
    };

    // delegate hiển thị thông điệp nhận được
    Action<string> printMessage = message => Console.Write(message);

    // delegate kiểm tra các kí tự trong câu có phải chữ hoa không
    Predicate<string> checkUpperCase = str => str.CompareTo(str.ToUpper()) == 0;

    Console.OutputEncoding = Encoding.UTF8;
    var msg = "Welcome to Branium Academy";
    Console.WriteLine($"Số từ của câu \"{msg}\" là {countWordFunc(msg)}");
    Console.WriteLine($"\"{msg}\" chỉ chứa các chữ cái hoa? {checkUpperCase(msg)}");
    Console.WriteLine("Thông điệp cần hiển thị là: ");
    printMessage(msg);
}
```

```
C:\WINDOWS\system32\cmd.exe
Số từ của câu "Welcome to Branium Academy" là 4
"Welcome to Branium Academy" chỉ chứa các chữ cái hoa? False
Thông điệp cần hiển thị là:
Welcome to Branium AcademyPress any key to continue . . .
```

# Tóm tắt

- ✓Delegate là kiểu dữ liệu tham chiếu trong đó có xác định nghĩa dấu hiệu nhận biết.
- ✓Delegate cho phép ta sử dụng phương thức làm đối số truyền vào các phương thức khác.
- ✓Sử dụng delegate để tạo các phương thức gọi lại(callback), xử lý sự kiện.
- ✓Ta có thể liên kết chuỗi các delegate vào 1 delegate tạo thành multicast delegate.
- ✓Thường sử dụng kết hợp giữa lambda expression và delegate.

# Nội dung tiếp theo

## Tìm hiểu về Danh sách liên kết đơn