

CHƯƠNG 11

NEW FEATURES IN C#

Giảng viên: Ths. Phạm Văn Tiệp

Nội dung

- Thuộc tính tự động
- Khởi tạo đối tượng
- Biến cục bộ tự suy
- Kiểu nặc danh
- Phương thức mở rộng
- Khởi tạo danh sách


Thuộc tính tự động

```
public class Student  
{
```

```
    private String _Name;  
    public String Name  
    {  
        get  
        {  
            return _Name;  
        }  
        set  
        {  
            _Name = value;  
        }  
    }  
}
```

```
public class Student  
{
```

```
    public String Name { get; set; }  
}
```



Tự sinh trường
để lưu dữ liệu
của thuộc tính

Khởi tạo đối tượng

```
public class Student
{
    public String Name { get; set; }
    public Double Marks { get; set; }
}
```

Cung cấp giá trị cho các thuộc tính cần thiết lúc khởi tạo

```
public ActionResult Index()
{
    Student sv = new Student
    {
        Name = "Nguyễn Nghiệm",
        Marks = 9
    };
    return View();
}
```

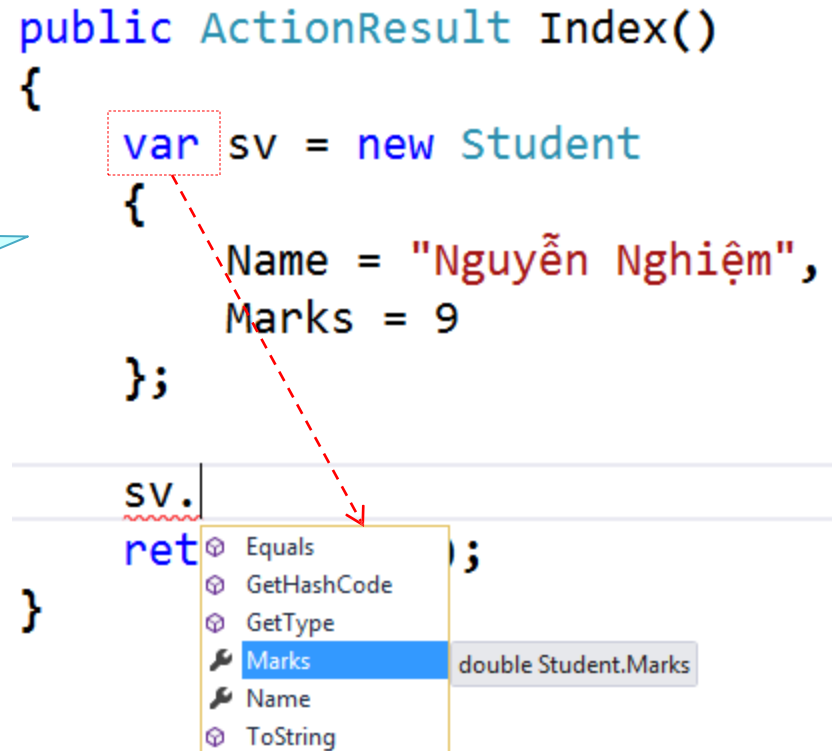
Biến cục bộ tự suy

```
public class Student
{
    public String Name { get; set; }
    public Double Marks { get; set; }
}
```

Tự nhận biết kiểu
thông qua giá trị
gán cho biến.

```
public ActionResult Index()
{
    var sv = new Student
    {
        Name = "Nguyễn Nghiệm",
        Marks = 9
    };

    sv.
    ret
}
```



double Student.Marks

Kiểu nặc danh

- ❑ Bạn có thể tạo đối tượng mà không cần định nghĩa lớp.
- ❑ Đối tượng có kiểu nặc danh không thể truyền cho view hoặc chia sẻ với các thành phần khác được

```
public ActionResult Index()  
{  
    var employee = new  
    {  
        Name = "Nguyễn Nghiêm",  
        Salary = 1000  
    };  
}
```

employee.

return Vi

}

- Equals
- GetHashCode
- GetType
- Name
- Salary
- ToString

int 'a.Salary

Anonymous Types:

'a is new { string Name, int Salary }

Phương thức mở rộng

- ❑ Bạn có thể viết các phương thức bổ sung cho một lớp đã tồn tại trước đó mà bạn không có mã nguồn.

Theo qui ước

```
public ActionResult Index()
{
    var name = "Nguyễn Nghiệm".T;
    return View();
}
```

- Split
- StartsWith
- Substring
- Sum<>
- Take<>
- TakeWhile<>
- ToArray<>
- ToBase64**
- ToCharArray

```
public static class XString
{
    public static String ToBase64(this String s)
    {
        byte[] data = Encoding.UTF8.GetBytes(s);
        return Convert.ToBase64String(data);
    }
}
```

**Lớp được bổ
sung phương
thức**

Khởi tạo List

```
var list = new List<Student>
{
    new Student {Name="Tuấn", Marks=5},
    new Student {Name="Hoa", Marks=7}
};
```

Danh sách có định kiểu

list[0].

- Equals
- GetHashCode
- GetType
- Marks** double Student.Marks
- Name
- ToString

Liệt kê các phần tử cách nhau bởi dấu phẩy

Danh sách không định kiểu

```
var list = new ArrayList
{
    new {Name="Tuấn", Marks=5},
    new {Name="Hoa", Marks=7}
};
```


Delegate

❑ Delegate là gì?

- Delegate là một kiểu tham chiếu dùng để bao bọc một phương thức có mẫu giống như delegate, sau đó chúng ta có thể thực thi phương thức bằng cách gọi delegate thay vì gọi phương thức (giống con trỏ hàm trong C)
- Delegate thường được sử dụng để tạo sự kiện và các hàm callback trong chương trình.

Sử dụng Delegate

Qua 3 bước:

1. Khai báo delegate
2. Khởi tạo
3. Thực thi

Khai báo Delegate

Cú pháp:

<chỉđịnhtừtruyxuất> **delegate** **<kiểutrảvề>** **têndelegate**(**danhsáchthamsố**);

Trong đó:

- **<chỉđịnhtừtruyxuất>**: là một trong các thuộc tính truy cập: private, public, protected, internal.
- **<kiểutrảvề>**: kiểu trả về của phương thức
- **têndelegate**: tên của delegate
- **danhsáchthamsố**: các tham số của phương thức

Ví dụ:

public **delegate** **int** MyDelegate(int p, int q);

Khởi tạo

Giả sử ta có một phương thức sau, có kiểu trả về và tham số tương ứng với delegate **MyDelegate** trên:

```
static int Cong(int a, int b)
{
    return a + b;
}
```

Khởi tạo Delegate theo 1 trong 2 cách sau:

//Cách 1

```
MyDelegate pheptinh = new MyDelegate(Cong);
```

//Cách 2

```
MyDelegate pheptinh = Cong;
```

Thực thi

Có hai cách để thực thi delegate:

- Coi delegate như một phương thức:

pheptinh(a, b)

- Coi như một đối tượng, bằng cách gọi phương thức Invoke():

pheptinh.Invoke(a, b)

Ví dụ

//1. Định nghĩa 1 delegate

```
public delegate int MyDelegate(int p, int q);
```

```
class Program
```

```
{
```

```
    //Khai báo hàm cộng static
```

```
    static int Cong(int a, int b)
```

```
    {
```

```
        return a + b;
```

```
    }
```

```
    static void Main(string[] args)
```

```
    {
```

```
        //2. Khởi tạo 1 delegate
```

```
        MyDelegate pheptinh = Cong;
```

```
        //3. Thực thi delegate "như là 1 phương thức".
```

```
        int ketqua = pheptinh(100, 200);
```

```
        Console.WriteLine("Ket qua: 100 + 200 = {0}", ketqua);
```

```
    }
```

```
}
```

Anonymous method

- Anonymous method là một khối code không có tên được khai báo với từ khóa delegate.
- Khi bạn tạo một phương thức mà tham số của nó là một kiểu delegate, lúc đó bạn có thể tạo một anonymous để truyền cho tham số của phương thức vừa tạo.
- Bạn không cần xác định kiểu trả về trong một phương thức nặc danh, nó được suy ra từ lệnh return bên trong thân phương thức nặc danh đó.

Ví dụ

```
public delegate int MyDelegate(int p, int q);  
MyDelegate pheptinh = delegate(int a, int b){return a + b;};
```


Lambda Expression

- **Lambda expression** là một cách viết rút gọn của **Anonymous method**.
- **Cú pháp:**
parameter-list => expression or statements

Ví dụ

Dùng Anonymous method

```
public delegate int MyDelegate(int p, int q);  
MyDelegate pheptinh = delegate(int a, int b){return a + b;};
```

Dùng lambda expression

```
public delegate int MyDelegate(int p, int q);  
MyDelegate pheptinh = (a, b) => {return a + b;}
```

Rút gọn một lambda expression

Start with an anonymous method

```
delegate(String text) { return text.Length; }
```



Convert to lambda expression

```
(string text) => { return text.Length; }
```



Single expression, so no braces required

```
(string text) => text.Length
```



Let the compiler infer the parameter type

```
(text) => text.Length
```



Remove unnecessary parentheses

```
text => text.Length
```