C# and .NET Framework
Bài 2: Hướng đối tượng trong C#

Đoàn Quang Minh
minhdqtt@gmail.com
http://www.VTPortal.net
Last update: 30. December 2006

Mục lục

- Kế thừa, hàm ảo
- Quá tải hàm
- Hàm tạo, hàm hủy
- Cấu trúc
- Quá tải toán tử
- Chỉ số
- Giao tiếp

Lớp và kế thừa (class & inherit)

- Định nghĩa một lớp: từ khóa class
- Kế thừa đơn giản: cú pháp.
  - Không hỗ trợ đa kế thừa
  - Không hỗ trợ phạm kế thừa (giảm phức tạp)
  - Bắt buộc phải kế thừa: lớp System.Object
- Quá tải hàm: cú pháp
  - Không hỗ trợ tham số mặc định
  - Hàm ảo: từ khóa virtual và override
- Hàm bị che
  - Lý do ra đời: cùng tên hàm nhưng khác ý nghĩa
  - Từ khóa new

Lớp và kế thừa (class & inherit)

- Hàm trừu tượng, lớp trừu tượng
  - Từ khóa abstract
  - Khác biệt với C++
- Lớp được đóng kín
  - Khái niệm
  - Từ khóa sealed
- Phạm vi truy cập
  - public, protected, private
  - internal, protected internal
- Lớp cục bộ
  - Là lớp mà mã lệnh của nó được đặt ở nhiều nơi.
  - Từ khóa partial

Thuộc tính (property)

- Thuộc tính
  - Là một phương thức hoặc một cặp phương thức, mà thể hiện của nó như là một trường dữ liệu
  - Cặp từ khóa get / set
  - Thuộc tính chỉ đọc, chỉ ghi
  - Phạm vi truy cập: thuộc tính get và set luôn có cùng phạm vi truy cập.
  - Thuộc tính trừu tượng: chỉ khai báo từ khoá mà không có thân hàm, do đó lớp kế thừa bắt buộc phải ghi đè
  - Thuộc tính ảo: cho phép lớp kế thừa ghi đè.

Lớp Object

- Là lớp cơ bản của .NET, mặc định mọi lớp nếu không nói gì thì hiểu là kế thừa từ Object
- Các phương thức của Object
  - public virutal string ToString()
  - public virutal int GetHashTable()
  - public virutal bool Equals(object obj)
  - public static bool Equals(object objA, object objB)
  - public static bool ReferenceEquals(object objA, object objB)
  - public Type GetType()
  - protected object MemberwiseClone()
  - protected virtual void Finalize()

Giao tiếp (interface)

- interface:
  - Một interface định nghĩa như một "hợp đồng", do đó, nếu một class hoặc một struct cài đặt một interface thì phải cài đặt tất cả các tính năng được khai báo trong interface đó.
  - Có thể hiểu interface như là một lớp trừu tượng hoàn toàn (tất cả các phương thức đều trừu tượng). Khi một class cài đặt một interface, thì coi như nó được kế thừa từ lớp trừu tượng kể trên
- Khai báo:
  - [*attributes*] [*modifiers*] **interface** *identifier* [**:***base-list*] **{***interface-body***}**[;]

Giao tiếp (interface)

- Đặc tính:
  - Một interface có thể là thành viên của một namespace hoặc một class.
  - Interface có thể chứa các thành viên sau:
    - Methods
    - Properties
    - Indexers
    - Events
  - Một interface có thể kế thừa từ một hay nhiều interface khác
- Tình huống thực tế:
  - Giả thiết chúng ta cần cung cấp chức năng Tìm kiếm cho hai loại đối tượng là văn bản và hình ảnh. Rõ ràng hai loại đối tượng này khác nhau, nên không thể có chung phương thức Tìm kiếm.
  - Sẽ đơn giản hơn nếu cả hai đối tượng này đều kế thừa interface ISearch: chúng ta có thể ép kiểu đối tượng về interface, việc gọi hàm Search() sẽ không phụ thuộc vào đối tượng ban đầu.

Giao tiếp (interface)

```
interface IPoint
{
    int x { get; set; }
    int y { get; set; }
}
class MyPoint : IPoint
{
    private int myX;
    private int myY;

    public MyPoint(int x, int y) {  myX = x;  myY = y;  }

    public int x { get { return myX; } set { myX = value; } }

    public int y { get { return myY; } set { myY = value; } }
}
```

Hàm tạo và hàm hủy (Construction and Disposal)

- Hàm tạo (Construction)
  - Định nghĩa và cú pháp: như C++
  - Khác biệt với C++: không nên khởi tạo biến thành viên trong hàm tạo.
  - Chú ý với hàm tạo có tham số: hãy luôn luôn có hàm tạo mặc định để tránh lỗi biên dịch.
- Hàm tạo tĩnh
  - Là hàm tạo, đồng thời là hàm tĩnh.
  - Được gọi khi sử dụng phương thức tĩnh của đối tượng.
- Gọi hàm tạo khi kế thừa
  - Thông qua từ khoá base.
  - Có thể truyền tham số cho lớp base.

Hàm tạo và hàm hủy (Construction and Disposal)

- Hàm hủy (Disposal)
  - Không quan trọng như C++, do bộ nhớ tự động được quản lý bởi bộ thu gom rác.
  - Nếu có định nghĩa, hàm hủy sẽ được gọi bởi bộ thu gom rác, nhưng không xác định được thời điểm gọi.
  - Có thể sử dụng giao tiếp IDisposable.
  - Hay dùng khi giải phóng các tài nguyên khác bộ nhớ (kết nối CSDL, tập tin,…)

Cấu trúc (Structs)

- Cấu trúc (struct)
  - Chỉ chứa các biến, không chứa phương thức
  - Khai báo và sử dụng cấu trúc: có thể dùng hoặc không dùng toán tử new
- struct và kế thừa
  - struct không thể kế thừa được.
  - Ngoại lệ: một struct coi như được kế thừa từ lớp Object
- Khởi tạo struct
  - Không thể khởi tạo các biến thành viên khi khai báo
  - Có thể có hàm tạo

Quá tải toán tử (Operator Overloading)

- Định nghĩa: như C++
- Cú pháp
- Ví dụ

Chỉ mục (Indexers)

- Mô tả:
  - Toán tử [ ] trong C# không thể quá tải được.
  - Chỉ mục là cách làm giống như việc quá tải toán tử [ ] trong C++, giúp cho việc truy cập vào một class hoặc một struct giống như truy cập vào một array.
  - Giống như thuộc tính, chỉ mục cũng gồm cặp phương thức get và set.
- *type* **this [***formal-index-parameter-list***]**
  - *type*: kiểu trả về
  - *formal-index-parameter-list*: danh sách các chỉ mục

Chỉ mục (Indexers)

- Ví dụ:
  - Giả sử có lớp Matrix (ma trận).
  - Khi dùng 2 chỉ số truy cập, ví dụ, a[i][j] sẽ nhận được một thành phần số. Nếu dùng 1 chỉ số truy cập, ví dụ, a[i] sẽ nhận được một vector

```
struct Matrix
{
   public double[][] x;
   public double this [uint i, uint j];
   public Vector this [uint i];
}
```

Tài liệu tham khảo

- Professional C#, Second Edition
- http://www.asp.net
- http://www.microsoft.com/net/default.mspx
- http://www.codeproject.com
- Địa chỉ download tài liệu http://www.thanglong.edu.vn/giang-day/tab.aspx
- Diễn đàn C# & .NET http://www.thanglong.edu.vn/forum/cmd/0/category/hoc-tap-nghien-cuu/dot-net/tab.aspx