C# and .NET Framework
Bài 1: C Sharp và kiến trúc .NET. C# cơ bản

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

Mục lục

- Mối quan hệ giữa C# và .NET
- The Common Language Runtime
- Assemblies
- Các ứng dụng sử dụng C#
- Nhắc lại kiến thức C# (cú pháp C++) căn bản

Mối quan hệ giữa C# và .NET

- C# là một ngôn ngữ lập trình mới nhất, hiện đại nhất, được thiết kế nhằm mục tiêu:
  - Phát triển, triển khai các ứng dụng trên nền .NET
  - Phát triển các ứng dụng dựa trên cách tiếp cận hướng đối tượng.
- *Chú ý*: C# là một ngôn ngữ lập trình, không phải là một thành phần của .NET. Tuy nhiên, hiệu quả trong lập trình với .NET phụ thuộc vào hiệu quả lập trình C#

The Common Language Runtime

- Hạt nhân của .NET là môi trường thực thi mã lệnh, gọi là **Common Language Runtime** (**CLR**) hay **.NET runtime**.
- Trước khi thực thi bởi CLR, mã nguồn cần được biên dịch ra ngôn ngữ trung gian gọi là **Microsoft Intermediate Language** (**MS-IL**)
- Ưu điểm:
  - Không phụ thuộc vào nền (platform): .NET có thể chạy trên Windows hoặc trên Linux;
  - Tăng hiệu suất thực thi: khác với Java, CLR là bộ biên dịch **Just-In-Time**: mã IL được biên dịch tức thời sang mã máy khi thực thi chương trình;
  - Khả năng biên dịch nhiều ngôn ngữ khác nhau: C#, VB.NET, C++ .NET, J#...

Xem thêm http://www.thanglongonline.net/forum/cmd/0/thread/28ddc5f6-acea-4190-a75f-076cb24e13e6/tab.aspx

Intermediate Language

- Kiểu dữ liệu dùng chung (**Common Type System – CTS**):
  - Là tập các kiểu dữ liệu được định nghĩa sẵn trong IL, dùng chung giữa các ngôn ngữ (C#, VB.NET,…).
  - Chia làm hai tập: dữ liệu kiểu tham trị và dữ liệu kiểu tham chiếu (**value type** và **reference type**)
- Đặc tả ngôn ngữ chung (**Common Language Specification – CLS**)
  - Tập con của CTS mà tất cả các ngôn ngữ đều phải hỗ trợ
- Đặc tính của IL:
  - Hỗ trợ hướng đối tượng và giao tiếp: bản thân IL là ngôn ngữ hướng đối tượng hoàn toàn. Có hỗ trợ giao tiếp (interface)
  - Xử lý tự động tham biến và tham trị (tính năng của CTS).
  - Ép kiểu tự động và an toàn
- Các tính năng đặc biệt:
  - Bộ thu gom rác: tự động quản lý bộ nhớ
  - An toàn: quản lý tiến trình dựa trên tài khoản

Assemblies

- Là một đơn vị logic chứa các mã đã được biên dịch bởi .NET.
- Chứa một metadata tự mô tả:
  - Type metadata: chứa các mô tả về kiểu dữ liệu và các phương thức có trong assembly
  - Manifest: chứa các assembly metadata là thông tin về assembly đó (version, copyright,…)
- Private Assemblies: gói này chỉ được thực thi bởi phần mềm sở hữu nó, thuộc thư mục hiện thời hoặc thư mục con.
- Shared Assemblies: chia sẻ giữa các phần mềm
  - Có thể gây nhập nhằng về tên gói
  - Có thể bị ghi đè bởi phiên bản mới hơn

.NET Framework Classes

- Các lớp cơ bản được cung cấp bởi Microsoft, cho phép thực thi hầu hết các tác vụ thường gặp
  - Các thủ tục triệu gọi đơn giản.
  - Cho phép truy xuất đến các hàm Windows API một cách đơn giản
- Namespaces: không gian tên, chứa một nhóm các kiểu dữ liệu hoặc lớp có mối liên quan qua lại với nhau.

Các ứng dụng sử dụng C#

- ASP.NET: là một công nghệ của Microsoft dùng để xây dựng website.
  - Là một hệ thống có cấu trúc;
  - Tốc độ thực thi nhanh;
  - Dễ dàng bảo trì mã lệnh.
- Web Forms: cách tiếp cận design các trang web tương tự như design các ứng dụng trên windows
- Web Services: cung cấp các dịch vụ cho website

Các ứng dụng sử dụng C#

- Windows Forms: hướng tiếp cận dành cho việc lập trình các ứng dụng trên nền Windows. Có cấu trúc tương tự Visual C++ hay VB 6
- Windows Services: chạy nền dưới dạng dịch vụ, đáp ứng các sự kiện không được kích hoạt trực tiếp bởi người dùng. Các dịch vụ này có thể được xây dựng đơn giản dựa vào .NET

C# cơ bản

- C# có nhiều nét tương đồng với C++, bao gồm các từ khóa, kiểu dữ liệu, cú pháp,…
- Cú pháp cơ bản
  - Biến: khai báo và sử dụng
  - if…else, switch
  - for
  - while, do…while
  - foreach
  - goto, break, continue, return

C# cơ bản

- Lớp (class)
  - Biến thành viên
  - Thuộc tính
  - Phương thức
- Phương thức
  - Cú pháp
  - Phạm vi
  - Cách triệu gọi
  - Truyền tham số
  - Từ khóa out

C# cơ bản

- Mảng: khai báo và sử dụng
- Toán tử
- Ép kiểu an toàn
- Kiểu liệt kê
- Không gian tên: từ khóa using
- Vào ra dữ liệu dựa trên console
- Chú thích mã lệnh

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

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

C# and .NET Framework
Bài 3: Lập trình nâng cao trong C#

Đoàn Quang Minh
minhdqtt@gmail.com
http://www.VTPortal.net
Last update: 21. Dec 2006

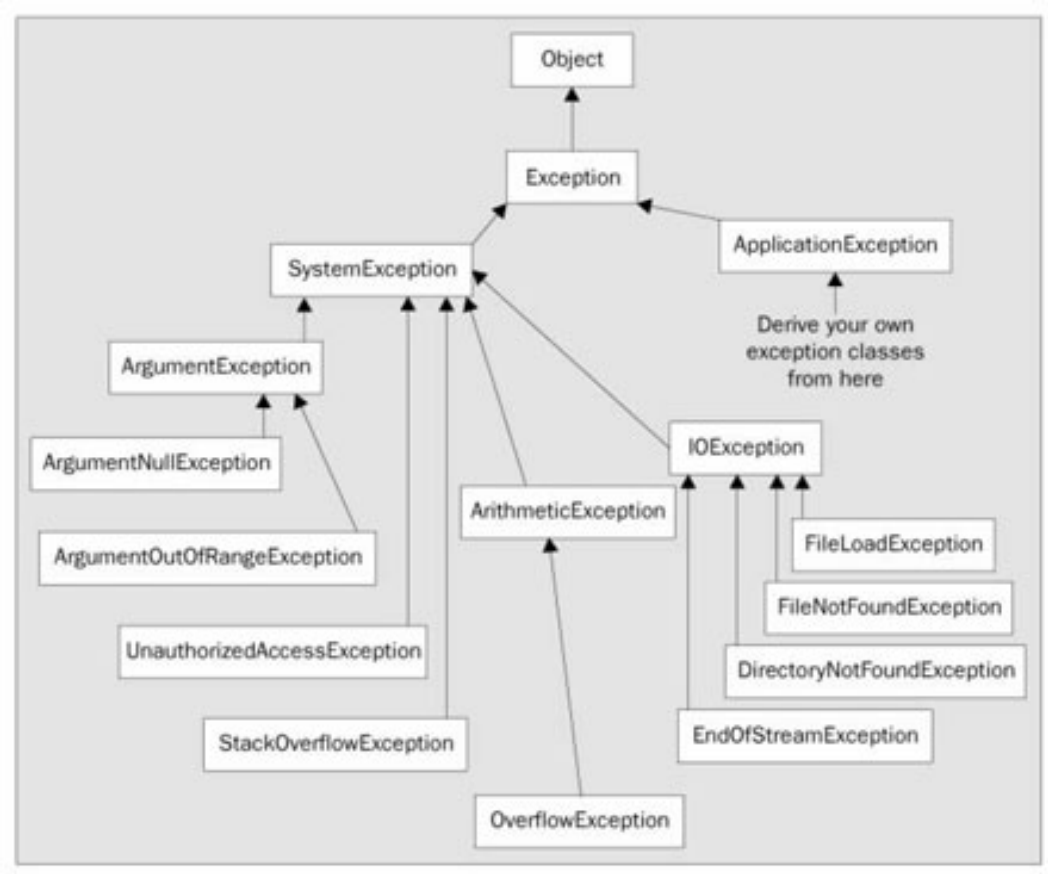
Mục lục

- Exceptions
- User-Defined Casts
- Delegates
- Events
- Generics
- Preprocessor Directive
- Unsafe code

Errors and Exception Handling

- Lỗi luôn luôn tồn tại, cho dù hệ thống được thiết kế tốt thế nào
  - Lỗi không được quyền truy cập
  - Lỗi do thiết bị hỏng (đía lỗi)
  - Lỗi do đường truyền mạng hỏng
- Khi một lỗi xuất hiện, .NET sẽ ném ra một ngoại lệ
  - Các ngoại lệ đều kế thừa từ lớp Exception
  - Tất cả các ngoại lệ cơ bản đều được cung cấp bởi .NET
  - Nếu gặp lỗi, chúng ta nên ném ra một ngoại lệ đặc biệt trong đó có mô tả thông tin rõ ràng về ngoại lệ đó
  - Nếu không tìm được lớp ngoại lệ phù hợp, có thể xây dựng lớp ngoại lệ của riêng mình

Errors and Exception Handling



Errors and Exception Handling

```
try
{
    …
}
catch (Exception e)
{
    …
}
finally
{
    …
}
```

Errors and Exception Handling

- Phần *try* thực thi các lệnh bình thường
- Phần *catch* xử lý các ngoại lệ có thể xuất hiện
  - Nếu không sử dụng ngoại lệ ném ra, có thể bỏ qua phần đối tượng đó.
  - Có thể có nhiều phần *catch* trong một khối *try catch*, khi đó mỗi phần catch xử lý một ngoại lệ khác nhau.
  - Ngoại lệ có thể được ném lại bằng từ khóa *throw*.
- Phần *finally* thực thi các lệnh kết thúc của khối lệnh.
- Ví dụ trong việc xử lý tập tin
  - Phần *try* thực thi các lệnh như mở file, đọc ghi bình thường
  - Phần *catch* xử lý lỗi.
  - Phần *finally*, nếu file được mở thì phải đóng lại

User-Defined Casts

- Toán tử ép kiểu *as*
- Chúng ta thường xuyên phải ép kiểu trong C#
  - Có 2 loại ép kiểu trong C#: implicit (an toàn tuyệt đối), explicit(có rủi ro)
  - Có thể ép kiểu từ lớp kế thừa về lớp cơ sở, song không thể làm ngược lại
  - Có thể ép kiểu qua lại giữa 2 lớp, khi đó cần khai báo hàm ép kiểu

public static implicit operator *conv-type-out* **(** *conv-type-in operand* **)**

public static explicit operator *conv-type-out* **(** *conv-type-in operand* **)**

Delegates

- delegate
  - Có những công việc không xác định lúc biên dịch, chỉ xác định lúc thực thi.
  - Các thuật toán tổng quát, ví dụ sắp xếp: không thể định nghĩa phương thức so sánh 2 đối tượng bất kỳ
  - delegate là kiểu tham chiếu, giống như class (trong C#), về ý nghĩa giống con trỏ hàm trong C++

[*attributes*] [*modifiers*] **delegate** *result-type identifier* **(**[*formal-parameters*]**);**

Delegates

```
delegate void MyDelegate(int i);

class Program
{
  public static void Main()
  {
    TakesADelegate(new MyDelegate(DelegateFunction));
  }

  public static void TakesADelegate(MyDelegate SomeFunction)
  {
    SomeFunction(21);
  }

  public static void DelegateFunction(int i)
  {
    System.Console.WriteLine("Called by delegate with number: {0}.", i);
  }
}
```

Events

- Sự kiện
  - Được sử dụng để báo hiệu một điều gì đó xảy ra.
  - Trong Windows, có rất nhiều sự kiện.
  - Trong C#, event là một dạng đặc biệt của delegate

  [*attributes*] [*modifiers*] **event** *type declarator*; [*attributes*] [*modifiers*] **event** *type member-name* {*accessor-declarations*};
- Phát sinh sự kiện
  - Định nghĩa tham số sự kiện, đặt tên là *EventName*EventArgs, kế thừa từ System.EventArgs.
  - Định nghĩa một delegates cho sự kiện, đặt tên là *EventName*EventHandler.
  - Phát sinh sự kiện
    - Khai báo sự kiện
    - Khai báo một phương thức On*EventName* để phát sinh sự kiện

Events

- Xử lý sự kiện
  - Nếu một component phát sinh một sự kiện, có thể bắt và xử lý sự kiện đó.
  - Để hander sự kiện trong Windows Form hoặc trong Web Form:
    - Khai báo component (ví dụ button)
    - Khai báo hàm xử lý sự kiện
    - Gắn hàm vào sự kiện

Generics

- Generics
  - Cho phép class, struct, interface, method sử dụng kiểu dữ liệu mà nó lưu trữ như là tham số đầu vào.
  - Khái niệm giống như template của C++.
  - Generics xuất hiện nhằm mục đích xử lí chính xác kiểu của dữ liệu. Ví dụ, với stack, nếu không có generics thì dữ liệu coi như các object, nên đòi hỏi phải ép kiểu khi xử lí, điều này có thể gây lỗi run-time.
- Tạo và sử dụng Generics
  - Khai báo giống như template trong C++: dùng cặp dấu < >
  - Sử dụng: phải chỉ định chính xác kiểu dữ liệu

Generics

- Ví dụ sử dụng Generics
  - Khai báo:

```
public class Stack<ItemType>
{
    private ItemType[] items = new ItemType[100];
    public void Push(ItemType data) {…}
    public ItemType Pop() {…}
}
```

  - Sử dụng

```
Stack<int> s = new Stack<int>();
s.Push(3);
int x = s.Pop();
```

Preprocessor Directives

- #define and #undef
- #if, #elif, #else, and #endif
- #warning and #error
- #region and #endregion
- #line

Memory Management

- C# tự động quản lý bộ nhớ nhờ vào bộ thu gom rác
  - Bộ nhớ ảo trong Windows
  - Stack và heap
- Có 2 loại kiểu dữ liệu trong C#
  - Value Data Types:
    - Dữ liệu chứa tại nơi nó được cấp phát vùng nhớ
    - Các kiểu số, bool, char, date, các cấu trúc, các kiểu liệt kê
  - Reference Data Types:
    - Chứa một con trỏ trỏ tới nơi cất giữ dữ liệu
    - Bao gồm kiểu string, mảng, class, delegate

Unsafe Code

- C# tự quản lý bộ nhớ, tuy nhiên có những lúc chúng ta cần sử dụng con trỏ.
  - Dùng từ khóa unsafe tại vùng lệnh muốn sử dụng con trỏ
  - Phải có tham số biên dịch unsafe khi dịch chương trình
- Con trỏ:
  - Các khai báo và sử dụng tương tự C++

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

C# and .NET Framework
Bài 4: .NET và các lớp cơ bản

Đoàn Quang Minh
minhdqtt@gmail.com
http://www.VTPortal.net
Last update: 28. December 2006

Mục lục

- System.Object
- Xử lý String
- Regular Expression
- Groups of Objects
- Reflection
- Threading

System.Object

- Là lớp cơ bản của C#
  - Nếu không nói gì, một lớp bất kỳ coi như được kế thừa từ Object
- Các phương thức
  - public virutal string ToString()
    - override phương thức này để chuyển một đối tượng thành xâu ký tự.
    - Thường dùng khi kết xuất thông tin về đối tượng.
  - public virutal int GetHashTable()
    - Trả về một giá trị băm của đối tượng
    - Thường dùng khi tạo khoá truy xuất cho đối tượng trong một tập dữ liệu như bảng băm hoặc từ điển.
  - public virutal bool Equals(object obj)
  - public static bool Equals(object objA, object objB)
  - public static bool ReferenceEquals(object objA, object objB)
    - So sánh hai đối tượng

System.Object

- Các phương thức (tiếp)
  - protected virtual void Finalize()
    - Mang ý nghĩa là hàm huỷ, được gọi bởi bộ thu gom rác. Mặc định không thực thi gì.
    - Chỉ override khi cần thiết, ví dụ đóng tập tin.
  - public Type GetType()
    - Trả về kiểu đối tượng, bao gồm lớp cha, các phương thức, thuộc tính,…
  - protected object MemberwiseClone()
    - Copy một đối tượng. Chú ý chỉ copy các tham chiếu bên trong đối tượng

Xử lý String

- Có 2 lớp hay được dùng để xử lý xâu
  - String: xử lý các xâu ký tự
  - StringBuilder: xây dựng một xâu ký tự
- String: chứa các phương thức cơ bản trong việc xử lý xâu ký tự.
  - Compare(): so sánh hai xâu.
  - CompareOrdinal(): so sánh, nhưng không tính đến văn hoá (culture)
  - Format(): định dạng xâu dựa trên biểu thức định dạng và các tham số đầu vào

Xử lý String

- Các phương thức của String (tiếp)
  - IndexOf(), IndexOfAny(), LastIndexOf(), LastIndexOfAny(): tìm kiếm chuỗi ký tự, hoặc một phần chuỗi ký tự trong một xâu cho trước.
  - PadLeft(), PadRight(): điền thêm vào đầu hoặc cuối xâu bởi ký tự cho trước.
  - Replace(): thay thế một mẫu trong xâu bởi một chuỗi ký tự khác.
  - Split(): cắt một xâu thành một tập hợp các xâu con dựa theo một ký tự phân cách cho trước.
  - Substring(): lấy một phần xâu con từ một xâu cho trước.
  - ToLower(), ToUpper(): biến các ký tự trong xâu thành ký tự thường hoặc ký tự hoa.
  - Trim(), TrimEnd(), TrimStart(): xoá các ký tự trắng ở đầu, cuối xâu.
  - Insert(), Remove(): chèn vào, xoá đi một xâu con trong một xâu cho trước.
  - StartsWith(), EndsWith(): kiểm tra xem xâu có bắt đầu, kết thúc bởi một xâu khác.

Xử lý String

- Để xử lý xâu, chúng ta hay dùng các phép toán như so sánh, gán, cộng thêm (+=)
  - Ưu điểm: Các phép toán đơn giản, dễ dùng
  - Nhược điểm: Hiệu suất quản lý bộ nhớ thấp
- Khi cần xây dựng một chuỗi văn bản phức tạp và có độ dài tương đối lớn, chúng ta dùng lớp StringBuilder
  - StringBuilder cho phép nối thêm các xâu mới vào trong một tập hợp các xâu có sẵn mà không cần quá nhiều các thao tác xử lý vùng nhớ
  - Ví dụ:

Xử lý String

- StringBuilder
  - Append(): nối thêm vào đuôi một xâu mới
  - Insert(): chèn vào một vị trí bất kỳ một xâu mới.
  - Remove(): xoá bỏ một xâu con tại vị trí hiện thời
  - ToString(): sau khi xây dựng tập các xâu xong, phương thức này biến đổi tập các xâu thành chuỗi văn bản duy nhất.

Xử lý String

- String.Format: định dạng xâu
  - Giống như hàm printf() của C, phương thức static Format của lớp String cho phép định dạng một chuỗi các tham số theo mẫu cho trước.
  - Cú pháp:

public static string **Format(** string *format*, object *arg0* **);**

Xử lý String

- Chuỗi format chứa một hoặc nhiều các đối tượng cần format, theo mẫu {index[,alignment][:formatString]}
  - index: chỉ số của đối tượng trong danh sách các đối tượng cần format
  - alignment: tuỳ chọn, là độ dài tối thiểu để chứa giá trij chuỗi của đối tượng đã được format
  - formatString: mã format.
  - Ví dụ, string.Format("I have {0,-4:G} computers", x), với x = 2 thì giá trị là "I have 2___ computers"

Biểu thức chính quy (Regular Expression)

- Regular Expression là lớp thực hiện các thao tác liên quan đến biểu thức chính quy:
  - Gồm một tập các ký tự đại diện;
  - Các phương thức phục vụ cho việc tìm kiếm và thay thế;
  - Sử dụng biểu thức chính quy, có thể thực hiện các công việc phức tạp về xử lý chuỗi
    - Kiểm định chuỗi đầu vào theo một tiêu chuẩn nào đó;
    - Định dạng lại chuỗi (thay thế các ký tự không hợp lệ);
    - TÌm kiếm và trích từ chuỗi đầu vào ra những thành phần đặc biệt.

Biểu thức chính quy (Regular Expression)

Ký tự	Ý nghĩa
^	Bắt đầu của chuỗi
$	Kết thúc của chuỗi
.	Tất cả ký tự, ngoại trừ xuống dòng \n
*	Lặp lại 0 lần hoặc nhiều hơn
+	Lặp lại ít nhất 1 lần
?	Lặp lại 0 hoặc 1 lần
\s	Khoảng trắng, bao gồm cả tab
\S	Tất cả các ký tự mà không là khoảng trắng
\b	Kết thúc nhóm
\B	Không kết thúc nhóm

Biểu thức chính quy (Regular Expression)

- Biểu thức chính quy trong C#
  - Nằm trong namespace System.Text.RegularExpression
  - Cung cấp các lớp Regex, Match,…
- Ví dụ
  - Kiểm tra xem 1 chuỗi đầu vào có là số nguyên hay không?

```
Regex re = new Regex(@"\d+");
Match m = re.Match(s);
if (m.Success)
{
    // match is found, s is a number
}
else
{
    // match not found, s isn't a number
}
```

- Bài tập: kiểm tra một chuỗi có biểu diễn một địa chỉ mail hay không?

Group of Objects

- Group of Objects là các lớp trong đó cho phép quản lý một tập hợp các đối tượng có cùng kiểu.
  - ArrayList: tương tự như mảng, nhưng có nhiều tính năng ưu việt: cho phép thêm, chèn, xoá, sắp xếp, tìm kiếm nhị phân,…
  - Collection: đối tượng tập hợp, trên đó có cài đặt giao tiếp IEnumerable cho phép duyệt từng phần tử trong tập hợp.
    - Stack: Cung cấp cơ chế FILO, có 2 phương thức đặc biệt là Push() và Pop()
    - Queue: cơ chế FIFO, có 2 phương thức đặc biệt là Enqueue() và Dequeue()
  - Dictionary: đối tượng từ điển, cung cấp cơ chế tìm kiếm đối tượng thông qua khoá
    - Hashtable: bảng băm, mỗi đối tượng sẽ được đại diện bởi một giá trị băm, gọi là khoá. Tác dụng tăng tốc trong các thao tác tìm kiếm.

Generics

- Gererics cung cấp các lớp cho phép sử lí một tập các đối tượng với kiểu của đối tượng như là tham số đầu vào.
  - List<T>: cho phép thao tác và xử lí một danh sách các đối tượng có kiểu T;
  - Stack<T>: cung cấp cơ chế FILO với kiểu dữ liệu T;
  - Queue<T>: cung cấp cơ chế FIFO với kiểu dữ liệu T;
  - LinkedList<T>: cung cấp một danh sách liên kết đôi xử lí các đối tượng có kiểu T;
  - Dictionary<K,T>: cung cấp một từ điển với kiểu dữ liệu là T, kiểu khóa là K.

Reflection

- Reflection là thuật ngữ chỉ các lớp trong .NET cho phép chúng ta có thể đọc được thông tin về các assembly. Chúng nằm trong namespace System.Reflection
  - Type: cung cấp thông tin về kiểu của một đối tượng. Bao gồm các thông tin như tên kiểu, tên đầy đủ (gồm cả namespace), tên lớp cơ sở,… Ngoài ra, có thể lấy được các thông tin khác như các phương thức, các trường, các sự kiện, các giao tiếp,…
  - Assembly: chứa thông tin về assembly, bao gồm các thông tin như tên assembly, tên công ty, phiên bản,…

Threading

- Các hệ thống thực đều là hệ đa tiến trình:
  - Các ứng dụng trong Windows chạy song song.
  - Với một công việc, thông thường sẽ gồm nhiều công việc nhỏ chạy song song.
  - .NET hỗ trợ lập trình song song dựa vào thread.
- Để tạo nhiều tiến trình chạy song song, chúng ta sẽ tạo ra các Thread
  - Các thuộc tính:
    - Name: tên của thread
    - Priority: mức độ ưu tiên của thread
  - Các phương thức:
    - Start(): khởi động thread
    - Suspend(): tạm ngưng thread
    - Resume(): kích hoạt lại thread đang tạm ngưng
    - Abort(): huỷ (ngắt giữa chừng) thread

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/nghien-cuu-phat-trien/thang-long/tab.aspx
- Diễn đàn C# & .NET http://www.thanglong.edu.vn/forum/cmd/0/category/hoc-tap-nghien-cuu/dot-net/tab.aspx

C# and .NET Framework
Bài 5: Windows Application

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

Mục lục

- Windows Application in .NET
- Windows Forms
- Windows Controls
- Custom Controls

Windows Application in .NET

- Lịch sử ứng dụng trên Windows
  - Có hai loại ứng dụng: rich client và thin client.
  - Ứng dụng rich client có nhiều ưu điểm: giao diện đẹp, tốc độ xử lý cao, tận dụng được các sức mạnh của hệ điều hành.
  - Các ứng dụng rich client được xây dựng dựa trên các hàm API (Application Programming Interface). Do tính phức tạp, nên xuất hiện một số môi trường lập trình khác (Visual Basic, MFC trên Visual C++).
  - .NET cung cấp một giải pháp đơn giản và hiệu quả trong việc lập trình các ứng dụng rich client trên Windows
- namespace System.Windows.Forms
  - Cung cấp các đối tượng để làm việc với Windows Form
    - Cho phép xử lý các sự kiện chuột và bàn phím
    - Cho phép xử lý các nút lệnh, các hộp soạn thảo, các thanh cuộn, các cửa sổ,…

Windows Forms

- Là các cửa sổ của ứng dụng chạy trên Windows
  - Tạo ứng dụng rich client
    - Có ít nhất một form kế thừa từ Form.
    - Trên form có thể chứa các control.
    - Một control có thể phát sinh sự kiện: để xử lý phải handler sự kiện.
    - Thay đổi diện mạo của form hoặc control bằng cách thay đổi các thuộc tính.
    - Thuộc tính quan trọng (mới): Anchor
  - Menu
    - Một ứng dụng có thể có hệ thống menu. Menu có thể có nhiều lớp.
    - Thuộc tính quan trọng: Name và Text.
    - Sự kiện quan trọng: click
    - Menu có thể thay đổi thuộc tính lúc run-time
  - Context menu
    - Một đối tượng có thể có menu ngữ cảnh.
    - Menu ngữ cảnh hoạt động giống menu thông thường.
  - Bài tập: xây dựng trình Notepad

Windows Forms

- Dialogs
  - Hiển thị thông báo và nhận trả lời của người dùng.
  - Có hai kiểu Modal và Modeless.
  - Có thể trả về các giá trị: Abort, Cancel, Ignore, No, Yes, None, Ok, Retry.
  - Để hiển thị hộp thoại, dùng phương thức Form.ShowDialog(). Phương thức này sẽ trả về giá trị thuộc kiểu DialogResult.
- Common Dialogs: hộp thoại thông dụng
  - Windows cung cấp sẵn một số hộp thoại thông dụng. Đó là các hộp chọn font, chọn màu, mở file, đóng file,…
  - Mỗi hộp thoại tương ứng với 1 lớp trong namespace Forms, và có tương ứng các thuộc tính đặc trưng riêng biệt. Các lớp bao gồm: ColorDialog, FontDialog, OpenFileDialog, SaveFileDialog, PageSetupDialog, PrintDialog, PrintPreviewDialog.
  - Bài tập: làm tốt hơn ứng dụng Notepad.

Windows Controls

- Có rất nhiều control trong Windows. Mỗi control đều có thể điều khiển thông qua các phương thức / thuộc tính với tên gợi nhớ.
  - Labels
  - Buttons
  - Checkboxes
  - Menus
  - Radio buttons
  - Combo boxes
  - Listboxes
  - Textboxes
  - Tabcontrols
  - Toolbars
  - Tree views

Windows Controls

- Data Binding
  - Là phương pháp gắn kết dữ liệu vào một control
    - Mỗi control đều chứa dữ liệu. Ví dụ: hộp checkbox sẽ chứa dữ liệu kiểu bool, hộp textbox chứa dữ liệu kiểu string, hộp listbox chứa dữ liệu kiểu tập hợp.
    - Khi có một nguồn dữ liệu, có thể gắn vào một control: thông thường control sẽ chứa một tập dữ liệu, nhưng chỉ có một mẩu dữ liệu được xác định là hiện thời.
- interface IList
  - Một đối tượng dữ liệu muốn trở thành nguồn dữ liệu thì phải cài đặt giao tiếp IList.
    - Hầu hết các đối tượng tập hợp của .NET đều cài đặt giao tiếp này.
  - Bài tập: viết hộp thoại chọn tên, nơi ở (thành phố, quận huyện) của một người.

Custom Controls

- Custom Controls
  - Rất quan trọng, cho phép lập trình viên mở rộng các đối tượng điều khiển.
  - Trong .NET, custom control được kế thừa từ lớp UserControl.
- Tạo custome control đơn giản
  - Gồm một tập các control có sẵn của windows
    - Thiết kế như một windows form
    - Mở rộng thêm các phương thức và thuộc tính
- Tạo custome control mở rộng
  - Nhằm mục đích mở rộng tính năng control sẵn có
    - Kế thừa từ control cần mở rộng
    - Thêm các phương thức và thuộc tính cần thiết
    - Có thể thêm sự kiện

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

C# and .NET Framework
Bài 6: Data Access and Viewing with .NET

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

Mục lục

- ADO.NET Overview
- Using Database Connections
- Commands
- Quick Data Access: The Data Reader
- Working with DataSet
- Viewing .NET data
- Example

ADO.NET Overview

- ADO.NET là gì
  - ADO - Microsoft's ActiveX Data Objects: thư viện các cho phép truy cập và xử lý CSDL.
  - ADO có một số hạn chế: luôn luôn giữ kết nối, chỉ làm việc với CSDL...
  - ADO.NET: làm việc với các đối tượng dữ liệu, hỗ trợ mạnh mẽ SQL Server, đồng thời hỗ trợ các kết nối OLE DB.
- namespace System.Data
  - Để truy cập và xử lý CSDL, sử dụng các namespace System.Data, System.Data.Common, System.Data.OleDb, System.Data.SqlClient, System.Data.SqlTypes.
  - Các lớp cơ bản trong System.Data: DataSet, DataTable, DataRow, DataColumn, DataRelation, Constraint
  - Các lớp đặc biệt: SqlCommand, OleDbCommand, SqlCommandBuilder, OleDbCommandBuilder, SqlConnection, OleDbConnection, SqlDataAdapter, OleDbDataAdapter, SqlDataReader, OleDbDataReader, SqlParameter, OleDbParameter, SqlTransaction, OleDbTransaction

Using Database Connections

- Muốn truy vấn CSDL, chúng ta phải có một kết nối đến CSDL
  - Sử dụng lớp SqlConnection, OleDbConnection
    - Cung cấp chuỗi kết nối: thông thường bao gồm tên server, tên CSDL, tên truy cập, mật khẩu.
    - Sử dụng các phương thức Open() và Close().
  - Sử dụng kết nối hiệu quả
    - Đóng ngay kết nối khi không dùng nữa: thông thường, chúng ta không duy trì một kết nối "cứng" đến CSDL. Khi cần truy vấn, chúng ta mở kết nối, truy vấn xong, đóng ngay kết nối lại.
    - Khối lệnh kết nối nên đặt trong khối try…catch
    - Từ khóa using: sử dụng một đối tượng. Ra khỏi phạm vi của using, đối tượng sẽ bị huỷ.
  - Transactions
    - Transactions là gì?.
    - Sử dụng thông qua SqlTransaction hoặc OleDbTransaction.

Commands

- Là đối tượng thực thi câu lệnh
  - Một đối tượng command thông thường được sử dụng để thực thi một câu lệnh SQL hoặc một thủ tục lưu.
  - Các bước thực hiện:
    - Khai báo và mở một connection.
    - Khai báo một chuỗi chứa câu lệnh SQL hoặc tên thủ tục lưu
    - Khai báo một đối tượng command với câu lệnh truy vấn và nguồn kết nối.
    - Chỉ định thuộc tính câu truy vấn: dạng text hay thủ tục lưu.
    - Thêm các tham số của câu truy vấn.
    - Thực hiện lệnh truy vấn: tuỳ theo yêu cầu thực hiện các lệnh khác nhau

Commands

- Các lệnh truy vấn với command.
  - ExecuteNonQuery()
    - Thực thi câu lệnh, không trả về kết quả.
    - Thường sử dụng trong truy vấn không cần quan tâm đến kết quả, ví dụ các lệnh delete, insert, update…
  - ExecuteReader()
    - Trả về một DataReader.
    - Thường sử dụng trong các truy vấn hiển thị dữ liệu.
  - ExecuteScalar()
    - Trả về một đối tượng duy nhất.
    - Thường sử dụng trong các truy vấn trả về một giá trị dữ liệu đơn, ví dụ các lệnh tính tổng, tính trung bình, tính min/max…
  - ExecuteXmlReader()
    - Trả về một XmlReader.
    - Thường được xử dụng khi hiển thị dữ liệu dưới dạng XML.
    - Nên dùng nếu CSDL hỗ trợ truy vấn XML, ví dụ SQL Server 2000.

Commands

```
public class ExecuteScalarExample
{
   public static void Main(string[] args)
   {
       string source = "server=(local)\\NetSDK;" +
   "uid=QSUser;pwd=QSPassword;" + "database=Northwind";
       string select = "SELECT COUNT(*) FROM Customers";
       SqlConnection conn = new SqlConnection(source);
       conn.Open();
       SqlCommand cmd = new SqlCommand(select, conn);
       object o = cmd.ExecuteScalar();
       Console.WriteLine ( o ) ;
   }
}
```

Quick Data Access: The Data Reader

- Data Reader
  - Chỉ được tạo ra bởi giá trị trả về của câu lệnh truy vấn.
  - Kết nối tới CSDL luôn mở, cho đến nhận được lệnh đóng.
- Tính chất
  - Một data reader giống như một record set chỉ tiến (forward only) trong ADO.
    - Chỉ có thể đọc, và đi đến bản ghi tiếp.
    - Không thể quay lại các bản ghi đã đọc.
  - Tốc độ cao:
    - Một data reader không giữ các bản ghi trong bộ nhớ.
    - Data reader chỉ có nhiệm vụ lấy dữ liệu từ CSDL và chuyển về.
    - Rất hay được sử dụng khi chỉ cần hiển thị dữ liệu, nhất là trong môi trường web.

Working with DataSet

- DataSet:
  - Có tác dụng giống như một CSDL offline:
    - Trong một DataSet có thể chứa các DataTable, DataRelation,…
    - DataSet có thể được xây dựng không chỉ từ các truy vấn CSDL, mà có thể từ các tập tin khác (text, Excel, CVS,…)
  - Để tạo DataSet
    - Truy vấn CSDL, dựa trên một DataAdapter
    - Xây dựng bằng cách thêm các DataTable

Working with DataSet

- Truy vấn CSDL
  - Mở một connection.
  - Tạo một DataAdapter, chỉ định câu lệnh truy vấn cho data adapter.
  - Tạo mới một data set.
  - Sử dụng phương thức Fill() của data adapter.
- Xây dựng bằng cách thêm các data table
  - Tạo mới một DataSet.
  - Tạo mới các DataTable.
    - Khởi tạo data table bằng cách thêm mới DataColumn
    - Thêm các dòng dữ liệu vào data table.
  - Add các data table vào data set bằng cách thêm vào thuộc tính Tables của data set

Working with DataSet – Example

```
DataSet ds = new DataSet();
DataTable dt = ds.Tables.Add("SampleData");

dt.Columns.Add("MonHocID", typeof(Guid));
dt.Columns.Add("TenMon", typeof(string));
dt.Columns.Add("MaMon", typeof(string));
dt.Columns.Add(“HeSoMon", typeof(int));

DataRow dr;
for (int i = 1; i <= 20; i++)
{
   dr = dt.NewRow();
   dr["MonHocID"] = Guid.NewGuid();
   dr["TenMon"] = "Mon hoc thu " + i.ToString();
   dr["MaMon"] = "MaMon00" + i.ToString();
   dr["HeSoMon"] = i;
   dt.Rows.Add(dr);
}
```

Working with DataSet – Example

```
SqlConnection conn = new SqlConnection(source);
SqlCommand cmd = new SqlCommand(select, conn);
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = cmd;
DataSet data = new DataSet();
try
{
   conn.Open();
   adapter.Fill(data);
}
catch (SqlException expSQL)
{
}
finally
{
   conn.Close();
}
```

Working with DataSet

- Sau khi xây dựng, điền thông tin vào data set, có thể thay đổi dữ liệu (insert, delete, update) và cập nhật trở lại vào CSDL.
  - Thuộc tính Rows của DataTable là một collection.
    - Thêm (insert) một row mới bằng phương thức Add()
    - Cập nhật (update) một row cũ bằng cách thay đổi các giá trị của row
    - Xoá (delete) một row bằng phương thức Delete()

Viewing .NET data

- Song song với việc xử lý dữ liệu là hiển thị dữ liệu:
  - .NET cung cấp các control rất hiệu quả cho việc hiển thị dữ liệu.
  - Việc hiển thị dữ liệu trên các control này chỉ đơn giản bằng cách chỉ ra nguồn dữ liệu, gọi phương thức DataBind()
  - Các đối tượng hiển thị dữ liệu bao gồm: DataGrid, DataList, Repeter

Viewing .NET data

- DataGrid (ASP.NET)
  - Cho phép hiển thị dữ liệu dưới dạng bảng.
  - Cho phép phân trang và sắp xếp dữ liệu.
- Hiển thị dữ liệu
  - Thiết kế form:
    - Thêm một DataGrid.
    - Thay đổi các thuộc tính cần thiết.
  - Thuộc tính quan trọng:
    - DataKeyField: khoá chính của lưới, thông thường là khoá chính trong bảng dữ liệu.
    - DataMember: bảng dữ liệu (trong DataSet) cần hiển thị.
    - DataSource: nguồn dữ liệu cần hiển thị
    - AutoGenerateColumns: nếu bằng true, các cột của grid sẽ tự động sinh theo các (tên) trường dữ liệu trong bảng.

Viewing .NET data

- DataGrid (ASP.NET)
  - Cột trong grid: cho phép hiển thị theo nhiều khuôn dạng khác nhau:
    - Bound Column: chỉ hiện thị dữ liệu dạng text
    - Button Column: cho phép thực thi lệnh (xoá, soạn thảo,…)
    - HyperLink Column: siêu liên kết
    - TemplateColumn: mẫu, chứa mô tả của cột
  - Với cột là button column
    - Như là một nút lệnh bình thường
    - Thông thường, cần xử lý dòng dữ liệu được click, dòng này sẽ được xác định dựa trên khoá của lưới

Viewing .NET data

- GridView (ASP.NET): Tương tự control DataGrid, nhưng có một số khác biệt
  - Thêm một số loại cột mới:
    - CheckBoxField: hiển thị dữ liệu dạng check;
    - ImageField: hiển thị dữ liệu dạng ảnh.
  - Hỗ trợ đa ngôn ngữ hoàn toàn;
  - Hỗ trợ Ajax thông qua khái niệm callback.
- Data Source: là các control kết nối và truy xuất dữ liệu từ nguồn dữ liệu
  - Các nguồn dữ liệu hỗ trợ: Access, SQL Server, Object, Site Map, XML file;
  - Cho phép thiết kế nhanh các nguồn dữ liệu phù hợp thông qua winzard;
  - Khi gắn nguồn dữ liệu vào control hiển thị, dữ liệu sẽ được tự động truy vấn và hiển thị mà không cần viết thêm mã lệnh;
  - Rất tối ưu nếu có sử dụng callback

Example

- Ví dụ làm việc với CSDL
  - Một cửa hàng bán thiết bị vi tính cần quản lý sản phẩm theo danh mục:
    - Các danh mục như: mainboard, chip, hdd,…
    - Trong danh mục có các sản phẩm: ví dụ trong danh mục chip có chip AMD, chip Intel (các dòng khác nhau)
  - Yêu cầu:
    - Hiển thị danh mục ở một bên, chi tiết các sản phẩm ở một bên
    - Khi người dùng chọn một danh mục, hiển thị các sản phẩm tương ứng.

Example

- Phân tích
  - Các yêu cầu đầu bài.
  - Thiết kế CSDL
    - Các bảng cần thiết
    - Các thủ tục lưu, nếu cần
  - Xác định môi trường ứng dụng (Windows hay Web)
  - Design form
  - Viết mã dựa trên phân tích yêu cầu

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

C# and .NET Framework
Bài 7: XML in .NET

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

Mục lục

- XML Overview
- Đọc XML
- Ghi XML
- DOM trong .NET
- Ứng dụng

XML Overview

- XML là gì
  - XML – Extensible Markup Language: ngôn ngữ định dạng mở rộng.
  - XML được định nghĩa bởi w3c (World Wide Web Consortium): tổ chức mạng toàn cầu.
  - Thực chất, XML là ngôn ngữ tổng quát dùng định nghĩa dữ liệu thông qua các thẻ.
- Ngôn ngữ định dạng
  - Bao gồm một tập các thẻ, và dữ liệu chứa trong các thẻ đó. Ví dụ, HTML là một loại ngôn ngữ định dạng.
  - Trong XML, các thẻ không hạn chế như HTML. Người sử dụng có thể tự do định nghĩa các thẻ của mình.
  - Các ưu điểm:
    - Dễ dàng trao đổi dữ liệu: do khuôn dạng XML rất dễ hiểu. XML còn được gọi là dữ liệu tự mô tả.
    - Khả năng tuỳ biến cao: việc cụ thể hoá các thẻ của XML sẽ tạo ra một loạt các ngôn ngữ mới, ví dụ MML, CML.
    - Có thể lưu dữ liệu có cấu trúc: dựa trên việc sử dụng các thẻ lồng nhau.

XML Overview

```
<? xml version="1.0" encoding="UTF-8" ?>
<Document>
   <Greeting>
        Hello from XML
   </Greeting>
   <Message>
        Welcome to the wild and woolly world of
   XML.
   </Message>
</Document>
```

XML Overview

- Các ứng dụng cụ thể của XML
  - XHTML: đây là mở rộng của HTML. Về bản chất, XHTML dùng các thẻ của HTML, các thẻ này phải viết thường, và tài liệu XHTML phải là tài liệu hợp khuôn dạng.
  - CML, MML: các ngôn ngữ biểu diễn công thức hoá học, công thức toán học. Các công thức này có thể được vẽ trên các trình duyệt đặc biệt, thông qua dữ liệu XML.
  - WML: ngôn ngữ định dạng mạng không dây, nhằm tạo các tài liệu web hiển thị trên máy điện thoại di động.
  - SOAP (Simple Object Access Protocol): giao thức truy cập đối tượng đơn giản, cho phép các ứng dụng web có thể trao đổi thông tin với nhau. Khi một ứng dụng web cần thi hành một chức năng do một web service cung cấp, nó sẽ gửi dữ liệu theo chuẩn SOAP tới web service. Đến lượt mình, sau khi thực thi yêu cầu xong, web service sẽ trả lại kết quả, cũng theo chuẩn SOAP.

Đọc XML

- Các đối tượng xử lý XML được cung cấp trong System.XML
  - Để đọc file XML, dùng lớp XmlTextReader
    - Hàm tạo với tham số là tên file XML cần đọc.
    - Phương thức Read() đọc 1 thành phần của file. Khi đọc thành công, đối tượng sẽ trỏ tới node hiện thời.
  - Tại một node, chúng ta có thể đọc được một số thành phần.
    - Kiểu của node: thuộc tính NodeType
      - Có những kiểu như Element, Text, CDATA, Comment,…
    - Giá trị của node: thuộc tính Value
    - Có thể sử dụng các phương thức cụ thể khác như ReadString(): đọc giá trị text của node, ReadAttributeValue(): đọc giá trị thuộc tính của node.

Đọc XML

```
public class Sample
{
  static void Main(string[] args)
  {
    XmlTextReader textReader  = new XmlTextReader("C:\\books.xml");
    textReader.Read();
    while (textReader.Read()  ) // If the node has value
    {
        // Move to fist element
        textReader.MoveToElement();
        Console.WriteLine("XmlTextReader Properties Test");
        Console.WriteLine("==================");
        // Read this element's properties and display them on console
        Console.WriteLine("Name:" + textReader.Name);
        Console.WriteLine("Base URI:" + textReader.BaseURI);
        Console.WriteLine("Local Name:" + textReader.LocalName);
        Console.WriteLine("Attribute Count:" + textReader.AttributeCount.ToString());
        Console.WriteLine("Depth:" + textReader.Depth.ToString());
        Console.WriteLine("Line Number:" + textReader.LineNumber.ToString());
        Console.WriteLine("Node Type:" + textReader.NodeType.ToString());
        Console.WriteLine("Attribute Count:" + textReader.Value.ToString());
    }
  }
}
```

Ghi XML

- Để ghi XML, dùng XmlTextWriter.
  - Hàm tạo với tên file cần ghi.
  - Các phương thức quan trọng.
    - WriteStartDocument(): ghi phần khai báo version XML.
    - WriteStartElement(string): ghi thẻ mở đầu của một nút.
    - WriteAttributeString(string, string): ghi thuộc tính và giá trị của nó.
    - WriteElementString(string, string): ghi một nút, trong đó có chứa một giá trị.
    - WriteEndElement(): ghi thẻ kết thúc của một nút.

Ghi XML

```
public class Sample
{
  public static void Main()
  {
    XmlTextWriter writer = new XmlTextWriter("titles.xml", null);

    //Write the root element
    writer.WriteStartElement("items");

    //Write sub-elements
    writer.WriteElementString("title", "Unreal Tournament 2003");
    writer.WriteElementString("title", "C&C: Renegade");
    writer.WriteElementString("title", "Dr. Seuss's ABC");

    // end the root element
    writer.WriteEndElement();

    //Write the XML to file and close the writer
    writer.Close();
  }
}
```

DOM trong .NET

- DOM (Document Object Model): mô hình đối tượng tài liệu cho phép xử lý XML một cách mềm dẻo
  - Khác với XmlTextReader, XmlTextWriter chỉ cho phép đọc và ghi XML theo kiểu tuần tự, DOM cho phép truy cập ngẫu nhiên vào tài liệu XML.
  - Các lớp quan trọng
    - XmlDocument: cho phép xử lý XML theo DOM
    - XmlNodeList: danh sách các node trong tài liệu XML.
    - XmlNode: một node đơn nhất trong tài liệu
  - XmlDocument cho phép thêm node mới, nối node vào đuôi tài liệu, xoá node khỏi tài liệu.

DOM trong .NET

```
void Sample()
{
  oXmlDoc = new XmlDocument();
  oXmlDoc.Load(Server.MapPath("xml_XmlDocument.xml"));
  XmlNode oNode = oXmlDoc.DocumentElement;
  Console.WriteLine("Node Name: " + oNode.Name);
  XmlNodeList oNodeList = oNode.SelectNodes("/books/category/title");
  Console.WriteLine("NodeList count=" + oNodeList.Count);
  for(int x = 0; x < oNodeList.Count; x++)
    Console.WriteLine("NodeList Item#" + x + " " +
     oNodeList.Item[x].InnerText);
}
```

Ứng dụng

- Tài liệu XML có thể dùng lưu trữ dữ liệu như là một CSDL.
- Một ứng dụng:
  - Xây dựng chương trình thi trắc nghiệm qua mạng, trên nền web:
    - Hiển thị câu hỏi, danh sách các đáp án
    - Số lượng đáp án trong một câu hỏi có thể khác nhau, và không hạn chế.
    - Một câu hỏi chỉ có một đáp án đúng.

Ứng dụng

- Phân tích
  - Các yêu cầu đầu bài.
  - Thiết kế CSDL dưới dạng file XML
    - Đề ra một mô hình lưu trữ phù hợp
    - Tạo thử file dữ liệu
  - Design form
  - Viết mã dựa trên phân tích yêu cầu

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

C# and .NET Framework
Bài 8: File and Registry

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

Mục lục

- Managing the File System
- Moving, Copying, and Deleting Files
- Reading and Writing to Files
- The Registry
- Ứng dụng

Managing the File System

- .NET hỗ trợ các thao tác làm việc với file
  - Các tác vụ thông thường như liệt kê file, sao chép, di chuyển, xoá.
  - Các lớp thao tác với file nằm trong namespace System.IO
  - Các lớp quan trọng: File, FileInfo, Directory, Path,...
- Làm việc với file và folder
  - Có 2 loại đối tượng làm việc với file và folder
    - Directory và File: chỉ chứa các phương thức tĩnh, không thể khởi tạo. Thường dùng khi chỉ thực hiện 1 thao tác với 1 file hoặc folder. Khi thao tác, chỉ cần cung cấp đường dẫn đến file hay folder cần làm việc
    - DirectoryInfo và FileInfo: cung cấp các phương thức như 2 đối tượng trên, nhưng yêu cầu phải tạo instance. Thường dùng khi thực hiện nhiều thao tác với 1 file hoặc folder.

Managing the File System

Tên	Ý nghĩa
CreationTime	Thời gian tạo file hoặc folder
DirectoryName (FileInfo), Parent (DirectoryInfo)	Đường dẫn đầy đủ của folder chứa file hoặc folder hiện thời
Exists	File hay folder có tồn tại hay không?
Extension	Phần mở rộng
FullName	Tên đầy đủ, cả đường dẫn
LastAccessTime	Thời gian lần truy cập cuối
LastWriteTime	Thời gian lần sửa đổi cuối
Name	Tên file hay folder
Root	Folder gốc (chỉ với DirectoryInfo)
Length	Dung lượng (bytes), chỉ với FileInfo

Managing the File System

```
// khởi tạo biến myFile trỏ đến một tập tin
FileInfo myFile = new FileInfo(@"C:\How to C Sharp.txt");
// sao chép sang ổ đĩa D
myFile.CopyTo(@"D:\");
// kiểm tra sự tồn tại
Console.WriteLine(myFile.Exists.ToString());
// ghi thông tin thời điểm tạo file
Console.WriteLine(myFile.CreationTime.ToString());
// cập nhật thời điểm tạo file
myFile.CreationTime = new DateTime(2001, 1, 1, 7, 30, 0);
```

Managing the File System

```
DirectoryInfo theFolder = new DirectoryInfo(folderFullName);
if (!theFolder.Exists)
    throw new DirectoryNotFoundException("Folder not found: " +
    folderFullName);
string currentPath = theFolder.FullName;

// Lấy tên các thư mục con của thư mục hiện thời
ArrayList folders = new ArrayList();
foreach(DirectoryInfo folder in theFolder.GetDirectories())
    folders.Add(folder.Name);

// Lấy tên các file trong thư mục hiện thời
ArrayList files = new ArrayList();
foreach(FileInfo file in theFolder.GetFiles())
    files.Add(file.Name);
```

Moving, Copying, and Deleting Files

- Có thể sao chép, di chuyển hoặc xoá tập tin.
  - Phương thức Path.Combine(string, string): trả về tên đầy đủ của file tạo từ đường dẫn và tên file.
  - Phương thức File.Delete(string): xoá tập tin.
  - Phương thức File.Move(string, string): di chuyển file từ vị trí cũ đến vị trí mới.
  - Phương thức File.Copy(string, string): sao chép file sang một thư mục mới.

Reading and Writing to Files

- Đọc và ghi file dựa trên khái niệm stream (luồng dữ liệu)
  - stream là đối tượng dùng để chuyển dữ liệu. Do đó stream có thể là luồng dựa trên bộ nhớ, trên tập tin, trên mạng,…
  - FileStream: đối tượng dùng để đọc ghi file nhị phân.
  - StreamReader và StreamWriter: đối tượng dùng để đọc ghi file text.
  - Chú ý: các tác vụ đọc ghi hầu hết đều sử dụng buffer. Do đó, với tác vụ ghi, phải đẩy dữ liệu từ buffer lên đĩa trước khi đóng file.

Reading and Writing to Files

- Đọc ghi file nhị phân: dùng FileStream.
  - Hàm tạo: cần chỉ ra filename, FileMode, FileAccess, FileShare.
    - FileMode: kiểu mở file, có thể là Append, Create, CreateNew, Open, OpenOrCreate, hoặc Truncate.
    - FileAccess: kiểu truy cập, có thể là Read, ReadWrite, hoặc Write.
    - FileShare: kiểu chia sẻ giữa các thread, có thể là Inheritable, None, Read, ReadWrite, or Write
  - Để đọc và ghi byte, dùng các hàm:
    - ReadByte(): đọc một byte từ stream
    - WriteByte(byte): ghi một byte vào stream
    - Read/Write(byte[], int off, int count): đọc/ghi một mảng byte bắt đầu từ off, độ dài count
  - Sau khi đọc/ghi, dùng Close() để đóng file

Reading and Writing to Files

- Đọc và ghi file text: dùng StreamReader và StreamWriter
  - Có thể khởi tạo StreamReader dựa trên
    - Tên file cần đọc
    - Một FileStream khác
    - Hoặc một FileInfo với phương thức OpenText()
  - Có thể khởi tạo StreamWriter dựa trên
    - Tên file cần đọc, mã encode
    - Một FileStream khác
    - Hoặc một FileInfo với phương thức CreatText()
  - Để đọc và ghi, dùng các hàm
    - Read()/Write(): đọc và ghi một ký tự
    - ReadLine()/WriteLine(): đọc và ghi một dòng
    - ReadToEnd(): đọc đến hết file

Reading and Writing to Files

```
void WriteToTextFile(string FileName, string strMessage)
{
	FileStream myFileStream = new FileStream(FileName, FileMode.Append, FileAccess.Write, System.IO.FileShare.None);
	System.IO.StreamWriter myWriter = new StreamWriter(myFileStream);
	myWriter.WriteLine(System.DateTime.Now.ToString() + " - " + strMessage);
	myWriter.Close();
	myFileStream.Close();
}

string ReadFileTextContent(string Filename)
{
	StreamReader myStreamReader = null;
	string FilePath = System.Web.HttpContext.Current.Server.MapPath(Filename);
	string result = string.Empty;
	try
	{
		myStreamReader = File.OpenText(FilePath);
		result = myStreamReader.ReadToEnd();
	}
	catch(Exception exc)
	{
		throw;
	}
	finally
	{
		if (myStreamReader != null) myStreamReader.Close();
	}
	return result;
}
```

The Registry

- Registry
  - Là một cấu trúc dạng cây cho phép các trinh ứng dụng có thể lưu thông tin. Được quản lý bởi Windows
  - Để soạn thảo registry, dùng trình regedit.
  - Các thành phần quan trọng:
    - HKEY_CLASSES_ROOT (HKCR): chứa các mô tả của các thành phần COM trong Windows.
    - HKEY_CURRENT_USER (HKCU): chứa các thông tin tuỳ biến của user hiện thời
    - HKEY_LOCAL_MACHINE (HKLM): các các thông tin về hardware và software
    - HKEY_USERS (HKUSR): chứa thông tin về các user
  - Để đọc và ghi registry, phải có quyền admin (mặc định)

The Registry

- Truy cập registry
  - namespace: Microsoft.Win32
    - Registry: chứa các mô tả về key trong registry
    - RegistryKey: cho phép thao tác với các key
  - Các phương thức của RegistryKey
    - OpenSubKey(): mở key con (tiếp tục đi xuống)
    - CreateSubKey()/DeleteSubKey(): tạo/xoá key con
    - GetValue()/SetValue(): lấy/đặt giá trị của key

Ứng dụng

- Ứng dụng các lớp thao tác với tập tin và registry
  - Tạo trình soạn thảo văn bản
  - Đọc và ghi cấu hình trong registry
    - File truy cập lần cuối
    - Các thiết lập như màu chữ, màu nền, font mặc định
    - Các thông tin lưu vết như các xâu tìm kiếm và thay thế…

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

C# and .NET Framework
Bài 9: Summary

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

Mục lục

- Nhắc lại các bài tập đã học
- Trả lời thắc mắc

Bài 1: C Sharp và kiến trúc .NET. C# cơ bản.

- Kiến thức cần nhớ
  - Ngôn ngữ C# cơ bản: cú pháp.
  - Khái niệm CLR.
  - Khái niệm Assemblies.
- Ứng dụng thực tiễn
  - Xây dựng các chương trình C# đầu tiên

Bài 2: Hướng đối tượng trong C#

- Kiến thức cần nhớ
  - Hướng đối tượng trong C#.
    - Kế thừa: các tính chất cơ bản. Khác biệt với C++.
    - Hàm ảo: khái niệm và ứng dụng.
    - Hàm tạo, hàm huỷ: khái niệm và cách dùng.
    - Thuộc tính: từ khoá get và set.
    - Chỉ số: khái niệm và ứng dụng.
    - Giao tiếp: từ khoá interface. Khái niệm và ứng dụng.
    - Đối tượng object.
- Ứng dụng thực tiễn
  - Xây dựng các lớp thực tế: lớp vector, lớp matrix,…

Bài 3: Lập trình nâng cao trong C#

- Kiến thức cần nhớ
  - Xử lý lỗi và ngoại lệ:
    - Khối try…catch…finally
    - Các ngoại lệ hay gặp
  - Ép kiểu: an toàn và không an toàn.
  - Mô hình chuyển giao:
    - Từ khoá delegate, khái niệm.
    - Cách dùng: trong các bài toán sắp xếp.
    - Sự kiện: khái niệm và cách dùng.
  - Khái niệm generic
  - Chỉ dẫn biên dịch
  - Quản lý bộ nhớ: khái niệm
  - Mã không an toàn: khái niệm
- Ứng dụng thực tiễn
  - Xây dựng các bài toán có dùng giải thuật sắp xếp

Bài 4: C# và các lớp cơ bản

- Kiến thức cần nhớ
  - Lớp Object: khái niệm, các phương thức quan trọng.
  - Xử lý chuỗi:
    - Lớp String: khái niệm, các phương thức quan trọng.
    - Lớp StringBuilder: khái niệm.
  - Biểu thức chính quy:
    - Cơ sở toán học.
    - Các lớp Regex, Match
    - Ứng dụng
  - Các lớp dạng nhóm
    - Khái niệm
    - Lớp ArrayList, lớp Stack và Queue.
    - Các lớp generics
- Ứng dụng thực tiễn
  - Xây dựng nghiệp vụ xử lý văn bản.
  - Kiểm tra tính hợp lệ của đầu vào dữ liệu
  - Lưu trữ và thao tác với tập hợp dữ liệu

Bài 5: Ứng dụng trên Windows

- Kiến thức cần nhớ
  - Khái niệm ứng dụng trên Windows
  - Các form trong Windows
    - Thiết kế form
    - Thêm menu và các control khác
    - Handle sự kiện
    - Sử dụng các hộp thoại dùng chung
- Ứng dụng thực tiễn
  - Xây dựng trình soạn thảo văn bản

Bài 6: Xử lý cơ sở dữ liệu trong .NET

- Kiến thức cần nhớ
  - Khái niệm về ADO.NET
  - Kết nối với CSDL: lớp connection
  - Thực thi câu lệnh truy vấn CSDL:
    - Lớp Commands: khai báo và khởi tạo
    - 3 thủ tục quan trọng hay dùng: ExecuteNonQuery(), ExecuteReader(), ExecuteScalar().
  - DataReader:
    - Khái niệm, khai báo, khởi tạo
    - Ưu, nhược điểm, tình huống sử dụng
  - DataSet:
    - Khái niệm, khai báo, khởi tạo
    - Ưu, nhược điểm, tình huống sử dụng
  - Hiển thị dữ liệu
    - Khái niệm binding data.
    - Các control DataGrid, Repeater, DataList
    - Control GridView
- Ứng dụng thực tiễn
  - Các ứng dụng về CSDL như thương mại điện tử, forum,…

Bài 7: XML trong .NET

- Kiến thức cần nhớ
  - Tổng quan về XML
    - Khái niệm
    - Ví dụ và ứng dụng
  - Đọc ghi XML trong .NET
    - Truy cập tuần tự: XmlTextReader và XmlTextWriter
    - Truy cập ngẫu nhiên:
      - DOM: mô hình đối tượng tài liệu
      - Các lớp XmlDocument, XmlNodeList,…
- Ứng dụng thực tiễn
  - Coi XML như một bảng dữ liệu, lưu trữ dữ liệu phức tạp trong một cột dữ liệu thực.

Bài 8: Tập tin và đăng ký

- Kiến thức cần nhớ
  - Xử lý thông tin tập tin: thư mục và tập tin.
  - Quản lý tập tin: sao chép, xoá, di chuyển.
  - Xử lý nội dung tập tin: tập tin nhị phân và văn bản.
  - Xử lý đăng ký trong Windows
- Ứng dụng thực tiễn
  - Quản lý tập tin từ xa qua ứng dụng web
  - Ghi thông tin cấu hình trong Windows.

Trả lời thắc mắc

- Thắc mắc về C# và .NET
  - ???
- Thắc mắc về bài tập lớn
  - ???
- Thắc mắc về kỳ thi hết môn
  - ???
- Đề nghị và góp ý
  - ???

Tạm biệt!

Chúc các bạn thi tốt.

Hẹn gặp lại tại môn C# và .NET nâng cao.

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