

CHƯƠNG 1: ÔN TẬP CÁC LỚP CƠ SỞ



**KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

NỘI DUNG

2

- Thao tác trên chuỗi (tự ôn tập)
- Biểu thức chính quy (regular expression)
- Nhóm đối tượng:
 - Danh sách mảng (ArrayList, List)
 - Collections
 - Từ điển

Biểu thức chính quy (Regular Expression)

3

- Ngôn ngữ biểu thức chính quy là ngôn ngữ được thiết kế đặc biệt cho việc xử lý chuỗi. Có 2 đặc tính :
 - 1 tập mã escape cho việc xác định kiểu của các kí tự.
 - 1 hệ thống cho việc nhóm những phần chuỗi con, và trả về kết quả trong suốt thao tác tìm.
- Dùng biểu thức chính quy có thể thao tác ở cấp cao và phức tạp trên chuỗi:
 - Xác định tất cả các từ lặp lại trong chuỗi , chuyển *"The computer books books"* → *"The computer books"*
 - Chuyển tất cả các từ theo title case, như là chuyển *"this is a Title"* → *"This Is A Title"*.
 - Chuyển những từ dài hơn 3 kí tự thành title case , ví dụ chuyển *"this is a Title"* → *"This is a Title"*
 - Bảo đảm các câu được viết hoa
 - Phân cách những phần tử của URL

Biểu thức chính quy (Regular Expression)

4

- Sử dụng đối tượng lớp **Regex**

```
public Regex(string pattern, RegexOptions options);
```

```
public Regex(string pattern);
```

- Pattern: là biểu thức so mẫu

- options: tùy chọn kiểu so khớp (có thể kết hợp nhiều tùy chọn bằng |)

```
RegexOptions.IgnoreCase
```

```
RegexOptions.Compiled
```

```
RegexOptions.None
```

```
.....
```

Biểu thức chính quy (Regular Expression)

5

- Method của **Regex**

- Matches: trả về tập MatchCollection đã so khớp

```
public MatchCollection Matches(string input)
```

```
public static MatchCollection Matches(string input,  
string pattern, RegexOptions options)
```

- ✦ Mỗi Match:

- **Index**: vị trí tìm thấy chuỗi khớp

- **Groups**: trả về GroupCollection chứa thông tin của chuỗi so khớp.
(**Value**: chuỗi tìm thấy, **Index**: vị trí của nhóm pattern so khớp)

- IsMatch: trả về giá trị true, false

```
public bool IsMatch(string input);
```

- Replace: thay thế chuỗi tìm thấy bằng delegate **MatchEvaluator**

```
public string Replace(string input, MatchEvaluator evaluator);
```

```
public static string Replace(string input, string pattern,  
MatchEvaluator evaluator);
```

Biểu thức chính quy (Regular Expression)

6

- Ví dụ: Tìm tất cả các vị trí xuất hiện của một chuỗi trong chuỗi khác

```
string input = "Xin chao cac ban. Chao";  
string patten = "chao";  
MatchCollection listmatch =  
    Regex.Matches(input, patten, RegexOptions.IgnoreCase);  
foreach (Match m in listmatch)  
    Console.WriteLine("Vi tri so khop: " + m.Index);
```

Chuỗi Pattern???

```
Vi tri so khop:4  
Vi tri so khop:18
```

```
string input = "Xin chao cac ban. Chao";  
string patten = "chao";  
Regex reg = new Regex(patten, RegexOptions.IgnoreCase);  
MatchCollection listmatch = reg.Matches(input); //(input, 10);  
foreach (Match m in listmatch)  
    Console.WriteLine("Vi tri so khop: " + m.Index);
```

Biểu thức chính quy (Regular Expression)

Ký tự	Mô tả	Ví dụ	Một số thể hiện
^	Bắt đầu của chuỗi nhập	^B	B, nhưng chỉ nếu kí tự đầu tiên trong chuỗi
\$	Kết thúc của chuỗi nhập	X\$	X, nhưng chỉ nếu kí tự cuối cùng trong chuỗi
.	Bất kì kí tự nào ngoại trừ kí tự xuống dòng(\n)	i.ation	isation, ization
*	Kí tự trước có thể được lặp lại 0 hoặc nhiều lần	ra*t	rt, rat, raat, raaat, and so on
+	Kí tự trước có thể được lặp lại 1 hoặc nhiều lần	ra+t	rat, raat, raaat and so on, (but not rt)
?	Kí tự trước có thể được lặp lại 0 hoặc 1 lần	ra?t	rt and rat only
\s	Bất kì kí tự khoảng trắng	\sa	[space]a, \ta, \na (\t and \n có ý nghĩa giống như trong C#)

Biểu thức chính quy (Regular Expression)

Ký tự	Mô tả
\w	Ký tự word (gồm chữ cái và chữ số, dấu gạch dưới _) tương đương [a-zA-Z_0-9]
\W	Ký tự không phải ký tự word tương đương [^a-zA-Z_0-9]
\A	Bắt đầu 1 chuỗi
\Z	Kết thúc 1 chuỗi
\S	Bất kì kí tự nào không phải là khoảng trắng
\b	Từ biên
\B	Bất kì vị trí nào không phải là từ biên
\d	Ký tự số 0-9

Biểu thức chính quy (Regular Expression)

Ký tự	Mô tả
	Ký tự ngăn cách so trùng tương đương với phép or
[abc]	Khớp với 1 ký tự nằm trong nhóm là a hay b hay c.
[a-z]	So trùng với 1 ký tự nằm trong phạm vi a-z, dùng dấu - làm dấu ngăn cách.
[^abc]	Sẽ không so trùng với 1 ký tự nằm trong nhóm, ví dụ không so trùng với a hay b hay c.
()	Xác định 1 group (biểu thức con) xem như nó là một yếu tố đơn lẻ trong pattern . Ví dụ: ((a(b)))c sẽ khớp với b, ab, abc.
{n}	n là con số, Khớp đúng với n ký tự đứng trước nó . Ví dụ A{2}: khớp đúng với 2 chữ A.
{n, }	Khớp đúng với n ký tự trở lên đứng trước nó , A{2,} khớp với AA, AAA ...
{m,n}	Khớp đúng với từ m->n ký tự đứng trước nó, A{2,4} khớp với AA,AAA,AAAA.

Biểu thức chính quy (Regular Expression)

10

- Ví dụ: Sử dụng biểu thức chính quy để:
 - Kiểm tra tiền tệ (USD).
 - Kiểm tra từ trùng nhau
 - Cắt khoảng trắng dư trong chuỗi
 - Chuyển chuỗi thành ký tự đầu mỗi câu là hoa

Biểu thức chính quy (Regular Expression)

11

- Kiểm tra tiền tệ (USD)

```
// Define a regular expression for currency values.
Regex rx = new Regex(@"^-?\d+(\.\d{2})?\s((USD) | (\$))");
// Define some test strings.
string[] tests = { "-42", "19.99 $", "0.001", "100 USD" };
// Check each test string against the regular expression.
foreach (string test in tests)
{
    if (rx.IsMatch(test))
        Console.WriteLine("{0} is a currency value.", test);
    else
        Console.WriteLine("{0} is not a currency value.", test);
}
```

```
-42 is not a currency value.
19.99 $ is a currency value.
0.001 is not a currency value.
100 USD is a currency value.
```

Biểu thức chính quy (Regular Expression)

12

- Kiểm tra từ trùng nhau

```
// Define a regular expression for repeated words.
Regex rx = new Regex(@"\b(?<word>\w+)\s+(\k<word>)\b",
    RegexOptions.Compiled | RegexOptions.IgnoreCase);
// Define a test string.
string text = "The the quick brown fox fox jumped over the lazy dog dog";
// Find matches.
MatchCollection matches = rx.Matches(text);
// Report the number of matches found.
Console.WriteLine("{0} matches found in:\n    {1}",
    matches.Count,
    text);
// Report on each match.
foreach (Match match in matches)
{
    GroupCollection groups = match.Groups;
    Console.WriteLine("' {0}' repeated at positions {1} and {2}",
        groups["word"].Value,
        groups[0].Index,
        groups[1].Index);
}
```

```
3 matches found in:
The the quick brown fox fox jumped over the lazy dog dog
'The' repeated at positions 0 and 4
'fox' repeated at positions 20 and 24
'dog' repeated at positions 49 and 53
```

```
public static string CatKhoangTrang()
```

Cắt khoảng trắng dư trong chuỗi

```
{  
    string pattern = @"\s{2,}";  
    string input = "    Xin chào    các bạn lớp CTK33";  
    return Regex.Replace(input, pattern, delegate(Match m)  
    {  
        if (m.Index == 0)  
            return "";  
        else  
            return " ";  
    });  
}
```

```
public static string CatKhoangTrang()
```

```
{  
    string pattern = @"\s{2,}";  
    string input = "    Xin chào    các bạn lớp CTK33";  
    return Regex.Replace(input, pattern, ReplaceKhoangTrang);  
}  
public static string ReplaceKhoangTrang(Match m)  
{  
    if (m.Index == 0) return "";  
    return " ";  
}
```

```
public static string ChuyenCauChuHoa()  
{
```

Chuyển chuỗi thành ký tự
đầu mỗi câu là hoa

```
    string pattern = @"(^[a-z]\w*)|(\. \s*[a-z]\w*)";  
    string input = "xin chao cac ban lop CTK33. chao. a";  
    return Regex.Replace(input, pattern, delegate(Match m)  
    {  
        string s1 = m.ToString();  
        if (m.Index == 0)  
            return char.ToUpper(s1[0]) + s1.Substring(1);  
        return s1.Substring(0, 2) + char.ToUpper(s1[2]) + s1.Substring(3);  
    });
```

Xin chao cac ban lop CTK33. Chao. A

```
public static string ChuyenCauChuHoa()  
{
```

```
    string pattern = @"(^[a-z]\w*)|(\. \s*[a-z]\w*)";  
    string input = "xin chao cac ban lop CTK33. chao. a";  
    return Regex.Replace(input, pattern, KyTuDauHoa);
```

```
}  
public static string KyTuDauHoa(Match m)
```

```
{  
    string s1 = m.ToString();  
    if (m.Index == 0)  
        return char.ToUpper(s1[0]) + s1.Substring(1);  
    return s1.Substring(0, 2) + char.ToUpper(s1[2]) + s1.Substring(3);  
}
```

Biểu thức chính quy (Regular Expression)

15

- Bài tập: Sử dụng biểu thức chính quy để:
 - Kiểm tra tên biến (ngôn ngữ: C++).
 - Kiểm tra số điện thoại di động
 - Kiểm tra địa chỉ email của yahoo hoặc gmail.
 - Loại bỏ chữ lặp lại liên tiếp.
 - Chuyển chuỗi hoa thành thường và ngược lại

Nhóm đối tượng

16

- Danh sách(tự xem lại)
 - ArrayList
 - List
- Từ điển
- Collections

Từ điển

17

- Từ điển trong .NET: Sử dụng lớp Hashtable

```
Hashtable employees = new Hashtable(53);
```



```

class Employee
{
    string Id;
    string Name;
    int Year;
    public Employee(string id, string name, int year)
    {
        this.Id = id;
        this.Name = name;
        this.Year = year;
    }
    public override string ToString()
    {
        return Id + "\t" + Name + "\t" + Year;
    }
}

Hashtable employees = new Hashtable(53);
//Insert Item to dictionary
employees.Add("001", new Employee("001", "John", 1982));
employees.Add("002", new Employee("002", "Marry", 1984));
//Get item with Key
Console.WriteLine(employees["001"] as Employee);
Console.WriteLine(employees["002"] as Employee);
//Count items in dictionary
Console.WriteLine("Count:" + employees.Count);
//remove Item with Key
employees.Remove("002");
Employee e=employees["002"] as Employee;
if(e!=null)
    Console.WriteLine(e);

```

Collection

19

- 1 đối tượng là 1 collection nếu nó có thể cung cấp 1 tham chiếu đến một đối tượng có liên quan, có thể duyệt qua từng mục trong collection.
- Đặc biệt, 1 collection phải thi hành 1 interface:
 - System.Collections.**IEnumerable**

```
interface IEnumerable
{
    IEnumerator GetEnumerator();
}
```

Collection

20

- Ngoài ra còn có một interface khác, được dẫn xuất từ IEnumerable. IEnumerator có cấu trúc sau:

```
interface IEnumerator
{
    object Current { get; }
    bool MoveNext();
    void Reset();
}
```

- Khi tạo đối tượng (collection) được tạo thì nó chưa trỏ đến phần tử nào trên collection vì vậy phải gọi **MoveNext()** để trỏ đến phần tử đầu tiên.
- Lấy phần tử gọi **Current**

Collection

21

- Ví dụ: với `string`

```
string[] MessageSet = new string[3];
MessageSet[0] = "Hello";
MessageSet[1] = "Good bye";
IEnumerator enumerator = MessageSet.GetEnumerator();
string nextMessage;
enumerator.MoveNext();
while ((nextMessage = enumerator.Current as string) != null)
{
    Console.WriteLine(nextMessage);
    // toNextMessage
    enumerator.MoveNext();
}
```

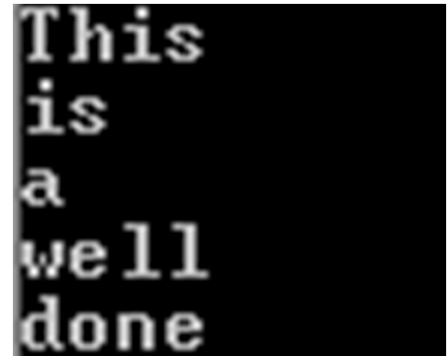
Tạo Collection cho Tokens

```
public class TokenEnumerator : IEnumerator
{
    private int position = -1;
    private Tokens t;
    public TokenEnumerator(Tokens t)
    {
        this.t = t;
    }
    // Declare the MoveNext method required by IEnumerator:
    public bool MoveNext()
    {
        if (position < t.elements.Length - 1)
        {
            position++;
            return true;
        }
        else
        {
            return false;
        }
    }
    // Declare the Reset method required by IEnumerator:
    public void Reset()
    {
        position = -1;
    }
    // Declare the Current property required by IEnumerator:
    public object Current
    {
        get
        {
            return t.elements[position];
        }
    }
}
```

```

public class Tokens : IEnumerable
{
    public string[] elements;
    Tokens(string source, char[] delimiters)
    {
        elements = source.Split(delimiters);
    }
    public IEnumerator GetEnumerator()
    {
        return new TokenEnumerator(this);
    }
}
static void Main()
{
    Tokens f = new Tokens("This is a well-done", new
        char[] { ' ', '-' });
    IEnumerator enumerator = f.GetEnumerator();
    string s;
    while (enumerator.MoveNext())
    {
        s = enumerator.Current as string;
        Console.WriteLine(s);
    }
}

```



This
is
a
well
done

```

public class Vector
{
    float a;
    float b;
    float c;
    public Vector()
    {
        a = 0;
        b = 0;
        c = 0;
    }
    public Vector(int a, int b, int c)
    {
        this.a = a;
        this.b = b;
        this.c = a;
    }
    public override string ToString()
    {
        return "("+a+", "+b+", "+c+")";
    }
}
public class Vectors : IEnumerable
{
    public Vector[] elements;
    public int num;
    public Vectors()
    {
        elements = new Vector[3];
        elements[0] = new Vector(1,2,3);
        elements[1] = new Vector(2,2,2);
        elements[2] = new Vector(3,3,3);
        num = 3;
    }
    public IEnumerator GetEnumerator()
    {
        return new VectorEnumerator(this);
    }
}

```

```

public class VectorEnumerator : IEnumerator
{
    private int position = -1;
    private Vectors v;
    public VectorEnumerator(Vectors v)
    {
        this.v = v;
    }
    public bool MoveNext()
    {
        if (position < v.num - 1)
        {
            position++;
            return true;
        }
        else return false;
    }
    public void Reset()
    {
        position = -1;
    }
    public object Current
    {
        get{ return v.elements[position];}
    }
    static void Main()
    {
        Vectors v = new Vectors();
        IEnumerator e = v.GetEnumerator();
        Vector s;
        while (e.MoveNext()) {
            s = e.Current as Vector;
            Console.WriteLine(s);
        }
    }
}

```

Tạo Collection cho vector

KẾT THÚC CHƯƠNG

25

Q&A