## 任务十 了解字符串操作

### 实验目的

理解正则表达式

掌握字符串的解析

### 第一部分 上机训练

1、下面的示例搜索输入字符串并输出所有 href="…" 的值和它们在字符串中的位置。它执行此操作的方式为，首先构造编译的 Regex 对象，然后使用 Match 对象来循环访问字符串中的所有匹配。

1）创建一个控制台应用程序

2）创建名为RegularExpressionsHandle的类，在该类中定义一个静态方法：

using System;

using System.Collections.Generic;

using System.Text;

using System.Text.RegularExpressions;

namespace DemoRegularExpressions

{

public class RegularExpressionsHandle

{

public static void DumpHrefs(String inputString)

{

Regex r;

Match m;

r = new Regex("href\\s\*=\\s\*(?:\"(?<1>[^\"]\*)\"|(?<1>\\S+))",

RegexOptions.IgnoreCase | RegexOptions.Compiled);

for (m = r.Match(inputString); m.Success; m = m.NextMatch())

{

Console.WriteLine("在原字符串的第"

+ m.Groups[1].Index + "个位置发现匹配href的字符串：\n" + m.Groups[1]);

}

}

}

}

注意using System.Text.RegularExpressions;

3）在Program.cs中填加如下代码：

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoRegularExpressions

{

class Program

{

static void Main(string[] args)

{

string href = "<a href=\"http://www.sina.com.cn\">新浪网</a>";

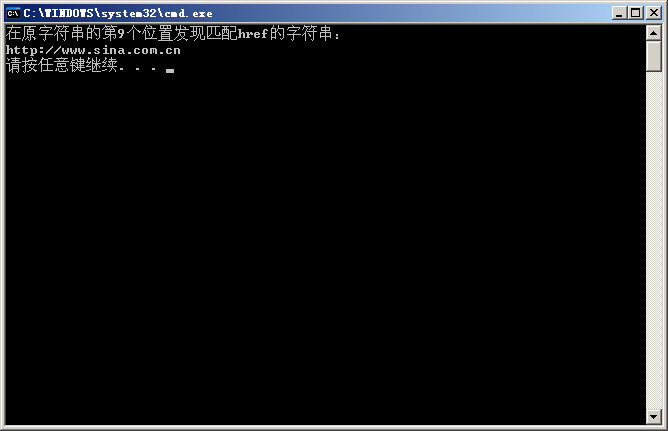
RegularExpressionsHandle.DumpHrefs(href);

}

}

}

4）按Ctrl运行程序，查看结果



2、下面的代码示例使用 Regex.Replace 方法来用 dd-mm-yy 的日期形式代替 mm/dd/yy 的日期形式

1）再在刚才建立的RegularExpressionsHandle类中继续填加静态方法：

public static String MDYToDMY(String input)

{

return Regex.Replace(input,

"\\b(?<month>\\d{1,2})/(?<day>\\d{1,2})/(?<year>\\d{2,4})\\b",

"${day}-${month}-${year}");

}

2）改写Program.cs为如下代码：

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoRegularExpressions

{

class Program

{

static void Main(string[] args)

{

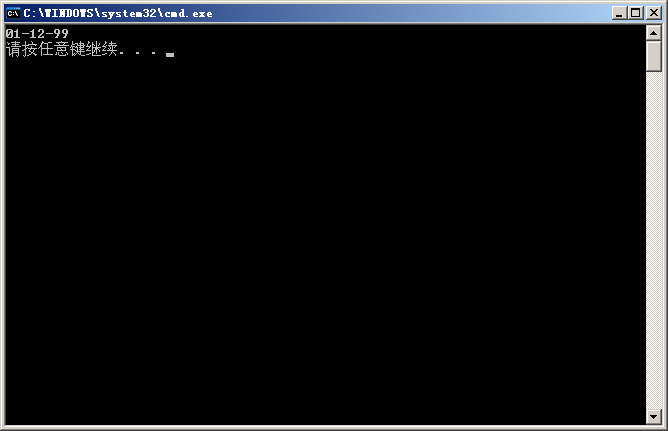
Console.WriteLine(RegularExpressionsHandle.MDYToDMY("12/01/99"));

}

}

}

3)按Ctrl+F5运行



3、下面的代码示例使用 Match.Result 来从 URL 提取协议和端口号。例如，“http://www.zjut.edu.cn:8080/index.html”将返回“http:8080”。

1）再在刚才建立的RegularExpressionsHandle类中继续填加静态方法：

public static String Extension(String url)

{

Regex r = new Regex(@"^(?<proto>\w+)://[^/]+?(?<port>:\d+)?/",

RegexOptions.Compiled);

return r.Match(url).Result("${proto}${port}");

}

2）改写Program.cs为如下代码：

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoRegularExpressions

{

class Program

{

static void Main(string[] args)

{

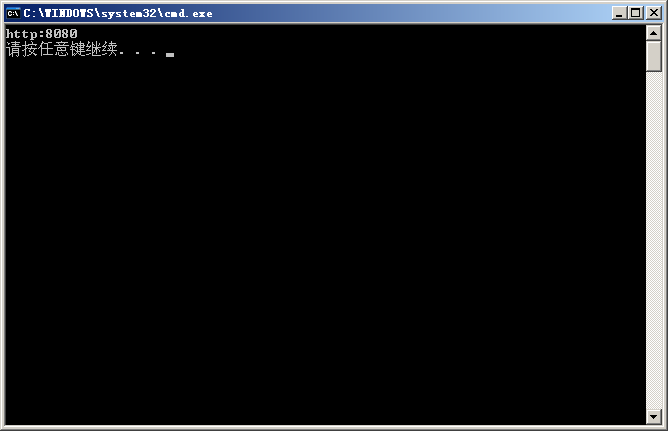
Console.WriteLine(RegularExpressionsHandle.Extension("http://www.zjut.edu.cn:8080/index.html"));

}

}

}

3)按Ctrl+F5运行



4、下面的代码示例使用静态 Regex.Replace 方法从字符串中抽出无效字符。您可以使用这里定义的 CleanInput 方法，清除掉在接受用户输入的窗体的文本字段中输入的可能有害的字符。CleanInput 在清除掉除 @、-（连字符）和 .（句点）以外的所有非字母数字字符后返回一个字符串。

1）再在刚才建立的RegularExpressionsHandle类中继续填加静态方法：

public static String CleanInput(string strIn)

{

// Replace invalid characters with empty strings.

return Regex.Replace(strIn, @"[^\w\.@-]", "");

}2）改写Program.cs为如下代码：

using System;

using System.Collections.Generic;

using System.Text;

namespace DemoRegularExpressions

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(RegularExpressionsHandle.CleanInput("mail!##\*&^@zjut"));

}

}

}

3)按Ctrl+F5运行



### 第二部分 实战提高

验证一个字符串是否为有效的电子邮件格式

代码如下：

在上述创建的类中添加静态方法MailVerify，

public static void MailVerify(String str)//验证邮箱格式

{

Regex r = new Regex("^\\s\*([A-Za-z0-9\_-]+(\\.\\w+)\*@(\\w+\\.)+\\w{2,5})\\s\*$");

if (r.IsMatch(str))

{

Console.WriteLine("{0} is definitely an email.",str);

}

else

{

Console.WriteLine("{0} is not an email.", str);

}

}

运行截图如下：



图1



图2



图3