The syntax

The Regex are therefore a succession of reasons which consist of metacharacters, of classes under .NET and aliases.

Symbole	Correspondence	Example
****	Escape character	[\.] contain an "."
^	Start the line	^b$ contain only b
.	Any character	^.$ contain one character .
$	End of line	er$ ended by "er"
|	Alternative	^(a|A) start by a or A
( )	Groups	^((a)|(er)) start by a or er
-	Range of characters	^[a-d] start by a,b,c or d
[ ]	a set of characters	[0-9] contents a number
[^]	All except a set of characters	^[^a] is not starting with a
+	1 time or more	^(a)+ start with one or more a
?	0 or 1 times	^(a)? start or not with a
*****	0 or more times	^(a)* can or not start with a
{x}	x times exact	a{2} 2 times "a"
{x,}	x times at least	a{2,} at least 2 times "a"
{x, y}	minimum x times, maximum y	a{2,4} 2, 3 or 4 times "a"

Alias	Correspondence	Equivalence
\n	Newline character	
\r	Newline character	
\t	Tab character	
\s	character space (space, tab , etc)	[\f\n\r\t\v]
\S	All but one space	[^\f\n\r\t\v]
\d	A number	[0-9]
\D	All but a number	[^0-9]
\w	One character	[a-zA-Z0-9_]
\W	All but a character	[^a-zA-Z0-9_]

```
using System.Text.RegularExpressions;
```

Examples

A string that contains letters from a to d or any uppercase letters: [a-dA-Z]
String starting (and ending) by y or z: ^(y | z)$
String containing no figure: [^0-9] or @[^\d] or [^\\d]
String containing the digits 1 or 2 or symbol ^ : [12\\^] or @[12\^]

You can use several metacharacters: ^[pP]hara(onix)?$
String containing: phara, Phara, pharaonix or Pharaonix

Regex methods: IsMatch(), Replace() or Split()

The IsMatch method is typically used to validate a string or to ensure that a string conforms to a particular pattern without retrieving that string for subsequent manipulation.(true/false)

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

Method IsMatch()is returning true or false
method RegexObject.IsMatch(string)

The Replace method in a specified input string, replaces strings that match a regular expression pattern with a specified replacement string.

```
public string Replace( string input, string replacement )
```

Method Replace()is replacing an sub-string following a specific pattern
method Regex.Replace(string, pattern, stringToReplace)

Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the Regex constructor

```
public string[] Split( string input)
```

Finding and replacing matched patterns

To	Use method
Validate match	Regex.IsMatch
Retrieve single match	Regex.Match (first) Match.NextMatch (next)
Retrieve all matches	Regex.Matches
Replace match	Regex.Replace
Divide text	Regex.Split
Handle char escapes	Regex.Escape Regex.Unescape

```
using System.IO;
```

Common methods of the Directory class

- **Exists(path)**
- **CreateDirectory(path)**
- **Delete(path)**
- **Delete(path, recursive)**

Common methods of the File class

- **Exists(path)**
- **Delete(path)**
- **Copy(source, dest)**
- **Move(source, dest)**

System.IO classes used to work with files and streams

- **FileStream**
- **StreamReader**
- **StreamWriter**
- **BinaryReader**
- **BinaryWriter**

Members in the FileMode enumeration

- **Append**
- **Create**
- **CreateNew**
- **Open**
- **OpenOrCreate**
- **Truncate**

Members in the FileAccess enumeration

- **Read**
- **ReadWrite**
- **Write**

Common methods of the StreamWriter class

Method	Description
`Write(``data`**`)`**	Writes the data to the output stream.
`WriteLine(``data`**`)`**	Writes the data to the output stream and appends a line terminator (usually a carriage return and a line feed).
`Close()`	Closes the StreamWriter object and the associated FileStream object.

Common methods of the StreamReader class

Method	Description
`Peek()`	Returns the next available character in the input stream without advancing to the next position. If no more characters are available, this method returns -1.
`Read()`	Reads the next character from the input stream.
`ReadLine()`	Reads the next line of characters from the input stream and returns it as a string.
`ReadToEnd()`	Reads the data from the current position in the input stream to the end of the stream and returns it as a string.

The exception classes for file I/O

- **IOException**
- **DirectoryNotFoundException**
- **FileNotFoundException**
- **EndOfStreamException**

```
using System.Xml;
```

Common methods of the XmlWriter class

Create(path**)**
Create(path, settings**)**
WriteStartDocument()
WriteComment(comment**)**
WriteStartElement(elementName**)**
WriteAttributeString(attributeName, value**)**
WriteEndElement()
WriteElementString(elementName, content**)**
Close()

Common properties of the XmlWriterSettings class

Indent
IndentChars

Examples

```
// create the XmlWriterSettings object
XmlWriterSettings settings = new XmlWriterSettings();
settings.Indent = true; settings.IndentChars = ("    ");

// create the XmlWriter object using settings object (indent)
XmlWriter xmlOut = XmlWriter.Create(dir + "FileName.xml", settings);

    xmlOut.WriteStartDocument(); // write the start of the document
    xmlOut.WriteStartElement("Root"); //start root element

      xmlOut.WriteStartElement("Child"); //content element
        xmlOut.WriteElementString("ElementTag1", elementValue1);//adding element tag 1 + value
        xmlOut.WriteElementString("ElementTag2", elementValue2);//adding element tag 2 + value
      xmlOut.WriteEndElement(); // write the end tag for the Child element

    xmlOut.WriteEndElement(); // write the end tag for the root element

xmlOut.Close();// close the XmlWriter object
```

Common indexer of the XmlReader class

[name]

Common properties of the XmlReader class

NodeType
Name
Value
EOF

Common methods of the XmlReader class

Create(path)
Create(path, settings)
Read()
ReadStartElement(name)
ReadEndElement()
ReadToDescendant(name)
ReadToNextSibling(name)
ReadElementContentAsString()
ReadElementContentAsDecimal()
Close()

Common properties of the XmlReaderSettings class

IgnoreWhitespace
IgnoreComments

Examples

```
// create the XmlReaderSettings object
  XmlReaderSettings settings = new XmlReaderSettings();
  settings.IgnoreWhitespace = true;
  settings.IgnoreComments = true;

// create the XmlReader object using settings object
XmlReader xmlIn = XmlReader.Create(dir + "FileName.xml", settings);

// read past all nodes to the first UserName node
  if (xmlIn.ReadToDescendant("Child"))
   {
     // create Element1 and Element2 string for each Child node
     string Element1 = "", Element2 = "", tempStr = "";
     do{
       xmlIn.ReadStartElement("Child");
       Element1 = xmlIn.ReadElementContentAsString();
       Element2 = xmlIn.ReadElementContentAsString();
       tempStr = Element1 + ", " + Element2 + "\n";
     }while (xmlIn.ReadToNextSibling("Child"));
   }
   MessageBox.Show(tempStr); //Show the elements of all the Childs in one MessageBox
// close the XmlReader object
xmlIn.Close();
```