Les expressions régulières

1. Définition

Les expressions régulières (regex ou expression rationnelle) représentent un système très élaboré et très puissant, permettant de trouver (retrouver, coupler, assortir) des motifs et de traiter (récupérer, extraire, remplacer) des éléments à l'intérieur d'une chaîne de caractères.
Elles sont utilisées essentiellement pour :

- Vérifier la validité des chaînes de caractères (email, entrée chiffrée, entrée texte, etc.)
- Extraire des parties d'une chaîne, d'un texte, d'une page, etc.

Une expression régulière se compose de caractères littéraux et de caractères spéciaux ou métacaractères (caractères ayant une signification particulière).

2. Les caractères spéciaux et les alias

Symbole	Correspondance	Exemple
\	Caractère d'échappement	[\.] contient un "."
^	Début de ligne	^b$ contient uniquement b
.	N'importe quel caractère	^.$ contient un seul caractère
$	Fin de ligne	er$ finit par "er"
|	Alternative	^(a|A) commence par a ou A
( )	Groupement	^((a)|(er)) commence par a ou er
-	Intervalle de caractères	^[a-d] commence par a,b,c ou d
[ ]	Ensemble de caractères	[0-9] contient un chiffre
[^]	Tout sauf un ensemble de caractères	^[^a] ne commence pas par a
+	1 fois ou plus	^(a)+ commence par un ou plusieurs a
?	0 ou 1 fois	^(a)? commence ou non par un a
*	0 fois ou plus	^(a)* peut ou non commencer par a
{x}	x fois exactement	a{2} deux fois "a"
{x,}	x fois au moins	a{2,} deux fois "a" au moins
{x, y}	x fois minimum, y maximum	a{2,4} deux, trois ou quatre fois "a"

Alias	**Correspondance**	**Equivalence**
\n	Caractère de nouvelle ligne	
\r	Caractère de retour à la ligne	
\t	Caractère de tabulation	
\s	Caractère d'espacement (espace, tabulation, saut de page, etc)	[\f\n\r\t\v]
\S	Tout ce qui n'est pas un espacement	[^\f\n\r\t\v]
\d	Un chiffre	[0-9]
\D	Tout sauf un chiffre	[^0-9]
\w	Un caractère alphanumérique	[a-zA-Z0-9_]
\W	Tout sauf un caractère alphanumérique	[^a-zA-Z0-9_]
\n	Caractère en octal ex:\001 ==> " 1 "	
\xn	Caractère en hexadécimal ex: \x41 ==>" A "	

3. Déclaration d'une expression régulières

Les Regex nécessitent l'utilisation de l'espace de noms **System.Text.RegularExpressions**

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

```
Regex uneExpression = new Regex(@"^([\w]+)@([\w]+)\.([\w]+)$");
```

les Regex sont à indiquer entre guillemets, si celles-ci comprennent des antislashs (" \ "), il faut alors les doubler. Néanmoins, il existe une astuce utile ici, qui consiste à placer le symbole @ avant les guillemets. Ainsi : @"^[\t]$" est équivalent à "^[\\t]$"

4. Utilisation dans les contrôles de données saisies côté client

Pour vérifier si le contenu d'une variable saisie dans un formulaire, il est souvent utile de recourir à une expression régulière.
La classe RegExp dispose de la méthode IsMatch qui retourne vrai si la valeur passée en paramètre respecte l'expression

```
private bool emailValide(string adresse) {
   Regex  uneExpression = new Regex(@"^([\w]+)@([\w]+)\.([\w]+)$");
   return uneExpression.IsMatch(adresse);
}
```

4.1. Contrôle d'un email

```
uneExpression = new Regex(@"^[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*@[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])*[.][a-zA-Z]{2,3}$");
```

Traduction :
^[0-9a-zA-Z] : commencer par une lettre ou un chiffre
([-_.]? : suivi par éventuellement un tiret ou un under score ou un point
[0-9a-zA-Z])* : suivi par éventuellement des lettres et des chiffres
@ : Ensuite on doit trouver la présence d'un @
[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])* : suivi par un motif identique à ce que l'on peut trouver devant l'arobase
[.][a-zA-Z]{2,3}$/ : La chaîne doit se terminer par un point suivi de deux ou trois lettres représentant l'extension.

On peut proposer une autre expression

```
^\w+([\.-_]?\w+)*@\w+([\.-_]?\w+)*(\.\w{2,3})+$
```

Traduction :
^\w+ ➔ doit commencer par au moins un lettre ou chiffre
[\.-]?\w+ ➔ Suivi éventuellement d'un . ou d'un tiret et d'au moins une lettre
([\.-]?\w+)* ➔ Le groupe défini par un . ou un – éventuel et d'au moins une lettre peut être répété 0 ou plusieurs fois
@ ➔ Un @ puis la même chose qu'avant l'@
\.\w{2,3})+$/ ➔ ce groupe doit se terminer par un '.' suivi de deux ou trois lettres (extension représentant le nom du domaine, deux lettres pour les domaines géographique .fr, … et trois lettres pour les domaines génériques .com .net).

4.2. Contrôle d'une URL

```
^(http|https):\/\/(\w+:{0,1}\w*@)?(\S+)(:[0-9]+)?(\/|\/([\w#!:.?+=&%@!\-\/]))?
```

Traduction :

^(http|https):\/\/ : commencer par http ou https suivi de ://
(\w+:{0,1}\w*@)? : suivi éventuellement par au moins un caratère
[0-9a-zA-Z])* : suivi par éventuellement des lettres et des chiffres
@ : Ensuite on doit trouver la présence d'un @
[0-9a-zA-Z]([-_.]?[0-9a-zA-Z])* : suivi par un motif identique à ce que l'on peut trouver devant l'arobase
[.][a-zA-Z]{2,3}$/ : La chaîne doit se terminer par un point suivi de deux ou trois lettres représentant l'extension.

5. Remplacement d'un motif basé sur une expression régulière dans un chaine

Les expressions régulières sont aussi très utiles pour effectuer une opération de "rechercher/remplacer" à l'intérieur d'une chaîne.
La méthode Replace applicable sur une chaine de caractères accepte une expression régulière comme motif :

```
Regex uneExpression = new Regex(@"\s{2,}");
string nomSansEspaceSuperflu = uneExpression.Replace(nom, " ");
```

Dans cet exemple, toutes les séquences d'au moins deux espaces consécutifs sont remplacés par un seul espace

L'utilisation de parenthèses est possible pour capturer certaines parties de la chaîne :

```
private string ToHtmlMail(string adresse) {
   Regex  myRegex = new Regex(@"([\w\-.]+@[\w\-.]+)");
   return myRegex.Replace(adresse, "<a href=\"mailto:$1\" >$1a>");
}
```

6. Webographie

- http://lgmorand.developpez.com/dotnet/regex/