Regex – Théorie

# Recapitulatif

Le terme Regex vient de l’anglais **Reg**ular **Ex**pression qui est, en français, traduit par expression régulière. Les regex sont utilisé plus que vous ne le penser : formulaire, jeu, forum, etc.

Elles permettent de contrôler des champs pour les valider par exemple ou encore filtrer certain mot !

# comment les utiliser

## C#

Il faudra ajouter le using :

« using System.Text.RegularExpressions; »

Ensuite il faudra créer la variable regex :

« Regex NomDeLaRegex = new Regex("METTRE LA REGEX ICI"); »

Puis, à l’endroit où on veut contrôler la valeur, il faudra créer une variable :

« bool NomDeVariable = NomDeLaRegex.IsMatch("ValeurAAnalyser"); »

Vous mettrez ensuite la variable bool dans un if et suivant l’information vous ferez qqch

## PHP

Créer une variable regex :

« **$NomDeLaRegex =** "votreRegex" »

Il faudra juste mettre la ligne dans un if :

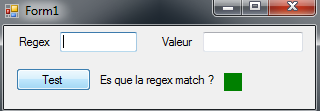
« **preg\_match** ( string $NomDelaRegex , string $ValeurAAnalyser)»

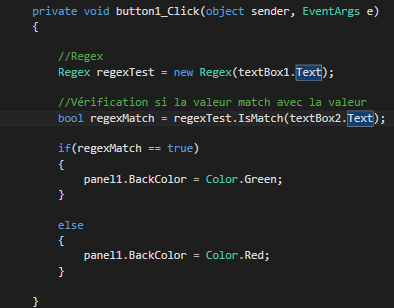
# Comment créer une regex

Vous allez à présent créer votre première regex. Celle-ci n’a pas besoin d’être compliquée, entre les guillemets, placez juste le un mot, ex : « cheval ». Si la valeur à analyser a comme valeur « cheval » celle-ci va match !

Si vous n’avez rien compris, c’est pas grave. Vous allez comprendre par la suite.

Si on utilise l’exemple suivant :





Dès que la regex matchera avec la valeur à tester on affichera une pastille verte sinon, elle sera rouge.

## Regex simple

Comment créer une regex simple ? Ce n’est pas compliqué, vous en avez déjà une dans l’exemple juste en dessus ; « cheval » est une regex mais elle n’a beaucoup d’utilité à part si vous voulez absolument « cheval » dans votre formulaire

|  |  |  |
| --- | --- | --- |
| Regex | Valeur à tester | Résultat |
| cheval | cheval |  |
| cheval | Cheval |  |
| cheval | Un cheval |  |

Il faut faire attention aux majuscules, minuscules. Les regex sont case sensitive (sensible à la casse).

Et aussi dans le cas où la regex n’est que « cheval », la valeur à tester doit se composer uniquement de la valeur « cheval » sinon ça ne matchera pas ! Nous verrons par la suite comment faire des

## Les Classes

Les classes sont comme des OU dans l’électronique, elles nous permettent de ne pas recréer des regex pour chaque variable. Regardez !

|  |  |  |
| --- | --- | --- |
| Regex | Valeur à tester | Résultat |
| [abc] | a |  |
| [abc] | cheval |  |
| [abc] | Un poney |  |

Comme on peut le voir sur l’exemple « cheval », la regex match dès qu’il a une valeur qui correspond. Si on écrit « kebab », la regex matchera dès le premier b.

## Les intervalles de classes

Les intervalles de classes permettent de raccourcir nos classes. Par exemple, si on veut une regex pour vérifier l’alphabet, en majuscules et minuscules, votre regex devrait ressembler à ça (enfin si vous avez juste suivit le document) : [abcdefghijklmnopqrstuvwyzABCDEFGHIJKLMNOPQRESTUVWXYZ]. Cette méthode fonctionne mais il y a une méthode beaucoup plus courte et moins ennuyeux à coder.

|  |  |  |
| --- | --- | --- |
| Regex | Valeur à tester | Résultat |
| [a-zA-Z] | Cette phrase matchera |  |
| [a-zA-Z1-8] | 9 |  |
| [a-zA-Z] | 1818 |  |

## Les exclusions

Pour exclure