# Stringy, regulární výrazy a parsování textu

*Předmluva: Nejdříve si přečtěte první stranu, poté si můžete vyzkoušet metody, které jsou zmíněné na straně 2. Regulární výrazy je ideální si v programování vyzkoušet (Visual Studio), a tím pochopit, jak fungují.*

*Vhodné videa pro pochopení regulárních výrazů:*

[*AJ video - 40m všechno vysvětlí*](https://www.youtube.com/watch?v=sa-TUpSx1JA)

[*AJ video z kterého jsem se učil já - 20m*](https://www.youtube.com/watch?v=rhzKDrUiJVk)

*Aleksandr Ščančar*

String

**String je řetězec znaků(chars), to znamená, že se objekt String chová jako pole. Lze na něj uplatnit hledání podle indexu(písmenka-znaku) jako v poli.**

**String je jediný objekt u kterého není potřeba při vytvoření nové instance objektu zadat příkaz new**, namísto toho je možné zadat přímo text z obou stran oddělený uvozovkami.

string s1 = new String(“asda”);

string s4 = new String(“asda”);

string s2 = “asd”;

string s3 = “asd”;

Bez new ukazují s2 a s3 na ten samý prvek na haldě. s1 a s4 mají díky new svůj vlastní vytvořený prvek.

String je **neměnný objekt** – po vytvoření instance objektu String jej není možné měnit. Třída sice obsahuje několik metod, které by mohly vypadat že objekt pozmění, ty ale vytvoří nový objekt a nahradí jím ten starý.

Parsování stringu

***práce jako s polem***

Se stringem můžeme pracovat jako s polem a tedy vybírat si jednotlivé znaky pomocí [index]

String text = "one\two three:four,five six seven";

Console.WriteLine(text[2]);//vrátí "e"

***text.*Split("regex")(char podle čeho dělí ‘,’)**

Tato metoda rozdělí text ve všech místech, kterým odpovídá regulární výraz. Jednotlivé rozdělené stringy vrací v podobě pole String[]. Regulární výraz není zahrnut do pole stringů.

//příklad 1

String text2 = "4F,48X,6T,47Q";

String[] splitText = text2.Split(",");

foreach (string s in splitText)

{

Console.WriteLine(s);

}

//příklad 2

char[] delimiterChars = { ' ', ',', '.', ':', '\t' };

String text = "one\two three:four,five six seven";

Console.WriteLine($"Original text: '{text}'");

String [] words = text.Split(delimiterChars);

Console.WriteLine($"{words.Length} words in text:");

foreach (var word in words)

{

System.Console.WriteLine($"{word}");

}

***text.*Replace("textInput","replacement")**

Tato metoda nahradí všechny zmínky daného stringu stringem novým. Metoda má minimálně dva vstupní parametry – string na úpravu a string, který jej má nahradit. Metoda vrací nový String(String je neměnný objekt, musí být proto vytvořen string nový).

String text3 = "20/04/2020";

text3 = text3.Replace(‘/’ , ‘.’);

Console.WriteLine(text3);https://puu.sh/vKy4u/b96aa865b5.png

**Regulární výrazy**

Regulární výrazy vznikly z důvodu potřeby práce s textovými řetězci určitým unifikovaným způsobem. Základní myšlenkou je porovnání textového řetězce s předpřipraveným vzorem. Nejčastěji se zjišťuje, zda řetězec odpovídá vzoru, lze také rozdělit řetězec na části (rozdělovačem je zadaný vzor) nebo části řetězce odpovídajícího vzoru nahradit jiným řetězcem.

Jednotlivé konkrétní znaky

|  |  |
| --- | --- |
| x | Konkrétní znak |
| [abc] | Jednoduchá **množina**(jeden ze znaků *a b c*) |
| [^abc] | Žádný ze znaků *abc*(**negace**) |
| [a-zA-Z] | **Rozsah** znaků |
| . (tečka) | **Libovolný znak** s výjimkou konce řádku |
| ^fugo | Najde „*fugo*“ které musí být **na začátku řádku** |
| fugo$ | Najde „*fugo*“ které musí být **na konci řádku** |
| X|Z | Najde *X* **nebo** *Z* |
| XZ | Najde *X* **následované** písmenem *Z* |
| [abc][vz] | Najde *a* nebo *b* nebo *c* následované písmenem *v* nebo *z* |

Zástupné znaky

|  |  |
| --- | --- |
| \d | Jakákoliv **číslice** |
| \D | Cokoliv kromě číslic(negace) |
| \s | Jakýkoliv bílý znak (mezera) |
| \S | Cokoliv kromě bílých znaků (mezer) |
| \w | Libovolný písmenný znak(vč. číslic) |
| \W | negace \w |

Znaky pro vyjádření počtu

|  |  |
| --- | --- |
| \* | Vyskytne se „nulakrát“ a více, zkratka pro {0,} |
| + | Vyskytne se jednou a více, zkratka pro {1,} |
| ? | Vyskytne se jednou nebo vůbec, zkratka pro {0,1} |
| {n} | Vyskytne se n-krát |
| {n,m} | Vyskytne se v rozmezí od n do m – krát |

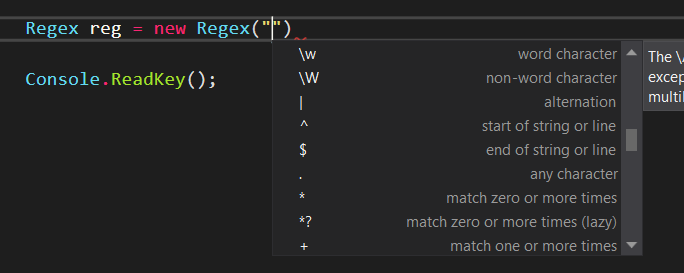
Třída Regex

Pro složitější operace s regulárními výrazy se používá třída Regex

String telefon = "+420 616 693 030";

Regex r = new Regex(@"^(\+420|\+421) ?[0-9]{3} ?[0-9]{3} ?[0-9]{3}$");

if (r.IsMatch(telefon))https://puu.sh/vKyN2/81189df7a8.png



Příklady užitečných reg. výrazů

**Devítimístné telefonní číslo s CZ nebo SK předvolbou**

Regex r = new Regex(@"^(\+420|\+421) ?[0-9]{3} ?[0-9]{3} ?[0-9]{3}$");

**IP adresa**

Regex r = new Regex(@"^((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))\.((2[0-5][0-5])|(0?[0-9][0-9])|((0{2})?[0-9])|(1[0-9][0-9]))$");