

Základy regulárních výrazů

Pokud vám pojem *regulární výrazy* skutečně mnoho neříká, může vám jako první přiblížení pomoci srovnání se zástupnými znaky `*` a `?`, které můžete znát už z DOSu (ikdyž za chvíli mladší ročníky už nebudou vědět o čem že to tu mluvím;-)) nebo všelijakých jiných nástrojů. Regulární výrazy disponují mnohem více speciálními znaky než `*` a `?`. Speciální znaky v regulárních výrazech nazýváme **metaznaky**.

V regulárních výrazech existuje několik typů metaznaků.

- Velmi důležitým metaznakem je `.` (tečka) – ta zastupuje právě jeden **libovolný znak**.
- Velmi významnou skupinou metaznaků jsou **kvantifikátory** (quantifiers) – ty určují kolikrát se smí opakovat znak předcházející kvantifikátoru. Na jednotlivé kvantifikátory se podíváme níže.
- Pokud chceme v určitém místě povolit právě jeden ze **skupiny znaků**, uzavřeme takovou skupinu znaků do hranatých závorek (`[]`). Pokud chceme naopak definovat, že v daném místě smí být libovolný znak s výjimkou několika určitých znaků, uzavřeme „zakázané“ znaky opět do hranatých závorek a za `[]` ještě doplníme `^`. Zápis `[^ab]`, tak bude znamenat, že se v daném místě má vyskytovat jeden libovolný znak kromě znaků `a` a `b`.
Pro často používané skupiny znaků existují speciální zkratky (například zápis `\d` zastupuje všechny číslice 0-9) – viz tabulku níže.
Pokud chceme definovat skupinu znaků, které následují v abecedě (resp. přesněji v tabulce znaků) za sebou, můžeme je zapsat jako interval, například `[1-5]`, `[a-e]` nebo třeba `[A-Z]`
- Pokud potřebujeme zajistit, opakování určité sekvence znaků (ne jen znaku jednoho), můžeme sekvenci znaků uzavřít do závorek (`()` a `)`) a pokud za pravou kulatou závorku doplníme kvantifikátor, bude se počet opakování vztahovat na celou sekvenci znaků uzavřenou do závorek.
- Pokud chceme dát na výběr několik variant textu (třeba `Petr` nebo `Pavel`), jako oddělovač variant použijeme metaznak `|` (výraz bude tedy `Petr|Pavel`)
- Pokud chceme přikázat, že hledaný textový řetězec se musí nacházet na začátku nebo konci prohledávaného textu, použijeme metaznaky, které nazýváme **hranice** (boundaries) nebo **ukotvení** (anchors) – na různé typy hranic se podíváme níže.

Následující tabulky zobrazují seznam kvantifikátorů a hranic.

Kvantifikátory

Kvantifikátor	Počet opakování
<code>?</code>	minimálně 0krát, maximálně 1krát

*	minimálně 0krát (maximálně neomezeno)
+	minimálně 1krát (maximálně neomezeno)
{ n }	právě nkrát
{ m, n }	minimálně mkrát, maximálně nkrát
{ m, }	minimálně mkrát (maximálně neomezeno)

Pro pořádek ještě doplním, že existují tzv. **liné kvantifikátory** – ty se od výše zmíněných (tzv. nenasytných) liší v zápisu tak, že výše uvedený kvantifikátor zprava doplníme o otazník (?). Liné kvantifikátory tedy budou `??`, `*?`, `+?`, `{m,n}?` a `{m,}?`.

Funkčně se budou liné kvantifikátory (od v tabulce uvedených nenasytných kvantifikátorů) lišit v tom, že pomocí líných kvantifikátorů je zachycen minimální počet znaků, které je třeba zachytit, aby došlo ke shodě s regulárním výrazem. Nenasytné kvantifikátory naopak zachytí co možná největší počet znaků vstupního textu.

Předdefinované skupiny znaků

<code>\d</code>	čísllice <code>0-9</code>
<code>\D</code>	jakýkoliv znak kromě číslic <code>0-9</code>
<code>\w</code>	znaky „slova“ (ekvivalentní zápisu <code>[a-zA-Z0-9_]</code>)
<code>\W</code>	jakýkoliv znak kromě znaků „slova“ (ekvivalentní zápisu <code>[^a-zA-Z0-9_]</code>)
<code>\s</code>	„bílé“ znaky (mezera, tabulátor, znaky pro zalomení řádků)
<code>\S</code>	jakýkoliv znak kromě „bílých“ znaků

Hranice

<code>^</code>	začátek řetězce (textu v němž se vyhledává)
<code>\$</code>	konec řetězce (textu v němž se vyhledává)

Původní význam speciálních znaků (metaznaků)

Možná vás napadlo, že když určité znaky (metaznaky) mají v regulárním výrazu zvláštní význam, jak je možné takový znak zapsat tak, aby nebyl chápán jako metaznak, ale jako obyčejný znak (třeba plus či hvězdička). Řešení je prosté – stačí před inkriminovaný znak doplnit v regulárním výrazu zpětné lomítko `\`. Pokud chcete například pomocí regulárního výrazu popsat rovnici $(a+b) * c = d$, je třeba použít regulární výraz `(a\b+ b) \ * c = d`.

Které znaky je třeba doplnit oním zpětným lomítkem (tzv. escapovat)? Mezi metaznaky patří `\`, `^`, `$`, `.`, `[`, `]`, `|`, `(`, `)`, `?`, `*`, `+`, `{`, `}`.

Příklady

Použití všech zmíněných metaznaků nejlépe pochopíte na několika příkladech.

Regulární výraz	Odpovídá...
<code>a+</code>	sekvence písmen <code>a</code> (1 a více znaků)
<code>a*</code>	sekvence písmen <code>a</code> (0 a více znaků)
<code>o?kov</code>	<code>okov</code> či <code>kov</code>
<code>tel(efon)?</code>	<code>tel</code> či <code>telefon</code>
<code>telef(on ax)</code>	<code>telefon</code> či <code>telefax</code>
<code>[0-9] [1-9][0-9]</code>	čísla 0 až 99
<code>\d{2}</code>	sekvence dvou číslic desítkové soustavy (<code>00</code> , <code>01</code> , ..., <code>98</code> , <code>99</code>)
<code>[0-9a-fA-F] [1-9a-fA-F][0-9a-fA-F]+</code>	hexadecimální čísla
<code>(19 20)\d{2}</code>	letopočty 1900-2099
<code>\d{2,6}</code>	sekvence dvou až šesti číslic
<code>[^ ,.]+</code>	neprázdná sekvence znaků mezi nimiž nesmí být mezera (<code> </code>), čárka (<code>,</code>) či tečka (<code>.</code>)
<code>^P.*</code>	řetězec, který začíná písmenem <code>P</code> za nímž následuje libovolný (i nulový) počet libovolných znaků
<code>\d+0\$</code>	řetězec, který končí znakem <code>0</code> (nula), kterému předchází minimálně jedna číslice
<code>a+b</code>	<code>ab</code> , <code>aab</code> , <code>aaab</code> atd.
<code>a\b+</code>	<code>a+b</code>

Poznámky závěrem

- Popsané konstrukce odpovídají regulárním výrazům vycházejícím z Perlu – budou tedy fungovat především v Perlu, .NETu, PHP (při použití Perl-compatible regular expressions funkcí) a Javascriptu.
- V tomto úvodním článku nebyly zmíněny pokročilejší konstrukce regulárních výrazů jako zpětné odkazy (backreferences), modifikátory (modifiers), pokročilejší hranice, komentáře, tvrzení (assertions) nebo podmíněné subvýrazy (conditional subexpressions). Některé pokročilejší konstrukce jsou závislé na konkrétní implementaci regulárních výrazů v daném programovacím jazyce, proto jsou popsány ve zvláštních tutoriálech (seriálech článků).