Category Escapes

A category escape matches a character from a set specified by a property or using a block:

\p indicates match any character in the set.

\P indicates match any character not in the set.

Categories and Properties

Any character can be matched by its properties using a category escape consisting of a Category code followed by an optional Property code:

\p{L}	Any Letter
\p{Lu}	Any Upper-case Letter
\p{Ll}	Any Lower-case Letter
\p{Lt}	Any Title-case Letter
\p{Lm}	Any Letter Modifier
\p{Lo}	Any "Other" Letter
\p{M}	Any Mark
\p{Mn}	Any Non-Spacing Mark
\p{Mc}	Any Combining Mark
\p{Me}	Any Enclosing Mark
\p{N}	Any Digit
\p{Nd}	Any Decimal Digit
\p{Nl}	Any Letter Digit
\p{No}	Any "Other" Digit
\p{P}	Any Punctuation Character
\p{Pc}	Any Connector Character
\p{Pd}	Any Dash Character
\p{Ps}	Any Open Character
\p{Pe}	Any Close Character
\p{Pi}	Any Initial Quote Character
\p{Pf}	Any Final Quote Character
\p{Po}	Any "Other" Punctuation
\p{Z}	Any Separator Character
\p{Zs}	Any Space Separator
\p{Zl}	Any Line Separator
\p{Zp}	Any Paragraph Separator
\p{S}	Any Symbol Character
\p{Sm}	Any Math Symbol
\p{Sc}	Any Currency Symbol
\p{Sk}	Any Modifier Symbol
\p{So}	Any "Other" Symbol
\p{C}	Any "Other" Character
\p{Cc}	Any Control Character
\p{Cf}	Any Format Character
\p{Co}	Any Private Use Character
\p{Cn}	Any "Not Assigned" Character

Character Blocks

Any character within a Unicode character block can be matched using a category escape consisting of "Is" followed by the block's name. For example: **\p{IsBasicLatin}**

Block Start	Block End	Block Name
0000	007F	BasicLatin
0080	00FF	Latin-1Supplement
0100	017F	LatinExtended-A
0180	024F	LatinExtended-B
0250	02AF	IPAExtensions
02B0	02FF	SpacingModifierLetters
0300	036F	CombiningDiacriticalMarks
0370	03FF	Greek
0400	04FF	Cyrillic
0530	058F	Armenian
0590	05FF	Hebrew
0600	06FF	Arabic
0700	074F	Syriac
0780	07BF	Thaana
0900	097F	Devanagari
0980	09FF	Bengali
0A00	0A7F	Gurmukhi
0A80	0AFF	Gujarati
0B00	0B7F	Oriya
0B80	0BFF	Tamil
0C00	0C7F	Telugu
0C80	0CFF	Kannada
0D00	0D7F	Malayalam
0D80	0DFF	Sinhala
0E00	0E7F	Thai
0E80	0EFF	Lao
0F00	0FFF	Tibetan
1000	109F	Myanmar
10A0	10FF	Georgian
1100	11FF	HangulJamo
1200	137F	Ethiopic
13A0	13FF	Cherokee
1400	167F	UnifiedCanadianAboriginalSyllabics
1680	169F	Ogham
16A0	16FF	Runic
1780	17FF	Khmer
1800	18AF	Mongolian
1E00	1EFF	LatinExtendedAdditional
1F00	1FFF	GreekExtended
2000	206F	GeneralPunctuation
2070	209F	SuperscriptsandSubscripts
20A0	20CF	CurrencySymbols
20D0	20FF	CombiningMarksforSymbols
2100	214F	LetterlikeSymbols
2150	218F	NumberForms
2190	21FF	Arrows
2200	22FF	MathematicalOperators
2300	23FF	MiscellaneousTechnical
2400	243F	ControlPictures
2440	245F	OpticalCharacterRecognition
2460	24FF	EnclosedAlphanumerics
2500	257F	BoxDrawing
2580	259F	BlockElements
25A0	25FF	GeometricShapes
2600	26FF	MiscellaneousSymbols
2700	27BF	Dingbats
2800	28FF	BraillePatterns
2E80	2EFF	CJKRadicalsSupplement
2F00	2FDF	KangxiRadicals
2FF0	2FFF	IdeographicDescriptionCharacters
3000	303F	CJKSymbolsandPunctuation
3040	309F	Hiragana
30A0	30FF	Katakana
3100	312F	Bopomofo
3130	318F	HangulCompatibilityJamo
3190	319F	Kanbun
31A0	31BF	BopomofoExtended
3200	32FF	EnclosedCJKLettersandMonths
3300	33FF	CJKCompatibility
3400	4DB5	CJKUnifiedIdeographsExtensionA
4E00	9FFF	CJKUnifiedIdeographs
A000	A48F	YiSyllables
A490	A4CF	YiRadicals
AC00	D7A3	HangulSyllables
E000	F8FF	PrivateUse
F900	FAFF	CJKCompatibilityIdeographs
FB00	FB4F	AlphabeticPresentationForms
FB50	FDFF	ArabicPresentationForms-A
FE20	FE2F	CombiningHalfMarks
FE30	FE4F	CJKCompatibilityForms
FE50	FE6F	SmallFormVariants
FE70	FEFE	ArabicPresentationForms-B
FEFF	FEFF	Specials
FF00	FFEF	HalfwidthandFullwidthForms
FFF0	FFFD	Specials

XSLT 2.0:
http://www.w3.org/TR/xslt20/

XQuery 1.0:
http://www.w3.org/TR/xquery/

XPath 2.0:
http://www.w3.org/TR/xpath20/

Unicode:
http://www.unicode.org

Regular Expression Examples

^[A-Za-z]
An Ascii letter at the start of a string or line.

^\p{Lu}
An upper-case Unicode letter at the start of a string or line.

\.$
A period at the end of a string or line.

\p{IsGreek}+
One or more Greek letters.

\p{IsGreek}{1,}
One or more Greek letters.

.*?;
Up to and including the next semicolon.

.*;
Up to and including the last semicolon.

^\c+$
Match only if the string consists entirely of XML name characters.

[-~-[\[\]]]+
Any Ascii printable character except the square brackets.

\w+
A "word".

[^\s]+
Non-white-space characters.

\S+
Non-white-space characters.

(["'])(.*?)\1
A string delimited by single or double quotes. $2 or regex-group(2) will return the unquoted substring. (\1 is the quote character used.)

\s*(\i\c*)\s*=\s*(["'])(.*?)\2
An XML-attribute-like name, equal and quoted value (with optional leading and intervening white space). $1 is the name and $3 is the value.

\((\d+|\p{L}+)\)
A parenthesized sequence either of digits or of letters (but not a mixture of both).

\p{Sc}(\d+(\.\d*)?|\.\d+)
A decimal number with a leading currency symbol.

Escaping Characters

Characters that have special meaning in regular expressions need to be escaped if they are to be represented "as is". These characters are:

\ | . ? * + ( ) { } [ ] - ^ $

In addition, the following escapes represent single characters:

\n	newline or line-feed character (
)
\r	carriage return character (
)
\t	tab character ( )

Multi-Character Escapes

.	(dot) Any Non-Line-End Character
\s	Any Space Character
\i	Any Initial Name Character (including '_' and ':')
\c	Any Name Character (including '.', '-', '_' and ':')
\d	Any Decimal Digit
\w	Any "Word" Character (anything other than Punctuation, Separator or "Other")

An upper-case multi-character escape matches any character not described by the lower-case escape. The upper-case escapes are:

\S \I \C \D \W

Character Class Expressions

A character class expression matches a single character. It's wrapped in square brackets and consists of three parts:

1. an optional negation indicator, ^.
2. one or more characters or ranges, and
3. an optional character class subtraction.

If the negation indicator is used, the single character matched is any character not given following it or in a given range.

A character range consists of two characters separated by a dash, as in:

[-a-zA-Z0-9_]

A leading dash (-) is a dash, not a range.

A character class subtraction consists of a dash followed by a character, category escape or nested character class expression, as in:

[a-z-[aeiou]]

i.e. Match lower-case letters but not the vowels.

XPath 2.0 and XQuery 1.0 Functions That Use Regular Expressions

matches(xs:string?, xs:string) **as** xs:boolean

matches(xs:string?, xs:string, xs:string) **as** xs:boolean

replace(xs:string?, xs:string, xs:string) **as** xs:string

replace(xs:string?, xs:string, xs:string, xs:string) **as** xs:string

tokenize(xs:string?, xs:string) **as** xs:string*

tokenize(xs:string?, xs:string, xs:string) **as** xs:string*

XSLT 2.0 Instructions That Use Regular Expressions

<xsl:analyze-string select = expression
regex = { string }
flags = { string }**>**
<xsl:matching-substring>
sequence-constructor
</xsl:matching-substring>
<xsl:non-matching-substring>
sequence-constructor
</xsl:non-matching-substring>
xsl:fallback*
</xsl:analyze-string>

One but not both of **xsl:matching-substring** and **xsl:non-matching-substring** can be omitted.

Inside **xsl:matching-substring**, the **regex-group(N)** function returns the **N**th group captured by the regular expression.

Regular Expression Matching Flags

Flags are letters used to indicate how Regular Expression matching is to be done:

s	Dot (.) matches any character, line-end characters included.
m	^ and $ match at the start and end of all lines, not just the start and end of the selected string as a whole.
i	Match case insensitive.
x	Remove white-space (space, tab and line-end) characters from the regular expression before using it.

Zero or more flags are specified as a string using the optional **flags=** attribute of **xsl:analyze-string** or the optional last argument of the **matches**, **replace** and **tokenize** functions.

2008-07-21

Regular Expressions in XSLT 2.0, XQuery 1.0 and XPath 2.0

Sam Wilmott
sam@wilmott.ca
http://www.wilmott.ca

and

Mulberry Technologies, Inc.
17 West Jefferson Street, Suite 207
Rockville, MD 20850 USA
Phone: +1 301/315-9631
Fax: +1 301/315-8285
info@mulberrytech.com
http://www.mulberrytech.com

Mulberry Technologies, Inc.

Regular Expression Basics

A regular expression is:

oneThing | anotherThing | yetAnother
Match one thing or another or another (one or more things).

oneThing anotherThing yetAnother
Match one thing followed by another etc. (one or more things)

atom quantifier
Match **atom** the number of times indicated by **quantifier**; once if **quantifier** is omitted.

Where **atom** is any of:

- an unescaped character,
- an escaped character,
- a parenthesized regular expression, or
- a character class expression.

Where **quantifier** is any of:

?	zero or one times (i.e. optional)
*****	zero or more times
+	one or more times
{N}	exactly N times
{N,}	N or more times
{N,M}	between N and M times inclusive.

An extra trailing **?**, as in **??**, **+?** or **{N,M}?** means match the shortest possible number of repetitions rather than the (default) longest.

Line Starts and Ends

A regular expression can be anchored at the start and/or end of a string using **^** (the start) and **$** (the end). If a regular expression is used with the **m** flag, **^** and **$** match at the start and end of each line.

In the absence of **^** or **$**, a regular expression matches unanchored: anywhere within the string.

Subexpressions and Back References

Each parenthesized group in a regular expression is assigned a group number counting unescaped left parentheses starting from the left.

Group numbers can be used in three ways:

1. Within a regular expression, to match what was matched by a previous subexpression. A previously matched group is identified by backslash and a number: **\1**, **\2** etc.
2. Within a **replace** replacement expression to match what was matched by a previous subexpression. A group is identified by a numeric name: **$1**, **$2** etc. As well, **$0** identifies the whole matched substring.
3. within a XSLT **regex-group(N)** to access the matched subexpression.