XPath für Fortgeschrittene

Ausgewählte XPath-Funktionen

***matches*(*string*,*regex*)**
- überprüft, ob ein String einem bestimmten Muster (regulärer Ausdruck!) entspricht

***replace*(*string*,*regex*,*string*)**
- ersetzt Teile des ersten Strings, die dem regulären Ausdruck entsprechen, durch den zweiten String

***tokenize*(*string*,*regex*)**
- teilt einen String in eine Reihe (Sequenz) von Strings, an der Stelle, die vom regulären Ausdruck bestimmt wird (das Muster gibt den „Teiler“ an)

***data*(*XPath*)**
- gibt die atomaren Werte aller Items in der Sequenz zurück, die der XPath-Ausdruck liefert

XQuery – XML Query Language

XQuery ist eine Abfragesprache, die auf die besonderen Eigenschaften von XML-Dokumenten und -Daten ausgerichtet ist. Sie dient zur Auswahl, Sortierung, Umstrukturierung und Verknüpfung von Daten und zur Rück- und Ausgabe von Ergebnissen.

Eine XQuery-Datei besteht aus einem optionalen „Prolog“ und einem „Hauptteil“ (body). Im Prolog können u. a. die XQuery-Version, Namensräume, globale Variablen und Funktionen deklariert werden. Im Hauptteil steht ein Ausdruck oder eine Sequenz von Ausdrücken (Elemente, XPath, FLWORs).

Regeln

- Deklarationen im Prolog mit `;` abschließen
- Variablennamen beginnen mit `$`; Zuweisung mit `:=`
- XQuery-Ausdrücke innerhalb von Elementen in geschweiften Klammern: `<elementname>{XQuery}</elementname>`
- Ein FLWOR-Ausdruck besteht aus folgenden Elementen:

`for`	For-Schleife (siehe XPath); mehrfach möglich
`let`	Deklaration einer Variablen, die im FLWOR verwendet werden kann; mehrfach möglich
`where`	Bedingung (XPath/XQuery-Ausdruck) für die Auswahl der Elemente in der For-Schleife
`order by`	Sortierkriterium für die Elemente in der For-Schleife (XPath/XQuery-Ausdruck)
`return`	Rückgabe eines Ausdrucks/einer Sequenz

  Die Reihenfolge von `for`- und `let`-Ausdrücken ist beliebig, bei den übrigen Ausdrücken steht sie fest (WOR am Ende). Ein FLWOR besteht mindestens aus einem `for`- oder `let`-Ausdruck und einer Rückgabe, alle anderen sind optional
- Syntax für Kommentare: `(: Kommentar :)`
- Die Rückgabe eines Queries muss eine Sequenz sein! (leer, mit einem oder mehreren Items)
- Im Übrigen gelten die XML-/XPath-Regeln

Beispiel

```
❶ xquery version "1.0";
❷ declare namespace ged="http://ged.de/nr";
❸ declare variable $ged:=doc("gedicht.xml");
❹ <h1> ❺ {data($ged/ged:titel)}</h1>, ❻
❼ <ul>{ ❽ (: FLWOR :)
   ❾ for $vers in $ged//ged:vers
     let $init:=substring($vers,1,1)
     where contains($vers,"Krethi")
     order by $init
     return <li>{data($vers)}</li>
  }</ul>
```

❶–❸ „Prolog“, bestehend aus:

❶ XQuery-Deklaration

❷ Deklaration des Namensraumes „gedicht“ (durch Zuweisung der URL `http://ged.de/nr` zum Präfix `ged`)

❸ Deklaration einer lokalen (in diesem XQuery gültigen) Variablen mit dem Namen `ged`, die das Dokument `gedicht.xml` enthält

❹–❾ „Hauptteil“ (body)

❹ Konstruktion eines Elements (hier: HTML-Überschrift `h1`), das Teil der Rückgabe des XQueries wird

❺ Darin ein XQuery-Ausdruck (von geschweiften Klammern umschlossen: `{...}`), mit dem der Inhalt des Elements `titel` ausgegeben wird

❻ Es wird eine „Sequenz“ von Dingen ausgegeben (hier: Elemente `h1`,`ul`), die durch ein Komma getrennt werden

❼ Konstruktion eines Elements (hier: HTML-Liste `ul`), darin ein XQuery-Ausdruck

❽ Ein Kommentar, der bei der Ausführung des Queries nicht berücksichtigt wird

❾ Ein FLWOR-Ausdruck. Für jeden Vers aus der Menge der `vers`-Elemente im Gedicht wird etwas getan (`for...in`), aber nur dann, wenn der Vers `"Krethi"` enthält (`where`): Sein erster Buchstabe wird in einer Variablen `$init` gespeichert (`let`), die Verse werden nach dem Anfangsbuchstaben sortiert (`order by`) und es wird ein HTML-Listeneintrag zurückgegeben, der den Inhalt des Vers-Elements enthält (`return`)

2015 – Autorin, Satz: Ulrike Henny; Gestaltung: Markus Schnöpf

Institut für Dokumentologie und Editorik e.V.
c/o Cologne Center for eHumanities (CCeH)
Universität zu Köln
Universitätsstraße 22
50923 Köln

i|d|e

XML

Kurzreferenz für Fortgeschrittene II

(mit Regex, XPath und XQuery)

Institut für Dokumentologie und Editorik

www.i-d-e.de
info@i-d-e.de

Regex – Regular Expressions

Reguläre Ausdrücke (Regular Expressions, Regex) sind Muster, die Zeichenketten beschreiben. Sie dienen dazu, Zeichenfolgen, die diesem Muster entsprechen, zu suchen und zu bearbeiten (z. B. sie aufzuspalten oder Teile von ihnen zu ersetzen). Reguläre Ausdrücke werden u. a. in bestimmten XPath-Funktionen zur Verarbeitung von Zeichenketten verwendet.

Beispiel

Zeichenketten:

```
Kreti,Pleti,Kreti?,Kreta,September 1918,Hinz und Kunz,Hinz und Hinz
```

	Regex	Treffer
❶	`Kreti`	`Kreti`,`Kreti?`
❷	`^Kret.$`	`Kreti`,`Kreta`
❸	`Kreti\?`	`Kreti?`
❹	`.*[0-9]+`	`September 1918`
❺	`\w{2}eti`	`Kreti`,`Pleti`,`Kreti?`
❻	`^[^e]+$`	`Hinz und Kunz`,`Hinz und Hinz`
❼	`Kreta|Kunz`	`Kreta`,`Hinz und Kunz`
❽	`([A-Za-z]+)\s([0-9]+)`	`September 1918`
❾	`(\w+)\sund\s\1`	`Hinz und Hinz`

❶ Ein regulärer Ausdruck mit den „Literalen“ `K`,`r`,`e`,`t`,`i`, die direkt übereinstimmen müssen. Treffer sind alle Strings, die diese Zeichenfolge enthalten

❷ Regex mit drei „reservierten Zeichen“: Zwischen Anfang (`^`) und Ende (`$`) des Strings sollen `Kret` und ein beliebiges Zeichen (`.`) stehen

❸ Ausdruck mit Literalen und einem „Quantor“ (`?`) , der durch die Maskierung (`\`) als normales `?` behandelt wird

❹ Ein beliebiges Zeichen beliebig oft (`.*`), gefolgt von Zahlen (beschrieben als „Bereich“ in einer in Klammern stehenden „Zeichenauswahl“:`[0-9]`), die ein- oder mehrmals (`+`) vorkommen können

❺ Gesucht werden alle Strings, die aus zwei (Quantor: `{2}`) Zeichen der Gruppe der Wortzeichen (`\w`) bestehen, gefolgt von den Buchstaben `eti`

❻ Zwischen Anfang und Ende des Strings sollen ein oder mehrere Zeichen stehen, die nicht `e` sind (`[^e]`)

❼ Entweder die Zeichenfolge `Kreta` oder (`|`) `Kunz`

❽ Ein oder mehrere Groß- oder Kleinbuchstaben (`[A-Za-z]`), gefolgt von einem Leerzeichen (`\s`), gefolgt von einer oder mehreren Zahlen.

❾ Es wird eine Gruppe von Zeichen gebildet (`(\w+)`), auf die weiter hinten im Suchstring mit `\1` Bezug genommen wird

Regeln

- „Literale“ stehen für sich selbst und müssen direkt übereinstimmen
- Es gibt „reservierte Zeichen“ mit besonderer Funktion, z. B.
  - `.` beliebiges Zeichen
  - `^` Anfang eines Strings
  - `$` Ende eines Strings
  - `\` maskiert reservierte Zeichen

  Die reservierten Zeichen müssen mit dem Backslash „maskiert“ werden, wenn sie wörtlich gemeint sind, z. B. `\?` und `\\`
- „Quantoren“ stehen nach dem Ausdruck, auf den sie sich beziehen:
  - *ohne* kommt genau einmal vor
  - `?` kein- oder einmal
  - `+` einmal oder mehrmals
  - `*` beliebig oft
  - `{n}` n mal
  - `{n,}` n bis mehr als n mal
  - `{,m}` weniger als m bis m mal
  - `{n,m}` n bis m mal
- Eine „Zeichenauswahl“ zeigt an, dass eines von mehreren Zeichen vorkommen kann. Sie wird in eckigen Klammern notiert. Neben Literalen (z. B. `[abc]`) und Zeichengruppen (z. B. `[\s]`) kann sie auch „Bereiche“ enthalten, z. B. `[a-z]`,`[0-9]`. Ein am Anfang in der Auswahl stehendes `^` bedeutet „Nicht...“
- „Zeichengruppen“ repräsentieren mehrere ‚verwandte‘ Zeichen oder ihre Umkehrung (in Großbuchstaben):
  - `\d`,`\D` digit: Ziffer bzw. alles, was keine Ziffer ist
  - `\w`,`\W` word: Buchstabe, Zahl oder Unterstrich
  - `\s`,`\S` space: Leerzeichen, Tab, Zeilenumbruch
- Gruppierung von Mustern mit `(...)`. Beim Suchen referenziert `\1` die erste Gruppe, `\2` die zweite, usw.; beim Ersetzen `$1` die erste Gruppe, `$2` die zweite, usw.
- Oder-Verknüpfungen von Mustern mit dem Pipe-Zeichen: `|`

Sequenzen

Ab der Version 2.0 ist in XPath das Konzept der „Sequenz“ zentral. Zum Beispiel können XPath-Funktionen eine Sequenz als Eingabe erwarten oder eine Sequenz zurückgeben. Eine Sequenz ist eine (nicht hierarchische) Reihe von Dingen, von so genannten „Items“. Die „Items“ wiederum können Knoten (Elemente, Attribute) oder atomare Werte (Strings, Zahlen, etc.) sein.

Beispiele

`(//titel,//strophe)` – Sequenz, die alle Elemente `titel` und alle Elemente `strophe` enthält

`(2,5,17)` – Sequenz mit drei Items (Zahlen)

`()` – leere Sequenz

XPath für Fortgeschrittene

Spezielle XPath-Ausdrücke und Operatoren

Beispiele

`//strophe[position(1 to 3)]`
– Range-Ausdruck in XPath: Es werden die Strophen-Elemente mit den Positionen 1 bis 3 zurückgegeben

`//strophe[position()=(1,2,5,7)]`
– Vergleich mit einer Sequenz: der Vergleich wird für jedes Item durchgeführt; die Vergleiche sind über eine ODER-Relation verbunden (hier: `//strophe[position()=1 or position()=2...]`)

`if (count(//gedicht/strophe) gt 3) then "lang" else "kurz"`
– If-Anweisung in XPath; hier: Wenn das Gedicht mehr als drei Strophen hat, wird der String `"lang"` zurückgegeben, sonst `"kurz"`

`for $num in 1 to 10 return //gedicht/strophe[$num]`
– For-Schleife in XPath; hier: Für jede Zahl von 1 bis 10 wird das Element `strophe` an der entsprechenden Position zurückgegeben

`for $str in //gedicht/strophe return substring-before($str," ")`
– For-Schleife; hier: von allen Strophen des Gedichts (= allen Elementen `strophe`) wird das erste Wort (= Teil-String bis zum ersten Leerzeichen) zurückgegeben

integer `to` *integer*
– Range-Ausdruck: eine Sequenz von Ganzzahlen, die nacheinander durchlaufen werden

`if` (*XPath*) `then` *XPath* `else` *XPath*
– If-Anweisung: Ausführung von Anweisungen in Abhängigkeit davon, ob eine Bedingung erfüllt ist

`for` *$variablenname* `in` *XPath/Sequenz* `return` *Sequenz*
– For-Schleife: gibt für jedes Item einer Sequenz etwas zurück; auf das aktuelle Item kann mit *$variablenname* Bezug genommen werden

Ausgewählte XPath-Funktionen

starts-with(*string*,*string*)
bzw. *ends-with*(*string*,*string*)
– beginnt bzw. endet der erste String mit dem zweiten?

substring-before(*string*,*string*)
bzw. *substring-after*(*string*,*string*)
– Teil des ersten Strings vor bzw. nach dem Vorkommen des zweiten Strings

upper-case(*string*) bzw. *lower-case*(*string*)
– gibt den String in Groß- bzw. Kleinbuchstaben zurück