24 tip: Validácia dokumentov XML oproti schéme XML

Pri svojej práci s dokumentmi XML sa môžete stretnúť s požiadavkou validácie XML oproti schéme. Ak sa pozriete do dokumentácie rozhrania API pre triedu **% XML.TextReader**, pri prvom pohľade ľahko nadobudnete dojem, že je to jednoduchá vec a nijaký zádrh v tom nie je. Stačí použiť niektorú z metód určených na prechádzanie dokumentom XML, napr. metódu **ParseFile ()**, správne nastaviť vstupné parametre (intuícia nám hovorí, že to budú pravdepodobne parametre **Filename** a **SchemaSpec**) a všetko prebehne na našu úplnú spokojnosť. Kým si to neotestujete, nemáte ani tušenie o tom, že je tu ukrytý malý zádrh. Predpokladajme pre jednoduchosť, že chceme validovať jednoduchý dokument XML, aký je uvedený ďalej:

```
|<?xml version="1.0" ?>↓
<p:Person xmlns:p="http://temp.org" ↓
              xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance " ↓
              xsi:schemaLocation="
http://temp.org
file:///c:\Home\Cache\Tmp\Person.xsd" >↓
       <P:Name>Black, Joe D. </p:Name>↓
</p:Person>←
```

oproti schéme XML:

```
|<?xml vision="1.0"  encoding="UTF-8"?>↓
<s:schema
xmlns:s="http://www.w3.org/2001/XMLSchema"↓
                xmlns:p="http://temp.org"↓

targetNamespace="http://temp.org" element-
FormDefault="qualified">↓
      <s:element name="Person"> ↓
          <s:complexType>↓
              <s:sequence>↓
        <s:element minOccurs="1"  name="Name"
type="s:string"/>↓
              </s:sequence>↓
       </s:complexType>↓
     </s:element>↓
</s:schema>←
```

Jednoduchá metóda, ktorá by mohla otestovať, či je uvedený dokument XML oproti schéme validný, by mohla vyzerať takto:

```
43
44  ClassMethod validateDocument(
45        pFile As %String = " "
46□     pSchema As %String = " ") as %Status
47   {
48         //definuj makra
49         #Define FILE
"c:\Home\Cache\Tmp\Person.xml"
50         #Define SCHEMA " "
51
52         set:´pFile pFile = $$$FILE
53         set:´pSchema tSchemaSpec =
$$$SCHEMA
54
55         set tResolver = $$$NULLOREF
56         set tFlags = $$$SAXDEFAULTS
57         set tMask = $$$SAXCONTENTEVENTS
58         set tSchemaSpec = $$$SCHEMA
59
60         // parsuj dokument
61        set
tSC=##class(%XML.TextReader).ParseFile(pFile,
.tReader,tResolver,tFlags,tMask, tSchemaSpec)
62
63        // otestuj chybovy stav
64        if $$$ISERR(tSC) do $System.Sta-
tus.DisplayError(tSC) quit
65
66        // prechadzaj dokument uzol po uzle
67        while tReader.Read()   {
68               if
(tReader.NodeType="error") ! (tRead.Node-
Type="warning"){
69                     Write !, "Node ", tRea-
der.seq, "is a(n) "
70                     Write tRead.NodeType, "
"
71
72                     If tReader.Name´=" " {
73                            Write "named:
", tReader.Name
74                     }
75                     Else {
76                            Write „and has
no name"
77                     }
78
79                     Write !, "   path: ",
tReader.Path
80
81                    If tReader.Value´=" " {
82                            Write !, "
value: ",tReader.Value
83                    }
84               }
85         }
86         quit tSC
87   }
```

Spustíte ju z terminálu a všetko vyzerá v poriadku, ale po chvíľke vo vás začne hlodať podozrenie, či naozaj validácia prebehla, lebo v dokumente XML síce je uvedená cesta k schéme a parameter **tFlags** je nastavený tak, že sa má validácia vykonávať, ale ako je to s parametrom **SchemaSpec**? Po dlhšom hľadaní v dokumentácii sa dozviete, že ak požadujete, aby sa vykonávala validácia oproti schéme, nestačí uviesť tieto informácie len v dokumente XML (zadaním cesty k schéme), ale ich treba zadať aj v parametri **SchemaSpec**. Ako však má táto informácia vyzerať, to už nikde nie je uvedené. A tak začnete experimentovať. Prvé, čo vás napadne, je, že tu zadáte cestu k súboru, v ktorom je schéma XML definovaná, t. j. skúsite upraviť kód tak, že zmeníte riadok 50 na tvar: *# Define SCHÉMA "c: \ Home \ Cache \ Tmp \ Person. xsd* ". Po spustení metódy v termináli prichádza prekvapenie:

```
Cache TRM>3564 (CACHE20102RC)
XMLSTUDY> 33class (XML.Utils.SAX) . validate-
Document ()

Node 1 is a(n) error and has no name
        path:
        value: schema dokument
´c:\Home\Cache\Tmp\Person.xsd´ has different
target namespace from the one specified in
instance dokument  ´´ while processing
c:=\Home\Cache\Tmp\Person.xml at line 4
offset 781
XMLSTUDY>
```

Teraz síce k validácii oproti schéme dochádza, to je zrejmé, ale Caché tvrdí, že váš dokument nie je platný, pretože sa nenachádza v rovnakom mennom priestore, aký je uvedený v schéme. Lenže vy viete, že to nie je pravda, pretože keď otestujete dokument XML iným validátorom, ten potvrdí, že všetko je v poriadku. Kde je teda chyba? Chyba spočíva vo formáte, akým bol parameter **ScheSpec** zadaný. Správny formát parametra **ScheSpec** je taký, že sa v reťazci najprv špecifikuje cieľový menný priestor a potom umiestnenie schémy XML, ktorá sa na daný menný priestor vzťahuje: *# Define SCHÉMA "http://temp.org c: \ Home \ Cache \ Tmp \ Person.xsd* ". Potom už validácia prebehne tak, ako by sme očakávali.

```
Cache TRM>3564 (CACHE20102RC)

XMLSTUDY>w  ##class (XML.Utils.SAX) . valida-
teDocument ()
1
XMLSTUDY>
```

DAN KUTÁČ, InterSystems

Ďalšie zaujímavé diely seriálu Tipy a triky s Caché, ktoré vychádzajú už od roku 2005, nájdete na stiahnutie tu:
http://www.intersystems.cz/education/university/serialy/tipyTrikyCache.html