Belastingdienst

Ontwerp-richtlijn XPath-controles in XSLT voor de loonaangifte

versie 1.0
01 Januari 2027

Inhoudsopgave

Inhoudsopgave 2

Inleiding 3

Aanleiding en uitgangspunten 3
Aanleiding 3
Uitgangspunten 3

Gebruik van de XSLT 4
De XSLT integreren in een softwaresysteem 4
Gebruik van de XSLT voor analyse en testen 4
Parameters bij het aanroepen van de XSLT 4
Relatie met de XSD 5

Structuur van de XSLT 6
Sectie A: gebruik van bronvariabelen 6
Sectie B: variabelen voor IKV rubrieken t.b.v. performance 6
Sectie C: gebruik van thematische hulpvariabelen 8
Sectie D: nominatieve controles 8

Opbouw van een controle 9
Eén foutmelding per foutcode per Inkomstenverhouding (IKV) 10
Regels voor contextgebruik 10
U8 Controles 10
Gebruik van normalize-space() 11
Gebruik van number() 12
Gebruik van sum() 13
Gebruik van count() 13
Boolean() 13
Gebruik van meerdere OR vergelijkingen versus contains(....concat(....)) 14

Performance 15
Gebruik centrale variabelen 15
Beperk het gebruik van XPath-functies zoals normalize-space(), number() en count() 15
Voorkom onnodige loops 15
Voorkom dubbel foutdetectie 15

Jaarovergang 16
Naamspace 16
Beoordelen van wijzigingen in de XSD 16
Actualiseren van codelijsten 16
Actualiseren van algemene variabelen en berekeningen 16
Actualiseren van XPath-controles 17

Inleiding

Deze ontwerprichtlijn bevat een aantal afspraken en toelichting over het ontwikkelen en onderhouden van de loonaangifte XSLT. Omdat de XPath controles worden ontwikkeld en onderhouden door meerdere personen, is het van belang dat er bepaalde afspraken bestaan . Daarmee wordt bewaakt dat:

- Controles op dezelfde manier worden ontwikkeld
- De code leesbaar en voorspelbaar blijft
- Reviewen en debuggen eenvoudig blijft
- Functionele fouten door stijlverschillen worden voorkomen
- We leren van onze fouten tijdens het ontwikkelen en deze documenten

Aanleiding en uitgangspunten

In dit hoofdstuk wordt los van de ontwerprichtlijn toegelicht wat het doel van de XSLT is, welke uitgangspunten hierbij zijn gekozen en waarom deze zijn gekozen. De eerste hoofdstukken beschrijven de achtergrond en uitgangspunten. De daadwerkelijke ontwerprichtlijnen beginnen bij hoofdstuk "Structuur van de XSLT".

Aanleiding

Directe aanleiding voor de XSLT is het komen vervallen van de terugkoppelingen aan de softwareontwikkelaars. Dit heeft een brainstorm op gang gebracht, waaruit alternatieven zijn gekomen om toch als softwareontwikkelaar inzicht te krijgen in de mate waarin controles afgaan en klanten hierin te kunnen helpen.
Eén van de alternatieven is een XSLT ontwikkelen, waarmee zoveel mogelijk controles kunnen worden uitgevoerd op de loonaangifte XML. Hoe sneller na het maken van de XML de XSLT wordt uitgevoerd, hoe eerder eventuele terugkoppelingen bekend zijn en kunnen worden gecorrigeerd.

Uitgangspunten

Er zijn een aantal uitgangspunten vastgesteld ten aanzien van de XSLT.

- Er wordt gebruik gemaakt van een zo generiek mogelijk opzet in XSLT 1.0.
  Hiervoor is gekozen omdat versie 1.0 door nagenoeg alle ontwikkelplatforms standaard wordt ondersteund met een XSLT processor. Dit houdt in dat het voor vrijwel iedereen mogelijk is om de in de release geleverde XSLT te gebruiken.
- De code is transparant en werkt op zichzelf staand, dus zonder externe connecties.
  In de XML zijn commentaarregels om toe te lichten wat de code doet. Ook dit document draagt bij aan deze transparantie. Dit is belangrijk vanwege algemene zorgen over scripts, omdat deze door kwaadwillende, misbruikt kunnen worden. Met onze uitgangspunten willen we deze zorg wegnemen en acceptatie van de XSLT verhogen.
- Er wordt een jaarlijkse versie gemaakt, zonder backward compatibiliteit. Dit is identiek aan de werkwijze van de XSD. De XSLT is ook nauw verbonden aan de XSD. Om een goede werking van de XSLT te krijgen moet de aangifte eerst tegen de XSD worden gevalideerd.
- Waarde gaat boven volledigheid. Zeker in de opstart zal een keuze moeten worden gemaakt welke controles als eerste beschikbaar komen. Daarbij is het uitgangspunt om meest voorkomende controles als eerste te ontwikkelen.
- Er kan sprake zijn van grote XML aangiftebestanden, daarom is performance is een belangrijk aandachtspunt.
  We streven ernaar om de XSLT voor zoveel mogelijk partijen werkbaar te houden.

Gebruik van de XSLT

De XSLT is op meerdere manieren te gebruiken. Hieronder worden de twee meest voorkomende toepassingen kort toegelicht.

De XSLT integreren in een softwaresysteem

De XSLT is een stylesheet die door een XSLT processor wordt uitgevoerd. Het XML loonaangiftebestand wordt, na de XSD-validatie, als invoer aangeboden aan de XSLT processor, waarna de XSLT validatieregels worden uitgevoerd en een resultaatbestand wordt gegenereerd. Hierdoor hoeft het softwaresysteem deze controles niet zelf te programmeren, maar alleen het XML loonaangiftebestand aan te leveren, de XSLT uit te voeren en het resultaat verder te verwerken of aan de gebruiker te tonen. De XSLT is onafhankelijk van de gebruikte programmeertaal en is ontwikkeld conform XSLT 1.0. Daardoor kan deze worden gebruikt in vrijwel iedere ontwikkelomgeving die XSLT 1.0 ondersteunt.

Gebruik van de XSLT voor analyse en testen

De XSLT kan ook zelfstandig worden gebruikt, zonder integratie in een softwaresysteem. Hiervoor is een hulpmiddel nodig dat een XSLT processor bevat. Er zijn hiervoor zowel betaalde als gratis hulpmiddelen beschikbaar. Dit is met name geschikt voor functioneel beheerders, analisten en testers die een XML loonaangiftebestand willen analyseren. Door een XML loonaangiftebestand samen met de XSLT aan te bieden aan een dergelijke omgeving, kunnen de validatieregels worden uitgevoerd en kan het resultaat direct worden bekeken.

Deze ontwerprichtlijn schrijft geen specifieke XSLT processor of testomgeving voor. De keuze hiervoor is aan de gebruiker, zolang de gekozen omgeving XSLT 1.0 ondersteunt.

Parameters bij het aanroepen van de XSLT

Bij het uitvoeren van de XSLT kunnen parameters worden meegegeven waarmee de werking van bepaalde controles wordt beïnvloed. Hierbij is het **aanroepende softwaresysteem verantwoordelijk voor de juiste parameterwaarden**. Vooral bij de 53e-weekcoulance is dit belangrijk, omdat de XSLT zelf niet vanuit het aangiftebericht kan vaststellen of er een aangiftefrequentie van vier weken is.

De parameters worden door het aanroepende softwaresysteem aan de XSLT-processor doorgegeven. Hierdoor kan de werking van de XSLT worden ingesteld zonder dat de XSLT zelf hoeft te worden aangepast.

parameter	*Waarden*	*Standaard*	*betekenis*
uitvoeren_poortcontroles	J/N	N	Bepaalt of de poortcontroles worden uitgevoerd.
coulance_53e_week	J/N	N	Bepaalt of de 53e-weekcoulance mag worden toegepast.

Uitvoeren poortcontroles

Met de parameter uitvoeren_poortcontroles wordt bepaald of de poortcontroles in de XSLT worden uitgevoerd.

- **J**: de poortcontroles worden uitgevoerd.
- **N**: de poortcontroles worden niet uitgevoerd.

Deze parameter maakt het mogelijk om de poortcontroles afhankelijk van de toepassing in of uit te schakelen.
Geeft u de parameter niet op, dan zal de XSLT worden uitgevoerd, volgens waarde N en worden de poortcontroles niet uitvoert.

53e-weekcoulance

Bij een aangiftefrequentie van vier weken kan rond de jaarwisseling een afwijking van maximaal drie kalenderdagen worden toegestaan voor een aantal controles. De coulance is verwerkt in de controles L0040, L2209, L2210 en L2211. Daarbij blijft de overige werking van deze controles ongewijzigd. Met de parameter coulance_53e_week wordt aangegeven of deze uitzondering mag worden toegepast.

- **J**: de 53e-weekcoulance mag worden toegepast.
- **N**: de 53e-weekcoulance wordt niet toegepast.

De parameter wordt door het aanroepende softwaresysteem ingesteld. De XSLT bepaalt vervolgens per aangifte- of correctietijdvak of de coulance daadwerkelijk van toepassing is. De coulance geldt uitsluitend voor:

- Het eerste tijdvak van het kalenderjaar, waarbij de betreffende datum maximaal drie dagen vóór 1 januari ligt.
- Het laatste tijdvak van het kalenderjaar, waarbij de betreffende datum maximaal drie dagen ná 31 december ligt.

Relatie met de XSD

De XSLT heeft een nauwe relatie met de XSD en gaat ervan uit dat het XML loonaangiftebestand eerst succesvol is gevalideerd aan de hand van de XSD. Valideer daarom altijd eerst het XML bestand tegen de XSD en voer pas daarna de XSLT uit.

De XSD definieert onder andere de structuur van het XML bestand, de gegevenstypen van de rubrieken en de toegestane waardebereiken. Omdat deze controles al tijdens de XSD-validatie worden uitgevoerd, hoeven zij in de XSLT niet opnieuw te worden geïmplementeerd.

De XSLT moet bij de verwerking wel rekening houden met het gegevenstype dat in de XSD is gedefinieerd. Wanneer een rubriek in de XSD bijvoorbeeld als numeriek is gedefinieerd, moet de XSLT deze rubriek ook als numerieke waarde behandelen. Dit is met name van belang wanneer numerieke waarden in de XML met voorloopnullen zijn opgenomen.

Een praktijkvoorbeeld:

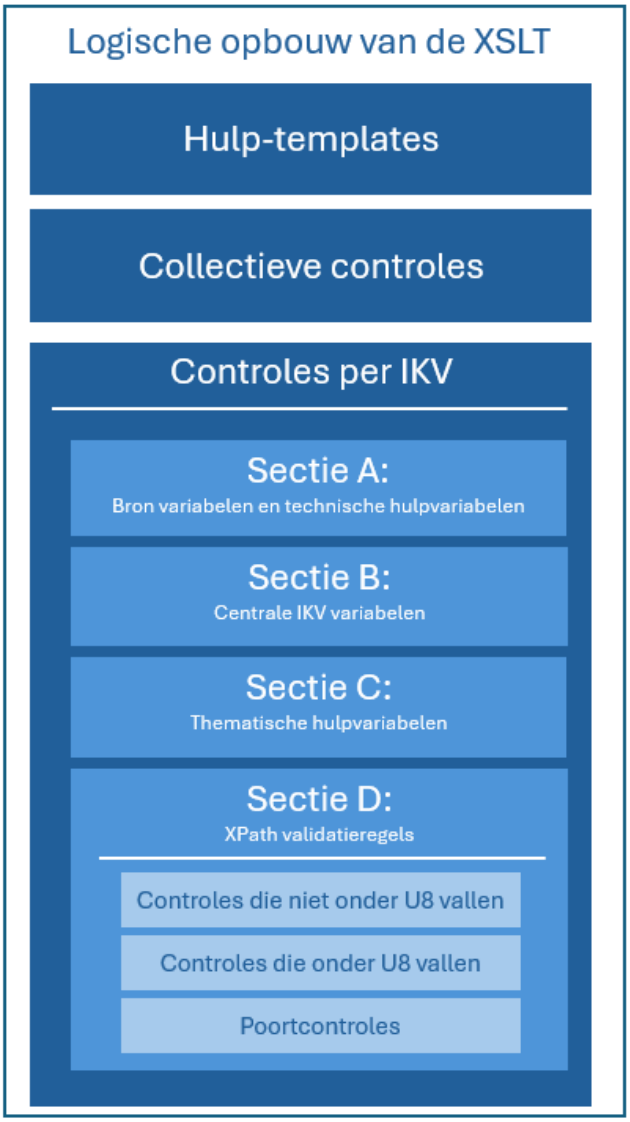
Bij controle L2203 wordt de rubriek CdAard gebruikt. In de XSD is deze rubriek gedefinieerd als xs:nonNegativeInteger en heeft dus een numeriek gegevenstype.

Wanneer deze rubriek in XPath als tekst wordt vergeleken, wordt de waarde '01' niet als gelijk beschouwd aan '1', terwijl beide waarden volgens de XSD dezelfde numerieke waarde vertegenwoordigen. De controle kan daardoor onjuist worden uitgevoerd.

Door de vergelijking numeriek uit te voeren (dus zonder quotes), worden zowel 1 als 01 als dezelfde waarde behandeld. De controle sluit daarmee aan op de definitie van de rubriek in de XSD.

Dit voorbeeld laat zien waarom het belangrijk is dat de XSLT de gegevenstypen uit de XSD respecteert en de gegevens overeenkomstig verwerkt.

Structuur van de XSLT



Nieuwe controles worden niet willekeurig in de XSLT geplaatst, maar altijd binnen de bestaande structuur. Zo blijft de XSLT overzichtelijk. De structuur van de XSLT is als volgt:

Sectie A: gebruik van bronvariabelen

In sectie A staan bronvariabelen. Dat zijn variabelen die een basis laag vormen voor vrijwel alle controles.
Bijvoorbeeld:

```
<xsl:variable name="numiv"      select="normalize-space(la:NumIV)"/>
<xsl:variable name="persnr"             select="normalize-space(la:PersNr)"/>
<xsl:variable name="bsn"                select="normalize-space(la:NatuurlijkPersoon/la:SofiNr)"/>
```

Sectie B: variabelen voor IKV rubrieken t.b.v. performance

In sectie B worden variabelen gemaakt van alle rubrieken uit de XML op IKV niveau. In veel gevallen kan er bij de controles geput worden

uit deze variabelen. Hiermee wordt de performance bevorderd, zie de paragraaf Gebruik centrale variabelen

Inkomstenperiode-rubrieken (nodes)
Rubrieken die voorkomen binnen de Inkomstenperiode worden centraal opgeslagen als node-sets. Dit gebeurt omdat binnen één inkomstenverhouding (IKV) meerdere inkomstenperiodes kunnen voorkomen.
Bij het ontwikkelen van controles is het belangrijk om rekening te houden met het feit dat een node-set waarden uit meerdere inkomstenperiodes kan bevatten.

Wanneer een controle meerdere rubrieken uit Inkomstenperiode combineert, mogen losse node-sets niet onafhankelijk van elkaar worden beoordeeld. In dat geval moet de controle binnen *$periodes[...]* worden uitgevoerd, zodat zeker is dat de betrokken waarden uit dezelfde inkomstenperiode afkomstig zijn.

Een voorbeeld waarbij het fout kan gaan is L2203:

ALS [*Code soort inkomstenverhouding / inkomenscode* gelijk is aan 11]
DAN [moet *Code aard arbeidsverhouding* gelijk zijn aan 1, 18 of 83]

Stel in de XML is opgenomen:

Inkomstenperiode	SrtIV	CdAard
Periode 1	11	7
Periode 2	15	18

Functioneel is dit fout, want beide inkomstenperiodes voldoen niet aan de conditie. Dus L2203 zou moeten afgaan.
Maar met losse node-sets kan dit ernstig misgaan:

```
$ikp_SrtIV_nodes = '11'
and
not($ikp_CdAard_nodes = 1 or $ikp_CdAard_nodes = 18 or $ikp_CdAard_nodes = 83)
```

XPath ziet:

Controledeel	Uitkomst
Zit er ergens SrtIV = 11 in de node-set?	Ja (in periode 1)
Zit er ergens CdAard = 18 in de node-set?	Ja (in periode 2)

Daardoor wordt de fout mogelijk **niet** gemeld, terwijl periode 1 wel fout is.
De goede werkwijze is in dit geval:

```
<xsl:variable name="fout_L2203"
   select="count($periodes[  la:SrtIV = '11'
                             and not(la:CdAard = 1
                             or la:CdAard = 18
                             or la:CdAard = 83)
                          ]) &gt; 0"/>
```

Er wordt nu door de beide periodes gelopen en per periode de controle uitgevoerd met de SrtIV en CdAard uit die periode.

Voor U8-controles vormt dit geen probleem. Deze controles worden uitsluitend uitgevoerd wanneer binnen de IKV precies één inkomstenperiode aanwezig is. In dat geval verwijzen de node-sets naar gegevens uit één en dezelfde periode.
Bij overige controles moet expliciet worden vastgesteld of een node-set rechtstreeks kan worden gebruikt of dat de inkomstenperiodes

afzonderlijk moeten worden beoordeeld. Dit laatste is met name van belang wanneer een controle een relatie legt tussen meerdere rubrieken uit de inkomstenperiode. In dat geval moet worden gewaarborgd dat de vergeleken waarden afkomstig zijn uit dezelfde inkomstenperiode.

Sectie C: gebruik van thematische hulpvariabelen

Soms zijn er variabelen die van belang zijn voor meerdere controles. Een voorbeeld hiervan is AOW gerechtigde leeftijd, die in meerdere controles van belang is.
Om niet voor iedere controles dan dezelfde variabele te declareren en bepalen gebeurt dat centraal in sectie C. Hiermee wordt geborgd dat alle controles met dezelfde waarde werken en in blijft onderhoud beperkt en overzichtelijk.

Sectie D: nominatieve controles

XPath controles bevatten de vertaling van de controles uit de gegevensspecificaties naar de XSLT. Omdat er honderden controles zijn is het belangrijk dat de XPath controles goed onderhoudbaar en interpreteerbaar blijven. Een uniforme werkwijze en opbouw hierin is daarom essentieel.

Binnen sectie D zijn 2 blokken, te weten controles zonder U8 beperking en controles met U8 beperking. Binnen deze blokken staan de controles in volgorde van het Lxxxx nummer. Dit verhoogt de zoekbaarheid van de controles.

U8-controle:

Een U8-controle wordt uitsluitend uitgevoerd wanneer binnen een inkomstenverhouding precies één inkomstenperiode **voorkomt.**

Opbouw van een controle

Elke controle is opgebouwd in het zelfde patroon met dezelfde onderdelen:

1. commentaarblok
2. eventueel controle-specifieke hulpvariabelen
3. één boolean variabele fout_Lxxxx
4. één xsl:if test="$fout_Lxxxx";
5. één melding via maakSpecificatie (of maakCollectieveSpecificatie bij een collectieve controle.

Een standaard controle ziet er dus als volgt uit:

```
<!-- ========================================================================
          1913:     ALS {[Datum einde inkomstenverhouding niet aangeleverd is]
                    EN  [Premie Ufo groter is dan  0]}
                    DAN [moet Indicatie verzekerd WW 'J' zijn]
========================================================================= -->
<xsl:variable name="fout_L1913"
  select="$ikvini_DatEind = ''
           and count($periodes) &gt; 0
           and count($periodes[
                    la:IndWW = 'N'
               ]) = count($periodes)
           and $wg_PrUFO != 0"/>

<xsl:if test="$fout_L1913">
  <xsl:call-template name="maakSpecificatie">
          <xsl:with-param name="bsn"    select="$bsn"/>
          <xsl:with-param name="persnr" select="$persnr"/>
          <xsl:with-param name="numiv"  select="$numiv"/>
          <xsl:with-param name="code"   select="'L1913'"/>
          <xsl:with-param name="msg"
           select="'De Indicatie verzekerd WW is N terwijl de Datum einde inkomstenverhouding niet
                    is aangeleverd en de Premie Ufo groter dan 0 is.'"/>
  </xsl:call-template>
</xsl:if></xsl:if>
```

In het groen staat een commentaarblok. Omdat er in deze controle geen hulpvariabelen worden gebruikt, staat meteen daaronder de boolean variabele *fout_L1913*, omdat het hier foutcode L1913 betreft. De *select* bepaalt de waarde van *fout_L1913*.
Vervolgens de *xsl:if test* voor *$fout_L1913* en als de boolean *$fout_L1913 = true*, dan wordt de template "*maakSpecificatie*" gevuld en aangeroepen.



In bovenstaande voorbeeld wordt *&gt* gebruikt in het select statement in plaats van >.
Dit is veiliger om eventuele parserproblemen te voorkomen.
Hetzelfde geldt ook voor het gebruik van *&lt* in plaats van < of het gebruik van *&amp* in plaats van &.

Eén foutmelding per foutcode per Inkomstenverhouding (IKV)

Een controle mag per inkomstenverhouding maximaal één melding met dezelfde foutcode genereren, ook wanneer meerdere inkomstenperiodes aan dezelfde fout voldoen.

Regels voor contextgebruik

Gebruik de juiste XML laag

De verwerking van de nominatieve controles vindt plaats in een loop waarbij de standaard XML laag *InkomstenverhoudingInitieel* is.

```
<xsl:for-each select="/la:Loonaangifte//la:InkomstenverhoudingInitieel">
```

Gebruik je rechtstreeks rubrieken andere lagen in de XML gebruik dan de juiste context.

- Gebruik $periodes[...] voor rubrieken die in Inkomstenperiode zitten.
- Gebruik $werknemergegevens/.. voor rubrieken die in periodegegevens zitten

De standaard variabelen uit Sectie B kun je rechtstreeks gebruiken en hebben ter indicatie de laag in de variabele naam staan

U8 Controles

In een aangifte kunnen binnen de gegevensgroep *'Inkomstenverhouding Initieel'* meerdere gegevensgroepen *'Inkomstenperiode'* voorkomen. In die aangifte is altijd maar 1 gegevensgroep *'Werknemersgegevens'* die hoort bij deze inkomstenverhouding. Sommige controles maken gebruik van rubrieken uit zowel de gegevensgroep '*Inkomstenperiode*' als de gegevensgroep *'Werknemersgegevens*'.

Als een rubriek uit de gegevensgroep *'Inkomstenperiode*' tijdens het aangiftetijdvak van waarde wijzigt, ontstaat een nieuwe inkomstenperiode. Zijn er in een aangifte meerdere inkomstenperiodes? En wordt er bij een controle zowel een rubriek uit de gegevensgroep *'Inkomstenperiode*' gebruikt als een rubriek uit de gegevensgroep '*Werknemersgegevens*? Dan zal een fout nooit leiden tot een terugkoppeling. Dat komt doordat rubrieken in de gegevensgroep '*Werknemersgegevens*' niet kunnen worden gesplitst per inkomstenperiode. Een terugkoppeling kan in deze situatie alleen als er maar 1 inkomstenperiode is.

Gebruik van normalize-space()

Wat doet normalize-space()?

Normalize-space() zorgt ervoor dat een tekst "opgeschoond" wordt:

- Spaties aan het begin worden verwijderd
- Spaties aan het einde worden verwijderd
- Meerdere spaties worden één spatie

Voorbeelden

Oorspronkelijke waarde	Na normalize-space()
" 11"	"11"
"11 "	"11"
" 11 "	"11"
"1 1"	"1 1"

Waarom is dit nodig?

In XML kunnen waarden:

- Leeg zijn
- Spaties bevatten
- Niet netjes gevuld zijn

Zonder normalize-space() kan dat tot verkeerde uitkomsten leiden.

Wanneer normalize-space() gebruiken?

Gebruik normalize-space() uitsluitend wanneer voorloop-, achterloop- of dubbele spaties geen betekenis hebben en je ze wilt verwijderen, zoals bij codes

Wanneer normalize-space() juist NIET gebruiken?

1. Wanneer je een centrale hulpvariabele gebruikt uit de sectie A, B of C.
   Deze moeten bij het definiëren in de betreffende sectie al genormaliseerd zijn, met uitzondering van de gegevens uit de inkomstenperiodes.
2. Tekstrubrieken waar spaties een betekenis hebben (namen bijvoorbeeld).
3. Niet "voor de zekerheid", zonder expliciet doel. Onnodig/overbodig gebruik maakt de code minder goed leesbaar/betrouwbaar en het gaat ook ten koste van de performance.

Gebruik van number()

Wat doet number()?

Number() zet een waarde om naar een getal.

Voorbeelden

Oorspronkelijke waarde	number() resultaat
"38"	38
" 38 "	38
""	NaN *
"ABC"	NaN *

* **NaN** = Not a Number (geen geldig getal)
NaN is een speciale waarde die ontstaat als iets niet naar een getal omgezet kan worden.
NaN geeft vaak geen zichtbare foutmelding, maar wél zorgt dat een controle verkeerd werkt.
Vergelijking met NaN is altijd false

Gebruik van de XSD kan veel NaN voorkomen, omdat deze controleert of er in datum- en bedrag rubrieken ongewenste tekens komen.

Waarom is dit nodig?

In XML kunnen waarden als tekst, ook als ze eigenlijk getallen zijn.

Aanbevolen patroon voor numerieke rubrieken

```
<LnSV>100</LnSV>
```

In XML is de inhoud van een element altijd tekst. Ook wanneer de XSD voorschrijft dat de waarde numeriek moet zijn. Als je wilt rekenen of vergelijken met > of <, moet je zeker weten dat het een getal is. Number() zorgt hiervoor.

Het aanbevolen patroon voor numerieke rubrieken is dan ook:

```
select="number(concat('0', translate(normalize-space($werknemergegevens/la:LnSV), '+', '')))"
```

number(...) zet de tekst om naar een numerieke waarde
concat('0', ...) voorkomt dat een lege waarde resulteert in een NaN
translate(...,'+','') verwijdert een eventueel '+'-teken vóór het getal.
normalize-space(...) verwijdert spaties aan het begin en einde van de waarde

Wanneer number() gebruiken?

- Bij berekeningen
- Bij numerieke vergelijkingen
- Bij datumvergelijk

Wanneer number() juist niet gebruiken?

- Gebruik number() niet automatisch bij codes. Bepaal aan de hand van het gegevenstype in de XSD of een code als tekst of als getal moet worden behandeld. Numerieke codes moeten numeriek worden vergeleken wanneer voorloopnullen geen betekenis hebben.
- Niet bij het risico op een NaN
- Niet bij tekstrubrieken

Gebruik van sum()

De functie sum() is bijzonder bruikbaar voor optionele numerieke elementen die bij afwezigheid als nul moeten meetellen. Dit geldt niet automatisch voor iedere optionele rubriek: de functionele betekenis van afwezigheid moet leidend blijven.

Voorbeeld

number(la:TotPrGediffWhk)

geeft NaN wanneer het element ontbreekt, terwijl:

sum(la:TotPrGediffWhk)

in dat geval 0 oplevert.

Gebruik van count()

Wat doet count()?

count() telt hoeveel XML elementen voldoen aan een selectie.

Gebruik count() wanneer het aantal XML-elementen van belang is voor de controle.

Bijvoorbeeld:

- Wanneer het exacte aantal elementen moet worden vastgesteld.
- Wanneer moet worden gecontroleerd of alle inkomstenperiodes aan een bepaalde voorwaarde voldoen.
- Wanneer een controle op meerdere plaatsen gebruikmaakt van het aantal elementen.

Voorbeeld

Count(($periodes)

Hiermee tel je hoeveel inkomstenperiodes er zijn. Bij drie inkomstenperiodes zal de uitkomst dan 3 zijn.

Boolean()

Wanneer alleen van belang is of er minimaal één XML-element bestaat dat aan een voorwaarde voldoet, gebruik dan bij voorkeur boolean() in plaats van count().

Voorbeeld

boolean($periodes[la:IndWW = 'N'])

Dit levert true op zodra er minimaal één inkomstenperiode is waarin IndWW = 'N'.

Het verschil is dat count() het aantal overeenkomende elementen bepaalt, terwijl boolean() uitsluitend vaststelt of er een overeenkomstig element bestaat. Omdat het bij boolean() niet nodig is het aantal te bepalen, heeft deze functie vanuit performanceoogpunt de voorkeur.

Uitzondering: wanneer het aantal overeenkomende elementen later binnen dezelfde controle toch nodig is, kan het efficiënter zijn om dit aantal eenmaal met count() te bepalen en de uitkomst te hergebruiken.

Gebruik van meerdere OR vergelijkingen versus contains(....concat(....))

In veel controles is sprake van een lijst van mogelijkheden bij een rubriek die moet worden getoetst, bijvoorbeeld SrtIV = 11, 13, 15 of 17.
XPath kent 2 mogelijkheden om een dergelijke controle te doen.

```
<xsl:variable name="fout_L2207"
   select="count($periodes[
                        not(normalize-space(la:SrtIV) = '11'
                        or normalize-space(la:SrtIV) = '13'
                        or normalize-space(la:SrtIV) = '15'
                        or normalize-space(la:SrtIV) = '17')
            and la:CdIncInkVerm]) &gt; 0"/>
```

of

```
<xsl:variable name="fout_L2207"
 select="boolean($periodes[
  not(contains( ' 11 13 15 17 ', concat(' ', la:SrtIV, ' ')))
  and la:CdIncInkVerm
 ])"/>
```

Beide doen precies hetzelfde, als er sprake is van reeds genormaliseerde waarden.

De *OR* variant is over het algemeen voor functionele mensen prettiger om te lezen en te begrijpen.
De *contains (.....) concat(.....)* variant is iets foutgevoeliger door de spaties die tussen de waardes moeten staan.
Daar staat dan weer tegenover dat de contains *(.....) concat(.....)* variant voor langere lijsten praktischer is.

Een goede richtlijn om de voordelen van beide te benutten is:

- Bij korte lijsten gebruik je de OR variant
- Bij langere lijsten (bijvoorbeeld L2339) de *contains (.....) concat(.....)* variant.

Performance

De XSLT is bedoeld voor het valideren van loonaangiftebestanden. Hierbij kan sprake zijn van bestanden met tienduizenden inkomstenverhoudingen (IKV's). Bij dergelijke bestanden kan de gekozen implementatie van een controle grote invloed hebben op de totale verwerkingstijd. Daarom is performance een belangrijk uitgangspunt bij het ontwikkelen van nieuwe controles.

Een kleine optimalisatie binnen één controle lijkt vaak verwaarloosbaar, maar kan bij grote XML bestanden duizenden of zelfs miljoenen extra XPath-evaluaties voorkomen.

De onderstaande richtlijnen dragen bij aan een efficiënte uitvoering van de XSLT.

Gebruik centrale variabelen

Daar waar mogelijk worden variabelen centraal gemaakt. Dit heeft een positief effect op de performance, omdat de declaratie éénmalig plaatsvindt en veel gebruikte rubrieken niet steeds uit de XML hoeven te worden gelezen. Iedere XPath-selectie vereist opnieuw een zoekactie in de XML. Door een waarde één keer op te slaan en daarna te hergebruiken, worden onnodige zoekacties voorkomen. Bij grotere XML bestanden kan dat heel veel zoekacties besparen. Daarnaast wordt het gebruik van de XSLT functies *number()* en *normalize-space()* in de controles beperkt, door het gebruik van centrale variabelen

Sectie B is hier een goed voorbeeld van, voor variabelen van alle rubrieken uit de XML op IKV niveau. In veel gevallen kan er bij de controles geput worden uit deze variabelen, dit verbetert de performance aanzienlijk.

Beperk het gebruik van XPath-functies zoals normalize-space(), number() en count()

XPath-functies zoals count(), normalize-space() en number() kosten verwerkingstijd. Gebruik deze functies daarom alleen wanneer zij daadwerkelijk nodig zijn. In grote XML bestanden kan de som van deze per functie beperkte verwerkingstijden serieus fors oplopen

Voorbeelden:

- gebruik count() alleen wanneer het aantal elementen relevant is;
- gebruik normalize-space() alleen wanneer witruimte geen betekenis heeft;
- gebruik number() alleen voor numerieke vergelijkingen of berekeningen.

Daarnaast wordt het gebruik van deze XPath-functies in de controles ook beperkt door het gebruik van centrale variabelen. Hierdoor hoeft niet meerdere keren dezelfde XML rubriek te worden uitgelezen en omgezet.

Voorkom onnodige loops

Vooral bij de inkomstenverhouding (IKV) is het oppassen met loops. De verwerking van de validatieregels vindt plaats per IKV, daarom bevat de XSLT één centrale loop waarin alle IKV's worden verwerkt. Wanneer een controle betrekking heeft op meerdere rubrieken uit Inkomstenperiode, is het noodzakelijk de inkomstenperiodes te doorlopen.

Voorkom echter dat dezelfde inkomstenperiodes binnen één controle meerdere keren worden doorlopen wanneer dit niet nodig is.

Houd de context zo klein mogelijk

Gebruik waar mogelijk de bestaande context (InkomstenverhoudingInitieel) en vermijd XPath-selecties vanaf de documentroot (//) wanneer de gegevens ook relatief kunnen worden benaderd.

Relatieve selecties zijn doorgaans efficiënter en maken bovendien duidelijk uit welke context de gegevens afkomstig zijn.
Zie ook Regels voor contextgebruik

Voorkom dubbel foutdetectie

Ontwerp controles zodanig dat een foutcode per IKV slechts éénmaal wordt bepaald. Hiermee wordt voorkomen dat dezelfde controle onnodig meerdere keren wordt uitgevoerd en blijft de verwerking efficiënt.

Jaarovergang

In dit hoofdstuk staan de aandachtspunten bij een jaarovergang beschreven. Ieder jaar wordt er, analoog aan de XSD, een nieuwe jaarversie gemaakt van de XSLT.

Naamspace

De namespace van de loonaangifte bevat een verwijzing naar het aangiftejaar. Bij een jaarovergang moet daarom de namespace in de XSLT worden aangepast. Voor het aangiftejaar 2027 is dit bijvoorbeeld:

```
xmlns:la="http://xml.belastingdienst.nl/schemas/Loonaangifte/2027/01"
```

De juiste namespace wordt overgenomen uit de XSD van het betreffende aangiftejaar.

Omdat de XPath-controles gebruikmaken van de prefix la:, is deze aanpassing essentieel voor het correct herkennen van de XML-elementen.

Beoordelen van wijzigingen in de XSD

Bij iedere jaarovergang wordt de nieuwe XSD vergeleken met die van het voorgaande jaar.
Daarbij wordt beoordeeld of:

- rubrieken zijn toegevoegd of vervallen;
- rubrieknamen of de XML-structuur zijn gewijzigd;
- gegevenstypen zijn gewijzigd;
- rubrieken verplicht of optioneel zijn geworden;
- toegestane waarden of waardebereiken zijn gewijzigd.

Deze wijzigingen kunnen gevolgen hebben voor de centrale variabelen en de XPath-controles.

Actualiseren van codelijsten

De XSLT bevat centraal gedeclareerde lijsten met toegestane codes, zoals CAO-codes en nationaliteitscodes. Bij de jaarovergang worden deze lijsten gecontroleerd en, indien nodig, vervangen door de lijsten die voor het nieuwe aangiftejaar gelden.
Bij het verwerken van codelijsten wordt gecontroleerd op:

- toegevoegde en vervallen codes;
- dubbele codes;
- behoud van eventuele voorloopnullen;
- correcte scheiding van codes door spaties bij gebruik van contains().

De bestaande variabelenamen blijven waar mogelijk behouden, zodat de controles die deze variabelen gebruiken niet hoeven te worden aangepast.

Actualiseren van algemene variabelen en berekeningen

In sectie B en C zijn centrale variabelen en berekeningen opgenomen die door meerdere controles worden gebruikt.
Bij iedere jaarovergang wordt beoordeeld of deze moeten worden aangepast.
Dit geldt bijvoorbeeld voor:

- AOW-leeftijden en de bijbehorende berekeningen;
- datumgrenzen en tijdvakafhankelijke bepalingen;
- jaarafhankelijke bedragen, percentages en grenswaarden;
- eventuele bijzondere overgangsregelingen.

Jaarafhankelijke gegevens worden bij voorkeur centraal vastgelegd, zodat aanpassing niet afzonderlijk in meerdere controles hoeft

plaats te vinden.

Actualiseren van XPath-controles

De gegevensspecificatie van het nieuwe aangiftejaar vormt de functionele basis voor de XPath-controles.
Bij de jaarovergang wordt beoordeeld welke controles:

- ongewijzigd blijven;
- inhoudelijk moeten worden aangepast;
- vervallen;
- nieuw moeten worden toegevoegd.

Ook bij bestaande foutcodes wordt gecontroleerd of de functionele conditie is gewijzigd.

Bij het aanpassen van controles wordt de werkwijze uit deze ontwerprichtlijn gevolgd. Daarbij wordt bijzondere aandacht besteed aan logische operatoren (and en or), de samenhang tussen voorwaarden en de juiste XML-context.