Geneste definities en namespacing

|  |  |  |
| --- | --- | --- |
| Joop Ringelberg | 15-12-17 | Versie: 1 |

# Introductie

De definitie van contexten, rollen, properties en property-perspectieven in tekstuele vorm vergt beantwoording van een aantal vragen.

# Situatie-specifieke definities

We gaan uit van het volgende:

1. Een context is exclusief gedefinieerd voor één domein. We bedoelen daarmee dat we in een ander domein dezelfde naam mogen gebruiken voor de definitie van een andere context (hetzelfde geldt voor rollen van het domein).
2. Een rol is exclusief gedefinieerd voor één bepaald type context. We bedoelen daarmee dat instanties van die rol slechts voor kunnen komen als onderdeel van instanties van die context.
3. Een property is exclusief gedefinieerd voor één bepaald type rol, met vergelijkbare interpretatie als voor rollen in context.
4. Een property-perspectief is exclusief gedefinieerd voor één bepaald type rol, idem.

De exclusiviteit van definities voor een bepaalde situatie (context of rol) heeft een direct gevolg, namelijk *dat in verschillende situaties verschillende rollen (of properties of property-perspectieven) dezelfde naam mogen dragen*.

Bijvoorbeeld: we kunnen de property ‘betrouwbaarheid’ toepassen in de context van getuigen (op een schaal van 1 tot 10), maar ook in de context van leveranciers (op een driepuntsschaal).

# Definities in tekst

De definitie van een type is zelf een context. Een context wordt in tekst “gedeclareerd” en wel als volgt:

:ContextType :ContextNaam

public :prop1 = <waarde1>

private :prop2 = <waarde2>

:rolNaam => <rol of context identificatie>

Bovenstaande tekst laat alle mogelijke syntactische constructies van een context zien[[1]](#footnote-1). De eerste regel noemen we de *context declaratie* en daaronder volgt een terugspringend blok van property-toekenningen en rol-bindingen.

# Namespacing

Hoe houden we situatie-specifieke definities uit elkaar als er naar verwezen wordt in tekst? De gangbare oplossing is door gebruik te maken van *namespacing*. Dat houdt in dat een bepaalde naam – een *lokale naam* – in verschillende namespaces gebruikt mag worden, maar dat als uit het voorkomen van een lokale naam niet duidelijk is welke namespace we bedoelen, een *gekwalificeerde naam* gebruikt moet worden. Gekwalificeerd betekent: de lokale naam voorafgegaan door zijn namespace.

# Domein

Een domein representeren we als een context met één type rol, namelijk psp:domein\_Item[[2]](#footnote-2). Elke definitie van dat domein is gebonden aan deze rol.

Een domein draagt een naam volgens deze regel:

model:<domeinNaam>

Om definities uit verschillende domeinen met dezelfde lokale naam uit elkaar te houden, prefixen we de lokale naam met de domeinnaam:

model:<domeinNaam>$<lokale naam>

Bijvoorbeeld: model:PolitieDomein$Aangifte. Domeinnamen kunnen we afkorten tot een prefix. In het geval van het PolitieDomein is dat pol: pol:Aangifte betekent hetzelfde als de volledige gekwalificeerde naam.

# Tekst

Een tekst heeft dezelfde structuur als domein. Het is een context met één type rol, namelijk psp:tekst\_Item. Maar teksten zijn geen namespaces. De items in een tekst ontlenen hun naam niet aan die van de tekst.

In de editor beelden we een tekst anders af dan welke andere context dan ook:

Tekst :Mijntekst

:ContextType :Aangifte

:ContextType :Misdrijf

Allereerst introduceren we de tekst met het sleutelwoord Tekst, gevolgd door de naam van de tekst. Vervolgens sommen we, aan de linkerkantlijn, alle items op. Daarbij laten we de rolnaam weg! Dat de items onderdeel zijn van de tekst, blijkt al wel uit het feit dat ze erin staan.

Een domein is ook een tekst. Dus als we een domein in tekst openen, ziet het er hetzelfde uit – afgezien van het sleutelwoord Domein in plaats van Tekst.

## Verwijzingen in een tekst

In een tekst is niet duidelijk over welk domein we het hebben. Er is geen manier om te zeggen: vanaf dit punt hebben we het over domein XYZ. Daarom moeten we gekwalificeerde namen gebruiken. Prefixes zijn daar een handig hulpmiddel voor. Een prefix wordt gedeclareerd in een tekst en geldt alleen voor die tekst:

Tekst :Mijntekst

prefix pol model:PolitieDomein

prefix psp model:Perspectives

defaultprefix model:Perspectives

De laatste regel stelt ons in staat om bijvoorbeeld :ContextType te schrijven, waarmee we model:Perspectives$ContextType bedoelen.

# Contextdefinitie

We gebruiken namespacing óók voor rollen en properties binnen een context[[3]](#footnote-3). Dat wil zeggen dat de gekwalificeerde naam van de rol Verbalisant binnen model:PolitieDomein$Aangifte luidt:

model:PolitieDomein$Aangifte$Verbalisant

of:

pol:Aangifte$Verbalisant

We kunnen op deze manier verwijzen naar deze rol, bijvoorbeeld om een mogelijke binding aan te geven (de binding van instanties van een rol te beperken tot een type). Maar net zoals we een default namespace kunnen gebruiken in een bepaalde tekst, hebben we een default voor de context als we een rol of property in context definiëren:

:ContextType pol:Aangifte

:publicProperty =>

:Property $Urgentie

public :isFunctioneel = true

Hier is duidelijk dat de property pol:Aangifte$Urgentie voor het contexttype pol:Aangifte wordt gedefinieerd. We kunnen daarom volstaan met $Urgentie. Dat is niet zo als de tekst er als volgt uitzag:

:ContextType pol:Aangifte

:publicProperty => $Urgentie

:Property pol:Aangifte$Urgentie

public :isFunctioneel = true

Omdat de definitie van pol:Aangifte$Urgentie hier tegen de linkerkantlijn staat, blijkt niet uit de tekstuele nesting dat het om een property van pol:Aangifte gaat. We moeten dus de gekwalificeerde naam gebruiken. Merk op dat we in de definitie van pol:Aangifte kunnen volstaan met verwijzing naar $Urgentie.

# Roldefinitie

Mutatis mutandis geldt hetzelfde voor definitie van rollen. Een rol introduceert een namespace en opnieuw gebruiken we een default voor inline definities van properties en propertyperspectieven. De gekwalificeerde namen worden één segment langer:

model:PolitieDomein$Aangifte$Aangever$Betrouwbaarheid

pol:Aangifte$Aangever$Betrouwbaarheid

Bij de opsomming van properties binnen een perspectief kunnen we altijd volstaan met de default aanduiding.

# Queries

Een querypad maakt de context altijd duidelijk. We kunnen dus volstaan met default namen.

# Lokale namen en unieke identifiers

## Van tekst naar data: correlation identifiers

Als de modelleur de model-tekst wijzigt, moeten we in staat zijn om intern gerepresenteerde rollen en contexten correct te relateren aan rollen en contexten die de parser uit de gewijzigde tekst afleidt. Dat betekent dat we idealiter de identificatie van rollen en contexten volledig uit de tekst zelf moeten kunnen afleiden[[4]](#footnote-4). Soms is een lokale naam daarvoor voldoende: soms hebben we een meer gekwalificeerde naam nodig, zoals boven bleek.

## Meerdere instanties van dezelfde rol

Een type rol kan meerdere keren geïnstantieerd worden in een context. Uiteraard moeten deze rol-instanties unieke identifiers hebben – èn we moeten ze uit de tekst kunnen afleiden. Het enige simpele unieke facet dat de ene instantie van de andere onderscheid in een tekst, is zijn positie in de rij van rolbindingen[[5]](#footnote-5). Maar de modelleur kan die naar believen wijzigen, zonder dat we met zekerheid tekst met data kunnen correleren.

Om die reden nemen we in de tekst een volgnummer op bij rolbindingen als hetzelfde roltype meerdere keren geïnstantieerd wordt (meerdere rolbinding-expressies met dezelfde rolnaam). Een context parseert slechts als de nummers kloppen (zie hieronder).

:rolNaam (1) => :Binding1

:rolNaam (2) => :Binding2

## Afleiding van identifiers

Contexten en rollen worden als documenten opgeslagen in Couchdb. Hoewel we beiden kunnen wijzigen, willen we de identifiers waarmee ze worden opgeslagen, liever nooit veranderen[[6]](#footnote-6). Maar de modelleur moet

* de naam van een context wel kunnen wijzigen, en
* instanties van hetzelfde roltype naar believen kunnen ordenen,

terwijl beiden onderdeel zijn van de identifiers die we voor ze afleiden! Om die reden

* geven we een context behalve een identifier óók een displaynaam;
* geven we een rol behalve een volgnummer in zijn identifier, óók een attribuut volgnummer.

Het volgnummer gebruiken we om volgens onderstaande regel identifiers voor rol-instanties af te leiden:

<context-instantie-naam>\_<rol-type-naam>\_n

De ingetikte lokale naam van een context expanderen we naar een volledige, gekwalificeerde naam:

model:<DomeinNaam>$<ingetikteNaam>

Als de modelleur de lokale naam in een context-declaratie wijzigt, kunnen we bepalen welke context een andere displaynaam moet krijgen, als we de versie van de betreffende regel vóór wijzigen bewaard hebben (want daaruit kunnen we de oude displaynaam afleiden, en daarmee in de data de context met zijn unieke identifier terugvinden).

Als de modelleur de volgorde van rolbindings-expressies wijzigt, kunnen we eenvoudig vaststellen welke volgnummers aangepast moeten worden. Immers, de tekst bevat alle informatie om de volledige identifiers van de rollen te construeren en uit de volgorde leiden we de nieuwe volgnummers af.

Merk op dat in de tekst de nummers dan niet meer op volgorde staan!

De modelleur mag de volgnummers niet wijzigen. Zij zijn immers onderdeel van de identificatie van de rollen!

## Complexe wijzigingen

Door een tekstblok ineens te plakken, kan vóórkomen dat het systeem een parseerbare context uit de gewijzigde tekst niet kan relateren aan een context die onderdeel is van de versie van de tekst gerepresenteerd als contexten.

Bijvoorbeeld: de modelleur plakt een blok tekst waardoor een context een nieuwe naam krijgt èn de betreffende context declaratie op een nieuwe regel staat.

In zo’n geval weigert het systeem de mutatie, met een informatieve melding.

Het simpele geval van een lokale naam editen, levert geen probleem. Evenmin is het een probleem om een hele definitie te overschrijven met een nieuwe versie, of een halve, zolang de context declaratie maar op dezelfde regel blijft staan.

1. Afgezien van query. Dit is een sleutelwoord zoals public en private en kan op dezelfde plek ingezet worden. Het wordt gevolgd door een query-expressie. Dit type expressie heeft alleen betekenis wanneer het gebruikt wordt in een rol-definitie. [↑](#footnote-ref-1)
2. De prefix psp staat voor Perspectives, het basis-domein van Perspectives. [↑](#footnote-ref-2)
3. Let op. Een context heeft geen properties, zijn binnen- en buitenrol wel. Maar in tekst schrijven we deze properties ingebed in een contextdeclaratie. Op vergelijkbare wijze definiëren we zulke properties rechtstreeks binnen een contextdefinitie. [↑](#footnote-ref-3)
4. Anders moeten we de tekstuele wijziging kunnen ‘reverse engineren’ op basis van oude tekst, nieuwe tekst, interne representatie en parseresult. Liever niet! [↑](#footnote-ref-4)
5. Ongetwijfeld zullen twee bindingen nooit precies dezelfde waarden hebben voor properties en de binding zelf. Maar liever laten we de matching tussen tekst en interne representatie niet afhangen van zo’n diepe vergelijking, temeer daar meerdere waarden tegelijk kunnen wijzigen als de modelleur een heel blok tekst ineens vervangt door een ander blok door te plakken. [↑](#footnote-ref-5)
6. Denk aan de problemen die ontstaan als tijdens het verkeer tussen Couchdb versies de volgorde waarin berichten aankomen, anders zijn dan waarin ze zijn verstuurd. [↑](#footnote-ref-6)