



Advies Overheid.nl

**IPM Samenwerkende
Catalogi: Technische
Architectuur en XML**

versie: 2.1
datum: 1 juni 2007

John Oldenhuizing
Projectleider Samenwerkende Catalogi

telefoon: 070-8887 850
e-mail: SC@overheid.nl
adres: Postbus 84011
2508 AA Den Haag
Wilhelmina van Pruisenweg 104
2595 AN Den Haag
internet: www.advies.overheid.nl
www.overheid.nl



Inhoudsopgave

1.	Inleiding.....	2
2.	Architectuur van Samenwerkende Catalogi	2
2.1.	Uitgangspunten	2
2.2.	Technische architectuur: back end.....	3
2.3.	Technische architectuur: front end	4
2.4.	Technische architectuur: integraal	7
2.5.	Routing zoekvragen	7
3.	XML-schema Samenwerkende Catalogi v2.1	8
3.1.	Controlled vocabularies en metadata	9
3.2.	Namespace en versie	10
3.3.	Global elements	10
3.4.	Algemeen gebruikte types	16
4.	Voorbeelden implementatie zoekfunctie	20



1. Inleiding

Dit document beschrijft de technische architectuur van de zoekdienst voor Samenwerkende Catalogi, en het XML-schema dat gebruikt kan worden voor de berichtuitwisseling in deze architectuur. U kunt dit schema gebruiken voor zowel het aanleveren van data aan de zoekdienst als het bevragen van de zoekdienst. Dit document is gebaseerd op de standaard voor Samenwerkende Catalogi versie 2.1.

Centraal in de standaard voor Samenwerkende Catalogi staat de zoekdienst voor producten en diensten. Hoe de berichtuitwisseling plaats vindt tussen deze zoekdienst en andere partijen, is beschreven in hoofdstuk 2 over de architectuur.

Hoofdstuk 3 gaat in detail in op het XML-schema. Deelnemers aan de standaard voor Samenwerkende Catalogi moeten hun informatie over producten en diensten conform dit XML-schema aanbieden. Ook moeten zij de zoekdienst raadplegen met zoekvragen die conform dit schema zijn opgesteld.

Voor meer informatie over de technische standaard en eerdere versies, kunt u contact opnemen met het projectteam Samenwerkende Catalogi. Stuur hiervoor een e-mail naar SC@overheid.nl.

2. Architectuur van Samenwerkende Catalogi

In dit hoofdstuk wordt de technische architectuur van het systeem van Samenwerkende Catalogi beschreven.

In de eerste paragraaf worden de uitgangspunten achter de architectuur weergegeven. In de daarop volgende paragrafen wordt de architectuur in delen weergegeven en uitgelegd. Vervolgens volgt de totale plaat die de integrale architectuur weergeeft. Tenslotte wordt het mechanisme geschetst waarmee bepaald wordt welke overheden doorzocht dienen te worden.

2.1. Uitgangspunten

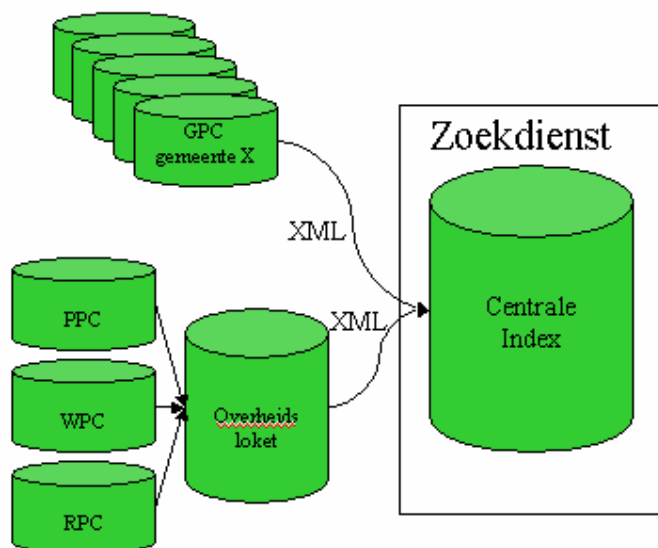
Centraal in de standaard voor Samenwerkende Catalogi staat de zoekdienst voor producten en diensten. Deze zoekdienst bevat een centrale, gemeenschappelijke index waar de producten van alle deelnemers (gemeenten, provincies, et cetera) in zijn opgenomen. In het verleden is gekozen voor een centrale index (in tegenstelling tot een stelsel van decentrale productcatalogi) om de beheerlast bij decentrale overheden te beperken en een goede performance te kunnen garanderen.

De bevraging van de zoekdienst vindt plaats door het verzenden van XML-berichten via het REST-protocol. Er wordt momenteel geen gebruik gemaakt van WSDL of SOAP. De reden hiervoor is het gemak en de laagdrempeligheid waarmee de zoekfunctie kan worden geïmplementeerd. Mogelijk wordt dit later toegevoegd ten behoeve van de standaardisatie over de transparantieprojecten van Advies Overheid.nl heen.



De structuur van de metadata is uniform door het hele stelsel heen. De metadata in de XML-feed is identiek aan die van de vraagkant. Deze uniformiteit maakt het stelsel eenvoudiger en maakt hergebruik van software mogelijk.

2.2. Technische architectuur: back end



Figuur 1: technische architectuur back end

De bronnen voor de gemeenschappelijke index zijn de gemeentelijke productencatalogi en het overheidsloket. Het Overheidsloket is een samengestelde index bestaande uit productencatalogi van de rijksoverheid (RPC), waterschappen (WPC) en provincies (PPC). Gemeenten en het overheidsloket zetten een XML-bestand (indexfeed) klaar dat wordt geïndexeerd door de indexer van de zoekdienst, die hier een centrale index uit samenstelt. Hierin staan alle producten uit de diverse catalogi.

De structuur van het XML-bestand staat beschreven in hoofdstuk 3. Dit XML-bestand dient op een URL op het internet beschikbaar te zijn. De URL van de Indexfeed kan bij Advies Overheid.nl worden aangemeld.

Valideer uw indexfeed voordat u deze aanbiedt. U moet dus zelf valideren als aanleverende partij. Hiervoor is de online validator beschikbaar op <http://validator.overheid.nl/21>. Zie hiervoor ook het document 'technisch stappenplan validatieservice'. Het maakt niet uit of de indexfeed via HTTP of HTTPS bereikbaar is. Wel dient een eventuele firewall de indexer van de zoekdienst door te laten.

De indexer haalt de indexfeed doorgaans eenmaal per dag op, en verwerkt de wijzigingen dan in de centrale index. Omdat de informatie in de productencatalogi geen hoge actualiteitseis kent, is een dagelijkse update voldoende. Door het gebruik van een indexfeed hoeft de overheid of leverancier zelf geen zorg te dragen voor het actueel houden van de producten in

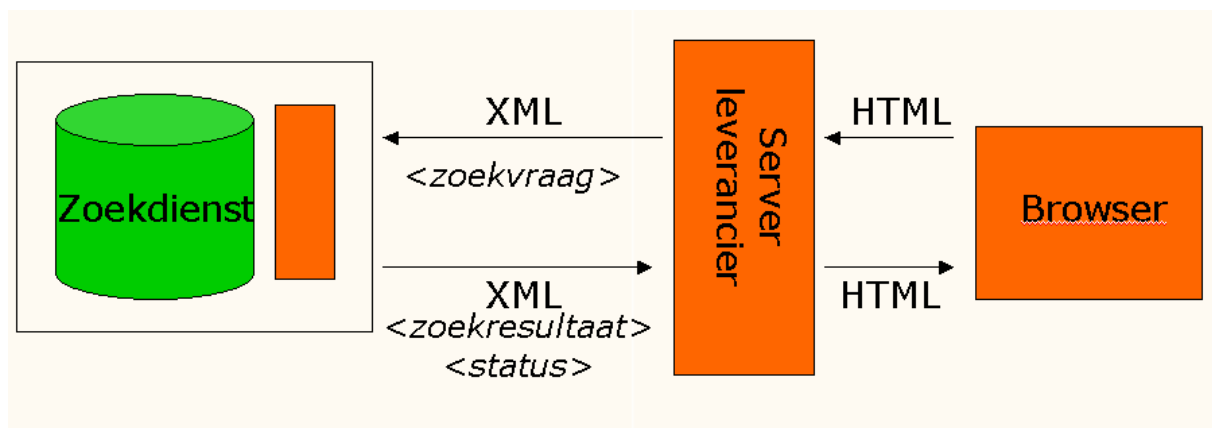


de centrale index, maar slechts te zorgen dat de indexfeed een actuele representatie van de productcatalogus is.

2.3. Technische architectuur: front end

Voor het bevragen van de zoekdienst is een aantal opties beschikbaar. In deze paragraaf worden de mogelijkheden beschreven met betrekking tot de wijze waarop de zoekfunctie kan worden ingericht. Belangrijk aandachtspunt bij deze scenario's is de vereiste manier van authenticatie, namelijk http basic authentication. Elke deelnemer krijgt hiervoor van het team Samenwerkende Catalogi een inlognaam en wachtwoord uitgereikt.

Front end (1): XML bevraging



figuur 2: technische architectuur front end (1)

De aanbevolen manier van communiceren met de zoekserver is met behulp van de XML-berichten 'zoekvraag' en 'zoekresultaat'. Het XML-zoekbericht bevat de meeste mogelijkheden voor het samenstellen van complexe zoekvragen. Het XML-zoekresultaat bevat alle benodigde gegevens over de gevonden producten, zonder dat dit in een specifieke presentatievorm is gegoten. U kunt er dus eenvoudig uw eigen opmaak aan toevoegen.

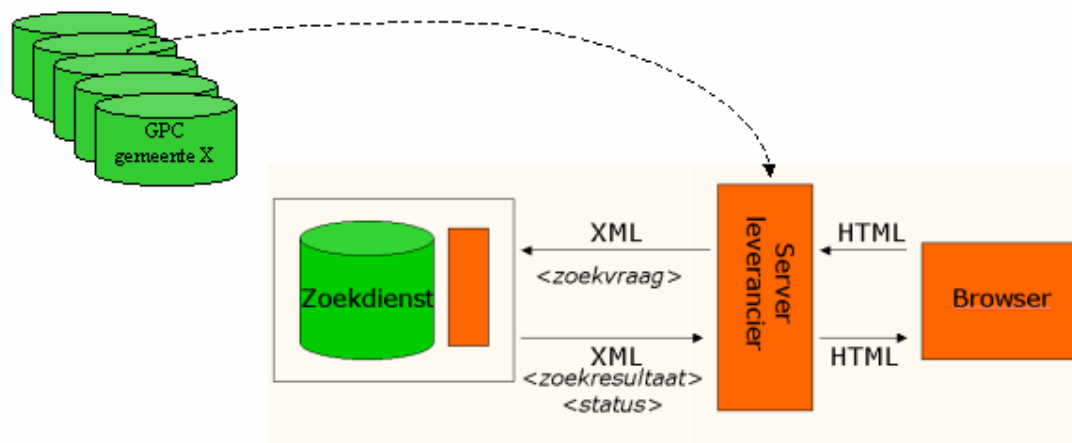
Deze manier van communiceren houdt in dat op de server van de leverancier (of gemeente) een vertaalslag plaats vindt van de input vanuit de browser naar een XML-bericht. Doorgaans zal deze input de vorm hebben van een gepost HTML form. De leverancier kan de gebruiker al dan niet in staat stellen om zelf een complexe zoekvraag samen te stellen (bijvoorbeeld met AND en OR constructies), en draagt zorg voor de dialoog met de gebruiker en de vertaling naar XML op de server. Daarnaast kan een leverancier aan de gebruiker ook slechts een eenvoudige zoekvraag toestaan, en deze aan de server kant uitbreiden, bijvoorbeeld met een zoekgebied of door toe te voegen dat zowel auteur als publicerende organisatie moet worden doorzocht. Op deze manier hoeft de gebruiker zelf geen notie te hebben van de diverse metadata velden.

Het antwoord op de zoekvraag is eveneens in XML opgesteld. Dit zal vertaald moeten worden naar HTML voordat het naar de browser van de gebruiker kan worden gestuurd. Het XML-zoekresultaat bevat geen presentatieloga. Op de server van de leverancier kan bepaald worden wat er precies met de resultaten gebeurt, hoe deze gepresenteerd worden en wat er in

geval van foutmeldingen moet gebeuren. Zo kan de leverancier bijvoorbeeld de resultaten over meerdere pagina's verdelen, of de resultaten in een bulleted list zetten. Ook de uiteindelijke vormgeving (huisstijl) die op de HTML wordt toegepast kan gebruik maken van CSS, naar wens van de leverancier. Voor de vertaling naar HTML kan bijvoorbeeld gebruik worden gemaakt van XSLT, of het bericht kan geparst / uitgelezen worden en expliciet per element vertaald worden.

De manier waarop de input van de browser vertaald wordt in de XML-zoekvraag, en het resultaat terug naar HTML voor de browser, is afhankelijk van de technologie die door de leverancier wordt gebruikt. In de bijlage is een beknopt voorbeeld opgenomen op basis van Java technologie, waarbij de input van een HTML form wordt verwerkt tot een XML-bericht dat gepost wordt naar de zoekserver. Het antwoord wordt met XSLT vertaald naar HTML dat weer naar de browser wordt gestuurd. Ook is een voorbeeld in ASP DOM opgenomen. Deze voorbeelden dienen uitsluitend ter illustratie. Voor andere technologieën (PHP, ASP.NET, et cetera) zal de werkwijze niet veel afwijken.

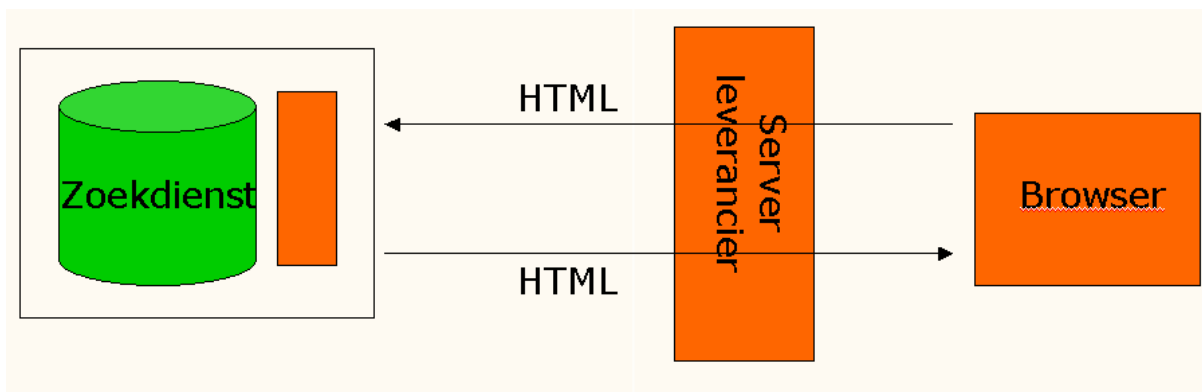
Front end (2): XML-bevraging en eigen bevraging GPC



figuur 3: technische architectuur front end (2)

Gemeenten die een eigen zoekfunctie implementeren kunnen direct hun eigen productencatalogus bevragen op de voor hen gebruikelijke manier, en de producten en diensten van andere overheden via de zoekdienst ophalen. Dit is optioneel, want de gemeentelijke productcatalogus kan uiteraard ook via de zoekdienst bevroegd worden.

Front end (3): Standaard zoekscherm

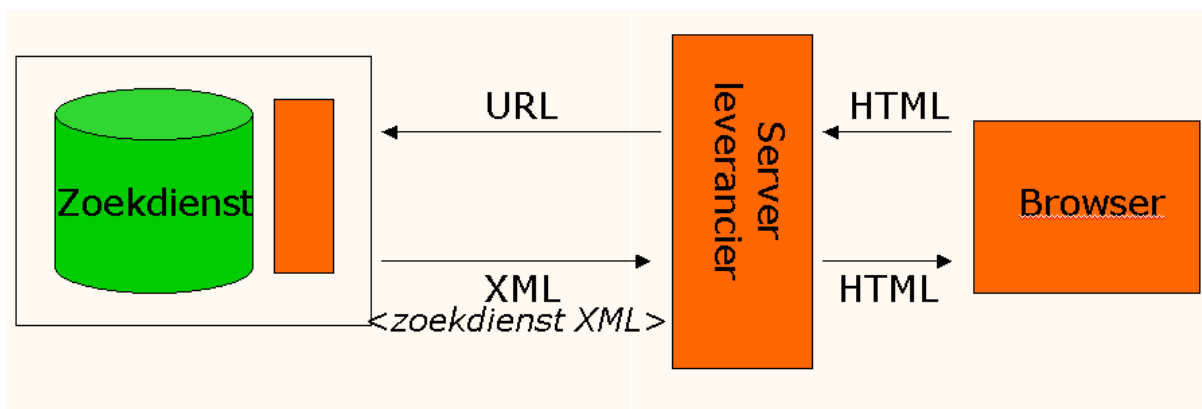


figuur 4: technische architectuur front end (3)

Een andere optie naast XML is het gebruik van een standaard zoekscherm en een standaard resultatenlijst. Deze schermen kunnen worden geïntegreerd in de website van de betreffende overheid, zonder dat de XML-berichten opgesteld en verwerkt hoeven te worden. Bij deze variant interacteert de browser rechtstreeks met de schermen van de zoekmachine (zij het geïntegreerd in de website van de betreffende overheid). Het overheidsloket (www.overheidsloket.nl) en de provincie- en waterschapsloketten werken ook op deze manier.

Voor elke gemeente is een standaard zoekscherm en resultatenscherm voorbereid. Het nadeel van deze optie is dat er minder invloed kan worden uitgeoefend op zowel de vraag als wat er met de resultaten gedaan wordt. Een vraag kunt u bijvoorbeeld niet uitbreiden met een eigen zoekgebied of andere additionele logica. Voor wat betreft de opmaak van beide pagina's kan wel gebruik gemaakt worden van Cascading Stylesheets (CSS). Een gedetailleerde toelichting op het gebruik van de standaard schermen is te vinden op de website van Samenwerkende Catalogi.

Front end (4): URL bevraging

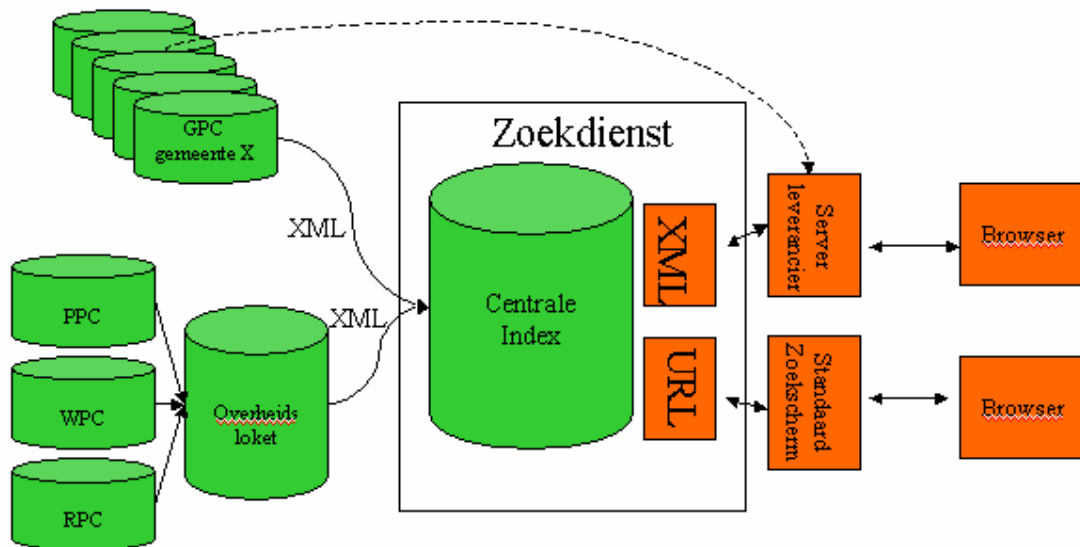


figuur 5: technische architectuur front end (4)

Naast de bovenstaande opties is het ook mogelijk een zoekvraag via een URL te stellen. Deze manier van zoeken is geen officieel onderdeel van de standaard Samenwerkende Catalogi, en

ondersteuning van deze variant is dan ook niet gegarandeerd. De URL bevraging is daarom uitsluitend te gebruiken na overleg met het team Samenwerkende Catalogi. De zoekresultaten komen in een zoekdienst specifiek XML-formaat terug. In deze XML zijn de productbeschrijvingen opgenomen, volgens de in dit document beschreven XML-standaard. De voordelen hiervan zijn dat hiermee eenvoudig, enkel gebruik makend van een browser, een zoekvraag kan worden gesteld, en dat een URL doorgaans gemakkelijker op te stellen is dan een XML-bericht. Nadeel van deze variant is dat met het XML-bericht mogelijk complexere zoekvragen zijn op te bouwen.

2.4. Technische architectuur: integraal



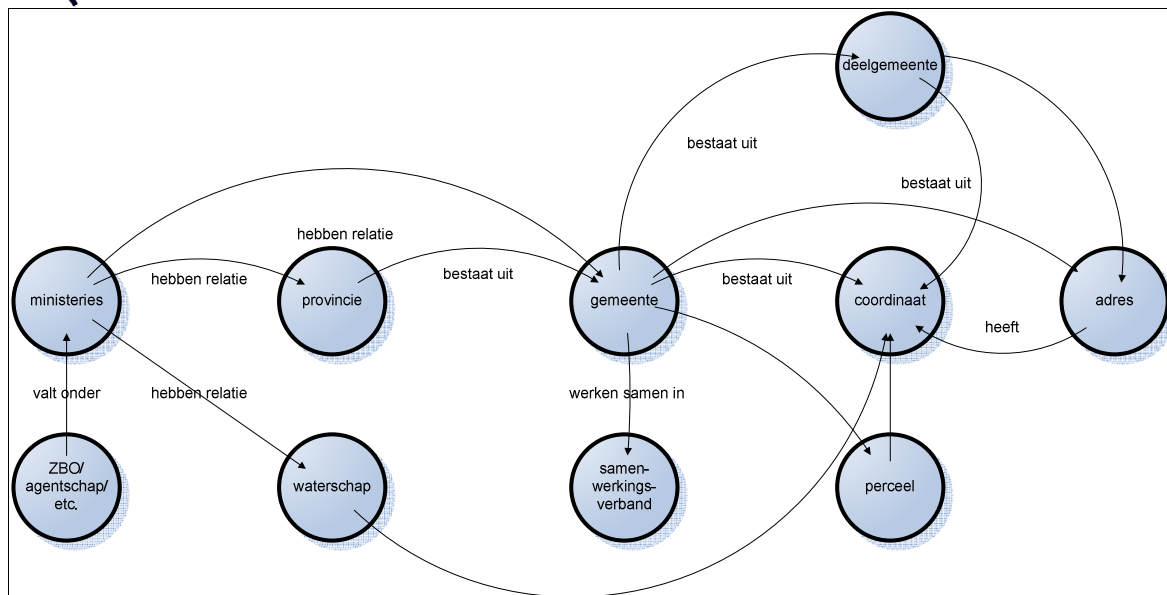
figuur 6: technische architectuur integraal

Bovenstaande figuur geeft de integrale architectuur weer van het stelsel van voorzieningen van Samenwerkende Catalogi. De afbeelding is opgebouwd uit samenvoeging van de figuren 1 tot en met 5.

De bronnen voor de gemeenschappelijke index zijn de gemeentelijke productcatalogi en het overheidsloket. Deze zetten een XML-bestand klaar dat geïndexeerd wordt. Websites (gemeente of overheid.nl) bevragen de gemeenschappelijke index via een eigen XML-zoekfunctie, of via het standaard zoekscherf. Zij kunnen dit combineren met het rechtstreeks bevragen van hun eigen catalogus.

2.5. Routing zoekvragen

Routing biedt de mogelijkheid om vanuit één loket producten uit de catalogi van één of meer andere overheden op te vragen. Hiervoor is inzicht in de samenhang tussen verschillende overheden en hun geografische afbakening nodig (zie de volgende figuur).



Figuur 5: overheden en hun onderlinge relaties

Deze figuur toont de samenhang tussen de diverse overheden, met coördinaten als gemeenschappelijk element voor alle overheden. Bepaalde overheden hebben een landelijke dekking (ministeries en ZBOs). Op dit moment wordt productinformatie alleen op het niveau van (deel)gemeenten, waterschappen, provincies en ministeries ontsloten.

In een zoekvraag kan de afzender meegeven op welke overheid de zoekvraag betrekking heeft door het specificeren van een organisatie (bijvoorbeeld gemeente Amsterdam) of een 4-cijferige postcode (bijvoorbeeld 1000). Tevens kan de verzender aangeven welke overheidslagen doorzocht dienen te worden (bijvoorbeeld alleen gemeenten en provincies), dit wordt het 'zoekbereik' genoemd. Organisatie, postcode en bereik vormen samen het 'zoekgebied'. In het volgende hoofdstuk wordt beschreven hoe deze velden in het XML-zoekbericht kunnen worden gespecificeerd.

De zoekdienst bepaalt welke overheden van toepassing zijn op een zoekvraag; in het geval van de gemeente Amsterdam is dat de provincie Noord-Holland en de gemeente Amsterdam zelf. Wordt het bereik weggelaten, dan worden alle overheidslagen doorzocht die voor de bevrager relevant zijn. Wordt ook de organisatie en postcode weggelaten, dan worden –alledeelnemende overheden doorzocht.

Omdat de zoekdienst slechts een relatie tussen gemeenten en waterschappen kent (dus niet postcode/huisnummer specifiek) kan het voorkomen dat een zoekvraag leidt tot resultaten van meerdere waterschappen. In dit geval dient de gebruiker zelf te bepalen welk waterschap op hem van toepassing is.

3. XML-schema Samenwerkende Catalogi versie 2.1

Het XML-schema specificeert de structuur van gegevens, zoals die in een XML-bericht of XML-document worden uitgewisseld. Het XML-schema is daarom de leidraad bij het opstellen van de berichten. XML-berichten zijn tegen hun XML-schema te valideren conform de regels die in dat



schema zijn vastgelegd. Succesvolle validatie is een indicatie voor de ontvangende partij dat het bericht juist is opgesteld en daarom ook zonder problemen te verwerken is.

De verschillende XML-berichten zijn vastgelegd in één XML-schema. Zo is er een zoekvraag, een statusbericht, een indexfeed en een zoekresultaat.

De verschillende onderdelen van het schema voor Samenwerkende Catalogi worden in dit hoofdstuk globaal toegelicht. Verdere details staan in het schema zelf. Een aantal voorbeeldberichten is op de website van Samenwerkende Catalogi te vinden.

3.1. Controlled vocabularies en metadata

In het schema (SamenwerkendeCatalogi.xsd) wordt gebruik gemaakt van een aantal andere schema's middels 'import' elementen. Deze externe schema's bevatten controlled vocabularies zoals die gepubliceerd zijn op <http://metadata.overheid.nl>. Op deze manier sluit de structuur van de berichten zoveel mogelijk aan bij andere transparantieprojecten binnen Advies Overheid.nl die XML-schema's gebruiken.

Het betreft de volgende generieke lijsten:

- Adviescollege.xsd: opsomming van adviescolleges in Nederland.
- Deelgemeenten.xsd: opsomming van alle deelgemeenten in Nederland.
- Gemeenten.xsd: opsomming van alle gemeenten in Nederland.
- Language.xsd: lijst met toegestane talen in Nederland voor (formele) publicaties. Dit zijn het Nederlands en het Fries als deel van alle mogelijke talen op de wereld. Als producten in een andere dan deze twee talen gepubliceerd worden, dan kan uit de overkoepelende lijst een selectie plaatsvinden.
- Ministeries.xsd: opsomming van Nederlandse ministeries.
- Openbaarlichaam.xsd: opsomming van openbare lichamen in Nederland.
- Provincies.xsd: opsomming van alle Nederlandse provincies.
- Rechtmacht.xsd: opsomming van organisatieonderdelen binnen de Nederlandse rechterlijke macht.
- Regiosamenwerking.xsd: opsomming van regionale samenwerkingsverbanden.
- Waterschappen.xsd: opsomming van alle waterschappen in Nederland.
- ZBO.xsd: opsomming van Nederlandse Zelfstandige Bestuursorganen.

Deze lijsten hebben momenteel geen versie. Dit is nu impliciet geregeld via de versie van het schema van Samenwerkende Catalogi. Verder geldt dat deze lijsten geen weergave zijn van de volledige structuur van de Nederlandse Overheid; agentschappen ontbreken bijvoorbeeld. Deze kunnen in een later stadium worden toegevoegd als dit relevant is voor publicatie en ontsluiting van producten.

Daarnaast kent Samenwerkende Catalogi de volgende specifieke lijsten (deze worden nog niet binnen een ander transparantieproject gehanteerd):

- Doelgroep (audience.xsd). Deze kan twee waarden hebben: particulier of organisatie/ondernemer.
- Indeling van producten (taxonomiesc.xsd). Dit is een nadere indeling van producten zoals die in het metadata element onderwerp1 te gebruiken is.

Tenslotte is het XML-schema van Samenwerkende Catalogi gebaseerd op de Dublin Core standaard. Dublin Core definieert een aantal standaard metagegevens voor publicatie van informatie. In het document 'technische documentatie metadata' wordt een lijst gegeven van de metagegevens die van toepassing zijn voor Samenwerkende Catalogi.



3.2. Namespace en versie

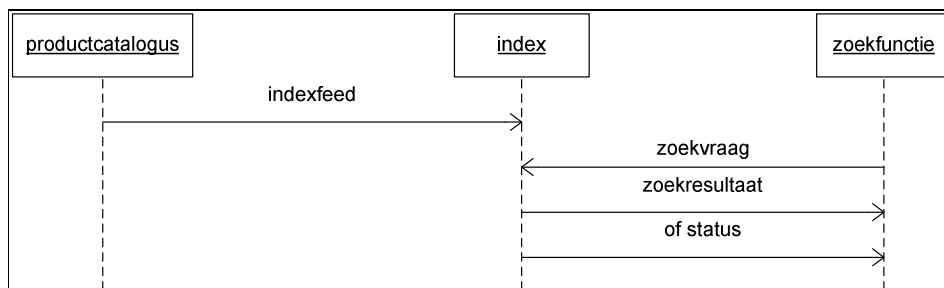
Een XML-schema start met een schema tag voor de declaratie van de namespaces die in een XML-document gebruikt worden. Een schema heeft zelf ook een namespace, de targetNamespace. Deze is vastgesteld op <http://www.adviesoverheid.nl/metadata-project/samenwerkende-catalogi/2007-1/>. De toevoeging '2007-1' geeft de versie van het te gebruiken schema. In de XML-berichten dient altijd dezelfde namespace gebruikt te worden.

3.3. Global elements

- Het schema kent vier mogelijke 'root' elementen:
- <indexfeed> voor het toevoegen van productinformatie aan de zoekdienst.
- <zoekvraag> voor het bevragen van de zoekdienst.
- <zoekresultaat> voor het verzenden van de gevonden producten conform een zoekvraag.
- <status> voor het weergeven van de status van een zoekvraag.
-

Een XML-bericht start altijd met één van deze vier tags die een verwijzing hebben naar het te gebruiken schema in de namespace. De rootelementen worden hieronder beschreven.

Het sequence diagram in Figuur 6 toont de berichten die tussen de verschillende componenten uitgewisseld kunnen worden.



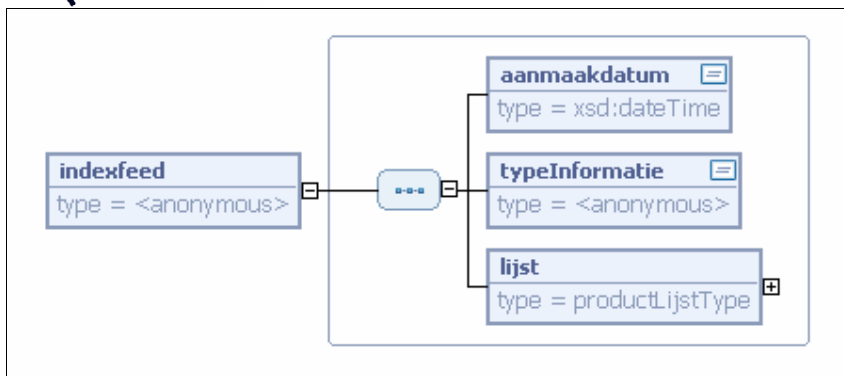
Figuur 6: Berichtuitwisseling tussen componenten

De globale elementen bestaan elk uit één of meer andere elementen. Deze andere elementen zijn van een type dat verderop in het schema is opgenomen. De opbouw van de globale elementen wordt hierna kort toegelicht.

Indexfeed

De volgende figuur toont de elementen van de indexfeed.



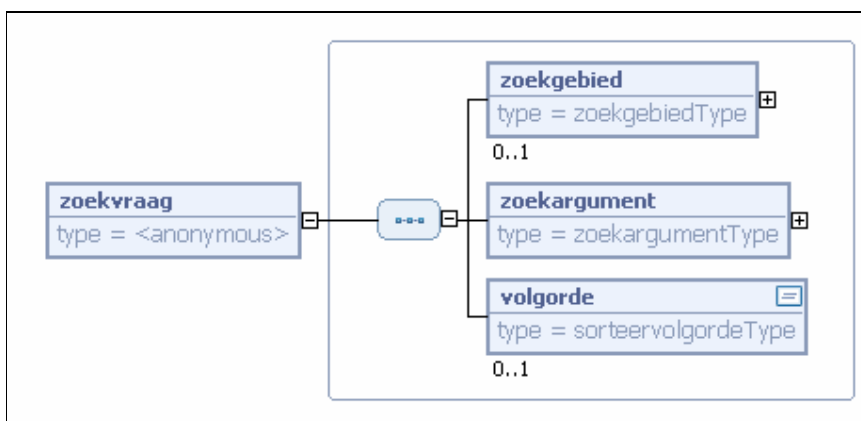


Figuur 7: elementen van indexfeed

De indexfeed bestaat uit een aanmaakdatum, het type informatie (waarde altijd 'productbeschrijving') en een lijst met producten. De lijst met producten is van het type 'productLijstType', zie verder.

Zoekvraag

De volgende figuur toont de elementen van de zoekvraag.



Figuur 8: elementen van zoekvraag

Naast sub-elementen is het toegestaan om in het zoekvraag element twee attributen te specificeren waarmee het aantal geretourneerde resultaten kan worden gespecificeerd:

- **results:** het maximaal aantal resultaten dat geretourneerd mag worden door de zoekdienst (default: 500)
- **start:** het eerste product dat geretourneerd moet worden

Om bijvoorbeeld product 1-10 van gemeente Amsterdam op te vragen kunnen de attributen als volgt gespecificeerd worden:

```
<zoekvraag start="1" results="10">
```

En om de volgende 10 producten op te vragen:

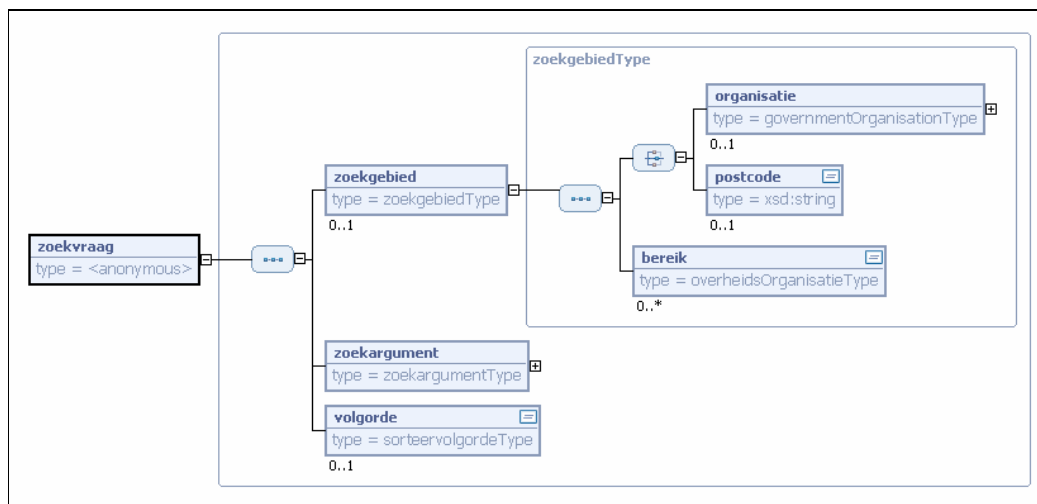
```
<zoekvraag start="11" results="10">
```

De sub-elementen van de zoekvraag zijn als volgt uitgewerkt:

- **Zoekgebied.** In dit element kan de *afzender* van de zoekvraag worden gespecificeerd in de vorm van een organisatie of een postcode. Deze informatie wordt gebruikt voor het doorzoeken van de juiste productencatalogus. Daarnaast is een *bereik* voor de zoekvraag op te geven, bijvoorbeeld alleen gemeente of naast gemeente ook provincie en



waterschap. Als het bereik bijvoorbeeld alleen gemeente is, worden alleen de gemeentelijke producten opgevraagd. Als het bereik tevens provincies of waterschappen bevat, worden ook de provinciale en waterschapsproducten opgevraagd.



Figuur 9: elementen van zoekvraag en zoekgebied

Voorbeeld XML voor het opvragen van *provinciale* producten die voor gemeente Amsterdam van toepassing zijn:

```

<zoekgebied>
  <organisatie>
    <gemeenten>Amsterdam</gemeenten>
  </organisatie>
  <bereik>provincies</bereik>
</zoekgebied>

```

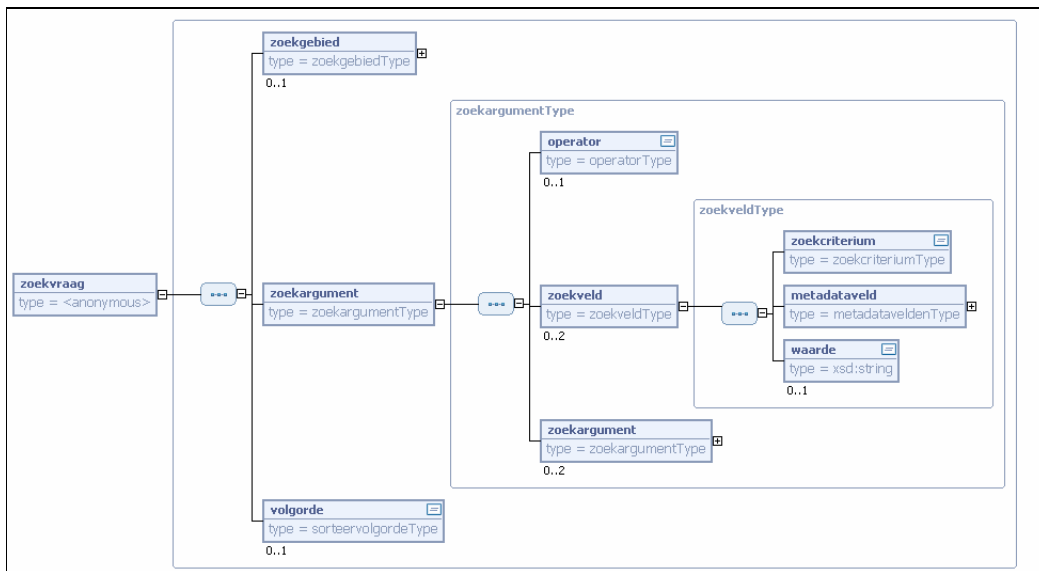
- **Zoekargument.** Een zoekargument bestaat uit een logische operator op één of meer metadata-elementen van Samenwerkende Catalogi met waarden voor die elementen. De operatoren 'AND' en 'OR' worden onderscheiden. Daarnaast kan per metadata veld de wijze van zoeken worden opgegeven:

Type zoekactie	Omschrijving
MATCH-EXACT	De waarde van een metadata-element in de zoekvraag moet exact overeenkomen met dat metadata-element in de index.
MATCH	De waarde in de zoekvraag wordt als 'full text search' gebruikt in het gespecificeerde metadata-veld.
MATCH-EXCEPT	De waarde van een metadata-element in de zoekvraag moet niet voorkomen in het zoekresultaat.
MATCH-EQ-LT	De waarden van het metadata-element van de gevonden resultaten moeten gelijk of kleiner zijn dan de waarde van het



	metadata-element in de zoekvraag. Deze constructie is bijvoorbeeld te gebruiken om alle producten die <u>voor</u> een bepaalde datum zijn toegevoegd te zoeken.
MATCH-EG-GT	De waarden van het metadata-element van de gevonden resultaten moeten gelijk of groter zijn dan de waarde van het metadata-element in de zoekvraag. Deze constructie is bijvoorbeeld te gebruiken om alle producten die <u>na</u> een bepaalde datum zijn toegevoegd te zoeken.

Het is mogelijk om een zoekargument op te bouwen uit een operator, veld en een ander zoekargument. Dit andere zoekargument kan zelf weer een operator, een veld en weer een ander zoekargument zijn (recursie). Een zoekargument kan ook alleen bestaan uit een operator en twee velden.



Figuur 10: elementen van zoekvraag en zoekargument

Voorbeeld XML:

```

<zoekargument>
  <operator>AND</operator>
  <zoekveld>
    <zoekcriterium>MATCH</zoekcriterium>
    <metadatumveld>
      <keyword>paspoort</keyword>
    </metadatumveld>
  </zoekveld>
</zoekargument>
<operator>OR</operator>
<zoekveld>
  <zoekcriterium>MATCH-EXACT</zoekcriterium>
  <metadatumveld>
    <publicerendeOrganisatie>

```



```

                <gemeenten>Amsterdam</gemeenten>
            </publicerendeOrganisatie>
        </metadataveld>
    </zoekveld>
    <zoekveld>
        <zoekcriterium>MATCH-EXACT</zoekcriterium>
        <metadataveld>
            <auteur>
                <deelgemeenten>Amsterdam - Bos en Lommer</deelgemeenten>
            </auteur>
        </metadataveld>
    </zoekveld>
</zoekargument>
</zoekargument>

```

Deze zoekvraag is gelijk aan de expressie:

(keyword MATCH paspoort) AND ((auteur MATCH-EXACT deelgemeente Amsterdam –Bos en Lommer) OR (publicerendeOrganisatie MATCH-EXACT gemeente Amsterdam))

Het is mogelijk om specifiek in één metadata-veld te zoeken, bijvoorbeeld auteur of onderwerp1. Daarnaast is een speciaal element gedefinieerd om in *alle* metadata-elementen van een product te zoeken. Dit is het element 'keyword'

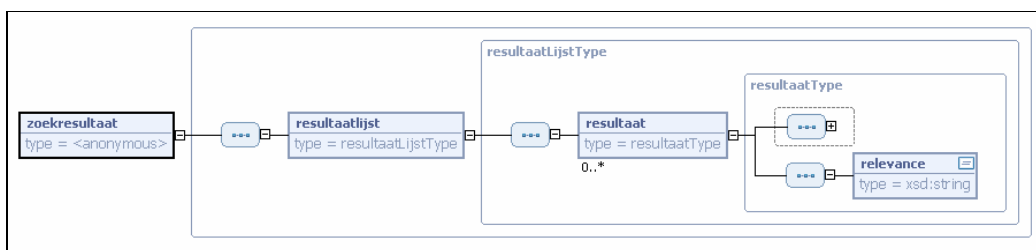
- *Volgorde*. De gewenste sorteervolgorde voor zoekresultaten wordt weergegeven met het sorteervolgordeType. Mogelijke waarden zijn 'datum-meest-recente-eerst', 'datum-minst-recente-eerst' en 'relevantie'.

Voorbeeld XML:

```
<volgorde>datum-meest-recente-eerst</volgorde>
```

Zoekresultaten

De zoekresultaten bestaan uit nul, één of meer producten. Als er geen producten met de zoekvraag worden gevonden, dan wordt alleen de tag 'zoekresultaat' uitgewisseld met een leeg element <resultaatlijst>. Anders volgt een opbouw zoals dit in de volgende figuur wordt getoond.



Figuur 11: elementen van zoekresultaat

Het element 'zoekresultaat' bevat drie attributen om aan te geven welke producten worden geretourneerd:

- results: het maximaal aantal resultaten dat geretourneerd mag worden door de zoekdienst (default: 500)
- start: het eerste product dat geretourneerd moet worden



- total: het totaal aantal producten

Als bijvoorbeeld resultaat 11-20 van in totaal 110 resultaten wordt getoond, ziet het element er als volgt uit:

```
<resultaat start="11" results="10" total="110">
```

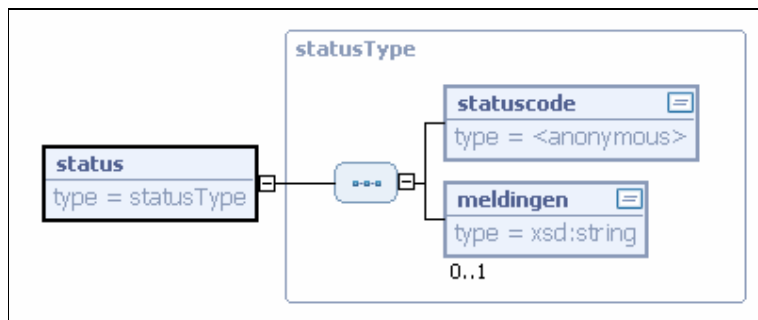
Ieder resultaat in de lijst met zoekresultaten is uit een resultaatType: een product met daarbij een element voor de relevantie van dat product. Hoe elementen van het type product er uit zien wordt verderop beschreven. De waarde voor 'relevance' wordt door de zoekdienst bepaald als een getal tussen 0 en 1.

Voorbeeld XML:

```
<resultaat>
  <auteur>
    <adviescollege>Adviescollege toetsing administratieve lasten</adviescollege>
  </auteur>
  <publicerendeOrganisatie>
    <gemeenten>Amsterdam</gemeenten>
  </publicerendeOrganisatie>
  <productnaam>product1</productnaam>
  <mutatiedatum>2006-05-12T20:00:01</mutatiedatum>
  <onderwerp1>bui.intern</onderwerp1>
  <onderwerp1>gez.handicap</onderwerp1>
  <onderwerp2>onderwerp2</onderwerp2>
  <doelgroep>organisatie/ondernemer </doelgroep>
  <taal>nl</taal>
  <url>http://advies.overheid.nl</url>
  <productID>id1</productID>
  <relevance>10</relevance>
</resultaat>
```

Status

Status wordt gebruikt voor het weergeven van de status van een zoekvraag.



Figuur 12: elementen van status

Voorbeeld XML:

```
<statuscode>fout</statuscode>
```



<meldingen>de waarde van een publicerende organisatie komt niet uit een lijst met gemeenten</meldingen>

Het statuscode element kan de waarden 'ok' en 'fout' bevatten. Het meldingen-element is een vrije tekst veld om foutmeldingen aan de gebruiker te tonen. Op het moment van schrijven wordt het status bericht alleen geretourneerd wanneer in de zoekvraag het zoekvraag element ontbreekt. Bij alle overige technische fouten komt alleen een HTTP foutcode terug.

3.4. Algemeen gebruikte types

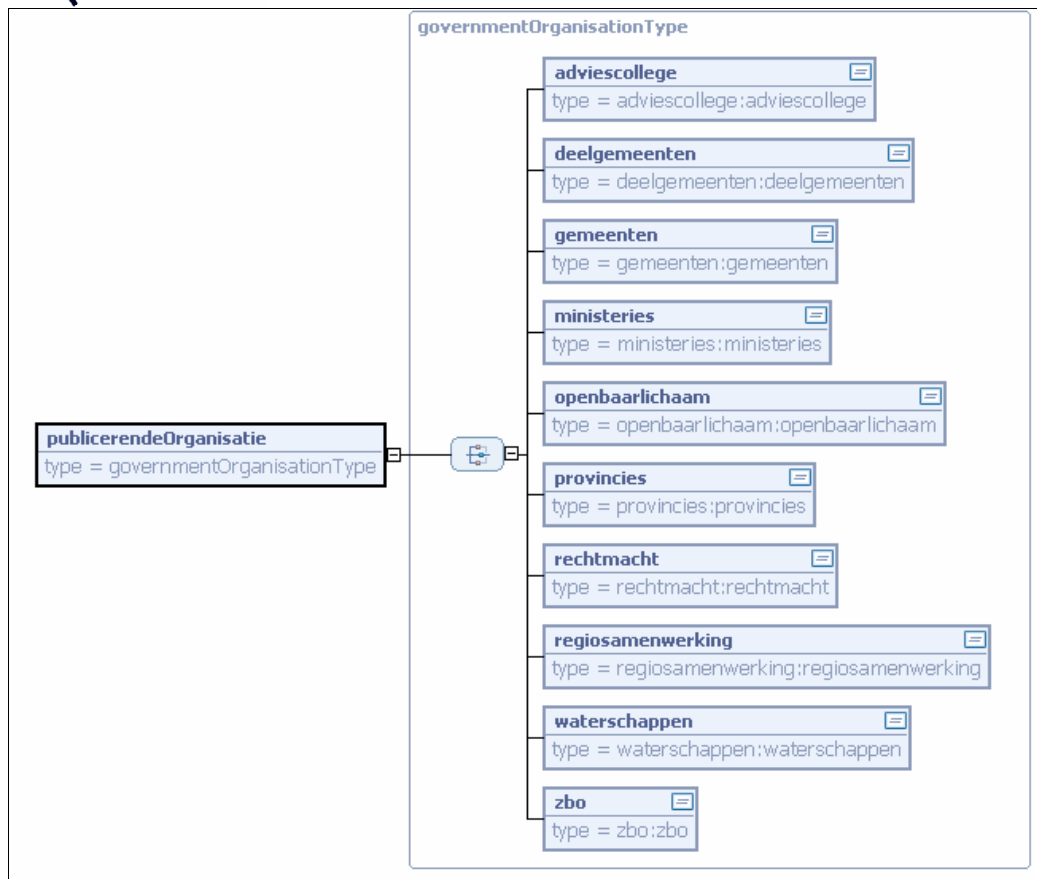
De meest gebruikte elementen zijn met een aantal generieke types gespecificeerd. Deze types zijn door meerdere elementen te gebruiken, zodat identieke elementen ook op dezelfde wijze gespecificeerd zijn. productType wordt bijvoorbeeld gebruikt in de lijst van producten die geïndexeerd moeten worden, maar ook in de lijst van zoekresultaten. Het doel van elk type wordt hieronder kort toegelicht. Deze type definities volgen in het schema bestand na de elementen.

Organisaties en hun typering

Een product heeft een auteur en een publicerende organisatie. Deze kunnen gelijk zijn, maar kunnen ook verschillen. Een deelgemeente is bijvoorbeeld een auteur, terwijl een gemeente het product publiceert.

Een publicerende organisatie en een auteur zijn van een organisatieType en hebben een naam. De naam komt uit een lijst met toegestane waarden (zie eerder). Beide elementen zijn van hetzelfde type, te weten 'governmentOrganisationType', en hebben daarmee dezelfde structuur.





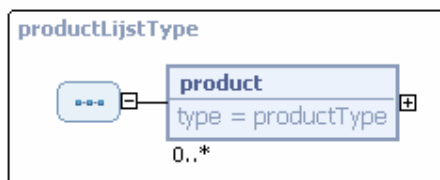
Figuur 13: opbouw van 'publicerendeOrganisatie'

Voorbeeld XML, governmentOrganisationType gebruikt in de definitie van <auteur> :

```
<auteur>
  <waterschappen>Waterschap Brabantse Delta</waterschappen>
</auteur>
```

productListType

Dit type geeft een lijst van producten. Deze lijst kan leeg zijn.



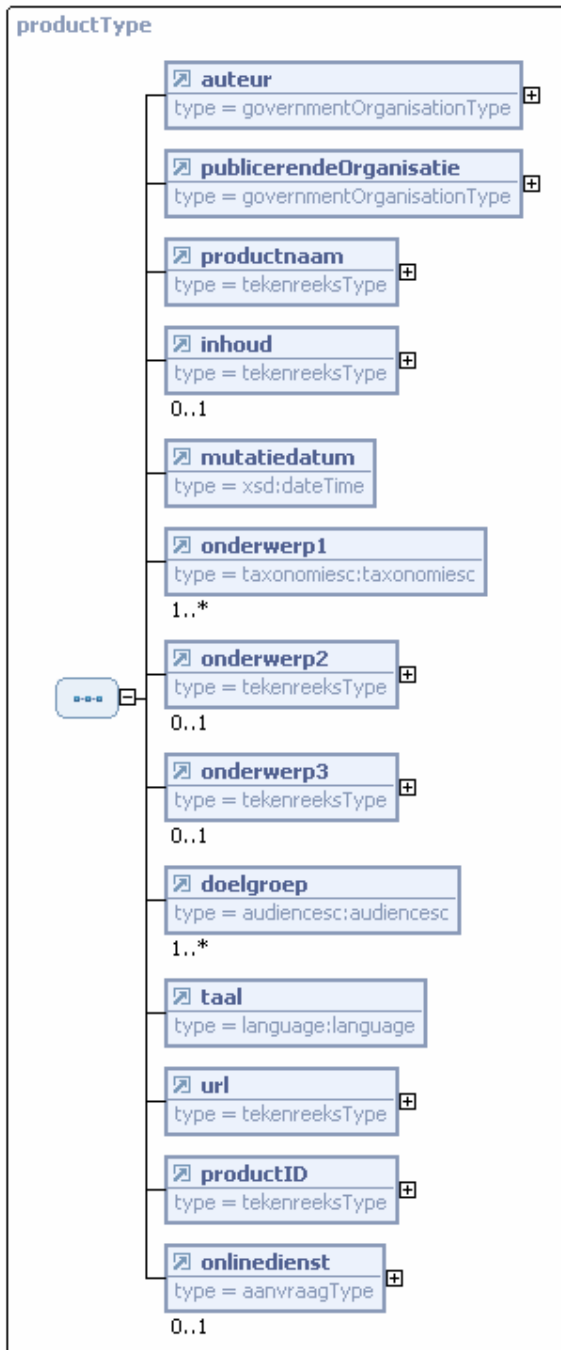
Figuur 14: opbouw van een lijst met producten



productType

productType is het belangrijkste type van het schema. Het is de basis voor de lijst van producten die geïndexeerd worden en de lijst van zoekresultaten van een zoekvraag. Dit type definieert alle meta-gegevens die in 'technische documentatie metadata' beschreven worden. Elk element van dit type mag in totaal (inclusief tags) niet meer dan 30.000 tekens groot zijn.





Figuur 15: elementen van een product

Hieronder volgt een voorbeeld uit een XML-bericht dat voldoet aan productType:

```
<product>
  <auteur>
    <adviescollege>Adviescollege toetsing administratieve lasten</adviescollege>
  </auteur>
  <publicerendeOrganisatie>
    <gemeenten>Amsterdam</gemeenten>
  </publicerendeOrganisatie>
  <productnaam>product1</productnaam>
  <mutatiedatum>2006-05-12T20:00:01</mutatiedatum>
  <onderwerp1>bui.intern</onderwerp1>
  <onderwerp1>gez.handicap</onderwerp1>
  <onderwerp2>onderwerp2</onderwerp2>
  <doelgroep>particulier</doelgroep>
  <taal>nl</taal>
  <url>http://advies.overheid.nl</url>
  <productID>id1</productID>
  <onlinedienst>
    <online>ja</online>
    <aanvraagurl>http://www.test.nl</aanvraagurl>
  </onlinedienst>
</product>
```

tekenreeksType

Type voor het weergeven van tekst waarin een deelverzameling van HTML-tags voor mag komen. In elementen van dit type mogen momenteel 'string' waarden voorkomen.

4. Voorbeelden implementatie zoekfunctie

Voorbeeld eenvoudig HTML-zoekscherm:

```
<html>
  <body>
    <form action="http://www.eengemeente.nl/zoeken" method="POST">
      Zoek op trefwoord: <input type="text" name="trefwoord" id="trefwoord"/>
      <input type="submit" value="OK"/>
    </form>
  </body>
</html>
```

Voorbeeld van een eenvoudige Java Servlet

Een eenvoudige servlet die achtereenvolgens:

- de input van het bovenstaande HTML-formulier ontvangt;
- de zoekterm uitleest;
- een XML-zoekvraag opbouwt;



Advies Overheid.nl bouwt mee aan de e-overheid

Titel: IPM Samenwerkende Catalogi: technische architectuur en XML
Versie: 2.1
Datum: 1 juni 2007

- de zoekvraag verstuurt naar de zoekdienst;
- het antwoord (zoekresultaat) van de zoekdienst ontvangt;
- het zoekresultaat vertaalt van XML naar HTML en terugstuurt naar de browser.

```
import java.io.IOException;

import java.io.OutputStream;

import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

public class SCXMLServlet extends HttpServlet {

    private static String ZOEKSERVER_URL = "http://www.dezoekserver.nl";

    private static String OPMAAK_STYLESHEET = "http://www.eengemeente.nl/zoekresultaat.xsl";

    public void doPost(HttpServletRequest req,HttpServletResponse res){
```



```
// lees de zoekterm die door de gebruiker is verstuurd
String zoekterm=req.getParameter("trefwoord");
Document zoekvraag=null;
try {
    //bouw het XML document op basis van de zoekterm
    zoekvraag = bouwDocument(zoekterm);
} catch(ParserConfigurationException e){
    handleError(e);
}
Document zoekresultaat=null;
try {
    //verstuur dit document naar de zoekserver en ontvang het antwoord
    zoekresultaat = verstuurZoekvraag(zoekvraag, ZOEKSERVER_URL);
} catch(Exception e){
    handleError(e);
}
//transformeer het resultaat met XSLT naar HTML en stuur dit naar de browser
try {
    verwerkResultaat(zoekresultaat, OPMAAK_STYLESHEET, res.getOutputStream());
} catch(Exception e){
    handleError(e);
}
}

private Document bouwDocument(String zoekterm) throws ParserConfigurationException {
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document document = builder.newDocument();
    Element root = document.createElement("zoekvraag");
    document.appendChild(root);
    Element zoekargument = document.createElement("zoekargument");
    root.appendChild(zoekargument);
    Element zoekveld = document.createElement("zoekveld");
    zoekargument.appendChild(zoekveld);
    Element zoekcriterium = document.createElement("zoekcriterium");
    zoekveld.appendChild(zoekcriterium);
}
```



```
Element metadataveld = document.createElement("metadataveld");
zoekveld.appendChild(metadataveld);

Element keyword = document.createElement("keyword");
metadataveld.appendChild(keyword);
keyword.appendChild(document.createTextNode(zoekterm));
return document;
}

private Document verstuurZoekvraag(Document zoekvraag, String urlString)
throws TransformerException, MalformedURLException, IOException, SAXException, ParserConfigurationException {
    //zet de authenticatie gegevens, zie verder voor MyAuthenticator class

    MyAuthenticator authenticator = new MyAuthenticator();
    authenticator.username = USERNAME;
    authenticator.password = PASSWORD;
    Authenticator.setDefault(authenticator);

    //bouw de URL op

    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setDoOutput(true);
    conn.setDoInput(true);
    conn.setUseCaches (false);
    conn.setRequestProperty("Content-Type", "text/xml");
    conn.setRequestProperty("Connection","keep-alive");
    conn.setRequestMethod("POST");

    //stuur het XML document via een stream naar de URL

    OutputStream out = conn.getOutputStream();
    DOMSource source = new DOMSource(zoekvraag);
    TransformerFactory tFactory = TransformerFactory.newInstance();
    Transformer transformer = tFactory.newTransformer();
    transformer.transform(source, new StreamResult(out));
    out.close();

    //parse het zoekresultaat tot een document

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder builder = factory.newDocumentBuilder();
    Document zoekresultaat = builder.parse(new InputSource(conn.getInputStream()));
    return zoekresultaat;
}
```



```
}

private void verwerkResultaat(Document zoekresultaat, String opmaakStylesheet, ServletOutputStream out)
throws IOException, MalformedURLException, TransformerConfigurationException, TransformerException {
    StreamSource stylesource = new StreamSource(new URL(opmaakStylesheet).openStream());
    StreamResult result = new StreamResult(out);
    DOMSource source = new DOMSource(zoekresultaat);
    TransformerFactory tFactory = TransformerFactory.newInstance();
    Transformer transformer = tFactory.newTransformer(stylesource);
    transformer.transform(source, result);
    out.flush();
    out.close();
}

private void handleError(Exception e){
    System.out.println(e.getMessage());
    e.printStackTrace();
    //verdere foutafhandeling
}
}

class MyAuthenticator extends Authenticator {
    public String username;
    public String password;
    // This method is called when a password-protected URL is accessed
    protected PasswordAuthentication getPasswordAuthentication() {
        // Get information about the request
        String promptString = getRequestingPrompt();
        String hostname = getRequestingHost();
        InetAddress ipaddr = getRequestingSite();
        int port = getRequestingPort();

        // Return the information
        return new PasswordAuthentication(username, password.toCharArray());
    }
}
}
```

Voorbeeld van een eenvoudige XSLT stylesheet

Deze stylesheet vertaalt het XML-zoekresultaat naar HTML met daarin productnamen en inhoud:



Advies Overheid.nl bouwt mee aan de e-overheid

Titel: IPM Samenwerkende Catalogi: technische architectuur en XML
Versie: 2.1
Datum: 1 juni 2007

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" omit-xml-declaration="no" standalone="yes" indent="yes"/>

  <xsl:template match="/">
    <html>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="zoekresultaat">
    <xsl:choose>
      <xsl:when test="count(resultaatlijst/resultaat)=0">
        Er waren geen zoekresultaten
      </xsl:when>
      <xsl:otherwise>
        Zoekresultaten:
        <xsl:for-each select="resultaatlijst/resultaat">
          <table class="resultatentabel">
            <tr>
              <td>Productnaam</td>
              <td><xsl:value-of select="productnaam"/></td>
            </tr>
            <tr>
              <td>Omschrijving</td>
              <td><xsl:value-of select="inhoud"/></td>
            </tr>
            <!-- etc -->
          </table>
        </xsl:for-each>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

</xsl:stylesheet>

```

Voorbeeld met gebruik van ASP DOM

In dit voorbeeld wordt een zoekvraag in XML verwerkt en verstuurd naar de zoekdienst. Het resultaat wordt ontvangen en met behulp van een XSLT-stylesheet getoond in de browser.

```
<%
```

```

'*****
'Voor de leesbaarheid zijn het declareren, schonen en verwijderen van variabelen weggelaten
'*****

```

```

'*****
'ZoekXML is de samengestelde XML string waarin de zoekvraag verwerkt is
'<?xml version="1.0" encoding="UTF-8"?>

```



Advies Overheid.nl bouwt mee aan de e-overheid

Titel: IPM Samenwerkende Catalogi: technische architectuur en XML
 Versie: 2.1
 Datum: 1 juni 2007

```
'<zoekvraag xmlns="http://www.adviesoverheid.nl/metadata-project/samenwerkende-
catalogi/2006-1/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.adviesoverheid.nl/metadata-project/samenwerkende-
catalogi/2006-1/ SamenwerkendeCatalogi.xsd">
'   <zoekargument>
'       <zoekveld>
'           <zoekcriterium>MATCH-EXACT</zoekcriterium>
'           <metadataveld><productnaam>Verhuizing van Amsterdam naar andere
plaats in Nederland</productnaam></metadataveld>
'       </zoekveld>
'   </zoekargument>
'</zoekvraag>
'*****
```

ZoekXML = *** Vervangen door XML string ***

```
Set source = Server.CreateObject("Microsoft.XMLDOM")
source.async = false
source.load ZoekXML
strUser = *** Vervangen door USERNAME ***
strPwd = *** Vervangen door PASSWORD ***
```

```
Set objXmlHttp = Server.CreateObject("MSXML2.ServerXMLHTTP")
objXmlHttp.open "POST",
"http://zoekdienst.overheid.nl/ZoekdienstXML/xml.sc.overheid.nl/sc", False, strUser,strPwd
objXmlHttp.send source.xml
```

If ObjXmlHttp.status = 200 Then

```
Set ObjXML = Server.CreateObject("Microsoft.XMLDOM")
Set ObjXML = objXmlHttp.responseXML
```

```
Set ObjXSL = Server.CreateObject("Msxml2.DOMDocument")
```

```
'*****
'Toon de resultaten mbv van een XSLT Stylesheet
'In het voorbeeld is een fictief Stylesheet bestand gebruikt: resultaat.xsl
```

```
'*****
```

```
ObjXSL.load Request.ServerVariables("APPL_PHYSICAL_PATH") & "\resultaat.xsl"
Response.Write (ObjXML.transformNode(ObjXSL))
Else
Response.Write "Er is een fout opgetreden, probeer het later nog eens."
End If
```

%>

Hieronder vind u een voorbeeld van een XSLT Stylesheet.
Microsoft gebruikers: let goed op het gebruik van de NameSpace.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:scPrefix="http://www.adviesoverheid.nl/metadata-project/samenwerkende-
catalogi/2006-1/">
<xsl:output method="html"/>

<xsl:template match="scPrefix:zoekresultaat">
<H1>Resultaatlijst</H1>
<table>

<xsl:for-each
select="scPrefix:resultaatlijst/scPrefix:resultaat/scPrefix:auteur/scPrefix:ministeries">
  <tr><td><xsl:value-of select="name()"/></td><td><xsl:value-of
select="."/></td></tr>
</xsl:for-each>

<xsl:for-each select="scPrefix:resultaatlijst/scPrefix:resultaat">

<xsl:for-each select="scPrefix:productnaam">
  <tr><td><xsl:value-of select="name()"/></td><td><xsl:value-of
select="."/></td></tr>
</xsl:for-each>

<xsl:apply-templates select="scPrefix:onderwerp1"/>

<xsl:for-each select="scPrefix:doelgroep">
  <tr><td><xsl:value-of select="name()"/></td><td><xsl:value-of
select="."/></td></tr>
</xsl:for-each>
  <tr><td>publicerende organisatie: </td><td><xsl:value-of
select="scPrefix:auteur"/></td></tr>
  <tr><td
colspan="2">*****</td></tr>
</xsl:for-each>

</table>

</xsl:template>

<xsl:template match="scPrefix:onderwerp1">
  <tr><td><xsl:value-of select="name()"/></td><td><xsl:value-of
select="."/>;</td></tr>
</xsl:template>
</xsl:stylesheet>
```

