e-Business voor Web4all

van Westreenen, Jurgen

Award date:
2003

TECHNISCHE UNIVERSITEIT EINDHOVEN

Faculteit Wiskunde en Informatica

AFSTUDEERVERSLAG

e-Business voor Web4all

door

Jurgen van Westreenen

Afstudeerdocent: Prof. dr. P.M.E. De Bra

november 2003

Voorwoord

Er zijn verschillende personen die ik zou willen bedanken voor hun steun en hulp tijdens mijn afstuderen waaronder mijn beide begeleiders (Paul de Bra en Paul Waasdorp) en verscheidene medewerkers van Centric e-Technology.
Maar er is één persoon in het bijzonder, die me altijd gesteund heeft en altijd vertrouwen heeft gehad in mijn kunne, en aan haar wil ik deze scriptie opdragen.

Bedankt Ma, voor alles.

Samenvatting

Via de moderne digitale communicatiemiddelen (internet, e-mail, SMS, enz.) is het mogelijk om met weinig inspanning een grote groep mensen van informatie te voorzien. Veel bedrijven, maar ook non-commerciele instellingen, zoals gemeentes zijn daarom (ook) vertegenwoordigd op het internet.
Het verschaffen van informatie en diensten is echter slechts één zijde van de medaille. Het is voor de aanbieder van deze informatie namelijk ook zeer prettig om meer over zijn gebruikers te weten te komen. Kennis over klanten is immers van onschatbare waarde voor met name bedrijven, omdat met die informatie beter op de wensen van de klant kan worden ingespeeld.
Het is dus zaak om zoveel mogelijk informatie over de gebruikers te vergaren.
Tijdens mijn afstuderen heb ik onderzocht hoe die informatie daadwerkelijk verzameld kan worden en globaal gezien kan dat op de volgende twee manieren:

Het mooiste zou zijn als de gebruiker zelf zou aangeven waar zijn voorkeuren liggen. Dit kan ook door gebruik te maken van paspoorten. Een paspoort is een soort formulier op de website waarin de gebruiker zelf zijn gegevens en voorkeuren kan invullen. Met deze gegevens kan vervolgens een gebruikersprofiel worden samengesteld.

Een andere manier om meer over de gebruiker te weten te komen, is door hem te 'volgen' tijdens zijn verblijf op de website. Door precies bij te houden wat een gebruiker allemaal ziet en doet op de website, kunnen er conclusies worden getrokken over wat de gebruiker interessant lijkt te vinden. Bovendien kan er op deze manier statistische informatie worden vergaard en kunnen eventuele 'bottlenecks' worden gedetecteerd.

Al deze informatie kan worden opgeslagen en verwerkt. Deze gegevens kunnen gebruikt worden om op de wensen van de gebruiker in te spelen. Het is zelfs mogelijk om de website aan te passen aan de gebruiker. Uit mijn onderzoek bleek dat dit personaliseren eveneens op verscheidene manieren kan, waaronder de volgende:

Een relatief eenvoudige manier om de website een persoonlijk tintje te geven, is door gegevens uit het profiel direct toe te voegen aan de inhoud van een pagina. Zo kan een bezoeker bijvoorbeeld persoonlijk begroet worden, doordat diens naam onderdeel is van het profiel.

Een meer diepgaande methode maakt het mogelijk dat de website zichzelf aanpast aan de gebruiker. Dit kan door de website zelf 'beslissingen' te laten nemen over het gedrag van de site op basis van het gebruikersprofiel.
Dit kan enerzijds door de navigatie aan te passen: zo kan een hyperlink naar meerdere bestemmingen wijzen. Afhankelijk van de gebruiker wordt er dan een keuze gemaakt uit de mogelijke bestemmingen.
Een andere mogelijkheid is om de inhoud van een pagina afhankelijk te laten zijn van het gebruikersprofiel: zo kan gebruiker A andere content te zien krijgen dan gebruiker B.

Al deze aspecten samen maken het mogelijk dat een 'gewone' website kan worden veranderd in een 'slimme' webomgeving voor het bedrijven van e-business.

Na mijn onderzoek heb ik een ontwerp gemaakt voor Web4all waarin wordt aangegeven hoe een uitbreiding met e-business-functionaliteit tot stand kan komen.

Web4all is een ontwikkelomgeving voor een intranet en internet-websites. Om nu bovengenoemde functionaliteit aan dit product toe te voegen, heb ik de volgende componenten gedefinieerd:

- Paspoorten
- Registrators
- Tags
- Routers

Zoals gezegd maken paspoorten het mogelijk dat de gebruiker zelf zijn voorkeuren aangeeft.

Registrators bieden de functionaliteit om de gebruiker te volgen tijdens zijn bezoek aan de website en zodoende de voorkeuren van de gebruiker af te leiden uit diens surfgedrag.

Tags maken het mogelijk om profielgegevens direct in de content van de website op te nemen.

Routers zorgen ervoor dat de website zich aan kan passen aan de gebruiker door beslissingen te nemen voor wat betreft de navigatie tussen en de inhoud van de verschillende pagina's.

Uiteindelijk is van de besproken onderdelen de router-functionaliteit ook daadwerkelijk geïmplementeerd. Door middel van een zogeheten content-router is het nu binnen Web4all mogelijk om het systeem een keuze te laten maken over de te tonen content op basis van het profiel van de gebruiker.

Inhoudsopgave

VOORWOORD **1**

SAMENVATTING **2**

INHOUDSOPGAVE **4**

1 INLEIDING **8**

1.1 CENTRIC E-TECHNOLOGY 8
1.2 WEB4ALL 8
1.3 DATA WAREHOUSING 9
1.4 OPDRACHT 9

2 INFORMATIEVERGARING **10**

2.1 NECKERMANN 10
2.1.1 Cookies *10*
2.1.2 Sessions *10*
2.1.3 Paspoorten *11*
2.1.4 Combinatie *11*
2.1.5 Thomas Cook *12*
2.1.6 Reclame & Advertenties *13*
2.2 RENAULT 14
2.2.1 Bezoekersstatistieken *14*
2.2.2 Pagehits *15*
2.2.3 Persoonlijke gegevens *16*
2.3 OVERIGE ASPECTEN 17
2.3.1 Surfgedrag buiten de website *17*
2.3.2 Andere digitale media *17*
2.3.3 Inloggen *18*

3 INFORMATIESCHEMA **19**

4 WEB4ALL **20**

4.1 WAT IS WEB4ALL? 20
4.2 GLOBALE ARCHITECTUUR 21
4.3 E-BUSINESS 22

5 REQUIREMENTS **23**

5.1 FUNCTIONELE REQUIREMENTS 23
5.2 PROFIEL REQUIREMENTS 24
5.3 BEZOEKERS REQUIREMENTS 24
5.4 BEPERKENDE REQUIREMENTS 24

6 CONCEPT **25**

6.1 VERGARING 25
6.1.1 Paspoorten *25*

6.1.2 Registrators ... 25
6.2 VERWERKING ... 26
6.2.1 DataShare ... 26
6.3 TOEPASSING ... 26
6.3.1 Routers ... 27
6.3.2 Tags ... 28
7 GLOBAAL ONTWERP ... 29
7.1 ATTRIBUTEN ... 29
7.1.1 Paspoort-attributen ... 29
7.1.2 Registrator-attributen ... 29
7.1.3 Systeem-attributen ... 29
7.1.4 Berekende attributen ... 30
7.1.5 DataShare ... 30
7.2 REGISTRATORS ... 31
7.3 ROUTERS ... 32
7.3.1 URL-router ... 32
7.3.2 Content-router ... 33
7.4 TAGS ... 33
7.5 EXPRESSIES ... 34
7.5.1 Attribuutnotatie ... 34
7.6 PREVIEWS ... 36
7.7 TEMPLATES ... 36
8 GRAFISCH ONTWERP ... 38
9 TECHNISCH ONTWERP ... 39
9.1 CONTENTHOSTING ... 39
9.1.1 Externe ContentHosting ... 41
9.2 DATABASE ... 41
9.3 CLASSES ... 41
9.3.1 Attributen ... 41
9.3.2 Expressies ... 42
9.3.3 Routers ... 42
9.3.4 Registrators ... 42
9.3.5 Tags ... 43
9.3.6 Preview Templates ... 43
10 IMPLEMENTATIE ... 44
10.1 INLEIDING ... 44
10.1.1 Attributen engine ... 44
10.1.2 Expressie engine ... 44
10.1.3 Content-router ... 45
10.1.4 Grafische User Interface ... 45
10.2 CLASSES ... 45
11 CONCLUSIES & SUGGESTIES ... 46
11.1 AFSTUDEERVERLOOP ... 46

11.2 HUIDIGE STAND VAN ZAKEN 46
11.3 EXPRESSIE EVALUATIE 47
11.4 DATABASE ACCESS 48
11.5 ZOEKFUNCTIONALITEIT 48
11.6 IMPORT/EXPORT 49
11.7 DATASHARE 49

12 REFERENTIES 50

12.1 LINKS 50

13 BIJLAGENOVERZICHT 51

14 BIJLAGE 1: INFORMATIESCHEMA 52

14.1 SYSTEEMLAGEN 52
14.2 COMPONENTEN 54
14.2.1 Banner / Website 54
14.2.2 E-mail Inbound 55
14.2.3 E-mail Outbound 55
14.2.4 SMS Inboud 56
14.2.5 SMS Outbound 56
14.2.6 I-mode 56
14.2.7 Call-center 57
14.2.8 Contact Registratie 57
14.2.9 Personalisatie 58
14.2.10 Procescache/CMS 58
14.2.11 CRM/Paspoort 58
14.2.12 Procesverwerking 58
14.2.13 Data Warehouse 58
14.2.14 Statistieken 59
14.3 INPUT EN OUTPUT 60
14.4 INFORMATIESTROMEN 62

15 BIJLAGE 2: ONTWERPBESLISSINGEN M.B.T. ROUTERS 65

15.1 URL ROUTER 65
15.2 CONTENT ROUTER 66

16 BIJLAGE 3: GRAFISCH ONTWERP 67

16.1 BEREKENDE ATTRIBUTEN 67
16.1.1 Opmerkingen 69
16.2 REGISTRATORS 70
16.2.1 Opmerkingen 72
16.3 URL ROUTERS 73
16.3.1 Opmerkingen 74
16.4 CONTENT ROUTERS 74
16.4.1 Opmerkingen 76
16.5 TAGS 76
16.6 EXPRESSIES 77
16.7 PREVIEWS 78

17 BIJLAGE 4: ER-DIAGRAM 79
17.1 TOELICHTING 79
17.2 OVERZICHT DATATABELLEN 82
17.3 INVULLING DATATABELLEN 84
17.3.1 Attributen 84
17.3.2 Registrators 85
17.3.3 Routers 86
17.3.4 Tags 87
17.3.5 Expressies 88
17.3.6 Preview Templates 89
18 BIJLAGE 5: CLASSES 90
18.1 ATTRIBUTEN 90
18.2 ROUTERS 91
18.3 EXPRESSIES 92
18.4 GRAFISCHE USER INTERFACE 93
18.5 CONTENTHOSTING 94

1 Inleiding

De moderne digitale communicatiemiddelen (www, i-mode, e-mail, SMS) maken het mogelijk om op een snelle en efficiënte manier informatie en diensten aan een groot publiek aan te bieden. Vooral bedrijven, maar ook non-commerciële instellingen hebben zeer veel baat bij het gebruik van deze media.
Behalve informatie *voor* de gebruikers, kunnen deze media echter ook informatie *over* de gebruikers opleveren. Deze gegevens kunnen gebruikt worden om een beter beeld te krijgen van de gebruiker en dat kan voor beide partijen zo zijn voordelen hebben. Als een bedrijf of instelling immers weet wat de gebruiker wil, dan zal daar dankbaar gebruik van worden gemaakt, maar ook voor de gebruiker zelf is het bijzonder prettig als hij zonder al te veel moeite die informatie voorgeschoteld krijgt waarnaar hij op zoek is.
Tijdens mijn afstuderen heb ik mij verdiept in de verschillende mogelijkheden om informatie over gebruikers te vergaren, te verwerken en vervolgens weer toe te passen. Alvorens hier in detail op in te gaan, zal ik eerst wat informatie geven over het bedrijf waar ik mijn afstudeerwerk heb gedaan (Centric e-Technology) en over één van de lopende projecten (Web4all). Verder zal ik kort stilstaan bij het begrip Data Warehousing aangezien dit nauw verbonden is met de uiteindelijke opdracht.
De laatste sectie van dit hoofdstuk behandelt de eigenlijke opdracht en de opbouw dan de rest van dit document.

1.1 Centric e-Technology

Centric e-Technology maakt deel uit van automatiseerder Centric en richt zich voornamelijk op e-toepassingen. Enkele projecten waar Centric-et zich momenteel mee bezighoudt, zijn de websites van Neckermann Reizen, Vrij Uit, Renault Nederland en ook enkele eigen producten, waaronder Web4all.
Al deze projecten bezitten in meer of mindere mate de functionaliteit om informatie over gebruikers op te slaan, maar in de meeste gevallen ontbreekt het nog aan de mogelijkheid om deze opgeslagen informatie effectief toe te passen.

1.2 Web4all

Web4all is een product voor de ontwikkeling en exploitatie van een webomgeving. Het bijbehorende Content Management Systeem (CMS) maakt het mogelijk om inter- en intranetsites te creëren. Tevens biedt het de mogelijkheid om door middel van kant-en-klare bouwstenen, zogenaamde modules, extra functionaliteit aan de sites toe te voegen. Deze modulaire opbouw maakt het relatief eenvoudig om nieuwe functionaliteit toe te voegen. Verderop in deze scriptie zal er nog uitgebreid worden teruggekomen op dit project.

1.3 Data Warehousing

Een definitie van Data Warehousing is de volgende:

"Data Warehousing describes the process of defining, populating, and using a collection of data designed to support management decision making. Data warehousing emphasizes the capture of data from diverse sources for useful analysis and access. Data from various transaction processing applications and other sources is selectively extracted and organized on the data warehouse database for use by analytical applications and user queries."

-- http://techdivas.com/data.htm

Met andere woorden: Data Warehousing is het proces van informatie vergaren, verwerken en toepassen. De informatie wordt zodanig gestructureerd, dat er makkelijk query's en analyses op losgelaten kunnen worden. Belangrijk hierbij is dat de gegevens van verschillende bronnen afkomstig kunnen zijn, maar dat, door zorgvuldige structurering, de data toch samengenomen en gecombineerd kunnen worden.

1.4 Opdracht

Zoals gezegd was er al wel de nodige functionaliteit om informatie over gebruikers op te slaan, maar vrijwel geen functionaliteit om deze informatie ook te kunnen toepassen. Om nu een overzicht te krijgen van de gehele situatie, was het allereerst nodig om het proces van informatie vergaren, verwerken en toepassen goed in beeld te brengen en te kijken naar nieuwe mogelijkheden. Hierbij is vooral gekeken naar het Neckermann project, maar ook aspecten van informatieverwerking en –toepassing zijn in het onderzoek meegenomen. De resultaten van dit onderzoek zullen in de eerstvolgende twee hoofdstukken uitvoerig worden behandeld.
Vervolgens zijn de resultaten van het onderzoek gebruikt om Web4all uit te breiden met functionaliteit voor het bedrijven van e-business. Wat dat precies inhoudt en hoe het gehele proces van ontwerp en implementatie is verlopen, zal in de daarop volgende hoofdstukken behandeld worden.
Tot slot zal er in het hoofdstuk 'Conclusies & Suggesties' worden teruggekeken op het gehele afstudeertraject en zullen er aanbevelingen worden gedaan voor wat betreft 'toekomstige benodigdheden' (future needs).

2 Informatievergaring

In de nu komende secties zal er besproken worden hoe in de verschillende projecten van Centric-et de vergaring van gebruikersinformatie is gerealiseerd.
Er zal achtereenvolgens gekeken worden naar de projecten van:

- Neckermann Reizen / Thomas Cook
- Renault Nederland

Tot slot zullen er nog enkele andere aspecten van informatievergaring behandeld worden.

2.1 Neckermann

De belangrijkste elementen m.b.t. informatievergaring binnen de Neckermann website zijn cookies, sessions en paspoorten. Hieronder zal ieder element belicht worden.

2.1.1 Cookies

Een cookie is een klein tekstbestandje (maximaal 4 KB) dat automatisch op de harde schijf van de gebruiker kan worden geplaatst, zodra deze de website voor het eerst bezoekt. De website kan tevens via de browser de cookie weer opvragen en zo bepalen of er sprake is van een terugkerende gebruiker. Daarnaast kan een cookie nog een beperkte hoeveelheid extra informatie bevatten.
In het geval van het Neckermann-project is de enige bruikbare informatie die de cookie bevat een nummer (User ID). Dit unieke nummer wordt in de database opgeslagen en gekoppeld aan eigenlijk alle handelingen die de gebruiker op de website verricht, zodat deze informatie bij een terugkerend bezoek is op te vragen.
Nu kunnen cookies door de gebruiker verwijderd of zelfs geweigerd worden. Dit zorgt ervoor dat een gebruiker niet herkend kan worden als terugkerend. In de praktijk blijkt echter dat slechts een klein deel van de gebruikers cookies verwijdert/weigert, vaak omdat ze niet eens van het bestaan van cookies afweten.
Meer informatie over cookies is te vinden op Cookie Central **[1]**.
Sinds Internet Explorer 6 is er echter sprake van de zogenaamde 'Compact Privacy Policy'. Dit maakt het aanzienlijk moeilijker om permanente cookies te gebruiken en dus om terugkerende gebruikers te kunnen detecteren. Meer informatie hierover is te vinden in de Microsoft Knowledge Base **[2]**.

2.1.2 Sessions

Zodra een gebruiker een website bezoekt, wordt er automatisch een zogenaamde "session" gestart. Tijdens zo'n sessie wordt er precies bijgehouden wat de gebruiker

allemaal op de website doet: welke pagina's hij bezoekt, hoe lang hij op een pagina verblijft, via welk "surfpad" hij de website doorzoekt, enz.
Aan het begin van een sessie krijgt de gebruiker een Session ID toegewezen. Vervolgens worden alle handelingen van de gebruiker bijgehouden en opgeslagen in de database gekoppeld aan dat SessionID. Zodoende kan er veel informatie worden verzameld over het surfgedrag en de voorkeuren van de gebruikers

2.1.3 Paspoorten

De website van Neckermann Reizen maakt gebruik van zogenaamde paspoorten. Een gebruiker kan zelf een paspoort aanmaken door op de website een formulier in te vullen met daarin zijn naam- en adresgegevens en eventuele voorkeuren m.b.t. vakanties (welk land, welk type vakantie, hotel of appartement, enz). Deze informatie wordt vervolgens opgeslagen in de database. Een paspoort vertoont veel overeenkomsten met wat in de literatuur wordt aangeduid als een 'User Model' [3]
Met de verstrekte gegevens is het mogelijk om meer op de wensen van de klant in te spelen. Overigens, als de gebruiker een boeking plaatst op de website, dan wordt automatisch een paspoort voor hem aangemaakt, tenzij deze er al één had.

2.1.4 Combinatie

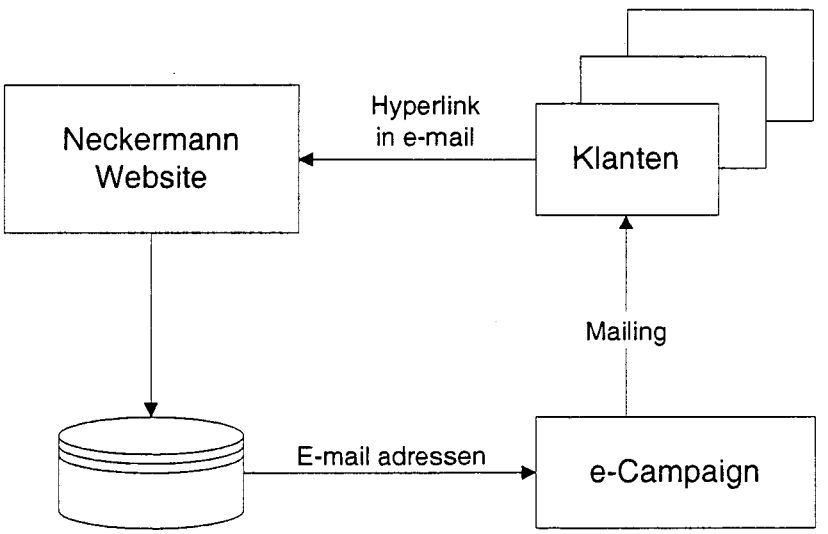
De combinatie van de drie hierboven genoemde elementen is genoeg om de gebruiker te herkennen en te volgen tijdens zijn bezoek aan de website:

1. De cookie zorgt allereerst voor de mogelijkheid om een gebruiker als terugkerend te kunnen bestempelen.
2. Tijdens de sessie worden de handelingen van de gebruiker opgeslagen in de database. M.b.v. de cookie kunnen bovendien eerdere sessies aan diezelfde gebruiker worden gekoppeld, zodat de verschillende sessies kunnen worden vergeleken en gecombineerd.
3. Tot slot zorgen de (door de gebruiker zelf opgegeven) persoonlijke gegevens en voorkeuren ervoor dat een gebruiker ook letterlijk geïdentificeerd kan worden. Door deze gegevens te koppelen aan de informatie uit de sessies, ontstaat er de situatie waarin de anonieme gebruiker ineens een "gezicht" krijgt.

Op deze manier kunnen er uitspraken gedaan worden over globale kenmerken van de gebruikers: "Jongeren tussen de 15 en 25 hebben een voorkeur voor Spanje als vakantiebestemming", maar ook over individuele gevallen: "Meneer A uit B heeft een duidelijke voorkeur voor wintersportvakanties in Oostenrijk".
Hier kan dan op ingespeeld worden door bijvoorbeeld in de zomer, wanneer veel jongeren vakantie hebben, meer aanbiedingen voor Spanje te plaatsen, maar ook door op het scherm van die individuele gebruiker een advertentie voor skivakanties in Oostenrijk te laten verschijnen of juist van een ander wintersportland.

2.1.5 Thomas Cook

Neckermann Reizen is onderdeel van Thomas Cook, waaronder ook Vrij Uit valt. Nu is het zo dat Neckermann en Vrij Uit gebruik maken van dezelfde database. Deze situatie maakt het mogelijk om de informatie vanuit de twee websites te combineren in één Data Warehouse.
Momenteel wordt er nog niet zoveel met die data gedaan, behalve mailingen. Hierbij worden uit de database de aanwezige e-mailadressen geëxtraheerd. Deze verzameling adressen wordt vervolgens doorgespeeld aan e-Campaign. Dat is het outbound mailing systeem van Thomas Cook (voor uitgaande e-mail dus).
Zo krijgen alle klanten een e-mail met de laatste nieuwtjes en aanbiedingen. Via een hyperlink in de e-mail kunnen ze naar de betreffende webpagina gaan. Deze pagehit kan dan weer geregistreerd worden als afkomstig van die betreffende e-mail. Op deze manier kan het rendement van de mailing worden nagegaan.
Dit mailing-proces staat schematisch weergegeven in **Figuur 1**. Hierin valt duidelijk te zien dat er sprake is van een lus.



Figuur 1: Schematische weergave mailing lus

Dit soort "lussen" zijn bijzonder nuttig en daarom ook een belangrijk onderdeel van deze afstudeeropdracht. Het is namelijk interessant om te kijken of, en zo ja hoe, er meer van dit soort lussen gecreëerd kunnen worden. Denk hierbij aan SMS en i-mode, maar ook aan lussen binnen de website zelf. In het laatste geval kan er gedacht worden aan personalisering van de website op basis van het surfgedrag van de gebruiker.
Als zo'n personalisering direct (on-the-fly) wordt toegepast, dan is er sprake van zogenaamde adaptiviteit. Hierbij wordt de inhoud van de website nog tijdens het bezoek van de gebruiker aangepast aan zijn (gebleken) voorkeuren. In zo'n situatie is de "lus" dus veel korter dan bijvoorbeeld de bovenstaande mailing-lus.

2.1.6 Reclame & Advertenties

De website van Neckermann Reizen heeft drie manieren om reclame en advertenties op andere websites te plaatsen. Deze zijn:

- Banners
- Feeds
- Co-branded sites

Deze drie methodes zullen nu kort besproken worden.

Banners

Allereerst is er de overbekende banner. Een banner is een soort reclame-poster die op andere websites geplaatst wordt. Als de gebruiker op zo'n banner klikt, dan wordt deze automatisch naar de bijbehorende website gebracht. Deze dienst wordt vaak door externe bedrijven aangeboden. Zo'n "bannerboer" plaatst dan, tegen betaling, de banners op de nodige webpagina's. Bovendien houdt deze bannerboer ook bij hoe vaak en op welke banners er geklikt wordt en speelt deze informatie door aan zijn klant. Op deze manier kan de effectiviteit en het rendement van de banners worden bepaald.

Feeds

Een andere mogelijkheid is het gebruik van zogenaamde "feeds". Een feed lijkt veel op een banner, maar is subtiel anders. Is een banner een soort reclame-boodschap, die door externe bedrijven op webpagina's worden geplaatst, een feed is meer een advertentie. Er wordt in dit geval een afspraak gemaakt met een andere website om een deel van diens pagina te reserveren voor een stukje inhoud van de eigen website. In plaats van alleen een afbeelding te tonen die naar de website linkt, wordt er een gedeelte van de website zelf aan de feed-website toegevoegd. Een voorbeeld hiervan is te vinden op www.reiskrant.nl. Het voordeel van deze methode is dat men niet afhankelijk is van externe bannerboeren, maar het vereist wel dat men zelf bijhoudt hoe vaak er via deze feeds de website wordt bezocht..

Co-Branded Sites

De derde optie is het concept van "co-branded sites". In dit geval wordt praktisch de gehele inhoud van de website ondergebracht bij een andere website. Vaak zijn dit soort sites te herkennen aan een afwijkende header (bovenste deel van de website). Hier zijn wel bepaalde voorwaarden aan verbonden. In het geval van Neckermann Reizen is het zo dat de externe website de Neckermann Reizen aanbiedt, maar dat de boeking alleen bij die betreffende website gedaan kan worden. Deze website is dan eigenlijk een soort reisbureau die Neckermann reizen aanbiedt. Een voorbeeld hiervan is te vinden op www.happytravel.neckermann-reizen.nl. Ook bij deze methode heeft men niet te maken met een derde partij, maar is men ook op zichzelf aangewezen voor het meten van de pagehits.

2.2 Renault

Ook bij het Renault-project houden ze zich bezig met soortgelijke elementen van informatievergaring. De makers van de website hebben een aantal interessante ideeën op dit gebied, die hieronder aan bod zullen komen.

2.2.1 Bezoekersstatistieken

Het bijhouden van bezoekersstatistieken is één van de meest voorkomende manieren om informatie te krijgen over het gebruik van een website. In principe zijn er drie methodes om dat te bewerkstelligen:

- Het analyseren van de logfiles van de webserver
- Het plaatsen van webcounters op de website
- Zelf registreren en de gegevens in een database opslaan

In de nu volgende secties van dit hoofdstuk zullen deze drie methoden behandeld worden.

Logfile analyse

De hosting server van de website houdt alle page requests (verzoeken van de browser om een pagina te tonen) bij in een log-bestand. Dit bestand bevat o.a. gegevens als:

- IP-adres van de gebruiker
- Tijdstip van de page request
- Naam van de opgevraagde pagina en/of bestand
- Een code die het resultaat van de aanvraag teruggeeft
- Type browser en besturingssysteem

Uit deze gegevens valt precies af te leiden hoe vaak, hoe laat en door wie een bepaalde pagina is bekeken. In dit verband moet "door wie" met een grote korrel zout genomen worden. Het enige wat men uit het IP-adres kan opmaken, is dat een site door eenzelfde gebruiker bekeken wordt, niet wie de gebruiker daadwerkelijk is (hooguit waar hij zich op de wereld bevindt).
De code waarover in puntje vier gesproken wordt, geeft aan of een page request succesvol was. Een succesvolle page request wordt aangeduid met code 200. De bekendste (en meest beruchte) code is 404 (File Not Found), welke aangeeft dat een opgevraagde webpagina niet bestaat of niet toegankelijk is. Indien er dit soort 404-codes in de logfile staan, is de kans groot dat de website zogenaamde "dead links" bevat.
Het moge duidelijk zijn dat dit soort data nuttige informatie verschaffen over de populariteit en kwaliteit van bepaalde onderdelen van een website.
Er is veel software beschikbaar die dit soort logfiles kunnen analyseren en omzetten in bruikbare statistieken. Eén daarvan is WebTrends (www.webtrends.com).

Webcounters

Webcounters zijn niets meer dan kleine stukjes code op een webpagina die een (verborgen) icoontje op de website plaatsen. Als de website door de browser wordt opgevraagd, wordt dat icoontje opgehaald van een (vaak externe) server. Deze server krijgt, via deze request, informatie over het bezoek aan de webpagina. Deze informatie kan dan vervolgens weer gebruikt worden om statistieken mee samen te stellen. Ook hiervoor zijn de nodige diensten beschikbaar, waaronder Stats4All (www.stats4all.nl). Stats4All is een eigen product van Centric-et dat werkt volgens bovenstaand principe. Een groot voordeel van deze methode is, dat ook zogenaamde Back-navigaties (gebruik van de Back-knop om terug te keren naar de vorige pagina) gedetecteerd worden. Indien de website zelf pagehits opslaat, worden deze events meestal niet opgemerkt, omdat de benodigde data uit de cache van de browser wordt opgehaald. De techniek van de webcounters maakt het wel mogelijk om dit soort events te detecteren.

Zelf registreren

Als de website gebruikt maakt van een achterliggende database, zoals veel moderne sites tegenwoordig doen, dan kan deze database natuurlijk ook gebruikt worden om bezoekers te registreren. Een groot voordeel van deze laatste methode is dat men zelf kan bepalen wat en hoe er geregistreerd wordt en hoe deze informatie vervolgens gebruikt wordt. Bovendien is het bij deze methode mogelijk om on-the-fly data te analyseren. Bij o.a. de logfile-methode kan dat niet, omdat hierbij de data pas achteraf geanalyseerd kunnen worden.
De methode van zelf registreren heeft sowieso de meeste potentie. De logfile-methode heeft namelijk teveel beperkingen. In principe bevatten ze genoeg informatie om te bepalen hoe populair een webpagina is, maar meer gedetailleerde gegevens over de gebruiker zijn hiermee niet te achterhalen.
Ook de methode van de webcounters kent zijn beperkingen. De gebruikte code die aan de website wordt toegevoegd, kan al meer gegevens verzamelen dan de logfile-methode, zoals gebruikte search-engines e.d., maar ook hierbij kan geen gedetailleerde informatie over de gebruiker worden vastgelegd.
Alleen met de zelf-registreren-methode kan praktisch alles van de gebruiker bijgehouden worden. Bovendien is men bij deze methode niet afhankelijk van externe partijen als Webtrends en Stats4All.

2.2.2 Pagehits

Het is in de praktijk vaak niet nuttig om alle afzonderlijke pagehits op te slaan. Er kan beter gekeken worden naar het totaal aantal bezoekers van een bepaalde pagina en dat aantal opslaan. Toch kan het voor individuele gevallen wel handig zijn om afzonderlijke pagehits te bewaren, zodat het surfgedrag van de gebruiker (o.a. surfpad) bepaald kan worden.
Echter ook in die individuele gevallen is het soms niet nodig om iédere pagehit te bewaren. Een beter idee is dan om de verschillende pagina's, of beter de verschillende

'versies' van de pagina (i.g.v. dynamische pagina's), te groeperen en een soort van hiërarchie te creëren. In het voorbeeld van Renault kan er dan gedacht worden aan hoofdgroepen als modellen, dealers, enz. Binnen de hoofdgroep modellen zijn er dan weer de subgroepen voor de verschillende modellen. Binnen die subgroepen zijn er dan bijvoorbeeld weer pagina's als gallery (om foto's te bekijken van het betreffende model). Op deze manier wordt er dus een hiërarchie gecreëerd, die zich uitstekend leent voor analytische doeleinden. Het gebruik van de website wordt zo veel overzichtelijker. Wil men weten hoe vaak de modellen sectie van de website geraadpleegd wordt, dan kijkt men naar de modellen groep. Is men meer geïnteresseerd in de populariteit van een specifiek model dan raadpleegt men de bijbehorende groep.

2.2.3 Persoonlijke gegevens

De Renault website kent niet een soortgelijk concept als het paspoort van de Neckermann website. Desalniettemin kan ook hier de nodige informatie over de gebruikers worden opgeslagen. Als een gebruiker bijvoorbeeld een brochure aanvraagt, dan is hij genoodzaakt zijn adresgegevens vrij te geven voor verzending of als hij een e-mail verstuurd is zijn e-mailadres ook gelijk bekend. Indien deze informatie in de database wordt opgeslagen, dan kan deze gekoppeld worden aan de bezoekersgegevens. Het surfgedrag wordt immers bijgehouden. Door nu dit surfgedrag te koppelen aan de persoonsgegevens, kan er het nodige gezegd worden over deze gebruiker, bijvoorbeeld dat zijn voorkeur uitgaat naar een bepaald model. Als er dan vervolgens een actie is met dat model, dan kan de gebruiker daarvan op de hoogte worden gebracht, bijvoorbeeld door een mailing.
Opgemerkt dient wel te worden dat Renault op dit moment (nog) niet zo te werk gaat. Verstrekte adresgegevens worden direct gebruikt voor het versturen van de brochures. De gegevens worden niet opgeslagen in de database, maar de mogelijkheid is er wel.
In zo'n geval is het echter wel wettelijk verplicht om de gebruiker te wijzen op het feit dat de adresgegevens worden opgeslagen.

2.3 Overige aspecten

Naast de technieken en methodes die in de besproken projecten gebruikt (zouden kunnen) worden, zijn er nog een aantal aspecten die voor dit soort projecten in het algemeen interessant zijn. In dit hoofdstuk zullen er een aantal behandeld worden.

2.3.1 Surfgedrag buiten de website

Tot nu toe is de aandacht vooral gericht op de handelingen binnen de website zelf. Het is echter ook interessant om te kijken wat er buiten de eigenlijke website gebeurt. Een belangrijke vraag is namelijk hoe de gebruiker überhaupt op de website terecht is gekomen. Er zijn namelijk verschillende manieren waardoor de gebruiker op een site terecht kan komen:

- Via advertenties op andere websites, vaak in de vorm van banners
- Via hyperlinks in e-mails naar de website
- Via de zoekresultaten van search engines als Google
- Door direct de URL in de browser in te typen

Deze gegevens zijn met name voor marketing doeleinden interessant: "Hoe effectief zijn de geplaatste banners geweest?" Het is dus nuttig om ook deze informatie op te slaan in de database.

2.3.2 Andere digitale media

Er is vooral gesproken over de informatie afkomstig van de websites zelf. Het is echter ook interessant om te kijken naar andere digitale media. De meest voor de hand liggende is e-mail. Via e-mail kunnen de gebruikers op de hoogte gehouden worden van de laatste ontwikkelingen. Bovendien is e-mail dé manier om contact op te nemen met de eigenaar van de website (het bedrijf, danwel de instantie).
Voor datzelfde doel kan ook SMS gebruikt worden. Het is natuurlijk minder uitgebreid dan e-mail, maar heeft als grote voordeel dat het bijna iedereen, bijna overal kan bereiken door zijn mobiele karakter.
Met de opkomst van diensten als i-mode is het ook interessant om deze nieuwe techniek in ogenschouw te nemen. I-mode combineert immers de kracht van internet en e-mail met de mobiliteit van SMS. I-mode maakt de interactie tussen klant en bedrijf/instelling ook op het mobiele vlak mogelijk, waar deze, tot voor kort, beperkt bleef tot het internet.

2.3.3 Inloggen

Het is inmiddels wel duidelijk dat de herkenning van een gebruiker allereerst via een cookie loopt. Indien de gebruiker geen cookie heeft, dan wordt hij beschouwd als een nieuwe gebruiker en kan eventueel al vergaarde informatie van deze gebruiker niet meer gebruikt worden, althans niet voor de gebruiker zelf.
De enige manier om dit te voorkomen, is de gebruiker te laten inloggen. Op die manier kan aan de hand van zijn gebruikersnaam en wachtwoord de gebruiker geïdentificeerd worden. Hoewel het een vrij effectieve manier is om gebruikers te herkennen en te identificeren, wordt deze methode over het algemeen minder door de gebruiker gewaardeerd. Veel gebruikers hechten namelijk veel waarde aan hun anonimiteit op het internet en zo'n verplichting tot inloggen kan sommige gebruikers afschrikken.
Het is dan ook beter om de noodzaak tot inloggen te beperken tot echt belangrijke zaken, zoals het boeken van een reis of het plaatsen van een bestelling.
Aan de andere kant kan het voor gebruikers ook voordelen hebben om in te loggen. Zo kan er namelijk beter worden ingespeeld op de wensen van een specifieke gebruiker. Dit kan voor de gebruiker zelf ook voordelen hebben, zodat de drempel om in te loggen wellicht lager komt te liggen. Het is dus zaak om de gebruikers te overtuigen van de voordelen die inloggen met zich meebrengt.

3 Informatieschema

Zoals al in de vorige hoofdstukken is aangegeven, zijn er verschillende stromen van informatie aan te wijzen in het proces van vergaren, verwerken en toepassen. Een 'standaard' web-based applicatie bestaat meestal uit een FrontEnd en een BackEnd. Het FrontEnd levert de verbindende schakel tussen de gebruiker en het systeem. Het is de kant van het systeem die de gebruiker te zien krijgt. In het BackEnd bevinden zich de databases en vindt de eigenlijke procesverwerking plaats.
Om de besproken functionaliteit aan zo'n standaard systeem toe te voegen zijn er een aantal uitbreidingen nodig:

- Allereerst moet er de mogelijkheid zijn tot contactregistratie. Dit maakt het mogelijk om klantcontacten te detecteren en op te slaan. Dit is essentieel om meer te weten te komen over de gebruikers.

- Daarnaast is het nodig om de eerder besproken paspoortfunctionaliteit te implementeren. Hiermee is het mogelijk om gebruikers zelf informatie en voorkeuren te laten aanbieden.

- Om de vergaarde informatie te verwerken is er behoefte aan een degelijk data warehouse waar alle informatie kan worden opgeslagen. Vanuit het data warehouse kunnen deze gegevens o.a. worden gebruikt voor het samenstellen van statistieken.

- Tot slot is er een speciaal component nodig dat de personalisatie verzorgt. Hiermee kan de informatie uit contactregistratie en data warehouse worden gebruikt om de website (of ander medium) aan te passen aan de gebruiker.

Een systeem dat met deze functionaliteit is uitgebreid, zal voldoen aan het schema in **Bijlage 1**. In deze bijlage wordt tevens uitvoeriger ingegaan op de verschillende componenten en informatiestromen.

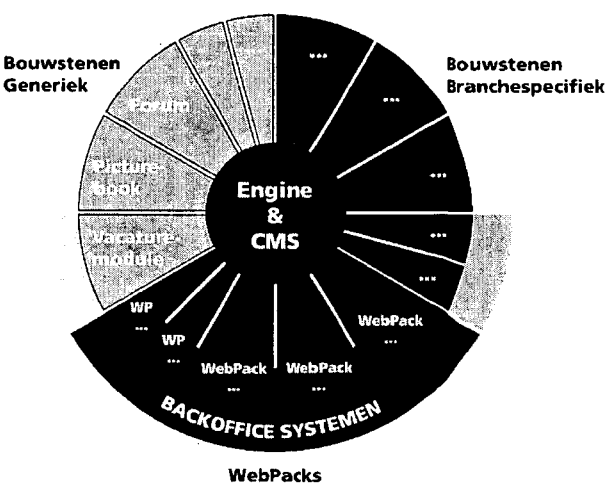
4 Web4all

Toen er eenmaal meer duidelijkheid bestond over het gehele proces van informatie vergaren, verwerken en toepassen, werd het tijd om de resultaten in de praktijk te brengen. Zoals gezegd is dat gedaan binnen het project Web4all. In dit hoofdstuk zal er een beschrijving worden gegeven van Web4all en de achterliggende architectuur. In de hierop volgende hoofdstukken zal verslag worden gedaan van het gehele proces van ontwerpen en implementeren.

4.1 Wat is Web4all?

"Web4all is de totaaloplossing voor de ontwikkeling en exploitatie van uw webomgeving. Het content management systeem biedt alle tools voor het ordenen, actualiseren en publiceren van uw informatie op uw internetsite en intra- of extranet."

Zoals bovenstaande quote al aangeeft, is Web4all een product voor de ontwikkeling en exploitatie van een webomgeving. Het bijbehorende Content Management Systeem (CMS) maakt het mogelijk om inter- en intranetsites te creëren.
Hierbij kan gebruik gemaakt worden van een verzameling bouwstenen die het mogelijk maken om op relatief eenvoudige wijze extra functionaliteit toe te voegen, zoals een forum, een picturebook, enz.
Onderstaande afbeelding geeft de opbouw van Web4*all* schematisch weer.

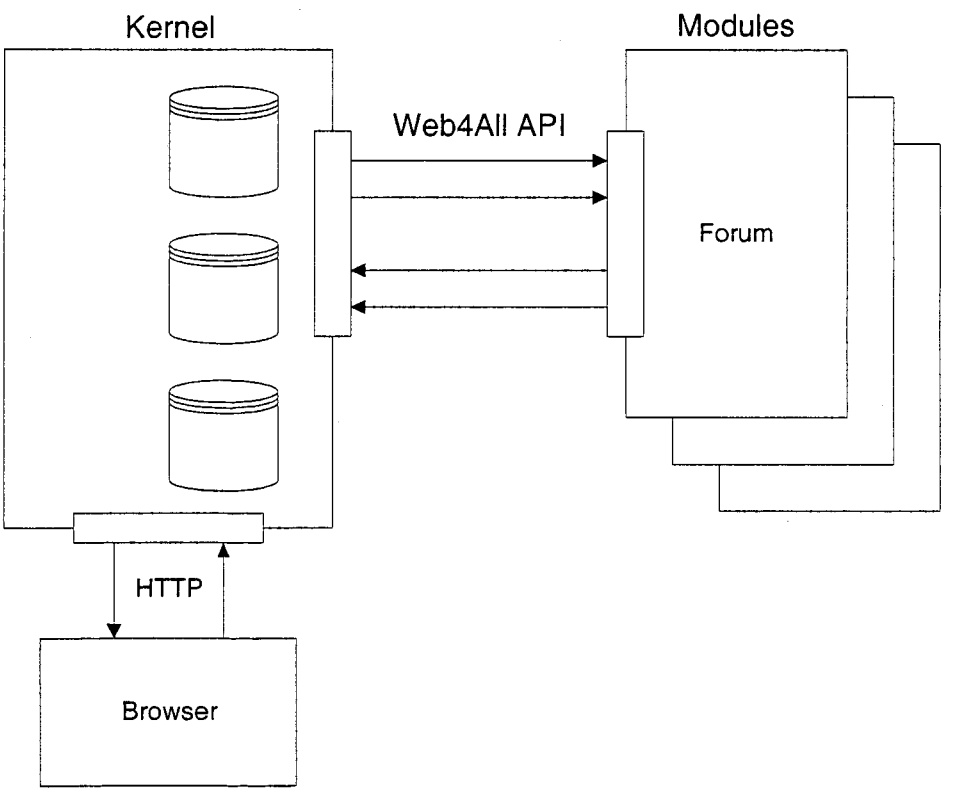


Figuur 2: Web4all

Het gebruik van deze bouwstenen kan de nodige informatie opleveren over de gebruikers. Deze informatie kan vervolgens nuttig gebruikt worden voor o.a. het personaliseren van een site. Er werd echter nog bijna niks met die informatie gedaan, waardoor het een interessant project was voor mijn afstudeerwerk.

4.2 Globale Architectuur

Standaard bestaat een Web4all implementatie uit een Intranet site, waarop werkgroepen aangemaakt kunnen worden. Daarnaast is het mogelijk om in Web4all één of meerdere targets te configureren, die vanuit het Web4all intranet beheerd worden. Onder een target wordt over het algemeen een publieke website verstaan. Denk hierbij aan een bedrijf met een Web4all intranet en een publieke website, met daarop content uit het CMS van Web4all.
Het product Web4all is modulair opgezet. In de basis bestaat Web4all slechts uit de Kernel. De Kernel is als het ware de 'kern' van het product. Alle communicatie tussen de browser en het systeem verloopt in de eerste plaats via de Kernel. Hieromheen worden Web4all modules gebouwd, die d.m.v. de Web4all API[1] communiceren met de Kernel. Hieruit blijkt dat zo'n module in ieder geval de Web4all API dient te ondersteunen.
Het doel van de Kernel en de Web4all API is om modules zodanig met elkaar samen te laten werken, dat alle modules samen zich als één geheel aan de gebruiker presenteren. **Figuur 3** laat schematisch zien hoe dit precies in zijn werk gaat.



Figuur 3: Globale architectuur Web4all

[1] Application Programming Interface

De kernel ontvangt, via de HTTP-interface, de requests van de browser en communiceert aan de hand daarvan met de verschillende modules. Deze communicatie gebeurt via de eerder genoemde Web4all API.
Deze API levert de functies en methodes die de kernel nodig heeft om de verschillende modules, de benodigde bouwstenen voor de pagina te laten leveren. Dit werkt als volgt:

1. De browser doet een page request bij de kernel
2. De kernel bepaalt vervolgens welke modules er nodig zijn om de pagina te kunnen opbouwen en roept de betreffende modules aan
3. Deze modules ontvangen deze opdracht van de kernel en geven op hun beurt de benodigde bouwstenen terug. Dit zal meestal HTML code zijn.
4. Met behulp van deze bouwstenen kan de kernel de gevraagde pagina samenstellen om deze vervolgens terug te geven aan de browser.

Eén van de belangrijkste modules is het al eerder genoemde CMS. Het CMS beheert namelijk alle content binnen Web4all. Zoals in **Figuur 2** te zien valt, is de combinatie Kernel (aangeduid als Engine) en CMS het kloppend hart van het systeem.

4.3 e-Business

Zoals al in de inleiding is aangegeven, is het de bedoeling om Web4all uit te breiden met e-business functionaliteit. De benaming e-business is gekozen om onderscheid te kunnen maken met e-commerce. Zoals de benaming al aangeeft is e-commerce met name gericht op het commerciële aspect. Aangezien Web4all niet alleen voor commerciële bedrijven bedoeld is, maar ook voor gemeentes en andere niet-commerciële instellingen, zou de term e-commerce niet ladingdekkend zijn. Omdat in dit geval wel sprake is van een 'zakelijke' situatie, is er gekozen voor de term e-business.
In de nu volgende hoofdstukken zal uitgebreid worden beschreven wat er, met betrekking tot Web4all, precies onder e-business functionaliteit wordt verstaan en hoe dit tot stand dient te komen.

5 Requirements

Om meer duidelijkheid te krijgen over wat e-business functionaliteit in dit verband inhoudt, wordt er in dit hoofdstuk een lijst van eisen (requirements) gegeven waaraan voldaan dient te worden. Het volgende hoofdstuk zal een concrete invulling aan deze requirements geven.

5.1 Functionele requirements

- Het surfgedrag van de bezoeker(s) moet kunnen worden bijgehouden. Dit houdt in dat er de beschikking moet zijn over de volgende functionaliteit:
    - Het moet mogelijk zijn om bij te houden hoe vaak en door welke bezoeker op een bepaalde link (knop, plaatje, tekst) wordt geklikt.
    - Het moet mogelijk zijn om bij te houden hoe vaak en door welke bezoeker bepaalde content wordt bekeken/bezocht.
    - Het moet mogelijk zijn om tevens bepaalde context-parameters op te slaan. Denk hierbij aan datum, tijd, enz.
- Er moet de beschikking zijn over een basisset van gegevens die altijd worden geregistreerd.
- De website (of intranet) moet kunnen worden gepersonaliseerd aan de bezoeker. Dit vereist de volgende functionaliteit:
    - Het moet mogelijk zijn om regels aan links te kunnen koppelen. De uitkomst van de evaluatie van zo'n regel bepaalt dan waar de link naartoe wijst.
    - Het moet mogelijk zijn om regels aan content op een webpagina te kunnen koppelen. De uitkomst van de evaluatie van zo'n regel bepaalt vervolgens welke content er op een specifieke locatie wordt getoond.
    - Het moet mogelijk zijn om informatie uit het bezoekersprofiel direct op een webpagina te kunnen plaatsen. Voorbeeld:
      "Beste <<voornaam van de bezoeker>>, welkom op onze website."
- Regels zijn expressies opgebouwd uit:
    - Profielgegevens
    - Systeemdata als tijd en datum
    - Data verkregen uit registratie van non-profielgegevens. Denk hierbij aan begrippen als 'duizendste bezoeker'.
    - Operatoren: =, >, <, +, -, AND, NOT
    - Constante waarden
- De vergaarde gebruikersinformatie moet kunnen worden geëxporteerd naar de Data Warehouse en/of een ander systeem voor verdere verwerking.
- Extern vergaarde informatie of informatie uit de Data Warehouse moet kunnen worden geïmporteerd om o.a. profielen mee samen te stellen en/of aan te passen.

- Binnen Web4all bestaat er al de mogelijkheid om een 'preview' te genereren van de webpagina, zoals deze er in gepubliceerde vorm uit komt te zien. Met de toevoeging van profielen wordt het noodzakelijk dat er gebruik kan worden gemaakt van profielgebonden previews. Dit betekent dat er de mogelijkheid moet zijn om, bij het genereren van een preview, aan te geven voor welk type profiel dit moet gebeuren.

5.2 Profiel requirements

- Een bezoeker moet een profiel kunnen hebben, waaruit zijn voorkeuren en interesses blijken.
- Een bezoekersprofiel moet kunnen bestaan uit:
    - Paspoortgegevens
    - Gegevens uit on-the-fly analyse van het surfgedrag
    - Gegevens uit externe analyse van het surfgedrag
    - Gegevens uit algemene profielen: bezoekers kunnen vanwege hun voorkeuren en surfgedrag voldoen aan een algemeen profiel. Alle kenmerken van dat algemene profiel worden daardoor automatisch kenmerken van de betreffende bezoeker.
- Een bezoeker heeft maximaal 1 profiel per website.
- Een bezoeker heeft maximaal 1 profiel per werkgroep op het intranet.

5.3 Bezoekers requirements

- Een bezoeker moet een paspoort kunnen aanmaken
- Een bezoeker moet een paspoort kunnen aanpassen
- Een bezoeker moet met behulp van zijn paspoort kunnen inloggen
- Het moet voor een bezoeker mogelijk zijn om op meerdere websites gebruik te maken van hetzelfde paspoort.

5.4 Beperkende requirements

- Zoals gezegd moet er binnen Web4all de mogelijkheid worden geboden om websites en intranetten te voorzien van functionaliteit voor het bedrijven van e-business. In dit geval blijft de functionaliteit dus beperkt tot websites en intranet, maar er moet wel rekening worden gehouden met eventuele uitbreidingen ten behoeve van e-mail en SMS.
- Bij het uitbreiden van de Web4all-functionaliteit moet er gebruik worden gemaakt van de ontwikkelgereedschappen waarmee de rest van Web4all ook is geïmplementeerd. Dit houdt o.a. in dat de gebruikte programmeertaal Visual Basic is.

6 Concept

Op basis van de requirements uit het vorige hoofdstuk, zal er in dit hoofdstuk een uitbreiding van Web4all worden behandeld die e-business mogelijk maakt. Het doel van deze uitbreiding is om beter op de wensen van de klant te kunnen inspelen. Om dit voor elkaar te krijgen zijn er in principe drie stappen nodig:

1) Allereerst wordt er informatie over de klant verzameld en opgeslagen.
2) Vervolgens wordt deze informatie verwerkt in een gebruikersprofiel.
3) Tot slot wordt zo'n profiel gebruikt om de site aan te passen aan de klant.

Het is duidelijk te zien dat er een sterke overeenkomst is met de drie stappen die in het begin van deze scriptie ter sprake kwamen.
In de nu volgende secties zal worden beschreven hoe deze drie stappen concreet kunnen worden uitgevoerd voor wat betreft Web4all.

6.1 Vergaring

Zoals al in het eerste deel van deze scriptie ter sprake kwam, zijn er verschillende manieren om klantinformatie te vergaren. In dit geval zullen de volgende methodes gebruikt gaan worden:

- Paspoorten
- Registrators

Deze begrippen worden hieronder kort toegelicht.

6.1.1 Paspoorten

Het begrip paspoort is al in de sectie over Neckermann ter sprake gekomen. Er zal dan ook in deze sectie niet verder op worden ingegaan. Bovendien zal deze manier van informatievergaring als aanwezig worden beschouwd, omdat de functionaliteit al in het Neckermann project is uitgewerkt.

6.1.2 Registrators

Een registrator is een stukje functionaliteit dat het mogelijk maakt om bepaalde 'events' te detecteren en op te slaan. Mogelijke 'events' zijn:

- Het aanklikken van een hyperlink
- Het getoond krijgen van een pagina

- Het getoond krijgen van een content-object

Met de eerste optie kan nauwkeurig het surfgedrag van de klant worden bijgehouden. Met deze informatie is het ook mogelijk om een surfpad te reconstrueren.
De tweede optie heeft tot taak om bij te houden hoe vaak een bepaalde pagina wordt bekeken. Hiermee kan o.a. worden nagegaan welke pagina's het meest worden bezocht. Hetzelfde geldt voor de derde optie met het verschil dat de focus niet op een gehele pagina ligt, maar op een deel van een pagina. Zo'n 'brok content' wordt een content-object genoemd. De reden dat deze optie is toegevoegd, zal verderop in het document worden gegeven.

6.2 Verwerking

De informatie die vergaard is met behulp van de paspoorten en registrators worden gebruikt om gebruikersprofielen mee samen te stellen. Deze profielen zijn opgebouwd uit zogenaamde attributen. Verderop in dit document zal er uitgebreid worden ingegaan op het begrip 'attribuut'.

6.2.1 DataShare

Het is goed denkbaar dat verschillende modules gebruikmaken van dezelfde gegevens. Zo kan bijvoorbeeld de module voor personeelsgegevens data delen met de picturebook-module (het picturebook bevat ook informatie over de verschillende werknemers). Als er wijzigingen optreden in de personeelsgegevens is het wenselijk dat deze wijzigingen ook worden doorgevoerd in het picturebook.
Voor dit doel is er DataShare ontwikkeld. DataShare zorgt ervoor dat gedeelde gegevens gesynchroniseerd worden en blijven. Het principe van DataShare is dat één module wordt aangewezen als de 'eigenaar' van de gegevens. Deze module wordt aangeduid als de publisher. De modules die deze gegevens delen, worden aangeduid als subscribers. Veranderingen binnen de publisher-module worden automatisch doorgevoerd binnen de subscriber-modules. Andersom worden wijzigingen in een subscriber-module ter goedkeuring aangeboden aan de publisher-module.
Wat de gevolgen zijn van DataShare voor de e-business-functionaliteit zal worden besproken in de sectie Attributen in het volgende hoofdstuk.

6.3 Toepassing

Voor het personaliseren van een website zijn ook verschillende methodes denkbaar. De methodes die binnen Web4all gebruikt gaan worden, zijn:

- Routers
- Tags

Beide methodes worden hieronder toegelicht.

6.3.1 Routers

Een router maakt het mogelijk om onderdelen van een website aan te passen aan de klant. Voor dit doel zijn er 2 type routers gedefinieerd, te weten:

- URL-routers
- Content-routers

In de nu volgende paragrafen zullen beide routers belicht worden. In het volgende hoofdstuk zal er dieper worden ingegaan op de achterliggende functionaliteit en hoe routers uiteindelijk geïmplementeerd zijn.

URL-routers

Normaal gesproken verwijst een hyperlink naar slechts één mogelijke 'bestemming'. Een URL-router maakt het mogelijk om één hyperlink naar verschillende bestemmingen te laten verwijzen. Op basis van het gebruikersprofiel wordt er door het systeem bepaald naar welke bestemming er 'gerouteerd' moet worden. Deze techniek is enigszins te vergelijken met 'Local Guidance' **[3]**. Hierbij worden op basis van de voorkeuren van de gebruiker suggesties gedaan voor verdere navigatie. De URL-router verschilt in dat opzicht van Local Guidance dat het systeem zelf de beslissing neemt en verder geen feedback geeft naar de gebruiker toe.

Content-routers

Een content-router maakt gebruik van de eerder genoemde content-objecten. Een content-object kan in principe van alles zijn, variërend van een stuk tekst tot een afbeelding, van een brok HTML tot (een verwijzing naar) een volledig document. Zo'n content-object kan worden ingevoegd in een webpagina. Een content-router maakt het echter mogelijk om, op basis van o.a. het gebruikersprofiel, een keuze te maken uit een selectie content-objecten. Dit maakt het mogelijk dat gebruiker A een ander content-object te zien krijgt dan gebruiker B. Het principe van Content-routers komt qua functionaliteit overeen met de 'Conditional Text'-techniek **[3]**. De achterliggende techniek van content-routers is echter subtiel anders (zie **Bijlage 2**). Bovendien beperkt de 'Conditional Text'-techniek zich, zoals de naam al aangeeft, tot tekst terwijl een content-object bijvoorbeeld ook een afbeelding kan zijn.
Vanwege deze router-functionaliteit moet een registrator ook de mogelijkheid bieden om (het tonen van) content-objecten te registreren: doordat er verschillende content-objecten in dezelfde pagina getoond kunnen worden, is het dus niet genoeg om alleen (het tonen van) de pagina te registreren.

6.3.2 Tags

Tags maken het mogelijk om attribuut-waarden direct in de content van een pagina op te nemen. Verderop in deze scriptie zal nader worden ingegaan op deze tags. Voorlopig wordt volstaan met een voorbeeld:
Om een klant persoonlijk te kunnen begroeten, kan een tag worden toegevoegd die de waarde van attribuut "achternaam" van de klant toont. Dit zou er ongeveer als volgt uit kunnen zien: "Beste meneer/mevrouw [achternaam], welkom op deze website."

7 Globaal Ontwerp

In dit hoofdstuk zal worden besproken hoe het voorgaande ook daadwerkelijk kan worden gerealiseerd.

7.1 Attributen

Het sleutelbegrip binnen het hele personalisatie-proces is 'attribuut'. Een attribuut is een object dat informatie verschaft over de huidige toestand van het systeem. Dit kan informatie zijn over de gebruiker die op dat moment is ingelogd, maar ook informatie over bijvoorbeeld het aantal bezoekers dat de website die dag heeft bezocht. Om het geheel overzichtelijk te houden, wordt er onderscheid gemaakt tussen de volgende types attributen:

- Paspoort-attributen
- Registrator-attributen
- Systeem-attributen
- Berekende attributen

7.1.1 Paspoort-attributen

Zoals gezegd is een paspoort niet veel meer dan een formulier waarin een klant persoonlijke gegevens en voorkeuren kan invoeren. Deze informatie wordt in de database opgeslagen en tegelijkertijd wordt voor ieder veld in het formulier een bijbehorend attribuut aangemaakt. Deze attributen corresponderen met de paspoortgegevens uit het gebruikersprofiel.

7.1.2 Registrator-attributen

Iedere registrator heeft een corresponderend attribuut. Eigenlijk is de registrator zelf het attribuut. Bij het aanmaken van een registrator wordt de gebruikte naam tevens de naam van het attribuut. Datgene wat de registrator detecteert, wordt dus opgeslagen in het registrator-attribuut. Deze attributen leveren informatie op die gebruikt kunnen worden voor on-the-fly en externe analyse. De resultaten hiervan kunnen gebruikt worden in het gebruikersprofiel.

7.1.3 Systeem-attributen

Dit type attribuut is enigszins afwijkend van de vorige twee attribuuttypen. Deze attributen zijn namelijk 'hard coded'. Dat houdt in dat deze attributen *niet* door de

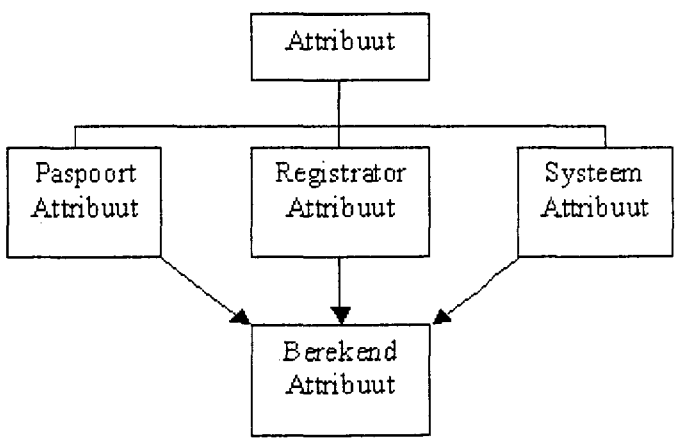
beheerder of de gebruiker kan worden aangepast. Deze attributen zijn globaal beschikbaar en kunnen ook als zodanig ingezet worden. Voorbeelden van systeem-attributen zijn:

- Datum/Tijd: Een attribuut dat de huidige datum/tijd weergeeft
- Random: Een attribuut dat een willekeurige waarde tussen 0 en 9 genereert
- Browsertype: Een attribuut dat aangeeft welk type browser de huidige bezoeker gebruikt

7.1.4 Berekende attributen

Berekende attributen zijn attributen die qua waarde afhankelijk zijn van andere attributen. Berekende attributen worden expliciet door de beheerder zelf gedefinieerd. Voor het definiëren van een berekend attribuut wordt gebruik gemaakt van een expressie. Zo'n expressie bepaalt de waarde van het attribuut op basis van de waarde van andere attributen. Op het begrip expressie zal verderop in dit verslag nader worden ingegaan. Deze attributen leveren een bijdrage aan de on-the-fly analyse van het gebruikersgedrag.

Figuur 4 geeft het geheel schematisch weer.



Figuur 4: Attributen

7.1.5 DataShare

Kijken we naar het paspoort, dan zien we dat met name de NAW-gegevens typisch zaken zijn die ook in andere modules zullen voorkomen, zoals bijvoorbeeld de picturebook-module. Hierdoor is het wenselijk dat de betreffende paspoort-attributen de mogelijkheid tot DataShare ondersteunen. Dit houdt in dat de attributen-engine (zie het hoofdstuk Implementatie) de functionaliteit moet bezitten om de DataShare API aan te spreken. Het moet dus mogelijk zijn dat de module die de attributen beheert, als publisher of als subscriber van deze data kan optreden.

7.2 Registrators

Zoals gezegd kan een registrator gekoppeld zijn aan

- links
- pagina's
- content-objecten

De functionaliteit is altijd aanwezig, dus de beheerder hoeft een registrator niet expliciet aan een pagina of link te verbinden. Elk van deze componenten heeft een eigenschappen-venster. Dit venster bevat een knop waarmee de registrator-editor kan worden geopend. Via deze editor kunnen er vervolgens één of meerdere registrators worden gedefinieerd, die automatisch aan de betreffende component zijn verbonden. Uiteraard kunnen registrators ook worden aangepast of verwijderd.

Registrators zijn onder te verdelen in twee groepen:

- Tellers
- Detectors

Tellers houden precies bij *hoe vaak* een bepaald 'event' plaatsvindt. Op deze wijze kan worden nagegaan hoe vaak een bepaalde pagina bezocht wordt of hoe vaak een bepaalde link wordt aangeklikt.
Een detector houdt bij *of* een bepaald 'event' plaatsvindt. In tegenstelling tot tellers doen detectors dus niet bijhouden *hoe vaak* een 'event' plaatsvindt. Met een detector kan bijvoorbeeld worden nagegaan of een gebruiker bepaalde content onder ogen heeft gekregen.
Iedere registrator heeft ook een scope. De scope geeft aan op welk niveau er geregistreerd wordt. De verschillende scopes zijn:

- Session
- User
- Workgroup
- Market

Een Session is eigenlijk hetzelfde als een bezoek aan de site (intra- danwel internet). Zodra de eerste pagina wordt opgevraagd begint de sessie. Deze sessie duurt voort totdat de gebruiker uitlogt of zodra de gebruiker gedurende een bepaalde tijd (meestal 10 a 15 minuten) niet meer op de site aanwezig/actief is geweest.

Een User is een gebruiker van de site. Het belangrijkste kenmerk van een gebruiker is, dat deze is ingelogd. Een gebruiker is dus niet een anonieme bezoeker die op de site rondkijkt, maar een bestaande 'klant' die zich, via een inlogprocedure, aan de site bekendmaakt.

Gebruikers kunnen tot een Workgroup behoren. Een werkgroep is een verzameling gebruikers met gedeelde interesses. Een gebruiker kan in een of meerdere werkgroepen worden ingedeeld op basis van deze gemeenschappelijke interesses. Een gebruiker *hoeft* overigens niet perse deel uit te maken van een werkgroep.

Een Market is een verzameling van sites die min of meer met elkaar verwant zijn. Concreet komt het erop neer dat een gebruiker op alle sites binnen een markt gebruik kan maken van hetzelfde profiel en dus ook op alle sites kan inloggen met dezelfde gebruikersnaam en wachtwoord. Het intranet wordt om die reden ook beschouwd als één markt.

Tot slot is er de mogelijkheid om een registrator (tijdelijk) uit te schakelen. De registrator wordt hierbij niet verwijderd, maar gedeactiveerd. Het bijbehorende registrator-attribuut wordt evenmin verwijderd. Het attribuut blijft gewoon beschikbaar, maar de waarde blijft (uiteraard) onveranderd.

7.3 Routers

De globale structuur van een router is als volgt:

- Een router is eigenlijk een lijst van bestemmingen (hyperlinks, danwel content-objecten).
- Aan elk van deze bestemmingen is een expressie verbonden.
- Wanneer de router wordt aangeroepen wordt de lijst van bestemmingen van boven naar beneden afgewerkt.
- De eerste bestemming, waarvan de expressie TRUE oplevert, wordt gekozen. Dit betekent dat bestemmingen, waarvan de expressie ook TRUE zou opleveren, maar lager in de lijst staan, worden genegeerd.

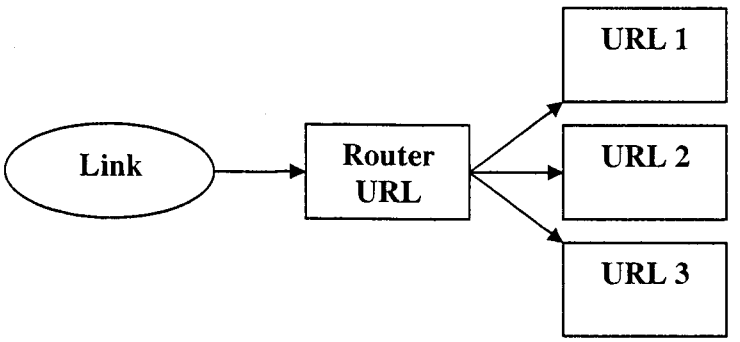
In het geval van URL-routers wordt er een keuze gemaakt uit een lijst van bestemmingen (URL's), waarnaar de link kan verwijzen.
De content-router maakt een keuze uit content-objecten die in de pagina kunnen worden ingevoegd.
De ontwerpbeslissingen die hierbij genomen zijn, zijn opgenomen in **Bijlage 2**.
Hieronder worden alleen de uiteindelijke resultaten besproken.

7.3.1 URL-router

Het eerste type router verzorgt de zogenaamde dynamische links. Een dynamische link is een link die naar meerdere 'destinations' kan wijzen. Dit wordt als volgt gerealiseerd:

De dynamische link bevat een speciale 'Router-URL'. Deze URL verwijst naar een soort 'dummy' pagina, alwaar een keuze wordt gemaakt uit de mogelijke bestemmings-URL's.

Vervolgens wordt er vanuit die pagina automatisch doorgelinkt naar de uiteindelijke pagina. Schematisch ziet dat er als volgt uit:



7.3.2 Content-router

Voor wat betreft het tweede type router, hebben we te maken met 'rule based content'. Dit wordt als volgt gerealiseerd:

De webpagina is in beginsel leeg en de uitkomst van de regels bepaalt welke content-objecten er in de pagina worden geplaatst. Hierbij wordt het *eerste* content-object dat aan een regel voldoet getoond. In pseudo-code:

```
IF <boolean expressie 1>
      THEN <plaats content-object 1>
ELSE
IF <boolean expressie 2>
      THEN <plaats content-object 2>
ELSE
IF …
…
ELSE <plaats default content-object>
```

Voor dit doel kan uiteraard ook de CASE-statement gebruikt worden.

7.4 Tags

Een tag is niets meer dan een verwijzing naar een attribuut. In 'design-mode' is deze tag te herkennen aan de attribuutnaam tussen rechte haken ("[" en "]"). Op de site zelf zal de tag vervangen worden door de waarde van het betreffende attribuut. Aangezien het mogelijk is dat een attribuut geen waarde heeft, moet de beheerder ook altijd een default waarde opgeven voor de tag (dit kan een lege string zijn).

7.5 Expressies

Zoals in de requirements al min of meer beschreven staat is een expressie binnen Web4all opgebouwd uit:

- Attributen
- Operatoren
- Constanten

Praktisch ieder attribuut dat besproken is, kan in een expressie gebruikt worden. Ieder attribuut (ook een berekend attribuut) heeft één van de volgende datatypen:

- Integer
- Float
- String
- Boolean
- DateTime
- Currency

Daarnaast is er de beschikking over de volgende operatoren:

- =
- >
- <
- +
- -
- AND
- NOT

Constanten zijn 'vaste' waarden die door de beheerder aan de expressie worden toegevoegd.

7.5.1 Attribuutnotatie

Er is voor gekozen om de notatie van attributen uniek te laten zijn op market-niveau.
Om dit voor elkaar te krijgen, moet er een notatie worden bedacht die het mogelijk maakt om ieder attribuut een unieke naam te geven.
Zoals gezegd komen attributen op verschillende manieren tot stand. De verschillende soorten attributen zijn:

- Paspoort-attributen
- Registrator-attributen
- Systeem-attributen
- Berekende attributen

Het is praktisch als aan de naam van het attribuut ook tevens deze 'oorsprong' valt af te leiden. Daarom is ervoor gekozen om attributen van een prefix te voorzien waaruit blijkt om wat voor type attribuut het gaat. De mogelijke prefixen zijn:

- Passport
- Registrator
- System
- Calculated

Deze prefixen worden alleen gebruikt binnen expressies. M.a.w. de beheerder zal het paspoort-attribuut **LastName** definiëren. Wordt dit attribuut in een expressie gebruikt, dan zal het aangeduid worden als **Passport . LastName**.

Attributen hebben, net als registrators, een scope. Welke scope een attribuut kan hebben hangt deels af van het soort attribuut:

- Paspoort-attributen hebben altijd de scope User. Paspoort-attributen zijn immers verbonden aan een gebruiker.
- Registrator-attributen kunnen in principe iedere mogelijke scope hebben variërend van Session tot Market.
- Systeem-attributen kunnen ook variërende scopes hebben. Systeemtijd heeft bijvoorbeeld scope Global, terwijl de scope van bijvoorbeeld Browserinfo juist per sessie kan verschillen.
- Berekende attributen zijn eigenlijk afhankelijk van de scope van de attributen die in de expressie voorkomen. Het attribuut met de 'laagste' scope in de expressie, bepaalt tevens de scope van het bijbehorende berekende attribuut. Dit betekent dus dat een berekende attribuut iedere scope kan hebben varierend van Session tot Global (bijvoorbeeld in het geval dat er alleen globale systeem-attributen in de expressie voorkomen).

Opgemerkt dient te worden dat registrator-attributen met scope User tot het profiel van de gebruiker behoren.

Uit het voorgaande blijkt dat attributen verschillende scopes kunnen hebben. Omdat het voor een beheerder aannemelijk is dat verschillende componenten verschillende namen dienen te hebben, is het ook diens taak om in de naamgeving rekening te houden met de scope. Een registrator die op user-niveau bijhoudt hoe vaak een pagina over Spanje bekeken wordt, zou in dit geval als volgt kunnen worden aangeduid:

Registrator . UserPageSpain

De beheerder is overigens vrij in de naamgeving van de attributen, dus andere noteringen zijn ook mogelijk. De beheerder kan zelf een notatie bedenken die het onderscheid tussen de verschillende (unieke) attributen maakt.

Hieronder staat een overzicht waarin wordt aangegeven hoe de verhouding is tussen attribuutnaam, -type en –scope.

Type	Benaming	Scope	Voorbeelden
Paspoort	**Passport . ***	**User**	**Passport . LastName** **Passport . LikesSpain**
Registrator	**Registrator . ***	**Market**	**Registrator . MarketPageSpain**
		Workgroup	**Registrator . WorkgroupLinkBooking**
		User	**Registrator . UserAdvertSpain**
		Session	**...**
Systeem	**System . ***	**Global**	**System . GlobalTime**
		Market	**...**
		Workgroup	**...**
		User	**...**
		Session	**System . SessionRandom**
Berekend	**Calculated . ***	**Global**	**...**
		Market	**...**
		Workgroup	**...**
		User	**Calculated . UserLikesSpain**
		Session	**...**

Om attributen duidelijk te kunnen onderscheiden in een expressie is er tot slot nog de eis dat er rechte haken ('[' en ']') om het attribuut komen te staan. Het attribuut komt er dus uiteindelijk als volgt uit te zien: **[Registrator . UserPageSpain]**

7.6 Previews

Binnen Web4all bestaat er al de mogelijkheid om een preview te genereren van de website, zoals deze er in gepubliceerde vorm uit komt te zien. In verband met de mogelijkheid tot dynamische content, is het nodig om ook de preview-functionaliteit uit te breiden. Dit gebeurt door middel van zogenaamde templates. Een template is een verzameling van attributen, waarbij ieder attribuut een vaste waarde heeft. Het template creëert als het ware een context waarbinnen de website getest wordt. Bij de evalutatie van expressies worden dan de attribuutwaarden gebruikt die in het template zijn vastgelegd.

7.7 Templates

De hierboven genoemde templates worden samengesteld in een overzichtsvenster van alle registrators en attributen.

In dit venster kunnen de waarden van alle (registrator-)attributen worden opgevraagd. Door in dit venster een selectie te maken van attributen, kan er een template worden samengesteld. Via een druk op de knop kan deze selectie vervolgens worden bewaard. Het is eventueel ook mogelijk om zelf waarden voor bepaalde attributen in te vullen, bijvoorbeeld als het attribuut geen eenduidige waarde heeft (bijvoorbeeld het systeem-attribuut Date).

8 Grafisch Ontwerp

Dit hoofdstuk beschrijft de grafische user interfaces van de verschillende componenten die aan Web4all zullen worden toegevoegd.

Routers

Het hoofdvenster van een router moet een overzicht tonen van de verschillende 'bestemmingen'. Daarnaast zijn er schermen nodig waarin nieuwe bestemmingen kunnen worden toegevoegd en bestaande kunnen worden aangepast. Vanuit deze schermen moeten vervolgens expressies en content-objecten/links kunnen worden toegevoegd.

Registrators

Allereerst is er een overzichtsvenster nodig waarin alle registrators worden getoond die aan een bepaald component (pagina, link, enz.) zijn verbonden. Vanuit dit hoofdvenster kunnen vervolgens schermen worden opgeroepen die de mogelijkheid bieden om registrators toe te voegen of aan te passen. In deze schermen kunnen o.a. type en scope worden opgegeven.

Berekende Attributen

Ook voor berekende attributen is een overzichtsvenster nodig. In de schermen voor het toevoegen en verwijderen van berekende attributen kunnen deze attributen aan expressies worden gekoppeld.

Tags

Het toevoegen van een tag gebeurt vanuit een venster waarin een specifiek attribuut kan worden geselecteerd.

Previews

Een preview venster bestond al. Om echter gebruik te maken van de eerder genoemde templates zijn er enkele uitbreidingen aan dat venster nodig, waaronder een dropdown die een overzicht van de verschillende templates bevat.

In **Bijlage 3** worden alle componenten in detail besproken.

9 Technisch Ontwerp

Zoals gezegd zijn de twee belangrijkste onderdelen van Web4all de kernel en het CMS. Het CMS beheert alle content van Web4all, terwijl de kernel de basisfunctionaliteit voor zijn rekening neemt.
Daarnaast zijn er de verscheidene modules die extra functionaliteit aan het systeem toevoegen. Deze modules kunnen *niet* direct worden aangesproken door de gebruiker, noch kunnen ze direct met elkaar communiceren. Alle communicatie tussen gebruiker (browser) en systeem en tussen modules onderling verloopt via de kernel. Functionaliteit die voor meerdere modules toegankelijk moet zijn, zal in principe altijd in de kernel aanwezig zijn.
Omdat de besproken componenten waarschijnlijk ook vanuit verschillende modules toegankelijk moeten zijn, is er besloten om deze componenten onderdeel te laten zijn van de kernel. De implementatie van de componenten zal dus een uitbreiding van de kernel-code inhouden. In het hoofdstuk 'Implementatie' zal hier preciezer op worden ingegaan.

Alvorens in detail in te gaan op de nieuwe architectuur, zal er eerst een moment worden stilgestaan bij een onderdeel dat met name voor de ontwikkeling van content-routers zeer belangrijk is gebleken: ContentHosting
In de daarop volgende secties zal beschreven worden hoe de nieuwe architectuur eruit komt te zien. Hierbij worden eerst de verschillende databasetabellen behandeld aan de hand van een ER-diagram. Vervolgens worden de benodigde classes behandeld.

9.1 ContentHosting

Voor de ontwikkeling van content-routers bleek het nodig om gebruik te maken van ContentHosting. In dit hoofdstuk zal worden uitgelegd wat ContentHosting inhoudt en wat de invloed ervan was op de ontwikkeling van content-routers.

Er is al eerder gesproken over content-objecten. Deze content-objecten worden gemaakt door de modules. Deze modules zijn als het ware de aanbieders (providers) van content-objecten. Ieder content-object is een instantiatie van een content-type. Er zijn verschillende content-types, zoals afbeeldingen, tekst, HTML en ook content-routers.

Het principe van contenthosting is dat een module (zeg A) de 'host' kan zijn van een content-object dat door een andere module (zeg B) wordt aangeboden. Dit houdt in dat module A de opdracht kan geven aan module B om een content-object te creëren van een bepaald content-type. De host wordt daardoor de eigenaar van dat content-object.
De functionaliteit hiervoor wordt geleverd door de kernel. In **Figuur 5** wordt het principe schematisch weergegeven.

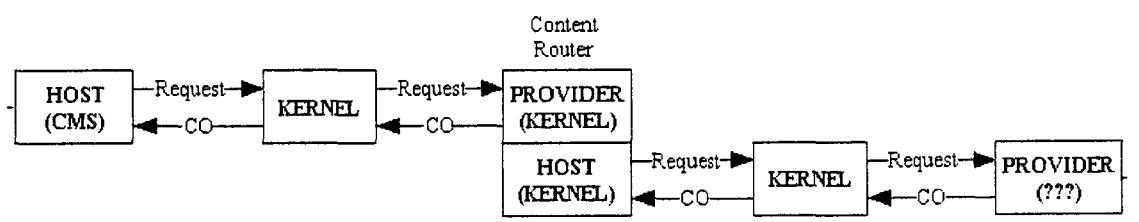
Opgemerkt dient te worden dat de kernel zelf ook een module is. Dit is van belang voor een goed begrip van content-routers.

Figuur 5: ContentHosting

Wat de gevolgen van contenthosting voor content-routers zijn, zal nu stap voor stap worden besproken:

- Een content-router is in de eerste plaats zelf een content-object.
- Aangezien de router-functionaliteit onderdeel is van de kernel, is de kernel dus de provider van het content-type 'content-router'.
- Omdat er sprake is van content is de CMS-module de host van het router-object.
- Het kenmerk van een content-router is echter dat het zelf ook weer een content-object oplevert, namelijk het bestemmings-object.
- Dat bestemmings-object wordt weer door een andere module geleverd en dus is deze module de provider van het bestemmings-object.
- Omdat de content-router deel uitmaakt van de kernel, is de kernel in dit geval de host van het bestemmings-object.
- Er ontstaat daardoor de bijzondere situatie dat de kernel tegelijkertijd provider (van het router-object) en host is (van het bestemmings-object)
- De kernel moet daardoor zowel de functionaliteit bezitten om host te zijn, als de functionaliteit om provider te zijn.

Om het geheel te verduidelijken, wordt in **Figuur 6** het geheel schematisch weergegeven.



Figuur 6: ContentHosting & Content-routers

Nu bezat de kernel al wel de functionaliteit om host te zijn, maar de functionaliteit om provider te zijn, ontbrak nog. Deze functionaliteit is daarom toegevoegd aan de kernel. Meer details hierover zijn te vinden in het volgende hoofdstuk ('Implementatie').

9.1.1 Externe ContentHosting

Behalve deze 'interne' contenthosting is er ook een vorm van 'externe' contenthosting. Deze vorm van contenthosting maakt het mogelijk voor externe (maatwerk)websites om ook gebruik te maken van content-objecten uit Web4all. De Neckermann website maakt daar bijvoorbeeld gebruik van. Nemen we **Figuur 5** als uitgangspunt, dan is de HOST in dit geval de betreffende website.
Globaal gezien verloopt externe contenthosting dus volgens hetzelfde principe als de interne variant. Onderhuids zijn er echter wel de nodige verschillen aan te wijzen.
Op deze technische details zal verder niet worden ingegaan, maar het is wel belangrijk om te beseffen dat ook maatwerk websites gebruik kunnen maken van bepaalde functionaliteit van Web4all. Dit maakt het namelijk in principe ook mogelijk om, met de nodige uitbreidingen, bijvoorbeeld router-functionaliteit te gebruiken in externe websites.

9.2 Database

Voor ieder van de besproken componenten zijn één of meerdere databasetabellen nodig. Deze tabellen maken onderdeel uit van de (reeds bestaande) kernel database.
Een overzicht van de verschillende tabellen is te vinden in **Bijlage 4** in de vorm van een ER-diagram. Verder bevat deze bijlage een toelichting op alle tabellen die in het diagram zijn opgenomen.

9.3 Classes

Deze sectie bevat een overzicht van de classes die nodig zijn om de e-business-functionaliteit mogelijk te maken. De classes zijn gegroepeerd op basis van de besproken componenten. Opgemerkt dient te worden dat dit overzicht de classes behandelt die minimaal nodig zijn voor het beoogde resultaat. Eventuele andere classes zullen tijdens de implementatiefase aan het licht komen en ook dan behandeld worden. Bovendien wordt hier alleen de functionaliteit besproken die de classes moeten leveren. De eigenlijke implementatie wordt in het volgende hoofdstuk behandeld.

9.3.1 Attributen

clsAttributeBuilder

Deze class verzorgt de functionaliteit om nieuwe attributen aan te maken en bestaande attributen te verwijderen.

clsAttribute

Dit is de attributen class. Ieder attribuut is een instance van deze class. De class bestaat uit een aantal properties, een functie om het attribuut te instantiëren en een functie om de waarde van het attribuut te bepalen.

9.3.2 Expressies

clsExpressionBuilder

Deze class verzorgt de functionaliteit om nieuwe expressies aan te maken en bestaande expressies aan te passen of te verwijderen. Daarnaast biedt deze class een functie aan voor het testen van een expressie op (syntactische) correctheid.

clsExpression

Dit is de expressie class. Iedere expressie is een instance van deze class. De class bestaat uit een aantal properties, een functie om de expressie te instantiëren en een functie om de expressie te evalueren en het resultaat terug te geven.

9.3.3 Routers

clsRouterBuilder

Deze class verzorgt de functionaliteit om nieuwe routers aan te maken en bestaande routers te verwijderen. Functionaliteit om een router aan te passen is niet nodig, omdat dit op een lager niveau gebeurt via clsOptionBuilder.

clsRouter

Dit is de router class. Iedere router is een instance van deze class. De class bestaat uit een functie om de router te instantiëren en een functie die de eigenlijke router-functionaliteit implementeert.

clsOptionBuilder

Iedere router bestaat uit één of meerdere opties. Deze class biedt de functionaliteit om nieuwe opties toe te voegen en bestaande opties te verwijderen of aan te passen.

9.3.4 Registrators

clsRegistratorBuilder

Deze class verzorgt de functionaliteit om nieuwe registrators aan te maken en bestaande registrators aan te passen of te verwijderen.

clsRegistrator

Dit is de registrator class. Iedere registrator is een instance van deze class. De class bestaat uit een uit een aantal properties, een functie om de registrator te instantiëren en een functie die de eigenlijke registrator-functionaliteit implementeert.

9.3.5 Tags

clsTagBuilder

Deze class verzorgt de functionaliteit om nieuwe tags aan te maken en bestaande tags te verwijderen. Functionaliteit voor het aanpassen van een tag is overbodig. Een tag is immers niets meer dan een verwijzing naar een attribuut. Het aanpassen van een tag staat dus gelijk aan het verwijderen van de oude verwijzing en het aanmaken van een nieuwe verwijzing.

clsTag

De enige functionaliteit die de tag class biedt is het tonen van de waarde van het betreffende attribuut.

9.3.6 Preview Templates

clsTemplateBuilder

Deze class verzorgt de functionaliteit om nieuwe templates aan te maken en bestaande templates aan te passen of te verwijderen.

clsTemplate

Deze class levert de functionaliteit om een preview te genereren op basis van de voorgedefinieerde attribuutwaarden van het template.

10 Implementatie

In dit hoofdstuk zal verslag gedaan worden van de daadwerkelijke implementatie van het besproken ontwerp.

10.1 Inleiding

Het implementeren van het bovenstaande ontwerp is behoorlijk veel werk. Aangezien er maar een beperkte hoeveelheid tijd beschikbaar was voor de implementatie, moest er een selectie worden gemaakt uit de verschillende componenten. Om toch een redelijk afgebakend geheel te hebben, is er gekozen voor de onderdelen 'attributen', 'expressies' en 'content-routers'. De keuze is gevallen op die drie onderdelen, omdat voor het implementeren van content-routers er namelijk de beschikking moet zijn over attributen en expressies.
In dit onderdeel van de scriptie zullen bovengenoemde drie onderdelen uitvoerig behandeld worden. Alvorens daarmee te beginnen, zal hieronder nog een korte opsomming worden gegeven van de benodigde functionaliteit.

10.1.1 Attributen engine

De attributen engine omvat alle functionaliteit die nodig is om het gebruik van attributen mogelijk te maken. Hieronder volgt een lijst met de verschillende mogelijkheden:

- Functionaliteit om nieuwe attributen toe te voegen
- Functionaliteit om attributen te verwijderen
- Functionaliteit om een instantiatie van een attribuut te creëren
- Functionaliteit om de waarde van een attribuut te bepalen. Dit is vooral van belang voor de zogenaamde berekende attributen

10.1.2 Expressie engine

De expressie engine omvat alle functionaliteit die nodig is om het gebruik van expressies mogelijk te maken. Hieronder volgt een lijst met de verschillende mogelijkheden:

- Functionaliteit om nieuwe expressies toe te voegen
- Functionaliteit om bestaande expressies zonodig aan passen
- Functionaliteit om expressies te verwijderen
- Functionaliteit om een instantiatie van een expressie te creëren
- Functionaliteit om een expressie te evalueren en het resultaat terug te geven

10.1.3 Content-router

Van de twee mogelijke routers is er gekozen voor de implementatie van de content-router. Overigens kan met die functionaliteit ook relatief eenvoudig de URL-router worden geïmplementeerd. Voor het gebruik van een router is er de volgende functionaliteit nodig:

- Functionaliteit om nieuwe content-routers toe te voegen
- Functionaliteit om bestaande content-routers zonodig aan passen
- Functionaliteit om content-routers te verwijderen
- Functionaliteit om een instantiatie van een content-router te creëren
- Functionaliteit om een content-router expressies te laten evalueren en op basis van de uitkomst een keuze te maken uit verscheidene "bestemmingen".

10.1.4 Grafische User Interface

Naast de "onderhuidse" functionaliteit dient er natuurlijk ook een grafische kant van het geheel gemaakt te worden. Deze grafische user interface omvat:

- Functionaliteit om aan het lijstje van bestaande content objecten het router object toe te voegen
- Functionaliteit om het router object te tonen als een lijst van bestemmingen (opties)
- Functionaliteit om een optie toe te voegen. Dit omvat:
    - Functionaliteit om een wizard op te roepen die het mogelijk maakt om een nieuwe expressie te definiëren of een bestaande aan te passen
    - Functionaliteit om de (reeds bestaande) wizard op te roepen waarin een nieuw "bestemmings-object" gemaakt kan worden of een bestaande kan worden aangepast.
- Functionaliteit om het gemaakte router object toe te voegen aan de pagina
- Functionaliteit om een toegevoegde router object zijn werk te laten doen en het resultaat (het betreffende "bestemmings-object") te tonen.

10.2 Classes

Zoals in het vorige hoofdstuk al werd aangegeven, zou vrijwel alle functionaliteit worden ondergebracht in de kernel. Omdat er sprake was van nieuw toegevoegde functionaliteit, waren er weinig aanpassingen aan de bestaande kernel code nodig. Dit was zeer gunstig, aangezien er daardoor in parallel met de 'bestaande' ontwikkeling van het product, aan deze uitbreiding gewerkt kon worden.
In **Bijlage 5** worden alle classes behandeld die aan de kernel code zijn toegevoegd. Hierbij is dezelfde onderverdeling gehanteerd als in het hoofdstuk 'Technisch Ontwerp' is gebruikt.

11 Conclusies & Suggesties

In dit laatste hoofdstuk zal een terugblik worden gegeven op het gehele afstudeertraject. Er zal aandacht worden besteed aan het verloop van het afstuderen, de huidige stand van zaken en suggesties ter verbetering van het ontwerp/implementatie.

11.1 Afstudeerverloop

De eerste fase van het afstudeertraject verliep voorspoedig. Door de vele gesprekken met verschillende personen binnen Centric-et is er een gevarieerd beeld ontstaan over de mogelijkheden tot informatievergaring.

Ook de tweede fase verliep aanvankelijk volgens de planning, maar de ontwerpfase heeft het geheel behoorlijk vertraagd. Er is veel tijd besteed aan het ontwerp zonder altijd even veel vorderingen te boeken. Achteraf bekeken was het misschien beter geweest om eerder met de implementatiefase te beginnen. Op die manier waren bepaalde knelpunten eerder aan het licht gekomen en had daar ook eerder adequaat op kunnen worden ingesprongen. De voornaamste reden dat de ontwerpfase zo lang heeft geduurd, was het streven om het ontwerp zo goed en volledig mogelijk te maken, zodat de implementatiefase relatief eenvoudig zou zijn. Immers al het denkwerk was dan al in de ontwerpfase gedaan. In de praktijk bleek echter wel dat veel ontwerpbeslissingen pas genomen werden tijdens de implementatiefase.
Dit was vooral te wijten aan een zeker gebrek aan programmeerervaring. Het bleek moeilijk in te schatten hoeveel tijd bepaalde implementatiestappen zouden vergen en welke mogelijke problemen er te verwachten vielen.
Enerzijds is er dus te lang gewacht met het beginnen van de implementatiefase, maar anderzijds is er ook het nodige inzicht verkregen in de 'problemen' die kunnen optreden. Bij een soortgelijk project zal er dus in de toekomst niet perse minder tijd aan de ontwerpfase worden besteed. Aangezien er nu meer kennis is over de valkuilen tijdens het implementatieproces, kan daar in het vervolg al in de implementatiefase op worden ingespeeld.

11.2 Huidige stand van zaken

Zoals in het vorige hoofdstuk is vermeldt, is er nu de beschikking over functionaliteit voor attributen, expressies en content-routers. Er zijn echter meerdere onderdelen behandeld in deze scriptie en ook die onderdelen zullen geimplementeerd moeten worden om volledige e-business-functionaliteit te realiseren.

Eén van de belangrijkste onderdelen zijn paspoorten en registrators. Deze maken het immers mogelijk om informatie over de gebruiker te verkrijgen.
Daarnaast moeten de URL-routers geïmplementeerd worden, zodat niet alleen de inhoud van pagina's maar ook de navigatie tussen de pagina's gepersonaliseerd kan worden.

Verder zijn er nog de tags en preview templates. Deze onderdelen zijn weliswaar niet zo onmisbaar als de vorige drie componenten, maar hebben wel degelijk een meerwaarde.

Een ander belangrijk aspect van e-business is de mogelijkheid om informatie te kunnen importeren en exporteren. Vooral voor externe analyse van gegevens is het van belang dat de data van en naar de Data Warehouse kunnen worden 'getransporteerd'. De gegevens uit externe analyse zijn op hun beurt weer van belang voor het samenstellen van het gebruikersprofiel. Ook het gebruik van algemene profielen is afhankelijk van met name goede import-functionaliteit. Het is dus duidelijk dat deze functionaliteit nodig is om een zo'n volledig mogelijk gebruikersprofiel te kunnen samenstellen

Behalve het toevoegen van de hierboven genoemde onderdelen, zijn er ook nog een aantal punten in het ontwerp en in de implementatie die voor verbetering vatbaar zijn. Deze punten zullen in de komende secties behandeld worden.

11.3 Expressie Evaluatie

Allereerst is er het aspect van expressie evaluatie. Voor dit doel is er gebruik gemaakt van een dll die geschreven is in C#.
De code hiervoor is te vinden op http://www.codeproject.com/csharp/runtime_eval.asp.
Door enkele aanpassingen in deze code is het mogelijk om deze dll vanuit VB code aan te spreken.
Er kleeft echter een nadeel aan de gebruikte C#-code. Er wordt namelijk gebruik gemaakt van een runtime compiler. Dit houdt in dit geval in dat er voor iedere evaluatie van een expressie een dll in het geheugen wordt geplaatst. Bij intensief gebruik zal er daardoor een geheugentekort onstaan. Bij kleinschalig gebruik zal dit niet zo snel optreden, maar op grootschaliger niveau is dit onacceptabel.
Een mogelijke oplossing voor dit probleem is om niet continu een constante expressie te laten evalueren, maar alle expressies als functies op te slaan in één dll en vervolgens de functies in deze dll aan te roepen met de benodigde attribuutwaarden als parameters. Iedere keer als de beheerder een expressie toevoegt, aanpast of verwijdert, wordt de dll opnieuw gecompileerd. 'At runtime' is er dan nog maar één dll nodig in het geheugen in plaats van een dll voor iedere expressie-evaluatie.
Zo'n aanpassing zal weinig impact hebben op de bestaande code, aangezien het evalueren van expressies in een aparte Evaluator class is ondergebracht.

Het mag duidelijk zijn dat de huidige implementatie dus niet geschikt is voor gebruik in een productversie van Web4all. De huidige implementatie levert echter wel een 'proof of concept'. Er kan dus wel een goede demonstratie van het concept worden gegeven. Vanuit het oogpunt van het afstudeerwerk is de huidige implementatie dus wel degelijk toereikend, maar voor grootschaliger gebruik is de aanpassing echter onontbeerlijk.

11.4 Database Access

Momenteel worden database queries 'on demand' uitgevoerd. Dit betekent dus dat er een tijdens een sessie een aanzienlijk aantal database accessen plaatsvinden. Daar tegenover staat wel dat er gebruik gemaakt wordt van 'stored procedures'. Hierdoor is een groot deel van de query-verwerking al van tevoren gedaan, zodat de queries 'at runtime' aanzienlijk sneller uitgevoerd kunnen worden.
Desalniettemin kan er overwogen worden om bepaalde data bij het begin van een sessie al op te halen uit de database en op te slaan in bijvoorbeeld sessie-variabelen. Bijvoorbeeld de profielgegevens van een (ingelogde) gebruiker zouden direct bij het inloggen al kunnen worden opgehaald uit de database. Hierdoor zijn de persoonlijke gegevens van de gebruiker direct beschikbaar op de momenten dat er naar 'gevraagd' wordt.

11.5 Zoekfunctionaliteit

Een interessant punt met het oog op het gebruik van routers, is de zoekfunctionaliteit. Routers zijn afhankelijk van de context waarin ze worden aangeroepen. Bij een zoekopdracht moet daar dus rekening mee worden gehouden.
Een zoekopdracht mag namelijk geen resultaten opleveren die voor een bepaalde gebruiker helemaal niet toegankelijk horen te zijn. Anderzijds, zij het minder problematisch, moet het ook niet zo zijn dat een zoekopdracht niks oplevert terwijl een router wel degelijk verwijst naar de gezochte informatie.
Bij een zoekopdracht binnen een router moet er dus bekend zijn waarnaar een router zal verwijzen. Hiervoor is het nodig om de benodigde contextparameters te kennen. Dit pleit voor de hierboven beschreven 'tactiek' om al bij het begin van een sessie de nodige contextinformatie uit de database op te halen.
Een mogelijkheid zou dan kunnen zijn om tijdens de zoekopdracht de router-functionaliteit aan te roepen en zodanig te achterhalen waarnaar de router zal verwijzen en waarbinnen dus gezocht kan worden.
Nu zijn er echter ook situaties denkbaar waarin de benodigde contextparameters simpelweg niet bekend (kunnen) zijn. Denk bijvoorbeeld aan router-expressies die gebruik maken van registrator-attributen. Deze kunnen tijdens een sessie een andere waarde krijgen, waardoor bepaalde content al dan niet toegankelijk wordt. Dit is op het moment van de zoekopdracht nog niet bekend.
Dit hoeft echter niet onacceptabel te zijn. In principe moet een zoekopdracht informatie opleveren die op dat moment beschikbaar is. Er hoeft dus alleen gekeken te worden naar bestemmings-objecten die op het moment van de zoekopdracht bereikbaar zijn.
Behalve de gebruiker zal de beheerder zeer waarschijnlijk ook gebruik willen maken van de zoekfunctionaliteit. Dat levert een andere situatie op. Indien de beheerder een zoekopdracht uitvoert, hoeft er geen rekening te worden gehouden met de router-functionaliteit. Het is dan zaak dat de expressies als het ware genegeerd worden en er dus direct gezocht kan worden in de 'bestemmings-objecten'.

11.6 Import/Export

In de requirements wordt er gesproken over de mogelijkheid tot import en export van gegevens. Ook deze functionaliteit is zeer belangrijk voor e-business. Vooral voor de Data Warehouse zijn goede import- en exportmogelijkheden van belang. Reeds vergaarde informatie moet namelijk in het nieuwe systeem kunnen worden geïmporteerd en nieuwe informatie moet naar de Data Warehouse kunnen worden weggeschreven voor verdere verwerking. Voor een efficiënt gebruik van de gegevens is het dus nodig dat deze functionaliteit nog wordt toegevoegd.

11.7 DataShare

Zoals in hoofdstuk 7 al werd aangegeven, is het wenselijk dat ook attributen gebruik kunnen maken van DataShare. Deze functionaliteit is echter nog niet aanwezig en vereist dus een uitbreiding van de attributen-engine.

12 Referenties

[1] Cookie Central: http://www.cookiecentral.com/content.phtml?area=2&id=1

[2] Cookies in IE 6: http://support.microsoft.com/default.aspx?scid=kb;nl;283185.

[3] Brusilovsky, P.: Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction* 6, 2-3 (1996) 87-129

12.1 Links

- C# expressie evaluator: http://www.codeproject.com/csharp/runtime_eval.asp
- Voorbeeld van een feed: http://www.reiskrant.nl
- Voorbeeld van een co-branded site: http://www.happytravel.neckermann-reizen.nl
- Logfile analyse: http://www.webtrends.com
- Webcounter: http://www.stats4all.nl

13 Bijlagenoverzicht

Bij deze scriptie behoren de volgende bijlagen:

- **Bijlage 1: Informatieschema**
- **Bijlage 2: Ontwerpbeslissingen m.b.t. Routers**
- **Bijlage 3: Grafisch Ontwerp**
- **Bijlage 4: ER-Diagram**
- **Bijlage 5: Classes**

14 Bijlage 1: Informatieschema

Deze bijlage bevat een schematische weergave van een systeem dat de functionaliteit bezit om gebruikersinformatie te vergaren, verwerken en toe te passen (zie de volgende pagina).
Om het schema te verduidelijken is er een toelichting toegevoegd. Allereerst worden de verschillende lagen en componenten in het schema behandeld. In de daarop volgende secties zullen achtereenvolgens de input en output van de verschillende componenten behandeld worden en tot slot de daadwerkelijke informatiestromen.

14.1 Systeemlagen

Het systeem uit het schema bestaat uit een aantal lagen. De verschillende lagen zullen nu kort worden behandeld.

Front Office (FO)

Het Front Office is de link tussen het systeem en de eindgebruiker. Alle communicatie van de gebruiker met het systeem verloopt via het Front Office. Het Front Office is dus het FrontEnd van het systeem.

Back Office (BO)

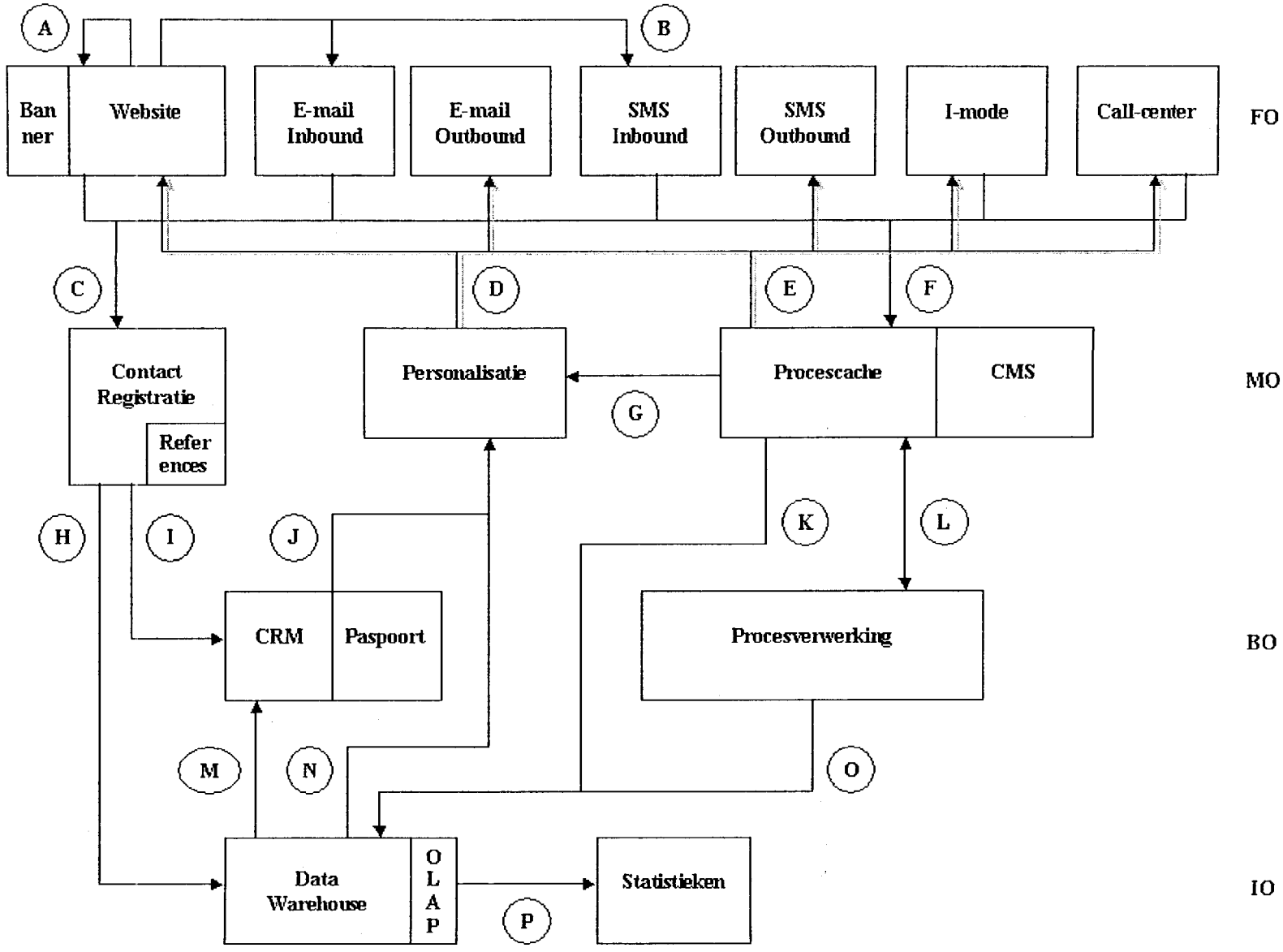
Het Back Office van het systeem is als het ware de kern van het systeem. In het Back Office vinden alle processen van het systeem plaats. Het Back Office is dus het BackEnd van het systeem.

Mid Office (MO)

Het Mid Office is de verbindende schakel tussen het Front en Back Office. De 'ruwe data' uit het Back Office wordt via het Mid Office omgezet in bruikbare content voor het Front Office. De ander kant op wordt data die via het Front Office het systeem inkomt, via het Mid Office doorgesluisd naar het Back Office voor verdere verwerking.

Intelligence Office (IO)

Tot slot is er een laag die als Intelligence Office wordt aangeduid. Deze laag verwerkt de ontvangen gegevens van de gebruikers en analyseert deze om te komen tot statistieken en bruikbare feedback voor hoger gelegen lagen, m.n. het onderdeel *Profiel*.


A
B
Ban ner
Website
E-mail Inbound
E-mail Outbound
SMS Inbound
SMS Outbound
I-mode
Call-center
FO
C
D
E
F
Contact Registratie
Refer ences
Personalisatie
Procescache
CMS
MO
G
H
I
J
K
L
CRM
Paspoort
Procesverwerking
BO
M
N
O
Data Warehouse
OLAP
Statistieken
P
IO

14.2 Componenten

Deze sectie geeft een toelichting op de verschillende onderdelen van het schema.
Bij de onderdelen e-mail, SMS en callcenter is er een onderscheid gemaakt tussen inbound en outbound. Onder inbound worden de e-mail- en SMS-berichten verstaan die het systeem binnen komen, alsmede de inkomende telefoongesprekken. De outbound communicatie omvat de e-mail- en SMS-berichten die het systeem naar de gebruikers stuurt, alsmede de uitgaande telefoongesprekken. De reden dat er bij websites en i-mode niet voor deze splitsing is gekozen, is dat deze media een meer interactief karakter hebben. De inbound en outbound communicatie zijn meer met elkaar verweven. In het geval van bijvoorbeeld e-mail is er een duidelijkere scheiding zichtbaar tussen inbound en outbound, dan bij bijvoorbeeld websites.

14.2.1 Banner / Website

Deze twee componenten zijn samengenomen, omdat beide betrekking hebben op het internetverkeer en banners eigenlijk altijd onderdeel zijn van een website.
Het doel van banners is om gebruikers naar de eigen website te krijgen. Om na te gaan hoe effectief deze banners zijn, moet bepaald worden vanaf welke banner is doorgeklikt naar de eigen website. Dit wordt gedaan in het Reference-gedeelte van Contact Registratie. Daar kan namelijk bepaald worden via welke banner men op de website terecht is gekomen.
De website zelf levert over het algemeen nog de meeste en tevens meest diverse informatie op met betrekking tot de gebruikers.
De meest eenvoudige manier om informatie over de gebruiker te vergaren, is door de gebruiker het zelf te laten aanbieden. Doordat de gebruiker een formulier invult met persoonsgegevens en/of voorkeuren, kan men al een heleboel over de gebruiker te weten komen. Deze methode is echter vrij direct en de gebruiker kan dus zelf bepalen of hij de informatie wilt verstrekken. Vaak is de gebruiker echter genoodzaakt om zijn gegevens prijs te geven, omdat hij anders simpelweg niet verder komt. Als een klant bijvoorbeeld online een reis wil boeken, komt hij er niet onderuit om persoonsgegevens in te voeren.
Men kan ook op een minder "opvallende" manier de nodige informatie over een gebruiker vergaren. Met behulp van cookies en sessions kan precies worden bijgehouden wat de gebruiker voor handelingen verricht op de website.
Daarnaast zijn er de pagehits. "Grote" sites hebben echter ook een groot aantal pagehits. Het is daarom veelal niet wenselijk om iedere afzonderlijke pagehit op te slaan. In plaats daarvan kan ervoor gekozen worden om de pagina's in te delen in groepen en zo een hiërarchie te creëren. Zodoende ontstaat er een situatie waarin niet gekeken wordt naar afzonderlijke pagehits, maar naar "grouphits" (zie ook het hoofdstuk over Renault).
Een alternatief is om de pagehits tijdelijk op te slaan. Door de pagehits te analyseren en de resultaten op te slaan, hoeven de afzonderlijke pagehits niet langer bewaard te blijven. Zodra de benodigde informatie is geëxtraheerd, kunnen de pagehits verwijderd worden. Dit vereist echter wel een grondige analyse van de gegevens, want eenmaal verwijderd kan deze informatie natuurlijk niet meer gebruikt worden.

Naast de pagehits kan ook naar andere aspecten van het surfgedrag van de gebruiker worden gekeken. Hoe lang verblijft hij op een bepaalde pagina, welk surfpad volgt hij, welke aspecten van een pagina zijn van belang om op te slaan, enz.

14.2.2 E-mail Inbound

Van inkomende (inbound) e-mail zijn eigenlijk 2 dingen van belang:

- Afzender
- Onderwerp/Inhoud

De afzender (het e-mailadres) gaat naar het onderdeel Contact Registratie, alwaar de informatie doorgestuurd wordt naar de Data Warehouse en eventueel het Paspoort van de afzender (als de persoon in kwestie er al één heeft natuurlijk).

De inhoud van het e-mailbericht kan verschillende formaten hebben. Enerzijds kan het correspondentie-mail zijn en heeft het (vaak) geen standaard formaat. In zo'n geval wordt de informatie doorgespeeld aan de klantenservice. Die informatie gaat via het onderdeel Procescache/CMS naar Procesverwerking. Anderzijds kan het bericht wel een standaard formaat hebben. Dit kan door de gebruiker zelf gedaan zijn (door middel van een voorgedefinieerde subjectregel) of door het systeem dat de e-mail gegenereerd heeft (dat kan het eigen systeem zijn). Ook in dat geval gaat de informatie via Procescache/CMS naar het onderdeel Procesverwerking, alleen nu voor een automatische afhandeling.

14.2.3 E-mail Outbound

Van uitgaande e-mail zijn de belangrijkste punten:

- Het e-mail adres waarna verzonden wordt (individueel of mailinglijst)
- Eventueel het onderwerp van de e-mail
- Een ID, die gebruikt kan worden voor referentie-doeleinden. Als de e-mail bijvoorbeeld een hyperlink naar de website bevat, kan op basis van de ID bepaald worden om welk e-mailbericht het gaat. Dit is handig om bijvoorbeeld het rendement van een mailing te bepalen: hoeveel extra bezoekers heeft de mailing opgeleverd?

Deze e-mail wordt veelal samengesteld op basis van processen binnen het systeem en kan eventueel worden gepersonaliseerd. Hierbij kan er een onderscheid worden gemaakt tussen 3 soorten (outbound) e-mail, namelijk:

- Campagne mail aan leden van mailinglist
- Mail op basis van profiel
- Individuele mail

De eerste mogelijkheid is het eenvoudigst in opzet. Alle e-mail adressen behorend bij een mailinglijst worden opgehaald uit de database en vervolgens wordt de campagne mail verzonden naar de leden van de lijst. Alle leden krijgen dan hetzelfde e-mail bericht. Een voorbeeld hiervan is de nieuwsbrief.

Bij de tweede mogelijkheid wordt er al specifieker naar de klant gekeken. Op basis van diens profiel wordt er bepaalde mail wel of niet verstuurd. Deze mail is dus nog steeds niet individueel, maar is wel bestemd voor een meer selecte groep dan bij bijvoorbeeld nieuwsbrieven het geval is.

Tot slot is er nog de individuele mail. Dit type mail is alleen bestemd voor één bepaalde klant. Hierbij kan gedacht worden aan bevestigingen, opgevraagde informatie, persoonlijke meldingen, enz. Deze mail wordt dus samengesteld en verstuurd op basis van de status en het profiel van de individuele klant.

De eerste twee types bevatten vaak de hierboven genoemde hyperlinks, welke naar een specifiek deel van de website leiden. Het derde type e-mail kan ook hyperlinks bevatten, maar deze hebben vaak de taak van verificatie en bevestiging. "Klik op deze link om uw aanvraag te bevestigen" is een voorbeeld van dit soort hyperlinks.

14.2.4 SMS Inboud

Voor inbound SMS geldt eigenlijk hetzelfde als voor inbound e-mail. In dit geval wordt i.p.v. het e-mailadres, het telefoonnummer van de afzender opgeslagen. Bovendien zal de inhoud van het SMS-bericht eerder een standaard formaat hebben en dus kan de inhoud vaak automatisch worden afgehandeld. Denk hierbij aan voorgedefinieerde commando's als "GAME ON". Deze commando's worden door het systeem geïnterpreteerd en uitgevoerd. Vaak levert dit dan weer informatie op die nuttig gebruikt kan worden.

14.2.5 SMS Outbound

Uitgaande SMS-berichten kunnen in principe niets meer bevatten dan notificaties of bevestigingen voor de gebruiker. Om die reden zijn de enige gegevens, geschikt voor opslag, het telefoonnummer van de ontvanger en eventueel het onderwerp van de SMS-mailing. Zodoende kan bijgehouden worden welke klantcontacten er per SMS hebben plaatsgevonden. Overigens kan ook bij SMS het onderscheid in 3 soorten berichten gemaakt worden, zoals dat bij de e-mail-berichten ook is gedaan.

14.2.6 I-mode

I-mode bezit veel functionaliteit die het World Wide Web ook bezit. In dat opzicht kan er dus op een soortgelijke manier naar gekeken worden met betrekking tot de vergaring en verwerking van gebruikersinformatie. Momenteel heeft I-mode echter voornamelijk een

informerende functie en in mindere mate een echt interactieve functie. Het boeken van een reis bijvoorbeeld, zal voorlopig nog niet via dit medium gedaan worden, maar het kan wel de nodige informatie over de vakantiebestemming verschaffen.
Desalniettemin biedt I-mode wel veel meer mogelijkheden dan e-mail en/of SMS. Dit maakt het daarom ook interessant voor de communicatie met klanten.

14.2.7 Call-center

Moderne call-centers zijn veelal semi-automatisch. De gebruiker wordt eerst door een computergestuurde stem door een aantal menu's geloodst om uiteindelijk, ofwel automatisch geholpen te worden (het noemen van openingstijden bijvoorbeeld), ofwel doorverbonden te worden met een medewerker van de helpdesk.
Een van de eerste dingen die opgeslagen kan worden is het telefoonnummer van de beller. Daarnaast kan aan de hand van de menukeuzes nagegaan worden wat voor type vraag/probleem de gebruiker heeft. Bovendien kan de helpdesk-medewerker zelf ook de nodige informatie invoeren, indien nuttig.
De meeste informatie zal via het onderdeel Contact Registratie verwerkt worden. Het is echter ook mogelijk dat bepaalde menu-keuzes processen aanroepen in het Backoffice. In dat geval zal de informatie via Procescache/CMS worden doorgestuurd naar Procesverwerking.
Behalve deze inbound-versie van het call-center, is er natuurlijk ook het outbound-equivalent in de vorm van telemarketing. Bij deze vorm van klantcontact is vooral van belang hoe groot de effectiviteit van een campagne is. Om dit na te gaan moet dus in ieder geval bijgehouden worden hoeveel en welke telefonische contacten er met de (potentiële) klant zijn geweest.

14.2.8 Contact Registratie

Het onderdeel Contact Registratie ontvangt in principe alle informatie die uit een klantcontact voortkomt. Bijna al deze informatie wordt doorgesluisd naar de Data Warehouse om vervolgens verder verwerkt te worden.
Een deel van de vergaarde gegevens gaat echter (ook) naar het onderdeel Customer Relationship Management (CRM).
Bovendien bevat Contact Registratie een onderdeel dat zich bezighoudt met zogenaamde "References". Dit onderdeel richt zich op de zogeheten broncode van een klantcontact. In dit zinsverband heeft broncode een andere betekenis dan die van programmacode (Engels: source code). In dit geval geeft de broncode aan 'waar' een gebruiker vandaan komt. M.a.w. hoe is de gebruiker op een website gekomen of door wie/wat is een e-mail verstuurd? Op deze manier kan nagegaan worden of een e-mailcampagne effect heeft gehad, maar ook hoe een gebruiker door de website navigeert.

14.2.9 Personalisatie

Personalisatie omvat alle handelingen die nodig zijn, om de verschillende media aan te passen aan de behoeften en wensen van de gebruiker. Dit onderdeel maakt gebruik van de verschillende andere onderdelen om tot een duidelijk beeld van de gebruikers te komen. Vervolgens wordt getracht zo effectief mogelijk de media aan te passen aan de gebruiker. Hierbij maakt Personalisatie gebruik van informatie uit de onderdelen CRM/Paspoort, Procescache/CMS en eventueel ook het Data Warehouse.

14.2.10 Procescache/CMS

Het onderdeel Procescache/CMS ontvangt van alle media-typen de informatie die nodig is om de verschillende taken van het systeem uit te voeren.
Het eigenlijke uitvoeren van de taken gebeurt in het onderdeel Procesverwerking. Dit onderdeel krijgt de benodigde input van de Procescache. De informatie (output) die Procesverwerking oplevert wordt vervolgens door het CMS (Content Management Systeem) gebruikt om feedback te leveren aan de Frontoffice laag.
Verder wordt een deel van de informatie ook gebruikt voor het Personalisatie-proces.

14.2.11 CRM/Paspoort

Customer Relationship Management omvat eigenlijk alle maatregelen om klantgedrag te registreren en analyseren om er vervolgens de klant ook weer mee van dienst te zijn. Om die reden is het onderdeel Paspoort hieraan toegevoegd. Immers, het Paspoort bevat nuttige informatie over de klant en zijn voorkeuren/interesses. Deze kunnen gebruikt worden om de klant gerichter te bedienen van informatie. Om de benodigde input te vergaren, doet het onderdeel CRM/Paspoort een beroep op de gegevens uit de Data Warehouse alsmede data die direct uit Contact Registratie afkomstig is.

14.2.12 Procesverwerking

Dit onderdeel verzorgt de daadwerkelijke uitvoering van de verschillende processen. Procesverwerking omvat eigenlijk alle Backoffice applicaties, uitgezonderd het CRM/Paspoort-aspect.
Over het algemeen zal dit onderdeel niet direct bruikbare informatie opleveren met betrekking tot klantcontacten. Deze informatie zal eerder door Procescache/CMS worden gegenereerd (Midoffice dus). Informatie die wel bruikbaar kan zijn, wordt doorgesluisd naar de Data Warehouse voor verdere verwerking.

14.2.13 Data Warehouse

Bijna alle informatie die vergaard wordt uit de verschillende media, komt uiteindelijk in het Data Warehouse terecht. Andere onderdelen (voornamelijk uit het Midoffice) gebruiken deze informatie voor de verschillende processen, waaronder Personalisatie. Een ander belangrijk onderdeel (met name voor marketing) is het onderdeel Statistieken. Gegevens uit de Data Warehouse worden m.b.v. OLAP-technieken (On Line Analytical Processing) geanalyseerd en vervolgens gebruikt om duidelijke statistische overzichten te genereren. Denk hierbij aan zaken als campagne-rendement (hoe "effectief" is een campagne geweest) en populariteit van pagina's (aantal pagehits).

14.2.14 Statistieken

Zoals hierboven vermeld, genereert het onderdeel Statistieken, met behulp van de OLAP-faciliteiten van het Data Warehouse, de benodigde statistische informatie. De informatie kan o.a. het management een duidelijk overzicht geven van de prestaties van het bedrijf of instelling. Aan de hand hiervan kunnen dan eventueel aanpassingen aan de Frontoffice worden gedaan.

14.3 Input en Output

Deze sectie geeft een overzicht van de input en output van de verschillende media, die nuttig zijn om op te slaan. Met input wordt in dit verband bedoeld de gegevens en gebeurtenissen die, door het gebruik van het medium, voor het systeem beschikbaar komen. De output is in dit verband de handelingen die het systeem, als reactie op de input uitvoert. De begrippen input en output zouden dus ook gezien kunnen worden als actie en reactie.

Website input:

- Click-events
- Inhoud van ingevulde formulieren
- De (content van de) pagina's zelf
- De verblijfsduur op de pagina's

Website output:

- Feedback naar de gebruiker (bijvoorbeeld een bevestiging van een betaling)
- Aangepaste content van de pagina's o.b.v. gebruiker(sprofiel)
- Externe acties (verzenden van e-mail en/of SMS)

Zowel de website input als output zijn het opslaan waard, omdat zo een volledig beeld ontstaat van wat er zich allemaal op de website heeft afgespeeld.

E-mail input (inbound):

- Afzender
- Onderwerp
- Inhoud

E-mail output (inbound):

- Bevestiging van ontvangst
- Uitvoeren van acties op basis van de content van de e-mail. Gestandaardiseerde e-mails kunnen opdrachten voor de website bevatten. Bijvoorbeeld "ABONNEMENT BEËINDIGEN" in subject van e-mail.

Ook hierbij kan de genoemde data worden opgeslagen. Zodoende krijgt men een beter beeld van het inkomende e-mail verkeer en de afhandeling ervan.

E-mail input (outbound):

- E-mail adres(sen)

- Onderwerp
- Inhoud
- Type: Campagne mail, Profiel mail, Persoonlijke mail

E-mail output (outbound):

- Verzending

Door deze gegevens bij te houden, kan precies worden nagegaan welke mail er aan wie verstuurd is. Zo kan bijvoorbeeld nagegaan worden hoe effectief een mailing is geweest.

SMS input (inbound):

- Telefoonnummer
- Commando; meestal bevatten inbound SMS-berichten enkel een commando (bijvoorbeeld CHECK ORDER, GAME ON, enz.). Inbound SMS wordt bijna nooit voor berichtgeving gebruikt.

SMS output (inbound):

- Bevestiging van of reactie op ontvangen SMS-bericht
- Interne actie, bijvoorbeeld het annuleren van een reservering

Het opslaan van deze gegevens levert in ieder geval het telefoonnummer van de klant op. Bovendien is het registreren van een uitgevoerde actie eveneens een nuttige handeling.

SMS input (outbound):

- Telefoonnummer(s)
- Inhoud bericht; vaak automatisch samengesteld

SMS output (outbound):

- Verzending SMS-bericht(en)

Dit soort outbound SMS-berichten is vaak alleen bestemd voor informering van de gebruiker, zoals melding van nieuwe aanbiedingen of producten.

I-mode input:

- Click-events / Toetsaanslagen
- Inhoud van ingevulde formulieren of andere getypte input
- De (content van de) pagina's zelf

I-mode output:

- Feedback naar gebruiker
- Aangepaste content van de pagina's o.b.v. gebruiker(sprofiel)
- Externe acties (verzenden van e-mail/SMS en/of bellen van een nummer)

Voor i-mode geldt ongeveer hetzelfde als voor internet, vandaar ook dat de in- en outputs min of meer overeenkomen.

Call-center input (inbound):

- Telefoonnummer
- Onderwerp/Probleem

Call-center output (inbound):

- Feedback naar beller

Bij call-centers (inbound) hebben we voornamelijk te maken met de helpdesks, die de klanten te woord staan en klachten/problemen afhandelen.

Call-center input (outbound):

- Telefoonnummers
- Onderwerp

Call-center output (outbound):

- Feedback van de gebelde

De outbound-variant van call-centers is in de praktijk vaak telemarketing. In principe is dit precies het omgekeerde van de helpdesks. I.p.v. dat de gebruiker contact legt met het bedrijf, legt het bedrijf contact met de gebruiker/klant.

14.4 Informatiestromen

Deze sectie geeft een toelichting bij de verschillende informatiestromen binnen het schema. In het schema staan verscheidene rondjes met letters erin. Deze 'tags' corresponderen met de verschillende informatie-stromen in het schema. Hieronder zal een toelichting gegeven worden op al deze stromen aan de hand van de bijbehorende tags.

A. Deze 'loop' geeft het adaptatie-aspect van de website weer. Op basis van het surfgedrag van de gebruiker, kan de website 'on-the-fly' worden aangepast aan die gebruiker.

B. Het is mogelijk om de website zelf e-mail-berichten te laten verzenden naar het eigen systeem. Het versturen van een vraag voor de klantenservice door een formulier in te vullen, is hierbij een goed voorbeeld. Het systeem zal dan zelf een e-mail-bericht genereren en versturen naar de betreffende klantenservice.

C. Alle informatie over de gebruikers die via de verschillende media het systeem binnenkomt, wordt verzameld in het onderdeel Contact Registratie. Van hieruit worden de gegevens doorgespeeld naar de Data Warehouse ofwel direct naar het CRM.

D. Deze 'verbinding' staat voor het proces van personalisering. Op basis van de gebruikersinformatie worden de verschillende media aangepast aan de (gebleken) voorkeuren van de gebruiker.

E. Na uitvoering van de verschillende processen worden de resultaten doorgesluisd naar de verschillende media om daar vervolgens gepresenteerd, danwel verstuurd te worden.

F. Naast gebruikersinformatie leveren de verschillende media ook data op om verwerkt te worden in de Backoffice. Deze data worden daarvoor eerst verstuurd naar het onderdeel Procescache/CMS.

G. De verschillende processen leveren nuttige informatie op die gebruikt kan worden bij het proces van personalisering.

H. Alle gegevens die via de klantcontacten zijn vergaard, worden doorgegeven aan de Data Warehouse alwaar de data worden opgeslagen om later verder verwerkt te worden.

I. De klantgegevens uit het onderdeel Contact Registratie zijn nuttig te gebruiken in het CRM. Uit de klantgegevens kan immers worden afgeleid hoe men het beste de gebruikers kan benaderen. NAW-gegevens en persoonlijke voorkeuren, die de gebruikers opgeven door bijvoorbeeld een formulier op de website in te vullen, worden doorgespeeld naar het onderdeel Paspoort.

J. Ook de output van het CRM is zeer bruikbaar voor de personalisering van de verschillende media-bronnen. Vandaar ook dat er een verbinding tussen het CRM en het onderdeel Personalisatie is opgenomen in het schema. De informatie uit het onderdeel Paspoort wordt eveneens gebruikt voor de personalisering van de verschillende media

K. Niet alleen de output van de processen, maar ook de processen zelf kunnen nuttig zijn. Vandaar dat dit onderdeel ook verbonden is met de Data Warehouse.

L. Het onderdeel Procescache/CMS in het Midoffice is als het ware de link tussen het Front en Backoffice. Het onderdeel Procescache/CMS sluist de input van het Frontoffice door naar de procesverwerking in het Backoffice. Het Backoffice op zijn beurt stuurt de output weer terug naar het Midoffice, dat de data weer aanbied aan het Frontoffice.

M. De gegevens die in de Data Warehouse zijn opgeslagen, kunnen weer nuttig gebruikt worden door het CRM.

N. Bovendien kunnen die data ook direct worden doorgesluisd naar het onderdeel Personalisatie.

O. De output van de verschillende processen worden ook in de Data Warehouse opgeslagen.

P. Uit alle gegevens die in de Data Warehouse zijn opgeslagen, kunnen ook de nodige statistieken worden afgeleid. Dit gebeurt door middel van de eerder genoemde OLAP technieken (On Line Analytical Processing).

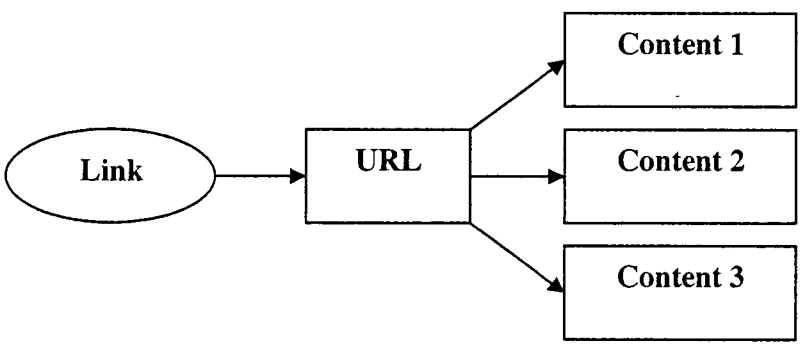
15 Bijlage 2: Ontwerpbeslissingen m.b.t. Routers

Deze bijlage behandelt de ontwerpbeslissingen die zijn genomen met betrekking tot routers.
Hierbij is onderscheid gemaakt tussen de URL-routers en Content-routers.

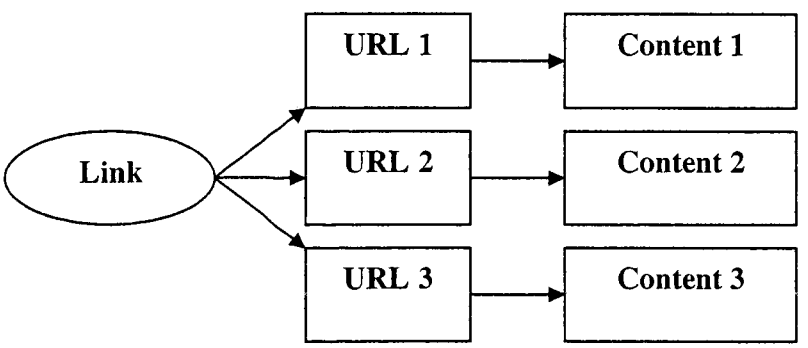
15.1 URL Router

Een dynamische link kan op verschillende manieren tot stand worden gebracht, waaronder de volgende twee mogelijkheden:

- Een link bevat een URL naar een pagina die afhankelijk van de uitkomst van de regel verschillende content toont. Dit ziet er schematisch als volgt uit:



- Een link kan naar meerdere pagina's verwijzen en afhankelijk van de uitkomst van de regel wordt een bepaalde URL gekozen. Dit ziet er schematisch als volgt uit:



De keuze is gevallen op de laatste optie, omdat de eerste optie al min of meer geleverd wordt door de hieronder besproken content router. Om deze laatste optie te realiseren, kan er als volgt te werk worden gegaan.

15.2 Content Router

Voor wat 'rule based content' zijn er de volgende opties:

- Alle content-objecten bevinden zich op de webpagina, maar de uitkomst van de regels bepaalt welke content-objecten zichtbaar zijn. M.a.w. content-objecten die niet van belang zijn, worden 'verborgen'. Deze methode staat bekend als de 'Conditional Text' techniek **[3]**. In pseudo-code ziet dat er als volgt uit:

  IF <boolean expressie>
  THEN <content-object is zichtbaar>
  ELSE <content-object is 'verborgen'>

- De webpagina is in beginsel leeg en de uitkomst van de regels bepaalt welke content-objecten er in de pagina worden geplaatst. Hierbij wordt het *eerste* content-object dat aan een regel voldoet getoond. In pseudo-code:

  IF <boolean expressie 1>
  THEN <plaats content-object 1>
  ELSE
  IF <boolean expressie 2>
  THEN <plaats content-object 2>
  ELSE
  IF ...
  ...
  ELSE <plaats default content-object> OR <plaats geen content-object>

  Voor dit doel kan uiteraard ook de CASE-statement gebruikt worden.

- De webpagina is in beginsel leeg en de uitkomst van de regels bepaalt welke content-objecten er in de pagina worden geplaatst. Het verschil met de vorige mogelijkheid is echter dat er nu een non-deterministiche keuze wordt gemaakt. Dat wil zeggen dat ieder content-object dat aan een regel voldoet, evenveel kans heeft om getoond te worden i.p.v. altijd het *eerste* content-object. De pseudo-code hiervoor is:

  IF <boolean expressie 1> → <plaats content-object 1>
  [] <boolean expressie 2> → <plaats content-object 2>
  [] ...
  ...
  FI

  Deze laatste optie is meer een theoretische implementatie, aangezien het niet mogelijk is om een volledig non-deterministische keuze te implementeren. De mogelijkheid is echter wel toegevoegd voor de volledigheid.

De keuze gevallen op optie 2. Optie 1 is vanuit grafisch oogpunt niet erg geschikt, omdat het dan vrij lastig wordt om een overzicht te bewaren voor wat betreft de layout van de webpagina. Optie 3 is, zoals reeds opgemerkt, praktisch niet haalbaar en daarom is gekozen voor optie 2.

16 Bijlage 3: Grafisch Ontwerp

In deze bijlage wordt er uitgebreid ingegaan op het grafische aspect van het ontwerp. Van ieder component is een soort flowchart van schermen gemaakt, die een visuele indruk geven van het uiteindelijke resultaat. Opgemerkt dient te worden dat de afbeeldingen conceptschetsen zijn van het uiteindelijke resultaat. Dit betekent dat de schermen aan verandering onderhevig kunnen zijn. De layout ligt bijvoorbeeld niet vast en eventueel kunnen er ook nog kleine aanpassingen aan de inhoud van de vensters gedaan worden indien tijdens het ontwikkelproces daartoe de noodzaak blijkt.

16.1 Berekende attributen

Het definiëren van berekende attributen gebeurt via een optie in de menubalk onder de knop 'Invoegen'. Er is voor deze locatie gekozen, omdat het definiëren van berekende attributen in de praktijk vaak zal gebeuren wanneer de beheerder bezig is met het aanmaken van routers. Om dan snel toegang te krijgen tot deze 'attribuut-editor' is de menubalk de meest logische en praktische locatie.
Het hoofdvenster van de attribuut-editor heeft de volgende onderdelen:

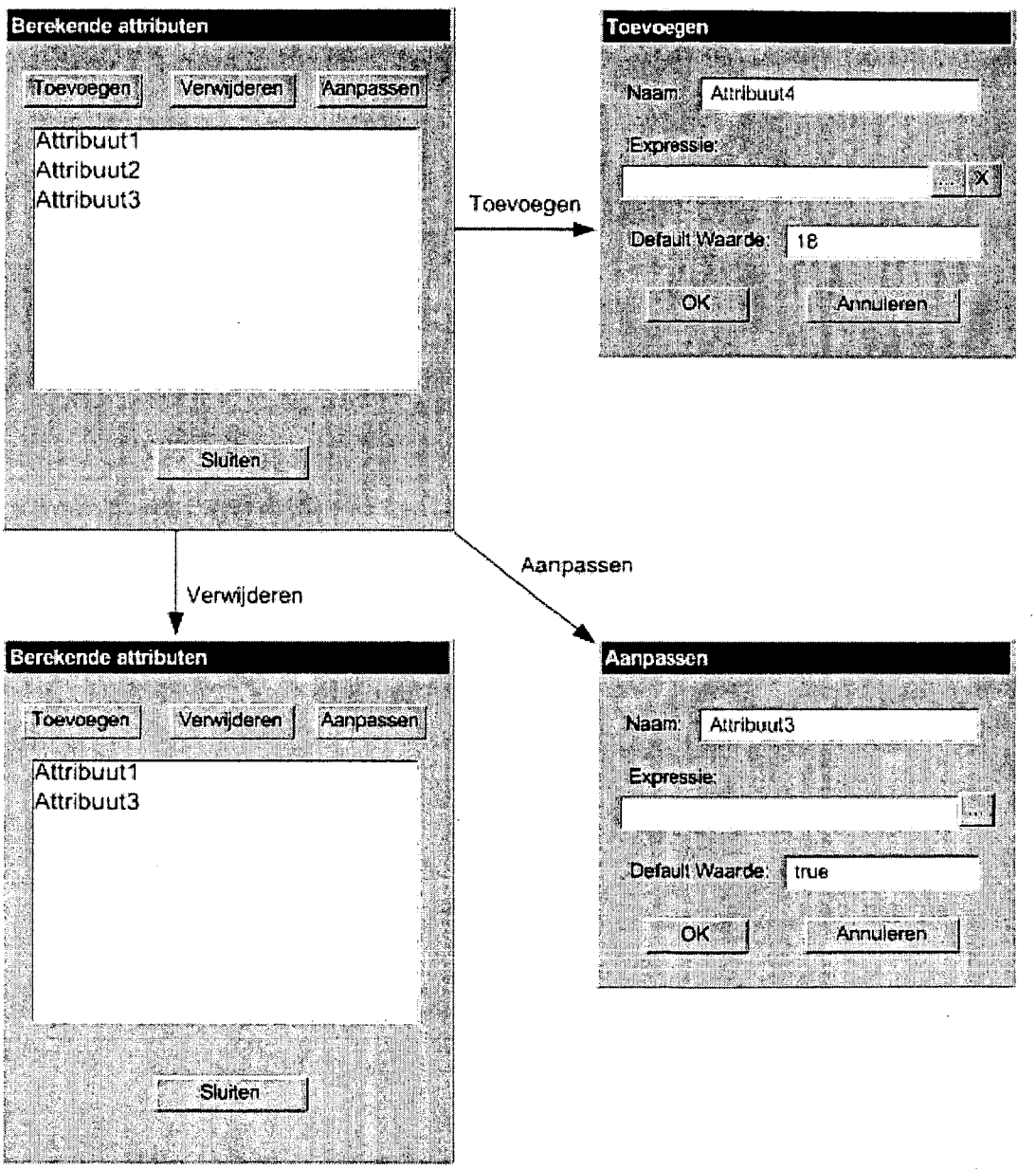
- Knoppen voor het toevoegen, verwijderen en aanpassen van berekende attributen. Indien onderstaand lijstje leeg is zijn de knoppen 'Verwijderen' en 'Aanpassen' niet actief (disabled).
- Een lijstje van berekende attributen
- Een knop om het venster weer te sluiten

De knop 'Toevoegen' leidt naar een scherm met de volgende onderdelen:

- Naam: een tekstveld met daarin de naam van het berekend attribuut
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knoppen voor het oproepen van de expressie editor (...) en voor het leegmaken van het veld (X)
- Default Waarde: indien het betreffende attribuut op een bepaald moment geen waarde heeft, zal de default waarde gebruikt worden.
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Aanpassen' leidt naar een scherm met de volgende onderdelen:

- Naam: de naam van het attribuut.
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knop voor het oproepen van de expressie editor (...) (zie ook de sectie 'Opmerkingen')
- Default Waarde: indien het betreffende attribuut op een bepaald moment geen waarde heeft, zal de default waarde gebruikt worden.
- Knoppen voor 'OK' en 'Annuleren'



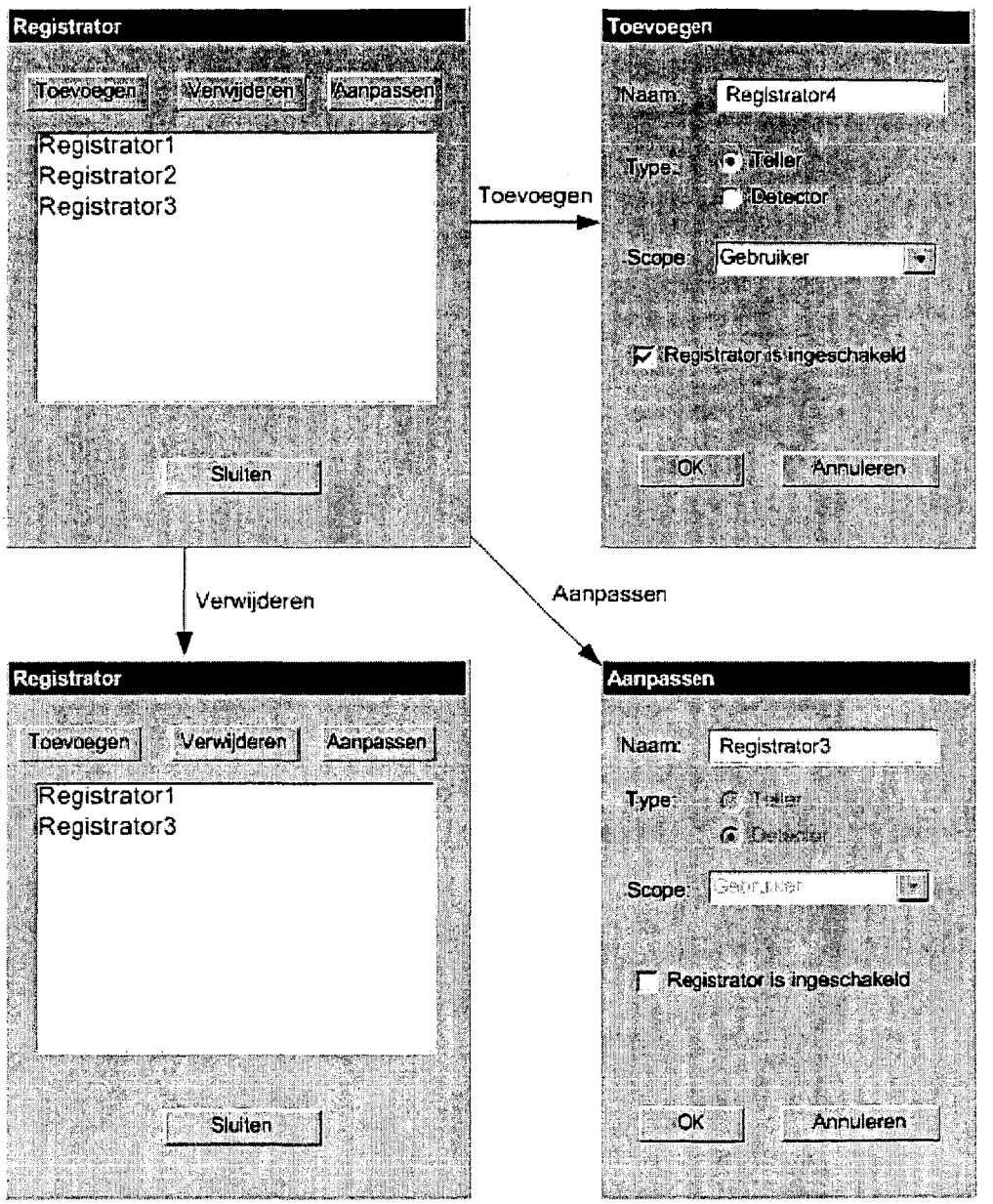
De knop 'Verwijderen' verwijdert het berekend attribuut. Er wordt echter eerst gecontroleerd of het betreffende attribuut niet elders in gebruik is (in een expressie,

danwel in een tag). Het attribuut zal slechts dan verwijderd kunnen worden als het niet in gebruik is.

16.1.1 Opmerkingen

Bij het aanpassen van het berekende attribuut is de mogelijkheid om het expressie-veld leeg te maken niet aanwezig. De reden hiervoor is dat het berekend attribuut al is aangemaakt en er dus een geldige expressie behoort te zijn. Het is dus niet gewenst dat deze expressie verwijderd kan worden.

16.2 Registrators



Registrator-functionaliteit is toegankelijk via het 'Eigenschappen'-venster van links, content-objecten of pagina's. In het geval van links en content-objecten kan dit venster benaderd worden door op de betreffende componenten te dubbelklikken.
Het 'Eigenschappen'-venster van een pagina is onderdeel van het scherm zelf. Alle 'Eigenschappen'-vensers bevatten een knop die leidt naar het hoofdvenster van de registrator-editor. Dat hoofdvenster heeft de volgende onderdelen:

- Knoppen voor het toevoegen, verwijderen en aanpassen van registrators. Indien onderstaand lijstje leeg is zijn de knoppen 'Verwijderen' en 'Aanpassen' niet actief (disabled).
- Een lijstje van registrators
- Een knop om het venster weer te sluiten

De knop 'Toevoegen' leidt naar een scherm met de volgende onderdelen:

- Naam: een tekstveld met daarin een benaming voor de registrator, zoals deze in de lijst komt te staan. Dit is tevens de naam van het corresponderende registrator-attribuut.
- Type: 2 radio-buttons voor het selecteren van het gewenste type registrator. De mogelijkheden zijn:
    - Teller: een teller houdt bij *hoe vaak* een pagina is bekeken of een link is aangeklikt
    - Detector: een detector houdt bij *of* een pagina is bekeken of een link is aangeklikt
- Scope: een dropdown waaruit de gewenste scope geselecteerd kan worden. De mogelijke scopes zijn:
    - Markt
    - Werkgroep
    - Gebruiker
    - Sessie
- Een checkbox voor het in- danwel uitschakelen van de registrator. Indien de registrator uitgeschakeld is (checkbox is niet aangevinkt) wordt er niks geregistreerd door de registrator. De registrator wordt echter niet verwijderd. Een registrator verwijderen kan alleen in het hoofdvenster.
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Aanpassen' leidt naar een scherm met de volgende onderdelen:

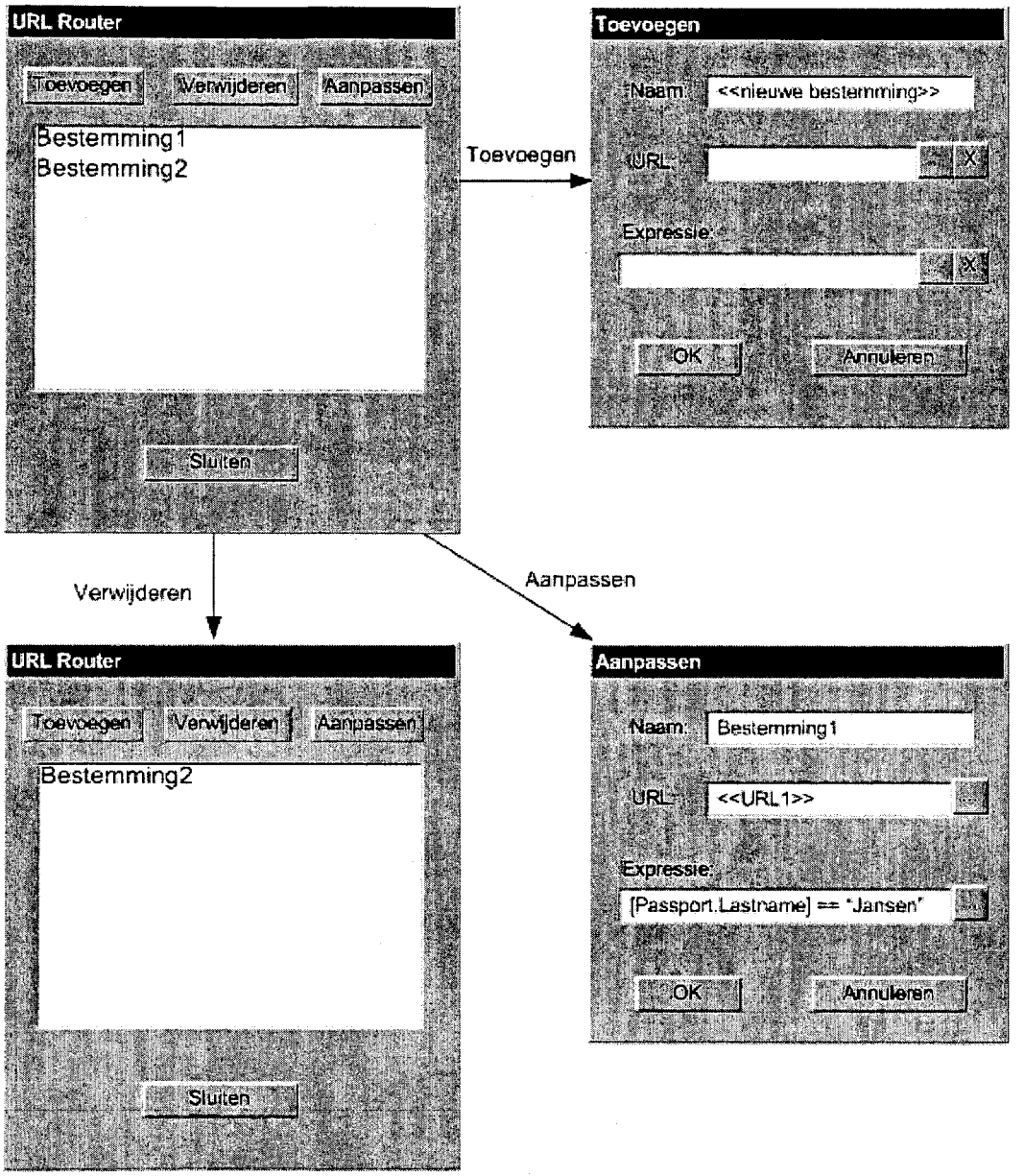
- Naam: de benaming voor de registrator, zoals deze in de lijst staat.
- Type: 2 radio-buttons die het type registrator aangeven. Deze keuze kan niet worden aangepast (de knoppen zijn 'disabled').
- Scope: de scope van de registrator. De scope kan niet worden aangepast.
- Een checkbox voor het in- danwel uitschakelen van de registrator.
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Verwijderen' verwijdert de registrator. Daarmee wordt tevens het registrator-attribuut verwijderd. Een registrator kan daarom slechts dan verwijderd worden als het corresponderende registrator-attribuut niet in gebruik is (bijvoorbeeld in een tag of een expressie).

16.2.1 Opmerkingen

Van een bestaande registrator kan het type en de scope niet worden aangepast. De reden hiervoor is, dat een bestaande registrator al de nodige informatie kan hebben vergaard. Het aanpassen van type of scope kan in zo'n geval ongewenste gevolgen hebben.
Een scope veranderen van werkgroep naar gebruiker bijvoorbeeld kan tot gevolg hebben dat werkgroep-informatie onterecht wordt toegepast op een gebruiker ofwel dat werkgroep-informatie ongewild verloren gaat.
Wil een beheerder toch een scope of type aanpassen, dan is deze genoodzaakt om de oude registrator te verwijderen en een nieuwe aan te maken. Dit omvat echter een geringe hoeveelheid extra inspanning voor de beheerder en wordt dus als acceptabel beschouwd.

16.3 URL Routers



De menubalk bevat al een knop 'link' om een hyperlink toe te voegen. Het is dus vrij voor de hand liggend om ook de URL Router functionaliteit onder die knop te plaatsen. In plaats van een knop kan er een dropdown gebruikt worden die de keuze biedt tussen een 'gewone' link of een URL Router. De laatste optie leidt dan naar het hoofdvenster van de URL Router. Dat hoofdvenster bevat de volgende onderdelen:

- Knoppen voor het toevoegen, verwijderen en aanpassen van bestemmingen. Indien onderstaand lijstje leeg is zijn de knoppen 'Verwijderen' en 'Aanpassen' niet actief (disabled).
- Een lijstje van bestemmingen: de locaties waar naartoe de link kan verwijzen.
- Een knop om het venster weer te sluiten

De knop 'Toevoegen' leidt naar een scherm met de volgende onderdelen:

- Naam: een tekstveld met daarin een benaming voor de bestemming, zoals deze in de lijst komt te staan.
- Link: een 'finder-veld' voor het selecteren van een bestemming
- Knoppen voor het oproepen van de content-object editor (...) en voor het leegmaken van het veld (X)
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knoppen voor het oproepen van de expressie editor (...) en voor het leegmaken van het veld (X)
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Aanpassen' leidt naar een scherm met de volgende onderdelen:

- Naam: de benaming voor de bestemming, zoals deze in de lijst staat. Deze benaming is niet aan te passen.
- Link: een 'finder-veld' voor het selecteren van een bestemming
- Knoppen voor het oproepen van de content-object editor (...)
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knoppen voor het oproepen van de expressie editor (...)
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Verwijderen' verwijdert de 'bestemming' uit de lijst. Dat houdt tevens in dat de achterliggende expressie en de URL-verwijzing worden verwijderd.

16.3.1 Opmerkingen

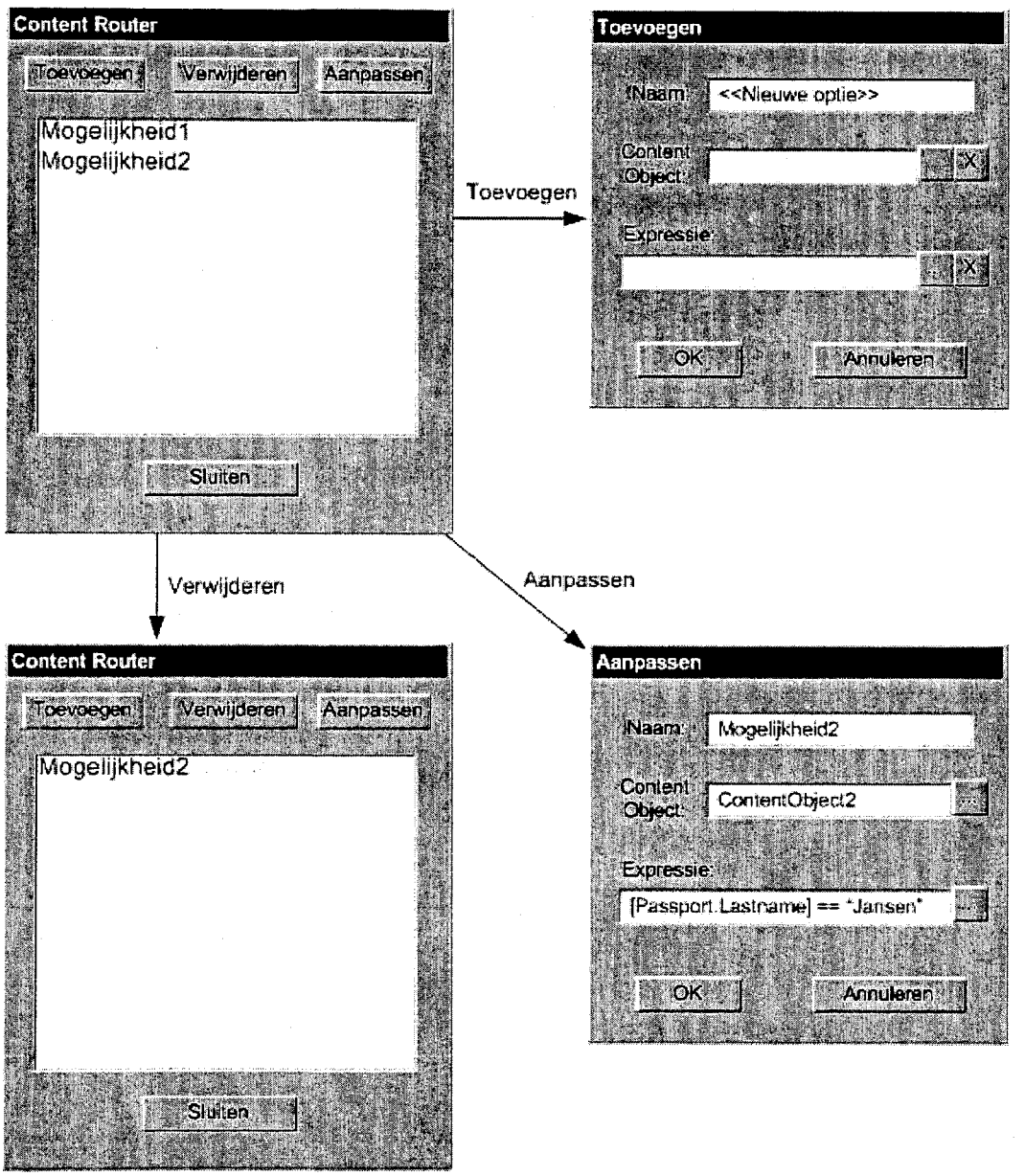
Net als bij de content router geldt ook hier dat, bij het aanpassen van de bestemming, de mogelijkheid om het expressie-veld of het content-object-veld leeg te maken niet aanwezig is. Hiervoor geldt exact dezelfde reden als voor de content router.

16.4 Content Routers

Een content-router kan worden toegevoegd via een optie onder de knop 'invoegen'. Via deze knop kunnen namelijk content-objecten aan pagina worden toegevoegd. Aangezien de router zelf ook een content-object is, is dit de meest logische plaats om deze

functionaliteit toegankelijk te maken. Deze optie leidt naar het hoofdvenster van de content-router-editor. Dat hoofdvenster bevat de volgende onderdelen:

- Knoppen voor het toevoegen, verwijderen en aanpassen van de content-objecten die in de placeholder kunnen worden geplaatst. Indien onderstaand lijstje leeg is zijn de knoppen 'Verwijderen' en 'Aanpassen' niet actief (disabled).
- Een lijstje van content-objecten: de mogelijke invullingen van de placeholder
- Een knop om het venster weer te sluiten



De knop 'Toevoegen' leidt naar een scherm met de volgende onderdelen:

- Naam: een tekstveld met daarin een benaming voor het content-object, zoals deze in de lijst komt te staan.
- Content Object: een 'finder-veld' voor het selecteren van een content-object
- Knoppen voor het oproepen van de content-object editor (...) en voor het leegmaken van het veld (X)
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knoppen voor het oproepen van de expressie editor (...) en voor het leegmaken van het veld (X)
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Aanpassen' leidt naar een scherm met de volgende onderdelen:

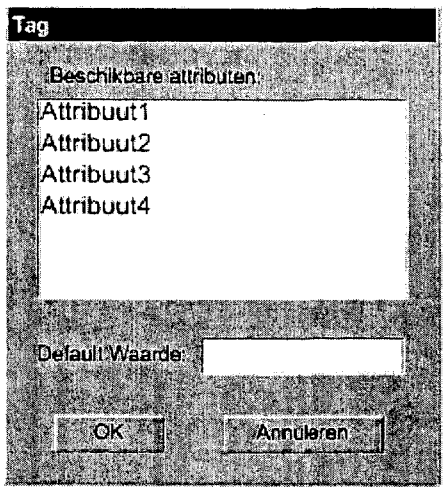
- Naam: de benaming voor het content-object, zoals deze in de lijst staat. Deze benaming is niet aan te passen.
- Content Object: een 'finder-veld' voor het selecteren van een content-object
- Knoppen voor het oproepen van de content-object editor (...)
- Expressie: een 'finder-veld' voor het samenstellen van een expressie
- Knoppen voor het oproepen van de expressie editor (...)
- Knoppen voor 'OK' en 'Annuleren'

De knop 'Verwijderen' verwijdert de 'bestemming' uit de lijst. Dat houdt tevens in dat de achterliggende expressie en het content-object worden verwijderd.

16.4.1 Opmerkingen

Net als bij berekende attributen is, bij het aanpassen van de bestemming, de mogelijkheid om het expressie-veld leeg te maken niet aanwezig. Hetzelfde geldt voor het content-object-veld. De reden hiervoor is dezelfde als bij berekende attributen: de router is al aangemaakt en het is dus niet gewenst dat een content-object of expressie verwijderd kan worden.

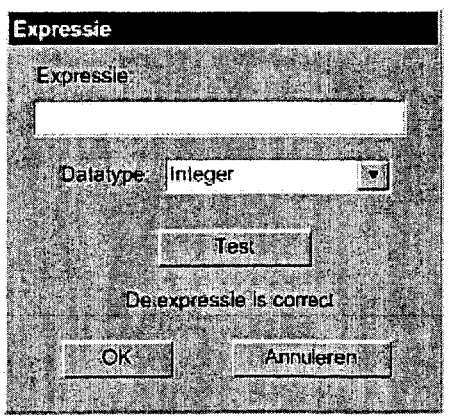
16.5 Tags



Een tag kan worden toegevoegd via een optie in de menubalk vergelijkbaar met de mogelijkheid om een hyperlink toe te voegen.
Het venster van de tag-editor bevat de volgende onderdelen:

- Een lijstje van beschikbare attributen, waaruit gekozen kan worden.
- Default Waarde: indien het geselecteerde attribuut op een bepaald moment geen waarde heeft, zal de default waarde gebruikt worden.
- Knoppen voor 'OK' en 'Annuleren'

16.6 Expressies

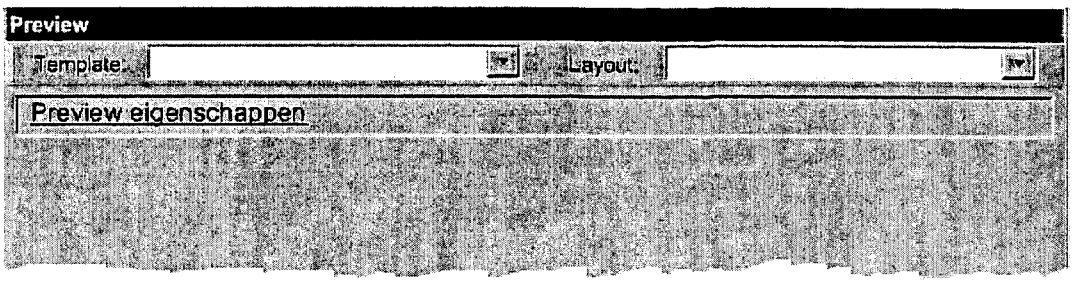


Het expressie-venster bevat de volgende onderdelen:

- Expressie: een tekstveld waar de expressie kan worden ingevuld
- Datatype: iedere expressie geeft een waarde terug. Het datatype van deze 'return value' wordt aangegeven via deze dropdown
- Een testknop om de expressie te evalueren op syntactische correctheid en te controleren of de return value van het opgegeven datatype is

- Een label dat aangeeft of de geteste expressie correct is
- Knoppen voor 'OK' en 'Annuleren'

16.7 Previews



Het preview-venster bestond al. Aan het venster zijn echter de volgende onderdelen toegevoegd:

- Template: via deze dropdown kan een specifieke template gekozen worden op basis waarvan de preview wordt samengesteld.
- Layout: via deze dropdown kan een specifieke layout gekozen worden op basis waarvan de preview wordt samengesteld.

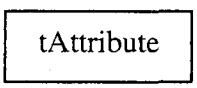
Preview eigenschappen: deze link opent een soort 'onderwaterscherm' waarin afzonderlijke attribuutwaarden handmatig kunnen worden aangepast.

17 Bijlage 4: ER-Diagram

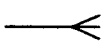
Het ER-diagram in deze bijlage (zie verderop) geeft een overzicht van de verschillende datatabellen en de onderlinge relaties. Per tabel zal kort de functie en de relatie(s) met andere tabellen besproken worden, maar eerst zal er een korte toelichting worden gegeven op de notaties in het schema:

17.1 Toelichting

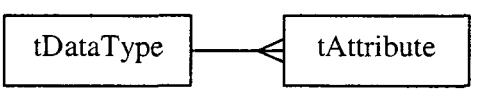
Iedere rechthoek in het schema representeert een datatabel. De tabel tAttribute wordt dus als volgt weergegeven:



Daarnaast lopen er lijnen tussen de verschillende datatabellen. Deze lijnen geven de relaties aan tussen de verschillende datatabellen. De meest voorkomende relatie is:



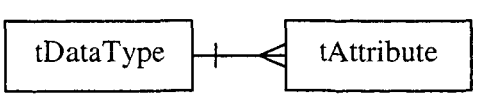
Dit geeft een 1-op-veel-relatie aan. Een 1-op-veel-relatie tussen de tabellen tDataType en tAttribute ziet er dus als volgt uit:



Dit geeft aan dat een attribuut hooguit 1 datatype kan hebben, maar dat een datatype wel bij meerdere attributen kan horen (verschillende attributen kunnen hetzelfde datatype hebben).

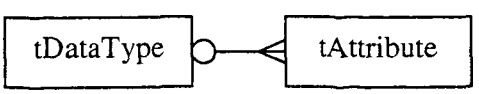
Daarnaast zijn er de 'rondjes' en 'streepjes'.

Een streepje staat voor 'verplicht' en een rondje voor 'optioneel'. Een uitbreiding van het vorige voorbeeld levert het volgende resultaat op:



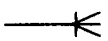
In dit voorbeeld heeft ieder attribuut dus altijd *precies* 1 datatype.

Had een attribuut ook *geen* datatype kunnen hebben, dan had het voorbeeld er als volgt uitgezien:

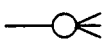


In dit geval heeft een attribuut *hooguit* 1 datatype. Dit geeft dus aan dat het ook mogelijk is dat een attribuut *geen* datatype heeft.

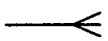
Naar analogie van het vorige geeft



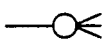
aan dat we te maken hebben met minimaal 1 mogelijkheid (1 of meer) en geeft

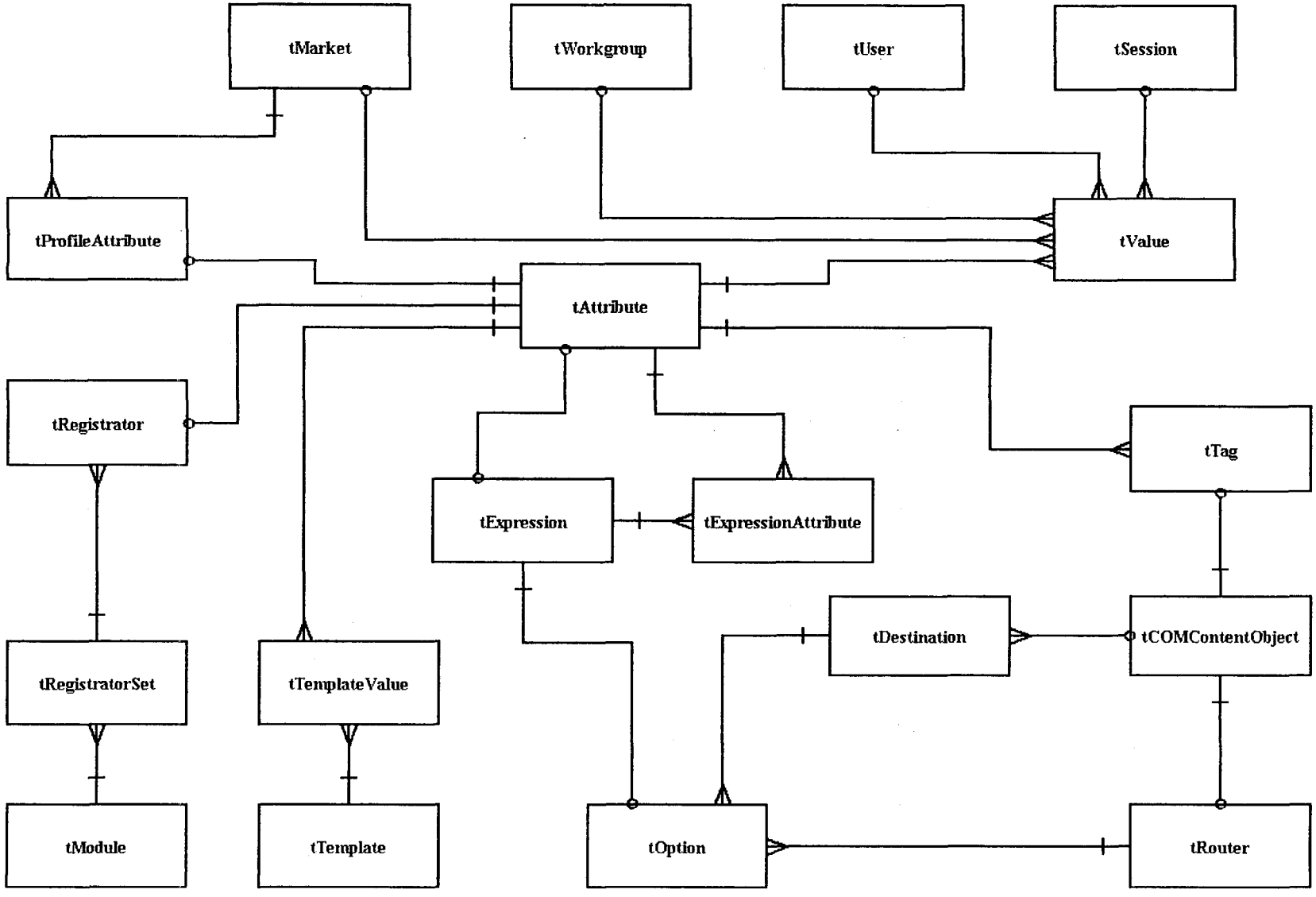


aan dat er sprake is van 0 of meer mogelijkheden. Bij deze laatste mogelijkheid wordt vaak het rondje weggelaten:



is in dit geval dus hetzelfde als




tMarket
tWorkgroup
tUser
tSession
tProfileAttribute
tValue
tAttribute
tRegistrator
tTag
tExpression
tExpressionAttribute
tDestination
tCOMContentObject
tRegistratorSet
tTemplateValue
tModule
tTemplate
tOption
tRouter

17.2 Overzicht Datatabellen

Nu er meer duidelijkheid is gegeven over de notaties in het schema, worden hieronder alle datatabellen uit het schema kort toegelicht.

tAttribute

Deze tabel bevat alle attributen die binnen het systeem beschikbaar zijn. Aan de prominente plaats in het diagram valt al af te leiden dat 'attribuut' het sleutelbegrip is binnen de e-business-functionaliteit.

tValue

Ieder attribuut heeft een waarde. De waarde kan echter afhankelijk zijn van de scope. Een attribuut met scope User bijvoorbeeld heeft per gebruiker een andere waarde. Daarom is er een koppeling tussen o.a. gebruikers (tUser) en attributen (tAttribute) in de vorm van de tabel tValue.

tMarket

Deze tabel bevat gegevens over de markten en bepaalt, in combinatie met tValue, de waarden van attributen uit tAttribute.

tWorkgroup

Deze tabel bevat gegevens over de werkgroepen en bepaalt, in combinatie met tValue, de waarden van attributen uit tAttribute.

tUser

Deze tabel bevat gegevens over de gebruikers en bepaalt, in combinatie met tValue, de waarden van attributen uit tAttribute.

tSession

Deze tabel bevat sessiegegevens en bepaalt, in combinatie met tValue, de waarden van attributen uit tAttribute.

De bovengenoemde vier tabellen in combinatie met de tabel tValue leveren dus de eigenlijke waarde van een attribuut op.
Een alternatief zou zijn geweest om voor iedere scope een aparte tabel aan te maken. In plaats van één tValue tabel, zou er de beschikking zijn over een tSessionValue tabel, een tUserValue tabel, enz.
Er is gekozen voor één tabel, omdat het opvragen van waarden daardoor eenvoudiger wordt. Om bijvoorbeeld een overzicht te creëren van alle waarden, hoeft er nu namelijk

maar één tabel uitgevraagd te worden. Bovendien is er op deze manier slechts één tabel in plaats van vier nodig.

tProfileAttribute

Profiel-attributen zijn alle attributen die tot een gebruikersprofiel behoren. Deze groep attributen is dus een sub-groep van de verzameling attributen, vandaar de koppeling van deze tabel met tAttribute.

tRegistrator

Deze tabel bevat alle registrators van het systeem. Iedere registrator heeft een corresponderend registrator-attribuut.

tRegistratorSet

In de registrator-editor kunnen meerdere registrators gedefinieerd worden. Al deze registrators samen behoren tot een registratorset. Alle componenten waaraan registrators verbonden kunnen worden, hebben zo'n registratorset. Deze registratorset kan ook leeg zijn indien er geen registrators voor het betreffende component zijn gedefinieerd.

tModule

Registrators worden altijd verbonden aan componenten die door een bepaalde module worden aangeleverd. Vandaar dat er ook een koppeling is tussen deze tabel en de tabel tRegistratorset.

tRouter

Deze tabel bevat alle routers van het systeem. Een router kan naar verschillende bestemmingen wijzen (waaronder content-objecten). Deze mogelijke bestemmingen worden in dit geval aangeduid als opties.

tCOMContentObject

Aangezien een content-router naar content-objecten 'routeert', is er een koppeling tussen de tRouter tabel en de tabel tCOMContentObject. Dit is een reeds bestaande tabel waarin alle content-objecten zijn opgeslagen.

tOption

Zoals gezegd heeft een router één of meerdere opties. Een optie is ofwel een link, ofwel een (verwijzing naar) een content-object. Deze opties zijn in de tabel tOption opgeslagen. De tabel tOption is eigenlijk de verbindende link tussen de router, de bestemmingen en de bijbehorende expressies.

tDestination

tDestination verbindt een optie (uit de tabel tOption) met een content-object (uit de tabel tCOMContentObject) mits er sprake is van een content router. Is er sprake van een URL router, dan bevat tDestination de URL van de bestemming.

tExpression

Expressies zijn vooral van belang voor routers en voor berekende attributen. Vandaar ook dat er een koppeling is met tOption (voor de routers) en met tAttribute (voor de berekende attributen).

tExpressionAttribute

Een expressie zelf is deels opgebouwd uit attributen. Via deze tabel worden de expressies en de bijbehorende attributen aan elkaar gekoppeld.

tTag

Aangezien tags niet veel meer zijn dan verwijzingen naar attributen is het logisch dat er een koppeling is tussen deze tabel en tAttribute. Omdat tags altijd worden ingevoegd in de content en dus eigenlijk content-objecten zijn, is er ook een koppeling met de tabel tCOMContentObject.

tTemplate

Een template is een verzameling attributen met een bepaalde waarde. Zo'n template kan gebruikt worden voor het genereren van previews. Alle templates worden in deze tabel opgeslagen.

tTemplateValue

De waarden die de verschillende attributen in een template hebben, worden in deze tabel opgeslagen.

17.3 Invulling Datatabellen

In deze sectie wordt er per component beschreven wat de precieze invulling van de tabellen is.

17.3.1 Attributen

tAttribute
nAttributeID
nExpressionID
nDatatype
nScope
sAttributeType
sName

De tabel tAttribute heeft de volgende records:

- nAttributeID: Bevat een uniek ID voor het attribuut
- nExpressionID: Indien het attribuut een berekend attribuut is, bevat deze record het ID van de corresponderende expressie
- nDataType: Een enumerator voor het datatype van het attribuut
- nScope: Een enumerator voor de scope van het attribuut
- nAttributeType: Een enumerator voor de type van het attribuut (registrator, berekend, enz.)
- sName: De benaming van het attribuut

tValue
nValueID
nAttributeID
nSessionID
nUserID
nWorkgroupID
nMarketID
sValue

De tabel tValue heeft de volgende records:

- nValueID: Bevat een uniek ID voor de waarde
- nAttributeID: Bevat de ID van het attribuut waarbij de waarde behoort
- nSessionID t/m nMarketID: Afhankelijk van de scope van het attribuut bevat één van deze records het ID van de bijbehorende cookie, sessie, gebruiker, enz. De overige records bevatten de waarde NULL
- sValue: Een string met daarin de eigenlijke waarde van het attribuut

17.3.2 Registrators

tRegistrator
nRegistratorID
nRegistratorType
nRegistratorSetID
nAttributeID
bActive
sName

De tabel tRegistrator heeft de volgende records:

- nRegistratorID: Bevat een uniek ID voor de registrator
- nRegistratorType: Een enumerator voor het type van de registrator
- nRegistratorSetID: Bevat de ID van de registratorset waartoe de registrator behoort
- nAttributeID: Bevat de ID van het corresponderende attribuut
- bActive: Bevat een boolean waarde welke aangeeft of de registrator geactiveerd is
- sName: De benaming van de registrator

tRegistratorSet
nRegistratorSetID
nModuleID
sName

De tabel tRegistratorSet heeft de volgende records:

- nRegistratorSetID: Bevat een uniek ID voor de registratorset
- nModuleID: Bevat de ID van de module die het component levert waaraan de registrator is verbonden
- sName: Een 'vriendelijke' benaming voor de registratorset

17.3.3 Routers

tRouter
nRouterID
nContentObjectID
nRouterType

De tabel tExpression heeft de volgende records:

- nRouterID: Bevat een uniek ID voor de router
- nContentObjectID: Een router is zelf ook een content object. De bijbehorende ID wordt in deze record opgeslagen
- nRouterType: Een enumerator voor het type van de router (content- of URL-router)

tOption
nOptionID
nRouterID
nExpressionID
nDestinationID
nOrder
sName

De tabel tOption bevat opties. Een optie is in dit verband een expression-destination-pair. Het geeft aan welke "bestemmingen" een router kan hebben en welke expressies daarbij horen. De tabel bevat de volgende records:

- nOptionID: Bevat een uniek ID voor de optie
- nRouterID: Bevat de ID van de router waar de optie deel van uitmaakt
- nExpressionID en nDestinationID: Zoals gezegd bestaat een optie uit een expressie en een bestemming. Deze twee records bevatten de ID's van respectievelijk de expressie en de bestemming
- nOrder: Aangezien de opties in een bepaalde volgorde moeten worden doorlopen, bepaalt deze record welk "volgnummer" een optie heeft: een hogere waarde voor nOrder houdt in dat de bijbehorende optie lager in de lijst staat.
- sName: Een 'vriendelijke' naam om de optie mee aan te duiden.

tDestination
nDestinationID
nContentObjectID
sDestinationURL

Een "bestemming" (destination) geeft aan waarna de router verwijst. In principe zijn er twee mogelijkheden: een ander content object of een URL. In dit geval ligt de focus op de content router en dus ook op het contect object. De tabel tDestination heeft de volgende records:

- nDestinationID: Bevat een uniek ID voor de bestemming
- nContentObjectID: Bevat het ID van het content object waarna de bijbehorende router verwijst.
- sDestinationURL: Indien er sprake is van een URL router wordt in dit record de URL van de bestemming opgeslagen. Bij afwezigheid van URL router functionaliteit wordt hier momenteel nog geen gebruik van gemaakt.

17.3.4 Tags

tTag
nTagID

nAttributeID
nContentObjectID
sDefaultValue

De tabel tTag heeft de volgende records:

- nTagID: Bevat een uniek ID voor de tag
- nAttributeID: Bevat de ID van het attribuut waarmee de tag correspondeert
- nContentObjectID: Een tag is ook een content object. De bijbehorende ID wordt in deze record opgeslagen
- sDefaultValue: Bevat de default waarde van de tag

17.3.5 Expressies

tExpression
nExpressionID
sExpression
nExprDataType
bCorrect

De tabel tExpression heeft de volgende records:

- nExpressionID: Bevat een uniek ID voor de expressie
- sExpression: Een string met daarin de expressie zelf
- nExprDataType: Een enumerator voor het datatype van de expressie. Het resultaat van de evaluatie van de expressie heeft het betreffende datatype.
- bCorrect: Geeft aan of de expressie syntactisch correct is. Een incorrecte expressie mag wel opgeslagen worden, maar zal geen corresponderende records in de tabel tExpressionAttribute (zie hieronder) bevatten.

tExpressionAttribute
nExpressionAttributeID
nAttributeID
nExpressionID

De tabel tExpressionAttribute heeft als functie attributen te koppelen aan expressies. Heeft geeft aan welke attributen in een bepaalde expressie voorkomen. De records zijn:

- nExpressionAttributeID: Bevat een uniek ID voor de tabel
- nAttributeID: Bevat de ID van het attribuut dat in de expressie voorkomt
- nExpressionID: Bevat de ID van de expressie waarvan het attribuut deel uitmaakt

17.3.6 Preview Templates

tTemplate
nTemplateID
sName

De tabel tTemplate heeft de volgende records:

- nTemplateID: Bevat een uniek ID voor de template
- sName: Een 'vriendelijke' benaming voor de template

tTemplateValue
nTemplateValueID
nTemplateID
nAttributeID

De tabel tTemplateValue heeft de volgende records:

- nTemplateValueID: Bevat een uniek ID voor de tabel
- nTemplateID: Bevat de ID van de template waartoe het attribuut behoort
- nAttributeID: Bevat de ID van het attribuut dat deel uitmaakt van de template

18 Bijlage 5: Classes

Deze bijlage behandelt alle classes die tijdens de implementatiefase zijn toegevoegd aan de reeds bestaande kernel code.

18.1 Attributen

clsAttributeBuilder

Deze class levert de mogelijkheid om attributen te creëren en te verwijderen.

Property/Method	Toelichting
Public	
Function nAddAttribute(ByVal v_nExpressionID As Integer, ByVal v_sName As String, ByVal v_eAttributeType As enumAttributeType, ByVal v_eDataType As enumDataType, ByVal v_eScope As enumScope) As Integer	Creëert een nieuw attribuut en voegt deze toe aan de tabel tAttribute in de Kernel database. De teruggegeven waarde bevat de ID van het nieuw aangemaakte attribuut.
Function nRemoveAttribute(ByVal v_nAttributeID As Integer) As Integer	Verwijdert het attribuut met betreffend ID uit de Kernel database.

clsAttribute

Deze class creëert een attributen object.

Property/Method	Toelichting
Public	
Get nAttributeID() As Integer	ID van het attribuut
Get sValue() As String	De waarde van het attribuut
Get eScope() As enumScope	De scope van het attribuut
Get eAttributeType() As enumAttributeType	Het type van het attribuut. Het gaat hier om het soort attribuut (paspoort, registrator, ...)
Get sName() As String	De naam van het attribuut
Get eDataType() As enumDataType	Het datatype van (de waarde van) het attribuut
Sub Create(ByVal v_nAttribID As Integer)	Instantieert het attribuut met de data uit de tabel tAttribute uit de Kernel database
Private	
Function sGetValue() As String	Bepaalt en retourneert de waarde van het attribuut
Function sGetSystemValue() As String	Bepaalt en retourneert de waarde van een

	systeem-attribuut
Function sGetCalculatedValue() As String	Bepaalt en retourneert de waarde van een berekend attribuut

18.2 Routers

clsRouterBuilder

Deze class levert de mogelijkheid om routers te creëren en te verwijderen.

Property/Method	Toelichting
Public	
Function nAddRouter(ByVal v_eRouterType As enumRouterType, ByVal v_nContentObjectID As Integer) As Integer	Creëert een nieuwe router en voegt deze toe aan de tabel tRouter in de Kernel database. De teruggegeven waarde bevat de ID van de nieuw aangemaakte router.
Sub RemoveRouter(ByVal v_nRouterID As Integer)	Verwijdert de router met betreffend ID uit de Kernel database

clsRouter

Deze class creëert een router object.

Property/Method	Toelichting
Public	
Sub Create(ByVal v_nRouterID As Integer)	Instantieert de router met de data uit de tabel tRouter uit de Kernel database
Function nRoute() As Integer	Verzorgt de daadwerkelijke router functionaliteit. Dit omvat het evalueren van de expressies, het bepalen van de geschikte 'bestemming' en het retourneren van de ID van de bestemming (uit tDestination)
Private	
Function nRouteContent(ByVal v_nDestinationID As Integer) As Integer	Verzorgt de router functionaliteit specifiek voor content routers.

clsOptionBuilder

Deze class levert de mogelijkheid om opties te creëren, aan te passen en te verwijderen. Deze opties maken deel uit van een router.

Property/Method	Toelichting
Public	
Function nAddOption(ByVal v_nRouterID As Integer, ByVal v_nExpressionID As	Creëert een nieuwe optie en voegt deze toe aan de tabel tOption in de Kernel database.

Integer, ByVal v_nDestiantionID As Integer, ByVal v_nOrder As Integer, ByVal v_sName As String) As Integer	De teruggegeven waarde bevat de ID van de nieuw aangemaakte optie.
Sub EditOption(ByVal v_nOptionID As Integer, ByVal v_sName As String)	Verwerkt de aanpassingen aan de optie in de tabel tOption van de Kernel database.
Sub RemoveOption(ByVal v_nOptionID As Integer)	Verwijdert de optie met betreffend ID uit de Kernel database

18.3 Expressies

clsExpressionBuilder

Deze class levert de mogelijkheid om expressies te creëren, aan te passen en te verwijderen.

Property/Method	**Toelichting**
Public	
Function nAddExpression(ByVal v_sExpression As String, ByVal v_eExprDataType As enumDataType) As Integer	Creëert een nieuwe expressie en voegt deze toe aan de tabel tExpression in de Kernel database. De teruggegeven waarde bevat de ID van de nieuw aangemaakte expressie.
Sub EditExpression(ByVal v_nExpressionID As Integer, ByVal v_sNewExpression As String)	Verwerkt de aanpassingen aan de expressie in de tabel tExpression van de Kernel database.
Function nRemoveExpression(ByVal v_nExpressionID As Integer) As Integer	Verwijdert de expressie met betreffend ID uit de Kernel database
Function bTestExpression(ByVal v_sExpression As String, ByVal v_eExprDataType As enumDataType) As Boolean	Test de expressie op syntactische correctheid en retourneert een boolean waarde welke aangeeft of de expressie correct is (True) of niet (False)

clsExpression

Deze class creëert een expressie object.

Property/Method	**Toelichting**
Public	
Get bCorrect() As Boolean	Geeft aan of de expressie syntactisch correct is.
Get eExprDataType() As enumDataType	Het datatype van de expressie. Dit geeft aan wat het datatype van het resultaat van de expressie moet zijn.
Get sExpression() As String	De expressie zelf
Get nExpressionID() As Integer	ID van de expressie
Sub Create(ByVal v_nExpressionID As	Instantieert de expressie met de data uit de

Integer)	tabel tExpression uit de Kernel database
Function sExecute() As String	Evalueert de expressie en retourneert de uitkomst ervan.

clsEvaluator

De eigenlijke functionaliteit voor het evalueren van expressies wordt geleverd door een dll geschreven in C#. Meer informatie over deze dll is te vinden in het hoofdstuk 'Conclusies & Suggesties' van het hoofddocument. Deze class bevat enkel een functie om de functionaliteit uit deze dll aan te spreken en het resultaat terug te geven.

Property/Method	**Toelichting**
Public	
Function sEvaluate(ByVal v_sExpression As String, ByVal v_eDataType As enumDataType) As String	Roept de functionaliteit uit de C# dll aan voor het evalueren van de expressie en retourneert het resultaat.

18.4 Grafische User Interface

Hieronder worden de classes besproken die nodig zijn voor het creëren van de grafische user interface. Deze classes leveren de functionaliteit voor het tonen van de benodigde wizards.

clsPLRouter

Deze class levert de grafische user interface voor de router-wizard.

Property/Method	**Toelichting**
Private	
Function sModalDialogue(ByVal v_sFormName As String, ByVal v_sReturnObject As String) As String	Creëert een dialoogvenster waarin de router-wizard wordt getoond.
Sub HandleRequest(r_eRequestStatus As EnumRequestStatus)	Verzorgt de afhandeling van alle post-request die in het dialoogvenster worden gedaan.
Function eGetRequest() As EnumRequest	Een extra functie om te bepalen welk post-request er precies is gedaan.
Sub HandleRequestShowRouter()	Toont het hoofdscherm van de router
Sub HandleRequestShowOptionFormToAdd(Optional ByVal v_bGotContentType As Boolean = False)	Toont het scherm voor het toevoegen van een optie.
Sub HandleRequestShowOptionFormToEdit()	Toont het scherm voor het aanpassen van een optie.
Sub HandleRequestAddOption()	Levert de functionaliteit om een optie toe te

	voegen door gebruik te maken van clsOptionBuilder
Sub HandleRequestEditOption()	Levert de functionaliteit om een optie aante passen door gebruik te maken van clsOptionBuilder
Sub HandleRequestDeleteOption()	Levert de functionaliteit om een optie te verwijderen door gebruik te maken van clsOptionBuilder
Sub HandleRequestDeleteExpression(ByVal bIsNew As Boolean)	Maakt het 'finder-veld' van de expressie leeg als gevolg van een klik op de knop "X"
Sub HandleRequestDeleteContentObject(ByVal v_bIsNew As Boolean)	Maakt het 'finder-veld' van het content-object leeg als gevolg van een klik op de knop "X"

clsPLExpression

Deze class levert de grafische user interface voor de expressie-editor.

Property/Method	**Toelichting**
Private	
HandleRequest(r_eRequestStatus As EnumRequestStatus)	Soortgelijke methode als de gelijknamige methode uit clsPLRouter
Function eGetRequest() As EnumRequest	Soortgelijke functie als de gelijknamige functie uit clsPLRouter
Sub HandleRequestShowExpression(ByVal v_bIsTest As Boolean, Optional ByVal v_bCorrect As Boolean = False)	Toont de expressie-editor
Sub HandleRequestTestExpression()	Test de expressie op syntactische correctheid door gebruik te maken van clsExpressionBuilder

18.5 ContentHosting

De functionaliteit die de kernel nodig heeft om als provider van content-types op te treden, wordt geleverd door clsPLRouter in combinatie met een aantal andere classes. Deze classes bestonden al in andere provider-modules maar moesten ook worden gemaakt voor de kernel. In de praktijk kon de bestaande code deels worden hergebruikt. Het vereiste echter ook de nodige aanpassingen om de code geschikt te maken voor gebruik met content-routers. Deze classes worden hieronder beschreven.

clsContentServer

Deze class maakt het mogelijk voor de kernel om als provider van content-types te functioneren.

Property/Method	Toelichting
Private	
Sub GetContentTypes(ByRef r_oProviderContentTypes As clsProviderContentTypes)	Geeft de content-types van de kernel terug. Op het moment is dat alleen het type content-router
Function oGetContentObjectDesigner(ByVal v_sContentType As String, ByVal v_oContentProperties As clsContentProperties, ByVal v_nContentObjectID As Long) As IProviderCODesigner	Deze functie levert een object op dat de functionaliteit biedt om een nieuw content-object aan te maken.
Function oGetContentObject(ByVal v_sContentType As String, ByVal v_oContentProperties As clsContentProperties, ByVal v_lWidth As Long, ByVal v_lHeight As Long, Optional ByVal v_sSubLabel As String = "") As IProviderContentObject	Deze functie geeft het eigenlijke content-object terug.
Sub DeleteContentObject(ByVal v_sContentType As String, ByVal v_oContentProperties As clsContentProperties)	Verwijdert een content-object

clsCODesigner

Deze class levert de functionaliteit om de editor voor het content-object te tonen.

Property/Method	Toelichting
Private	
Get sModalEditorString() As String	Deze property levert een string op die de HTML bevat om de editor mee te tonen
<Overig>	Deze class bevat nog meerdere properties, die bepalen hoe de editor eruit komt te zien, maar hebben verder geen invloed op de functionaliteit van het geheel. Ze zullen hier dan ook verder niet behandeld worden.