Onderzoek Templating en i18n

Het kunnen selecteren van verschillende stijlen en opmaken voor de webapplicatie

Afstuderen

Bert Gritter
Rob Juurlink
2004

Laatste wijziging:
maandag 26 april 2004
20:38:22 uur.

Onderzoek Templating en i18n

Het kunnen selecteren van verschillende stijlen en opmaken voor de webapplicatie

Versiebeheer

Datum	auteur	Versie	Status/Wijziging
14-04-2004	Bert	0.1	Start document
19-04-2004	Rob	0.2	Beschrijven internationalisatie en SiteMesh templating
21-04-2004	Bert	0.3	Beschrijving adressering.
23-04-2004	Bert	0.4	Verder documenteren, ontwerp, conclusie.
25-04-2004	Rob	0.5	Beschrijving foutafhandeling
26-04-2004	Bert	0.6	Document afronden
26-04-2004	Rob	1.0	Reviewen en kleine verbeteringen aanbrengen

INHOUDSOPGAVE

1. INLEIDING

De webapplicatie moet door meerdere makelaars gebruikt kunnen worden. Het is de bedoeling dat er verschillende opmaken voor dezelfde applicatie ingesteld kunnen worden. Iedere makelaar heeft z'n eigen logo's en stijl.

Het is niet de bedoeling dat er voor iedere makelaar een nieuwe webapplicatie gemaakt wordt. Alle makelaars werken op hetzelfde systeem, maar het resultaat wordt steeds in de opmaak en stijl van de makelaar weergegeven.

Het kunnen selecteren van een andere stijl en opmaak voor dezelfde applicatie wordt templating genoemd. Hoe dit werkt wordt beschreven in hoofdstuk 2.1 op bladzijde 5.

Naast het kunnen instellen van een andere stijl en opmaak (kortweg een template genoemd) moet per template de taal ingesteld kunnen worden. Het beschikbaar stellen van een applicatie in meerdere talen wordt internationalisatie genoemd. Het correct weergeven van bijvoorbeeld de notatie van getallen en datums, wordt lokalisatie genoemd. Internationalisatie en lokalisatie zijn nauw verwant. Dit wordt allemaal besproken in hoofdstuk 2.2 op bladzijde 10 van dit document.

In het eerstvolgende hoofdstuk “Templating en i18n” wordt eerst onderzocht wat de mogelijkheden zijn en hoe de werking zal zijn in de web applicatie.

Het ontwerp, de testen en de conclusie worden als laatste daarna beschreven in dit verslag.

2. TEMPLATING EN I18N

2.1. TEMPLATING

Eerst een korte omschrijving uit het requirementsdocument:

> De opmaak losgekoppeld van de code. Er moet gekozen kunnen worden uit verschillende sjablonen. De Engelse term is **templating**. De objecten in de database kunnen van verschillende makelaars zijn. Elke makelaar kan z'n eigen sjabloon hebben. Implementeer twee verschillende sjablonen om te demonstreren dat het omschakelen tussen verschillende opmaken werkt.

De vastgoedobjecten van alle makelaars bevinden zich in dezelfde database. Per object is bekend bij welke makelaar deze hoort. Er kan alleen een overzicht worden afgedrukt waarin zich objecten van één makelaar bevinden.

Op deze manier kan het overzicht van vastgoedobjecten naadloos ingepast worden in een bestaande website van een makelaar. Elke makelaar kan een eigen lay-out hebben. Door de lay-out los te koppelen van de inhoud, kan er tussen verschillende opmaken gekozen worden.

De lay-out is niet door de makelaar zelf in te stellen of te wijzigen en wordt gekoppeld aan de hostnaam waarmee een pagina opgevraagd wordt.

Figuur 1, een object uit een overzicht. Een lay-out van een overzicht kan er zo uitzien.

2.1.1. Per makelaar een template

De applicatie weet welke stijl geselecteerd moet worden door naar de hostnaam van de aanroep te kijken. Door naar deze hostnaam te kijken wordt ook bepaald welke objecten afgedrukt mogen worden. (Altijd de objecten van maximaal één makelaar tegelijk)

De opmaak van elk type pagina, zoals het lijstoverzicht of de detailpagina is vastgelegd in een apart bestand. De opmaak wordt beschreven door gebruik te maken van de Velocity templating language. Deze beschrijvingstaal is vergelijkbaar met JSP. De precieze werking van Velocity staat beschreven in het document “Onderzoek Raamwerk”.

De verschillende templates staan geordend in verschillende bestandsmappen.

De werking

In de browser wordt via onderstaande URL een overzicht van objecten opgeroepen.

http://localhost:8080/vastgoedonline/

Figuur 2, overzicht van objecten afgedrukt door de template die gekoppeld is aan de hostnaam "localhost".

Via de hostnaam “localhost” weet de applicatie welke template gebruikt moet worden en welke gegevens van welke makelaar uit de database gehaald moeten worden.

Hieronder nogmaals hetzelfde overzicht, maar nu aangeroepen met een andere hostnaam “10.0.20.4”. Deze pagina werd aangeroepen via de volgende verwijzing.

http://10.0.20.4:8080/vastgoedonline/

Figuur 3, overzicht van objecten afgedruk door de template die gekoppeld is aan de hostnaam "10.0.20.4".

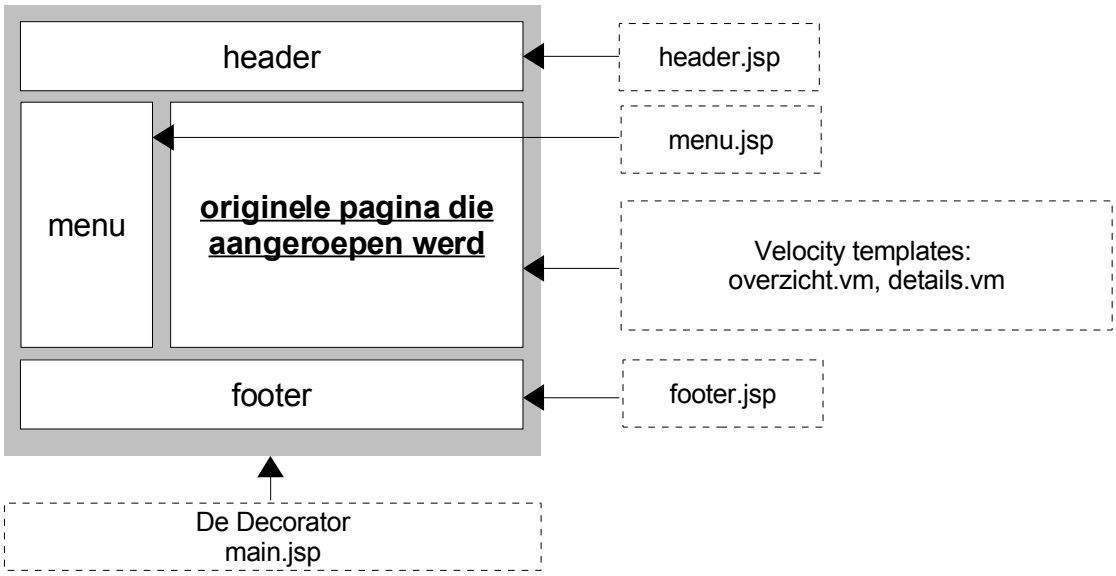
De gegevens gelijk, in de praktijk zal dit niet voorkomen. Het gaat hier alleen om de verschillende templates.

2.1.2. Decorator pattern

In een webapplicatie is het scherm vaak op te delen in verschillende onderdelen. Bijvoorbeeld een header, een footer, een gebied voor het menu en de inhoud.

Op de verschillende pagina's zullen veel onderdelen vaak dezelfde zijn, terwijl alleen de inhoud verandert. In dat geval kan het gebruik van het decorator pattern code besparen en het ontwerp eenvoudiger houden.

Het is vaak handig om de lay-out te laten bepalen door een hoofdtemplate. Deze template kan dan door de gehele applicatie gebruikt worden. In het figuur hieronder is de ontworpen template afgebeeld. Zoals te zien verandert alleen de inhoud van de originele pagina.



Figuur 4, een scherm van de webapplicatie bestaat uit verschillende onderdelen. Tijdens het navigeren verandert alleen het deel met de inhoud (de decorator).

SiteMesh

SiteMesh van OpenSymphony[1] is een implementatie van het Decorator pattern voor web applicaties.

SiteMesh onderschept de aanvraag van elke statische of dynamische HTML pagina doordat het als filter geconfigureerd staat in het `web.xml` configuratie bestand van de webserver. Zie onderstaande code.

Nadat een aanvraag onderschept is, wordt deze geparst. De inhoud van de pagina op de locatie van de originele aanvraag wordt gebruikt in nieuw te genereren pagina. De lay-out van de nieuwe pagina, die Decorator genoemd wordt in SiteMesh, bevat naast de data van de originele aanvraag ook een header, footer en een menu.

De Decorators zijn de pagina's die de originele pagina decoreren nadat SiteMesh deze onderschept heeft.

1 http://www.opensymphony.com/sitemesh

De configuratie van SiteMesh als filter in `web.xml`.

```
<!-- Start of SiteMesh stuff -->
<filter>
    <filter-name>sitemesh</filter-name>
    <filter-class>
        com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>sitemesh</filter-name>
    <url-pattern>*.html</url-pattern>
</filter-mapping>
```

2.1.3. Foutafhandeling

Bij de foutafhandeling wordt er onderscheid gemaakt tussen een niet gevonden pagina (dit levert een HTTP 404 foutmelding op) en een fout bij een verzonden formulier (bijvoorbeeld een verplicht veld dat leeg gelaten is).

Pagina niet gevonden (HTTP code 404)

Dat een pagina niet gevonden kan worden is een situatie die niet mag voorkomen. Doordat de gebruiker in de adres balk van de browser het adres kan veranderen, is het toch mogelijk dat er een niet bestaande pagina opgeroepen wordt.

Als zo'n niet bestaande pagina of niet bestaande actie opgeroepen wordt, stuurt de Servlet in het antwoord een fout code van 404 mee. In het bestand voor de globale instellingen `web.xml` is te definiëren wat er moet gebeuren in het geval er zo'n fout optreedt. In onderstaande fragment wordt ingesteld dat in zo'n geval de pagina genaamd `error.jsp` die in de root van de webapplicatie staat, afgebeeld moet worden.

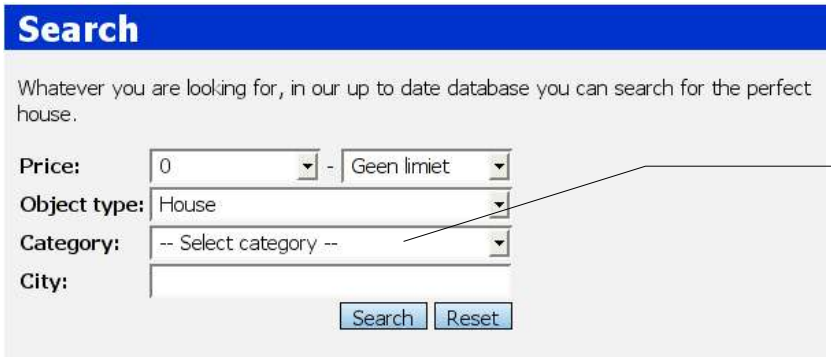
```
<!-- Standaard error pagina -->
<error-page>
  <error-code>404</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>500</error-code>
  <location>/error.jsp</location>
</error-page>
```

De foutpagina wordt niet in de opmaak van de makelaar weergegeven, maar in een eenvoudige opmaak die toont dat er iets fout gegaan is en een aantal mogelijk vervolgstappen voor de gebruiker, zoals een verwijzing naar de index pagina van de makelaar of een zoekpagina.

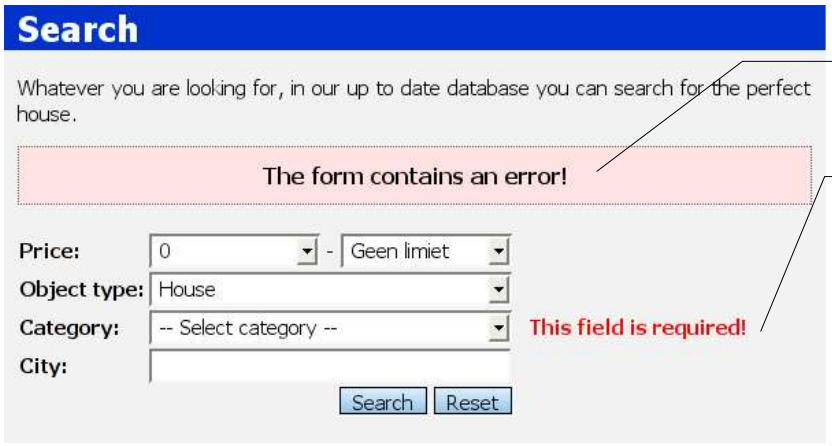
Fouten in het verzonden formulier

Als na het valideren van een verzonden formulier blijkt dat deze fouten bevat, dan wordt het formulier opnieuw getoond met bovenaan in het scherm een globale melding.

Daarnaast komt naast elk veld dat een fout bevat een korte omschrijving van de fout te staan, bijvoorbeeld bij een niet ingevuld verplicht veld, zie figuur 5 en 6 hieronder.



Figuur 5, het niet volledig ingevulde zoekscherm. Categorie is een verplicht veld.



Figuur 6, het verzonden formulier bevatte een fout. Categorie is een verplicht veld.

2.2. INTERNATIONALISATIE

Ook hier eerst een stukje uit het requirementsdocument:

Ondersteuning voor verschillende talen mogelijk maken in het ontwerp (**internationalisatie** en **lokalisatie**). De applicatie hoeft voor het afstudeerproject alleen in het Nederlands. Voeg om te demonstreren dat meerdere talen mogelijk zijn een deel in het Engels toe.

De getoonde website wordt standaard zichtbaar in het Nederlands. Daarnaast moet deze taal op de website te wijzigen zijn naar een andere beschikbare taal.

De geselecteerde taal moet dan opgeslagen worden (in bijvoorbeeld een cookie) zodat bij een volgende aanroep van de web applicatie de pagina in de laatst geselecteerde taal zichtbaar wordt.

Naast internationalisatie moet de website ook rekening houden met de plaatsing van komma's en punten in getallen. Dit wordt lokalisatie genoemd. De valuta is altijd de Euro.

Voor dit afstudeerproject hoeft de applicatie alleen beschikbaar gemaakt te worden in de Nederlandse taal. Om de internationalisatie te kunnen testen wordt er één extra taal toegevoegd, het Engels.

Figuur 7, een fragment van de Solweb website. De website is beschikbaar in het Engels, Duits en Spaans. Rechtsbovenin het scherm is de taal te selecteren.

2.2.1. Resource bundels

Er wordt geadviseerd om tekst nooit rechtstreeks in de code te plaatsen, maar om te werken met zogenaamde resource bundels. Een resource bundel is een lijst met sleutels en de bijbehorende beschrijving. Voor elke ondersteunde taal is er dan een lijst aanwezig.

Internationalisatie wordt vaak afgekort tot i18n, dat staat voor internationalisation, oftewel een i, 18 letters en een n.

Om nu een tekst af te drukken, is het voldoende om alleen de betreffende sleutel op te geven. Afhankelijk van de ingestelde taal wordt de beschrijving die bij de opgegeven sleutel hoort opgezocht en afgedrukt.

Hieronder is een deel van de Engelse en Nederlandse resourcebundel afgedrukt. In de linker kolom staat de lijst met sleutels.

Sleutel	Nederlands	Engels
overzicht.label.title	Overzicht van objecten	Objectlist
overzicht.label.totalcount	Totaal aantal objecten	Total object count
details.label.title	Details object	Object details
details.label.city	Plaats	City

Tabel 1, de resource lijst met de sleutels en een Nederlandse en Engelse vertaling.

De naam van een sleutel van een resource bundel is als volgt opgebouwd:

`[naam van de view].[type label].[naam label]`

Bij algemene labels die op meerdere views kunnen voorkomen, kan de naam van de view weg gelaten worden.

Lokalisatie

Naast i18n, bestaat er lokalisatie, dat wordt afgekort tot l10n.

Lokalisatie regelt de manier waarop bijvoorbeeld een datum afgedrukt wordt. Het Nederlandse formaat is `dd-mm-jj` terwijl het Amerikaanse formaat `mm-dd-jj` jaar is en de Japanse standaard `jj-mm-dd`.

Naast de datum wordt om getallen leesbaarder weer te geven een . (punt) gebruikt in het Nederlands terwijl dat in het Engels een , (comma) is. Voor gebroken getallen geldt weer precies het omgekeerde.

Nog een opmerkelijk verschil wat betreft de datum is dat een week in Europa officieel op maandag begint, terwijl in de Verenigde Staten een week officieel op zondag begint. Dit is ook de reden dat een weeknummer kan afwijken.

2.2.2. Internationalisatie in Java

I18n is eenvoudig toe te voegen aan een applicatie, omdat deze extra functionaliteit altijd al standaard in Java aanwezig is.

Instellingen voor i18n kunnen allemaal bijgehouden worden in een zogenaamde Java `ResourceBundle`. In een object van dit type is bijvoorbeeld vastgelegd waar de taallijsten staan en welke taal en land ingesteld is.

Door deze `ResourceBundle` object beschikbaar te maken voor de view, kan de view door middel van een sleutel de tekst opzoeken die afgedrukt moet worden. De ingestelde taal bepaalt uit welke taallijst de beschrijving moet komen. Als er voor de opgevraagde sleutel geen waarde aanwezig is, valt de `ResourceBundle` terug op de standaard lijst.

In de Velocity view kan rechtstreeks een Java object benaderd worden. Het afdrukken van een titel ziet er dan als volgt uit:

```
<head>
  <title>$resource.getString("overzicht.label.title")</title>
</head>
```

2.2.3. Opslaan taal instelling

Nadat de gebruiker een taal anders dan de standaard taal geselecteerd heeft, kan deze bewaard worden in een cookie. Deze methode heeft als nadeel dat als de gebruiker geen cookies ondersteunt, deze instelling niet bewaard wordt.

Een andere manier zou kunnen zijn om de instellingen via (`HTTP GET`) parameters mee te sturen met de URL. De instellingen wordt vastgehouden door deze extra parameters achter elke verwijzing op de pagina te plakken. Een probleem waar dan tegenaan gelopen kan worden, is dat er met verschillende talen voor de view; HTML, JSP, Velocity gewerkt wordt. Dit zou dan weer opgelost kunnen worden door in elke view een methode aan te roepen die de bestaande request parameters achter elke link plakt.

Nog een andere manier zou kunnen zijn om de taal in te stellen in een sessie. Op deze manier hoeft er niet gekeken te worden of de gebruiker wel cookies ondersteunt. URL rewrite voor het onthouden van een sessie als de gebruiker geen cookies ondersteunt gaat dan automatisch voor elke type view, Velocity en JSP. Een nadeel hiervan is, is dat een sessie vertraagt en dat op de JSP pagina het gebruik van sessies aangeschakeld moet zijn. Ook als de sessie niet gebruikt wordt, wordt er een nieuwe sessie gecreëerd, wat overhead is.

Een laatste manier die we is om de taal te verwerken in de URL. Door bijvoorbeeld de taal in de URL naam te coderen als een directory, bijvoorbeeld:

`http://localhost/nl/NL/overzicht.html`

Deze naam wordt dan in bijvoorbeeld een Filter, server side herschreven naar een URL met HTTP Get parameters, bijvoorbeeld:

`http://localhost/overzicht.html?l=nl&c=NL`

Doordat alle verwijzingen in een pagina in principe relatief zijn, blijft de taal in de URL staan (aan de browserkant). In bovenstaande verwijzing is de eerste instelling de taal (nl) en de tweede het land (NL).

2.2.4. Adressering

Iedere land heeft zijn eigen adressering. De notatie van de adressering komt vaak niet met elkaar overeen. De adresseringen van de woningen zijn dus op een andere manier geschreven. Bijvoorbeeld in Engeland schrijft men eerst het nummer van de woning en daarna de straat waar de woning staat. Dit hangt dus af in welk land de woning zich bevindt. Er mag niet gekeken worden naar de taal- en landinstellingen van de browser, want dat zou kunnen veroorzaken dat de weergave van de adressering niet goed is.

Een oplossing voor het adresserings probleem zou kunnen zijn om de adressering in de resourcebundle te zetten. Maar dit betekent dat er nieuwe resourcebundle gecreeerd moet worden bij iedere aanroep.

Een andere manier zou kunnen zijn om de gegevens van het adres in een bepaalde volgorde in een array te zetten. Deze array kan men dan uitlezen en zichtbaar maken in de view. Nadeel hiervan is dat de view misschien niet helemaal naar de wens van de makelaar wordt.

Nog een andere manier is om gewoon in de view te controleren in welk land de woning staat.

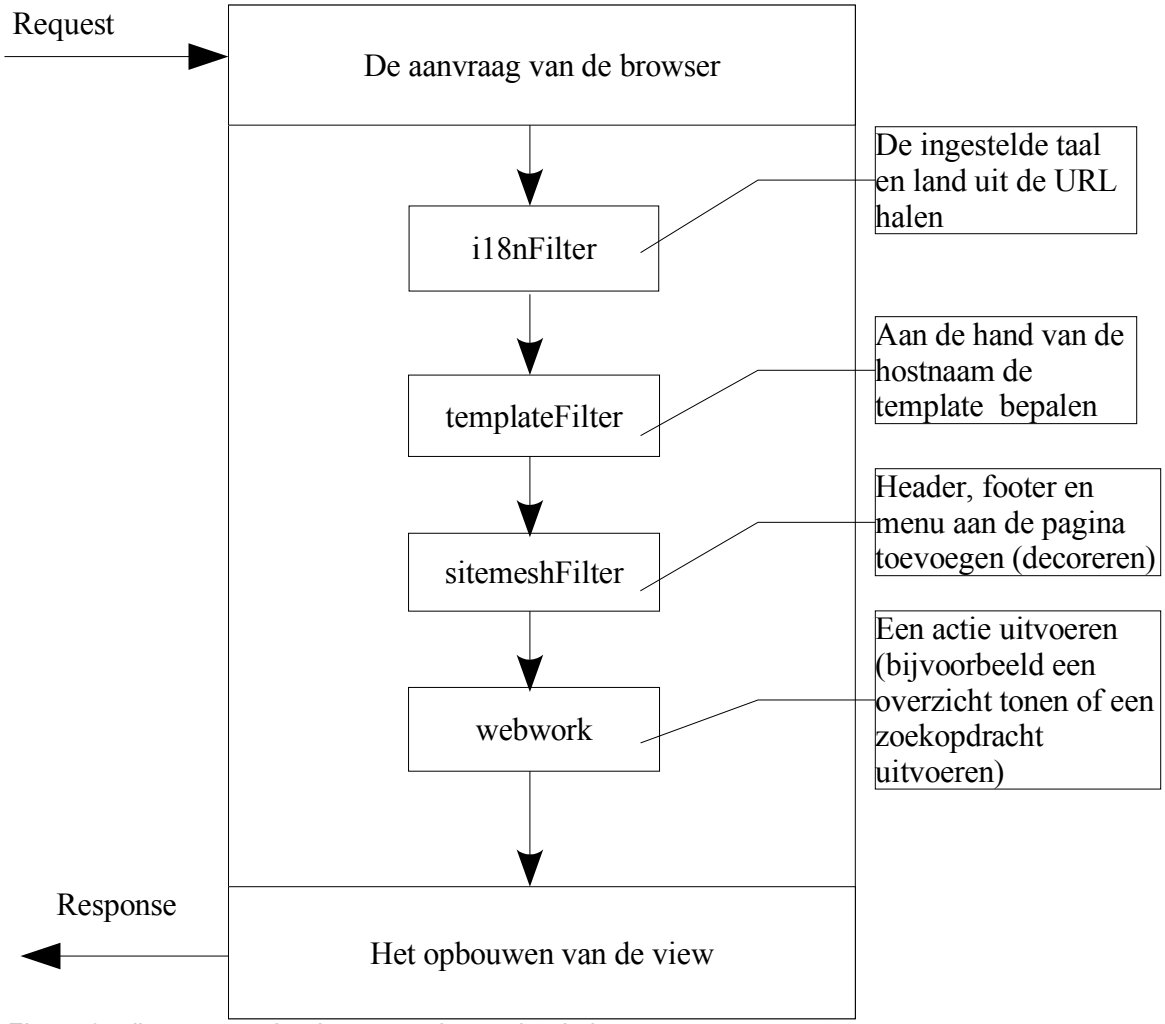
```
#elseif ($vastgoedObject.land == "EN")
  $vastgoedObject.categorie<br />
  $vastgoedObject.straatnummer$vastgoedObject.straatbijvoegsel----------
  ---------------$vastgoedObject.straatnaam <br />
  $vastgoedObject.postcode<br />
  $vastgoedObject.plaatsnaam<br />
  &euro; $formatter.format($vastgoedObject.vraagprijs)
#end
```

Hierboven zie een fragment uit de code van de view waar bepaald wordt in welk land de woning staat en hoe de bijbehorende adressering er uit moet komen te zien.

3. ONTWERP EN IMPLEMENTATIE

3.1. HET ONTWERP

Als eerste een schema van de verschillende stappen die doorlopen worden nadat de browser een request verzonden heeft, zie figuur 8. Na dit globale overzicht wordt in de volgende hoofdstukken dieper ingegaan op elk deel.



Figuur 8, alle stappen doorlopen worden nadat de browser een request verzonden heeft en de response die daarop ontvangen wordt.

Nadat de view is opgebouwd, wordt het resultaat in een response terug gestuurd naar de browser.

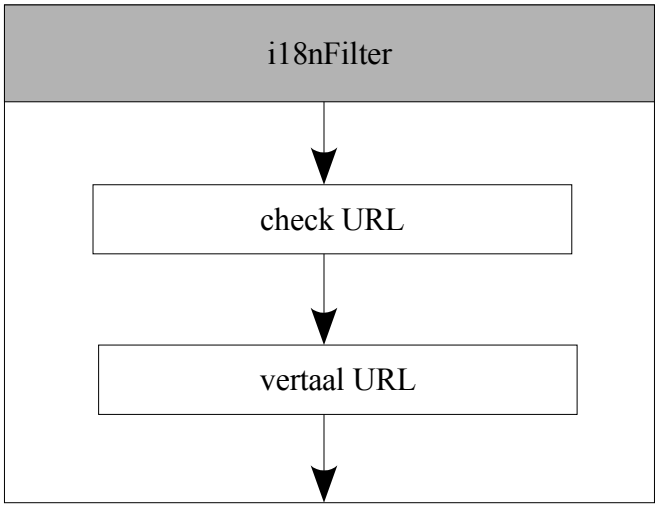
3.1.1. I18n Filter

Als er een taal of land ingesteld is, zitten deze verwerkt in de URL van de aanroep als virtuele directories. Gewoonlijk wordt als eerste de taal aangegeven en als 2e het land, maar deze volgorde is niet verplicht. Het is wel verplicht dat de taal in kleine letters geschreven wordt en het land in hoofdletters. bijvoorbeeld:

`http://localhost/nl/NL/overzicht.html` wordt:
`http://localhost/overzicht.html?l=nl&c=NL`

Taal en land instelling zijn niet verplicht. Zie voor de codering van talen en landen respectievelijk ISO-639 en ISO-3166.

Belangrijk: Dit Filter moet als eerste aangeroepen worden in een chain, voordat de URL herschreven wordt. Verder moet het gebruik van bestandsmappen met een lengte van 2 vermeden worden, omdat deze aangezien worden als een taal- of land instelling en verwijderd worden uit het path van de request.



Figuur 9, het i18n filter controleert de URL op een taal instelling. Als er een taal instelling gevonden is, wordt de URL herschreven naar een bestaand path en eindigt de taak van dit filter.

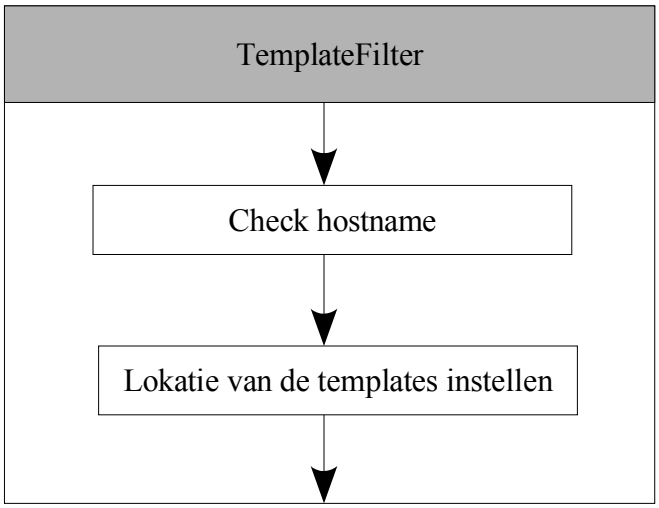
Nadat dit filter uitgevoerd is, wordt het volgende filter in werking gesteld.

3.1.2. Template Filter

Het template Filter is het volgende filter. De naam zegt het eigenlijk al. Deze bepaalt via de hostnaam welke basis directory ingesteld moet worden. In deze directory staan de plaatje (logo's), de views en de style van de makelaar.

Herschrijf de URL, zodat de resources (plaatjes, css enz) van de juiste template geladen worden. Die "rewrite" is niet zichtbaar in de browser. Het is "serverside".

Voorbeeld: `/overzicht.html -> /templates/default/overzicht.html`



Figuur 10, het template filter bepaalt aan de hand van de hostnaam welke template gebruikt moet worden om de view op te bouwen. Aan de kant van de server wordt de URL herschreven naar de locatie waar zich de templates bevinden.

3.1.3. SiteMesh Filter (decoratie pattern)

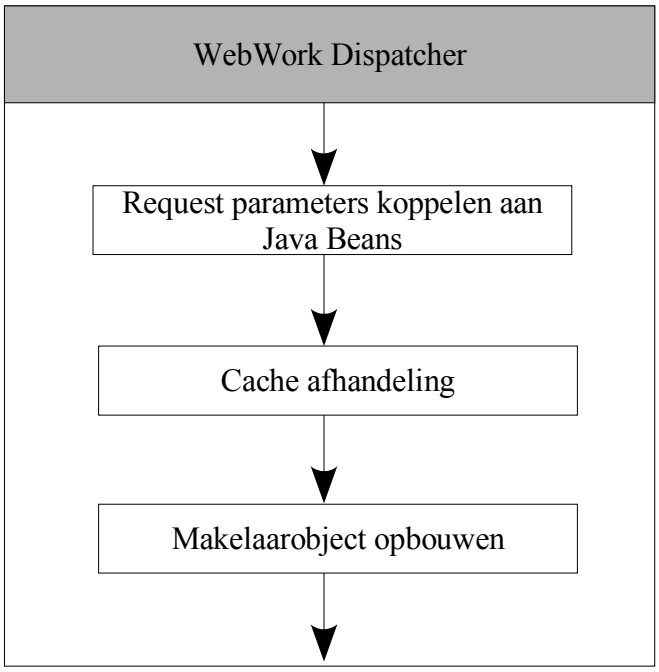
In de directory die is ingesteld door het Template Filter staan ook de overige pagina's uit de template die geen onderdeel zijn van een actie. Zoals het menu, een kop- en voettekst en de pagina's die geen actie uitvoeren en alleen info bevatten.

Het menu en de kop- en voettekst moeten samen met de actie- of info pagina samengevoegd worden tot één geheel. Hiervoor wordt SiteMesh gebruikt. De resulteren pagina is ook echt één pagina en geen pagina die bestaat uit verschillende frames. De werking van SiteMesh is eerder in dit document uitgelegd. Zie hoofdstuk 2.1.2 op bladzijde 7 voor de details.

Het bestand dat bepaalt hoe de verschillende onderdelen als één pagina afgedrukt moeten worden, wordt een `decorator` genoemd.

3.1.4. WebWork versie 2

WebWork wordt aangeroepen als er een actie uitgevoerd moet worden. In de applicatie zijn alleen de bestanden die eindigen op `*.html` gekoppeld aan WebWork acties. Een actie bestaat uit een stuk business code waarin bijvoorbeeld de benodigde data uit een databank gehaald wordt. De af te beelden informatie wordt in de context gezet. Deze context kan door de view benaderd worden.



Figuur 11, de WebWork2 dispatcher.
In WebWork bestaan er zogenaamde interceptors, dit zijn filters die uitgevoerd worden voordat de code van een actie uitgevoerd wordt. In onze implementatie zijn er drie van deze interceptors. Eén om de variabelen uit een formulier te koppelen aan een JavaBean, één om te controleren of voor een pagina de versie in de cache van de browser gebruikt kan worden en als laatste één om die een object opbouwt die aan de hand van de hostnaam bepaalt met welke pagina van welke makelaar we te maken hebben.

3.2. IMPLEMENTATIE

Tijdens het ontwikkelen van de pagina's bleek er verschil te zitten in hoe Velocity en hoe JSP niet bestaande ResourceBundle messages afbeeldt.

Velocity geeft een foutmelding weer als het een bepaalde sleutel (i18n – key) niet kan vinden. De view wordt niet opgebouwd en afgebeeld en in de log verschijnt een nietszeggende foutmelding. JSP daarentegen bouwt de view op een normale manier op en drukt de naam van een sleutel af op de plek waar anders de tekst uit de ResourceBundle zou komen.

4. TESTEN

4.1. TESTPLAN EN TESTRAPPORT

4.1.1. Internationalisatie

Beschrijving	Instructies	Verwachte uitvoer	Check
Het instellen van een taal.(Nederlands)	Stuur de volgende link: http://localhost:8080/vastgoedonline/nl/NL/overzicht.html	De taal is in het Nederlands	√
Het instellen van een taal (Engels)	Stuur de volgende link: http://localhost:8080/vastgoedonline/en/EN/overzicht.html	De taal is nu in het Engels.	√

4.1.2. Templating

Beschrijving	Instructies	Verwachte uitvoer	Check
Het instellen van een andere template	Stuur de volgende link: http://localhost:8080/vastgoedonline/nl/NL/overzicht.html	De template van localhost	√
Het instellen van een andere template	Stuur de volgende link: http://10.0.20.4:8080/vastgoedonline/en/EN/overzicht.html	De template van 10.0.20.4	√

5. CONCLUSIE

Voor het gebruik van templating, wordt er gebruik gemaakt van een combinatie van Velocity, JSP en SiteMesh.

Voor het opbouwen van de view nadat er door WebWork een actie uitgevoerd is, wordt Velocity gebruikt.

Voor het bouwen van de makelaars specifieke pagina's die geen onderdeel zijn van de applicatie, zoals informatie pagina's kan Velocity, JSP of HTML gebruikt worden. Bij het gebruik van statische HTML is niet de mogelijkheid aanwezig voor meerdere talen.

Voor het decoreren van een pagina met een header, footer en een menu wordt SiteMesh gebruikt. SiteMesh onderschept een bestaande pagina en plakt de header, footer en het menu er later bij aan. Voor het beschrijven van de lay-out in SiteMesh wordt JSP gebruikt, net zoals voor het menu, de header en de footer. In de volgende versie van SiteMesh is ook ondersteuning voor Velocity templates aanwezig.

Bij het gebruik van SiteMesh zijn we tegen een klein probleem aangelopen. SiteMesh blijkt onder water een sessie aan te zetten en dat is niet de bedoeling bij de web applicatie. Dit probleem wordt nog verholpen. Verder waren er geen problemen bij het uitvoeren van de testen. Alles bleek na twee weken hard werken te werken zoals de bedoeling is volgens het ontwerp.

Via de hostnaam wordt bepaald met welke makelaar we te maken hebben. Op dezelfde manier wordt ook de template voor de opmaak bepaald. De verschillende templates staan netjes geordend in verschillende bestandsmappen in de applicatie.

Voor het onthouden van de taal instelling wordt gebruik gemaakt van de manier waarop relatieve URL's werken. De taal wordt verwerkt als een virtuele directory in de URL. Dit bleek in de praktijk de de meest flexibele oplossing, omdat het op deze manier bewaren van de taal instelling op elke browser zal werken. Het opslaan van de taal instelling in een cookie valt af, omdat deze uitgeschakeld kunnen zijn. Ook het gebruik van sessies willen we vermijden omdat dit het gebruik van cache in de war schopt en de applicatie vertraagt.

De oplossing wat betreft het afbeelden van een adres volgens een bepaalde land notatie is eenvoudig gehouden. Het formaat van de adressering wordt bepaald door de view. De andere oplossing waren te uitgebreid of niet toepasbaar in onze applicatie.

6. REFERENTIES

[1] **i18nGuy**, *Internationalisation*, maart 2004, http://www.i18nguy.com/

[2] **Sitemesh**, *Opensymphony SiteMesh Overview*, maart 2004, http://www.opensymphony.com/sitemesh/

[3] **Java i18n**, *Tutorial i18n Java*, maart 2004, http://java.sun.com/docs/books/tutorial/i18n/