SmartElement - Automatiserad anpassning av webb-innehåll

Staffan Enberg

Examensarbete
Informations- och medieteknik
2014

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4669
Författare:	Staffan Enberg
Arbetets namn:	SmartElement - Automatiserad anpassning av webb-innehåll
Handledare (Arcada):	Göran Pulkkis
Uppdragsgivare:	Developer's Helsinki Oy

Sammandrag:
Avsikten med detta arbeet är att redogöra för implementationen av ett system för automatiserad anpassning av webbinnehåll.

Nyckelord:	Anpassning, Innehåll, Webb-analys, Webbsidor
Sidantal:	TBD
Språk:	Svenska
Datum för godkännande:	TBD

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	4669
Author:	Staffan Enberg
Title:	SmartElement - Automatic personalization of web content
Supervisor (Arcada):	Göran Pulkkis
Commissioned by:	Developer's Helsinki Oy

Abstract:
The purpose of this thesis is to describe the implementation of a system for automatic personalization of content on a website.

Keywords:	Personalization, Content-delivery, Analytics, Websites
Number of pages:	TBD
Language:	Swedish
Date of acceptance:	TBD

INNEHÅLL

TABELLER

FIGURER

FÖRKORTNINGAR

DOM Document Object Model. 7, 12

JSON JavaScript Object Notation. 7, 12, 14, 15, 36

JSONP Json-with-padding. 7, 12, 37

SaaS Software-as-a-Service. 7, 10

SQL Structured Query Language. 7, 18

1 INLEDNING

Detta arbete dokumenterar systemet SmartElement som utvecklades för företaget Developer's Helsinki Ab. Systemets syfte är att möjliggöra enkel konfiguration av anpassat innehåll för webbsidor, vilket tillåter webbsidors ägare att själva kunna konfigurera regler som bestämmer vad för innehåll som skall lyftas fram för olika användare.

1.1 Målsättning och syfte

Målsättningen med arbetet är att kartlägga den arkitektur som bäst stöder målet av ett effektivt system för innehålls-filtrering och leverans. som även är lätt att vidare utveckla och utvidga i form av nya filter och nya algoritmer för filtrering.

Arbetet redogör för den information som systemet kan registrera om besökare till en webbsida. Olika problem relaterade till pålitligheten med informationen beaktas. Utgående från den information som samlas in definieras 14 filtertyper som implementerats i systemet, samt hur de fungerar.

Arbetet redogör även för den tekniska arkitekturen bakom systemet och hur den utformats. Mjukvaruarkitekturen förklaras och systemets uppbyggnad presenteras utgående från den centrala funktionaliteten, vilket är filtrering av innehåll.

1.2 Avgränsningar

Detta är en redogörelse för SmartElements tekniska implementation, inte användningen av anpassat innehåll på webbplatser. Arbetet tar därmed inte ställning till hur anpassat innehåll skall användas, endast hur det kan implementeras.

Efter som det inte finns någon klar standard för vad som kan anses vara bra prestanda för ett liknande system, kommer arbetet inte att innehålla prestanda analyser av systemet.

2 BAKGRUND

Projektet började våren 2013 som ett internt verktyg på Developer's Helsinki efter ett behov uppstått för ett system som skulle tillåta snabb implementation av anpassat innehåll på kunders webbsidor. Det bestämdes att det skulle uvecklas en mjukvara först för internt bruk men med målsättningen att lanseras som en SaaS produkt. Mjukvaran skulle utvecklas för att vara möjligast anpassningsbar, för att tillåta flera olika sorters applikationer av anpassning.

2.1 Developer's Helsinki

Developer's Helsinki Oy är ett finskt företag som utvecklar såväl applikationestjänster (jmfr. engelskans Software-as-a-Service (SaaS)) som webbsidor. Företaget grundades 2009 för att vidareutveckla SaaS tjänsten Netmonitor, en produktfamilj av webbanalys och marknadsföringsverktyg.

Företaget har en stark kännedom av marknadsföring på webben efter att ha jobbat med både utvecklingen av webbsidor samt uppföljningen av resultat genom webbanalys. På basis av denna erfarenhet började SmartElement projektet planeras. Kunskapen fanns inom företaget och genom att anpassa processer från webbanalys sågs en möjlighet för automatisering av anpassning av innehåll.

2.2 Anpassat innehåll

Anpassat innehåll, eller personaliserat innehåll (eng. *Personalized content*), är innehåll som väljs ut på basis av egenskaper hos användaren. I fallet av webbsidor kan det handla om information som användarens geografiska läge, användarens språk, användarens webbläsare, antalet besök som användaren gjort till sidan m.m. Tanken är att förse användaren med det den behöver, eller vill ha, utan att denne behöver be om det. (Mulvenna et al. 2000)

Figur 1. Amazons användarsida med anpassat innehåll

Anpassat innehåll har bland annat användts inom nätbutiker för att visa reklam anpassad för kunden på basis av dennes beställningshistorik. Bild 1 visar användarsidan som visas då man loggar in på Amazons webb-butik. Inom social media används anpassat innehåll för att lyfta fram innehåll som antas vara intressant för användaren, som till exempel uppdateringar från vänner som användaren ofta är i kontakt med eller reklam från företag användaren har gillat. (Guy et al. 2010)

Anpassningen av innehåll kan åstadkommas på olika sätt. Ofta handlar det om användning av statistik för att välja ut innehåll som motsvarar en besökares uppvisade intressen. Det kan handla om en så enkel sak som att använda statistiken för att visa en liten hälsning för återkommande besökare. Det finns dock mera avancerade metoder för anpassning, som att använda information som besökaren förser webbsidan med i kombination med användarens beteende på sidan för att klassificera denne. (Albanese et al. 2004)

2.3 Behov

I takt med att stora aktörer införde allt mer anpassat innehåll så började även mindre aktörers kunder fråga om möjligheten att använda tekniken på sina webbsidor. Det visade sig att det inte fanns något företag på finska marknaden som erbjöd ett tillräckligt flexibelt system för anpassning av innehåll och därmed bestämdes det att Developer's Helsinki skulle utveckla ett sådant.

Systemet skulle kunna leverera innehåll dels i form av text (HTML eller rå text) för direkt injicering i webbsidans Document Object Model (DOM), dels i form av data i JavaScript Object Notation (JSON) format och dels som Json-with-padding (JSONP) svar, vilket tillåter exekvering av JavaScript vid svaret. För filtrering skulle det samlas in data om besökares webbläsare och dator, men det skulle även vara möjligt att tillägga egen data som sedan kunde användas för filtrering.

2.4 Befintliga lösningar

I takt med att anpassning av webbinnehåll blivit mer populärt har det utvecklats olika lösningar för ändamålet. Det finns i dagens läge så väl komersiella produkter som open-sourceprojekt som erbjuder olika typer av anpassningssystem. På open-source fronten handlar det oftast om webbinnehållsplatformer som innehåller en anpassningsmodul som en del av helheten. På den komersiella sidan finns det många lösningar för optimering av webbutiker genom anpassning av innehållet på sidan.

2.4.1 Nosto.com

Vid tillfället för beslutet att utveckla produkten var den enda komersiella lösningen på finska marknaden nosto.com som levererar anpassat innehåll med inriktning på webbutiker.

Systemet är designat att förse upprätthållare av webbutiker med verktyg for optimering

av konversioner samt för att uppehålla kundrelationer. (Nosto Website 2014)

Nosto har valt en tydlig kundgrupp, systemet utvecklas för att passa in i webbutikers verksamhet. SmartElement utformades, till skillnad från Nosto, som ett generellt system för anpassning av innehåll på webbsidor av alla typer. Nosto är en större helhet, som inte bara innefattar anpassat innehåll.

2.4.2 Pimcore

På open-soucefronten finns det bl.a. webbinnehållsplatformen Pimcore som innehåller en anpassningsmotor. Funktionaliteten baserar sig på att skapa besökarsegment. Efter konfiguration kan upprätthållaren, medan denne editerar en sida, välja ett besökarsegment att anpassa innehållet för. (Pimcore - On-site Behavioral Targeting and Personalization Platform)

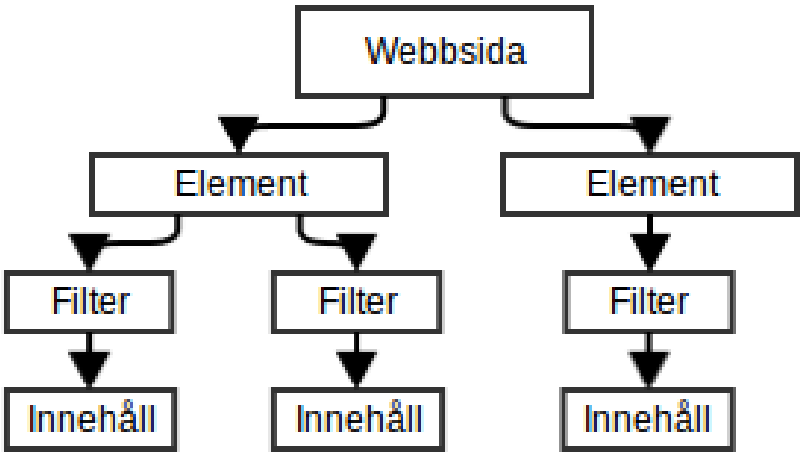
Till skillnad från SmartElement är Pimcore en hel platform för hantering av webbinnehåll. För att använda platformen måste upprätthållaren ta i bruk systemet samt upprätthålla en server var systemet kan köras. SmartElement är däremot tänkt som en tjänst för upprätthållare att lägga till på sin befintliga sida, oberoende av underliggande system.

3 SMARTELEMENT

SmartElement är ett back-endsystem som sköter filtrering av innehåll på basis av information som samlas in om besökare på en webbsida. Upprätthållarem av en webbsida kan genom att integrera SmartElements JavaScript-tag i sin sida låta back-end systemet fylla i specificerade element med anpassat innehåll.

Systemet är i sig bara en motor för identifikation av användare och filtrering av innehåll. Det enda gränssnitt som SmartElement har är ett JSON-gränssnitt genom vilket ett separat grafiskt gränssnitt tillåter användare att konfigurera sitt innehåll.

3.1 Beståndsdelar



Figur 2. Exempel på en hierarki i SmartElement

SmartElement bygger på några grundläggande koncept; webbsidor, element, filter och innehåll. Med dessa fyra byggstenar konfigureras objekt, visualiserat i figur 2, som back-end systemet använder för att filtrera och leverera innehåll.

3.1.1 Webbsidan

Webbsidan är det högsta elementet i hierarkin som SmartElement handskas med. Det representerar en hel webbplats, under vilken man kan definiera element med innehåll. Varje sida har ett unikt identifikationsnummer som används för att ladda SmartElement-tagen.

3.1.2 Element

Elementen är hållare för den data som returneras från back-end systemet, de knyter innehållet till webbsidan. Elementet har i sig fler innehåll och filter. Filtren används för att välja ut det innehåll som skall visas i elementet. Element har en inom webbsidan unik kod som används för att specificera vilka element som behöver processeras när man anropar back-end systemet.

3.1.3 Filter

Filter består i SmartElement av flera olika regler som ställer en fråga om besökaren och har en länk till ett innehåll, vilket visas om filtret matchar. För att ett filter skall matcha krävs det att besökaren passar alla regler i filtret. För att hantera fall var två eller flera filter passar för en besökare, har filter en prioritets-ordning inom elementet som användaren själv kan definiera.

3.1.4 Innehåll

Innehåll är den konkreta data som levereras för ett element efter att ett filter valts som vinnare. Innehållet kan bestå av text, HTML kod eller JSON data, vad som helst för data som kan sparas som en textsträng. Det är upp till användaren att definiera vad för data som sparas.

3.2 Funktion

SmartElement har två huvudsakliga funktionsområden. Konfiguration av systemet genom API gränssnittet, och hantering av sidvisningar och filtrering av innehåll genom innehålls-gränssnittet.

3.2.1 Konfiguration

Bild av konfigurationsprocessen.

För att SmartElement skall kunna leverera innehåll, måste användaren konfigurera sidan och dess element på förhand. Processen består av att registrera element under sidobjektet, skapa innehåll samt att konfigurera filter genom att kombinera regler om vilka användare innehållet passar för.

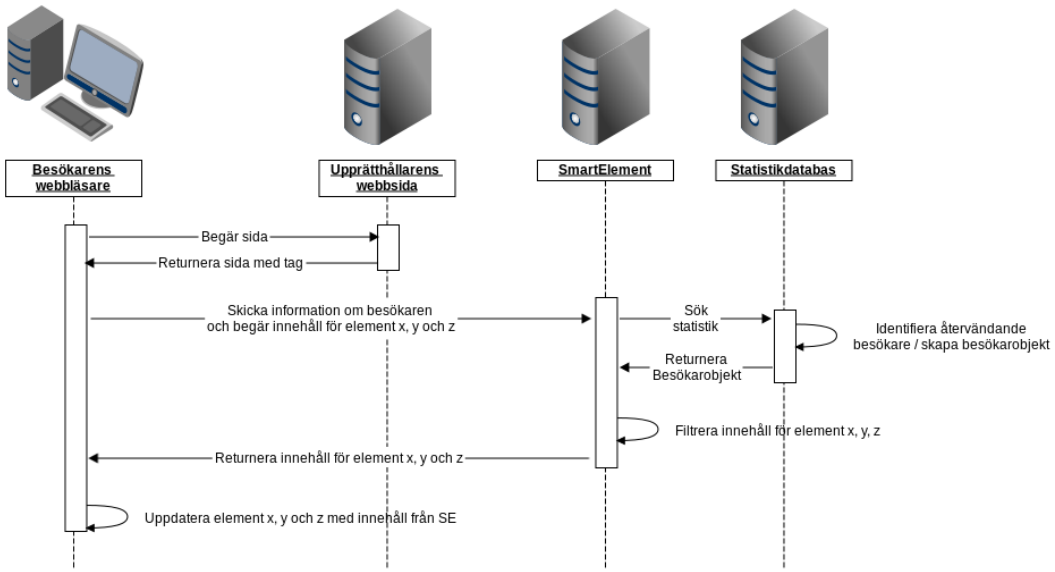
I praktiken sker all konfiguration av systemet, då det är i produktion, genom ett JSON gränssnitt. En klient kopplar upp sig, auktoriserar sig genom en nyckel och kan sedan göra förfrågningar mot gränssnittet.

Konfigurationsprocessen börjar från att registrera en webbsida i systemet. En användare kan ha flera olika sidor registrerade i systemet, så gränssnittet jobbar alltid under en utvald webbsida.

Under webbsidan kan användaren registrera de element för vilka innehållet skall anpassas. Elementen är i grund och botten bara en nyckel som används för att representera ett objekt som kan se ut på olika sätt (ha olika innehåll) beroende på de filter som associeras med det.

Efter konfiguration av element konfigureras ett eller flera innehåll under dessa. I samband med innehållen lägger användaren till filter i innehållets filter-set. Alla filter som registreras måste matcha för att ett innehåll skall visas. Det är genom dessa filterkombinationer som avancerade villkor kan implementeras.

3.2.2 Sidvisning



Figur 3. Processen vid en sidvisning

Figur 3 representerar kommunikationen vid en sidvisning. När en användares webbsida laddas, inkluderar den en JavaScript fil som innehåller SmartElements tag. Efter att sidan laddats klar körs tagen, som samlar ihop information om besökaren, samt eventuell extra data som användaren specificerat i tagen, och skickar informationen till back-enden.

Backenden tar emot data från tagen och söker upp sidan som motsvarar det id som skickats. Efter att en sida har hittats går back-enden igenom de element som tagen begär innehåll för och filtrerar deras innehåll på basis av den data som tagen skickat. Allt innehåll kompileras till en lista med element-id länkat till innehåll och skickas tillbaka till besökarens webbläsare.

Om användaren inte valt att ändra på beteendet av tagen så anropar back-enden på tagen

för att fylla elementen med den data som returnerats.

4 TEKNISK ARKITEKTUR

SmartElement bygger i stort sett på fem centrala tekniker, programmeringsspråket PHP som används för att skriva koden bakom systemet, MySQL som databaslager för användar- och systemdata, Memcache som caching lager, vilket tillåter att information som är tung att beräkna kan sparas mellan förfrågningar, dokument databasen MongoDB för lagring av information om besökare och statistik, och till sist JavaScript som används för att skapa det informationspaket om användaren som skickas till servern för användning vid filtreringen av innehåll.

Många av de val som gjorts i planeringen av den tekniska arkitekturen bakom SmartElement baserar sig på en möjlighet att kunna skala systemet för att hantera förändrade användarmängder. De grundläggande principerna har varit att det skall vara lätt att utveckla, det skall vara lätt att underhålla, det skall vara lätt att installera och det skall vara lätt att skala upp om det skulle behövas.

4.1 PHP

PHP är ett skriptspråk som utvecklats med målet att snabbt kunna skapa dynamiska webbsidor. (PHP Manual 2013) Språket valdes för att det är enkelt att arbeta med och det var språket som företagets andra produkter var skrivna med.

PHP är ett av de mest populära språken för webbutveckling (TIOBE Programming community index 2014) och det finns en mängd information och färdiga kodpaket till förfogande för programmerare, vilket underlättar och försnabbar utvecklingen. Genom sin popularitet har språket även den fördelen att det är lätt att hitta utvecklare som kan fortsätta utvecklingen av projekt skrivna i PHP.

4.2 Databas

Eftersom SmartElement hanterar dels intern data som är starkt bunden till systemet och påverkar funktionaliteten av systemet, och dels extern data, som är närmare kopplad till upprätthållarens webbsida och påverkar hur en dennes sida fungerar, så finns det två tydliga data-set som båda har egna krav.

På grund av denna delning var det möjligt att använda två skilda databaser, var med sitt ansvarsområde. Databas lagret inom SmartElement splittrades därvid i två skilda lager, dels en relationsdatabas för systemets interna data och en dokumentdatabas för lagring av extern data. Denna lösning valdes för att på bästa vis stöda de egenskaper som dataseten har.

4.2.1 MySQL

MySQL är en relationsdatabashanterare som sköter om lagring av data samt modifikation och sökning genom Structured Query Language (SQL) språket. MySQL utvecklades som ett öppen-källkods projekt av det svenska företaget MySQL AB och är ett av de populäraste databassystemen för utveckling av webapplikationer. (DB-Engines Ranking 2014)

MySQL används i SmartElement för lagring av intern data relaterad till systemets funktionalitet. Databasen lämpar sig bra för lagring av data med starka länkar mellan objekten, vilket passar bra in på den användardata som SmartElement behandlar. Det finns en klar hierarki bland objekten och länkar mellan dessa.

Relationsdatabasen gör sig bra i denna roll då den ger en viss garanti på data strukturen bakom de olika objekt som bygger upp SmartElement. Genom att binda modellerna till en databasstruktur har man en sorts garanti att existerande data uppdateras och transformeras i samband med att processer i systemet uppdateras. På så vis ansågs en mer rigid databasmodell bättre anpassad för denna uppgift.

4.2.2 MongoDB

MongoDB är en dokumentdatabas som utvecklas av företaget MongoDB, Inc. Databasen är en av de så kallade NoSQL databaserna, vilket är en term som används för att bemärka databaser som inte använder sig av SQL språket för data-manipulering. I MongoDB använder man i stället ett objektorienterat gränssnitt för att ge kommandon till databasen. Utöver det kan man även skicka JavaScript kod som exekveras i databasen för att vidare filtrera data. (MongoDB Manual - Querying Documents)

För lagring av extern data, relaterad till besökare av användares webbsidor, valdes dokumentdatabasen MongoDB. MongoDB använder inte ett strikt schema för datalagring utan dokument i samma samling kan ha skiljande attribut associerade med sig. Detta underlättar vidareutveckling av systemet genom att tillåtta gammal data att existera i databasen tillsammans med ny data tills den gamla uppdateras och kompletteras då besökare återvänder till en användares webbsida.

I fallet av denna data så är den strikta datastrukturen inte lika viktig. Datan föråldras dels snabbt samt att designen är sådan att datan inte skall behöva vara komplett och innehålla strikta datastrukturer. Tanken med besökarobjekten är att de skall vara levande och öppna för modifikation och utvidgning. Om ett nytt filter läggs till i systemet så skall det inte kräva en migration av gammal data, utan statistiken för det nya filtret läggs helt enkelt till på objektet. Om inte datan finns på objektet så passar filtret ej. Denna flexibilitet är lättare att åstadkomma med en dokumentdatabas eftersom scheman inte behöver uppdateras, vilket potentiellt skulle vara en väldigt tung operation i fallet av besökarobjektet, som är den snabbast växande samlingen inom systemet.

4.3 Memcached

Memcached är ett object caching system, en mjukvara som tillåter program att spara objekt i minnet och senare läsa tillbaka det utan att behöva bygga upp datastrukturer igen. Systemet fungerar som ett nyckel-värde system, det vill säga att information sparas i en

nivå med unika nycklar som id, en enkel princip som bidrar till att systemet är mycket effektivt. (Memcached Wiki 2014)

Eftersom SmartElements datamodell består av många länkade objekt är det relativt tungt att bygga upp ett sidobjekt från databasen då det måste göras många förfrågningar. På grund av detta, samt för att minska risken för (n+1)-problem (var man gör en databas-förfrågan för varje element i en samling som man itererar över), så bestämdes det att ett caching lager skulle användas för att hålla dessa datamodeller, och på så vis låta mjukvaran gå direkt till filtrerings-logiken då en förfrågan från tagen behandlas.

Målet med cache lagret är att hålla sidobjekten i minnet på servern så mycket som möjligt, och endast läsa från databasen när ett objekt byggs om i samband med en uppdatering. För detta ändamål valdes Memcache, en väletablerad mjukvara för caching. Memcache valdes för att det var en teknik som utvecklarna var vana vid. Den var redan i bruk i andra projekt och dess enkla modell av horisontell skalning skulle tillåta systemet att snabbt reagera på en växande kundskara.

4.4 JavaScript

JavaScript är ett skriptspråk som tillåter utvecklare att implementera kod i webbsidor som exekveras i en besökares webbläsare.

För att kunna identifiera så mycket information som möjligt om besökaren, samt för att leverera innehållet som valts ut för besökaren är det nödvändigt att köra kod på klienten. För att stöda detta krav valdes scriptspråket javascript, vilket stöds av så gott som godtyckliga webbläsare och tillåter logik att köra på besökarens maskin i samband med en sidvisning.

5 PROCESSER

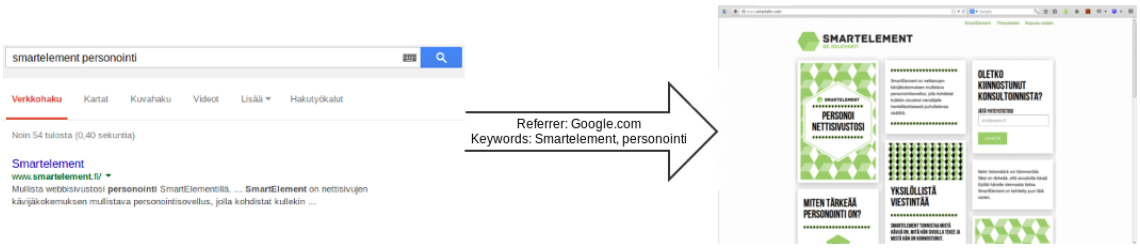
För att kunna leverera anpassat webbinehåll till besökaren av en webbsida, måste systemet ha ett sätt att om denne kunna identifiera information. SmartElement använder sig av några olika tekniker för att samla in information om besökare då de anländer på en webbsida, dels teknisk information och dels statistik.

Denna information används sedan för att söka igenom tillgängligt innehåll och testa om informationen passar för de filter som registrerats. Filtren bygger i sig på olika villkor som kan appliceras på den insamlade informationen.

5.1 Information om användaren

SmartElement använder sig till stor del av statistik som systemet samlar ihop med hjälp av en JavaScript tag som körs i besökarens webbläsare då denne besöker en sida som använder systemet. Denna information skickas till back-endsystemet som vidare processerar datan och sparar den i en dokument databas under ett besökarobjekt. På detta vis kan systemet komma ihåg användare mellan sessioner och på så vis skapa historik om användarens beteende och vanor, vilket i sin tur är till nytta för att hitta det bästa innehållet att presentera.

5.1.1 Hänvisningsinformation



Figur 4. Hänvisningsinformationen berättar varifrån beökaren kommer

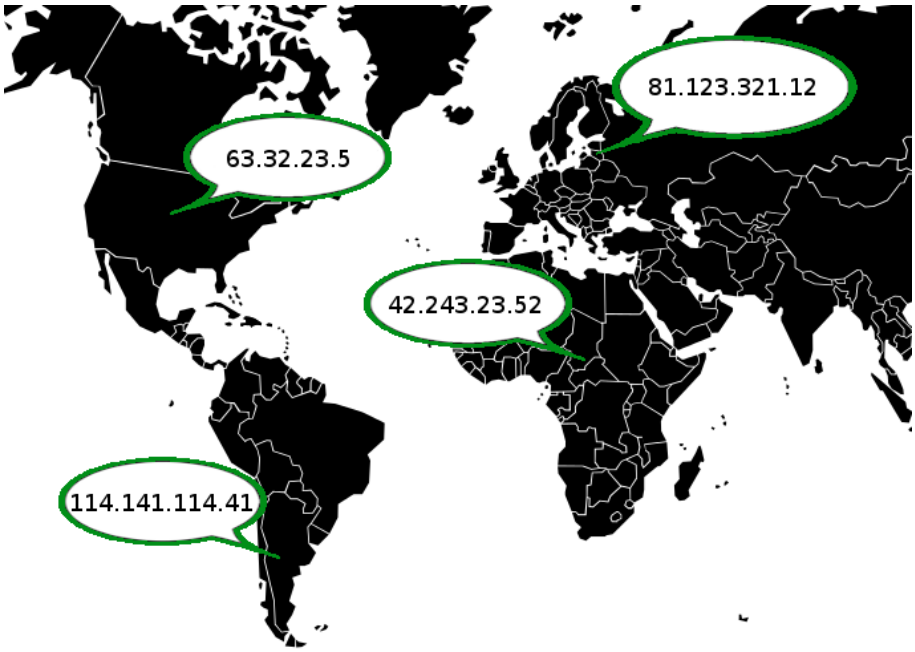
Den första informationen som registreras om användaren är varifrån denne anländer till sidan. Webbläsare skickar oftast en så kallad referrer variabel när man navigerar till en

webbsida genom att klicka på en länk. Denna variabel innehåller adressen på sidan som visade länken.

Genom att spara denna information får man dels en indikation av varifrån besökare hittar till sidan, men även en indikation av vilken typs sidor den aktuella besökaren ofta besöker.

En nackdel är att hänvisningsinformation inte alltid finns tillgänglig. Avsaknad av denna information ger en inte ett definitivt svar om hur användaren hittat till sidan. Detta på grund av att hänvisningsinformation inte skickas t.ex. när man navigerar till webbsidan från en sida med krypterad anslutning, om användarens webbläsare har konfigurerats att inte sända informationen eller om användaren helt enkelt skrivit in webbsidans adress i sin webbläsare och är en så kallad direkt träff. (Fielding et al. 1999)

5.1.2 IP Geolokalisering



Figur 5. IP-geolokalisering handlar om att associera plats med IP-address

Nästa bit av information som samlas in om besökaren är en uppskattning av dennes position på basis av den information som registrerats för IP-adressen som förfrågan kommer från.

SmartElement använder sig av en databas som köps in av ett företag som specialiserar sig i att uppehålla så noggrann information som möjligt om var i världen IP-adresser egentligen är registrerade. Från denna databas söker systemet sedan fram information om besökarens läge. Information som är tillgänglig är bland annat vilket land användaren befinner sig i, vilken region inom landet samt vilken stad.

Trots att informationen i databasen är av rätt hög kvalitet, kan man inte helt och hållet lite på information som genereras genom IP geolokalisering. Dels så är IP-addressen som sänds till servern inte nödvändigtvis besökarens egna IP-address då denne kan vara uppkopplad via en VPN-anslutning eller eventuellt använda sig av en proxy-server som inte vidare förmedlar ursprungsadressen. Detta betyder i sin tur att man får platsen var VPN- eller proxy-serverns IP är registrerad. Ett annat problem är att den information som finns tillgänglig beror på vad besökarens internet leverantör rapporterar. Även om IP-adressen är användarens egna så kan leverantören ha registrerat adressen på en annan plats än den var användaren befinner sig.

5.1.3 Besökarstatistik

För varje sidvisning på en webbsida med SmartElement sparas det statistik. Användarobjektet som sparas i systemet innehåller information om hur ofta besökaren varit på webbplatesen, hur många sidor denne besökt inom webbplatsen, samt hur ofta denne sett en specifik sida.

Genom att skapa statistik som är kopplad till användaren kan man generera information om användarens beteende då denne använder webbsidan. Man kan räkna hur ofta denne besöker specifika delar av sidan, om den inte sett en viss sida på länge, om den börjat besöka en specifik del mer eller mindre o.s.v. All denna information kan hjälpa att söka fram information som kan tänkas vara relevant för besökaren.

De största problemen med statistik är att den är beroende av möjligheten att identifiera besökaren då denne återvänder till webbsidan. Som systemet är byggt för tillfället hänger detta på användningen av kakor som innehåller ett unikt id-nummer, vilket betyder att sy-

stemet tappar användaren så fort denne raderar denna kaka. Det finns sätt att rundgå detta genom att använda sig av till exempel caching information för att identifiera besökare trots att denne raderat eller blockerat kakor. (Soltani 2011)

5.1.4 Tid

Vid varje sidvisning registreras tiden för besöket på basis av den tid som användarens webbläsare rapporterar. Utöver tiden för sidvisningen så beräknar javascript tagen hur lång tid användaren spenderat på sidan under det pågående besöket.

Valet att registrera tiden som rapporteras av webbläsaren gjordes för att undvika problem med olika tids-zoner och för att det är troligare att man vill göra ett beslut utgående från användarens tid, och eftersom tagen registrerar användarens tid så får systemet samtidigt möjligheten att registrera besökets längd.

5.1.5 Användardefinierad information

För att tillåta så flexibel användning som möjligt, tillåter SmartElement även att användaren själv definierar information som skickas till servern för processering. Detta sker genom att användaren lägger till egna dataelement i tagen då den inkluderas på sidan.

Genom att förse tagen med egen information kan upprätthållaren av en webbsida använda information son SmartElement inte i sig kan ta reda på, information som besökaren matar in på webbsidan eller information som uppräthållaren genererar.

Det går även att skapa statistik över användardefinierad information. Funktionen är den samma som för SmartElements egna statistik, man kan lägga till nya värden för varje sidvisning och sedan göra filtrering baserad på denna informationssamling.

Genom denna teknik kan användaren i princip använda vad helst information som är relevant i dennes fall. Exempel på data som kunde vara intressant att spara är shoppingvagnens innehåll i en webbutik, någon form av id för registrerade användare, aktiva kampan-

jer och annan information om webbsidans status vid sidvisningen. Systemet sätter ingen begränsning på vad som skickas förutom att det måste sändas i formen av en samling av värden med nycklar.

5.2 Filtrering

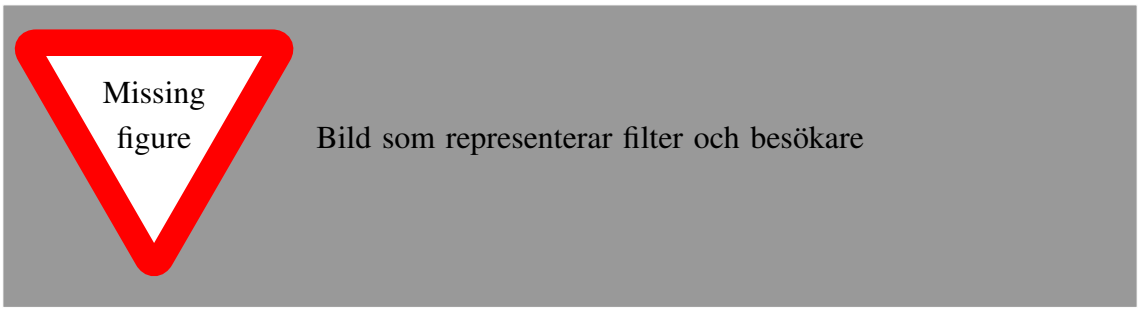


Bild som representerar filter och besökare

För att välja ut vilket innehålls-element som skall skickas till besökaren, använder sig systemet av informationen som samlats in och ett eller flera filter som användaren definierat för de olika elementen som registrerats i systemet. Filtersystemet bygger på användningen av enkla test som i kombination med olika typers data bildar en fråga om en användare.

5.2.1 Villkor

I grunden bygger SmartElements filter system på 12 enkla villkorsfunktioner som utför en jämförelse mellan den information tagen skickat och den information som användaren sparat i filtret.

De tolv villkoren som kan användas samt de datatyper de kan hantera listas i tabell 1.

Genom att kombinera dessa simpla villkor med den data som samlas in vid sidvisningar skapas filter som kan göra meningsfulla beslut om besökaren.

Namn	Funktion	Datatyp
Större än	Testar om värdet är större än det i filtret	Alfanumerisk
Mindre än	Negation av större än villkoret	Alfanumerisk
Lika med	Testar om värdet är lika med det i filtret	Godtycklig
Inte lika med	Negation av lika med villkoret	Godtycklig
I samling	Testar om samlingen som registrerats i filtret innehåller värdet som sänts	Samling
Icke i samling	Negation av "i samling"villkoret	Samling
Innehåller	Testar om värdet i filtret innehåller värdet som sändts	Text
Innehåller ej	Negation av innehåller villkoret	Text
Börjar med	Testar om värdet som skickats börjar med värdet i filtret	Text
Slutar med	Testar om värdet som skickats slutar med värdet i filtret	Text
Tom	Testar om värdet som skickats är tomt	Text, Samling
Icke tom	Negation av tomhetsfiltret	Text, Samling

Tabell 1. Villkorstyper

Under utvecklingen av SmartElement valdes några färdiga filtrerings-metoder ut för implementering i systemet som en grund för användare att börja bygga sin anpassning. Genom att studera den tillgängliga informationen valdes några frågor ut som var lätta att identifiera. Dessa utgör de inbyggda filtren i SmartElement.

5.2.2 Stadsfiltret

Stadsfiltret använder sig av informationen som samlats in genom IP-geolokalisering och tillåter användaren att visa specifikt innehåll för besökare från en viss stad.

Motivationen för att implementera stadsfiltret kom från ett användningsfall var användaren kan köra kampanjer för olika kontor i olika städer eller visa kontorsspecifika öppethållningstider.

5.2.3 Landfiltret

Landsfiltret använder sig av IP-geolokaliseringsinformationen för att tillåta användare att avgränsa innehåll på basis av vilket land besökare kommer ifrån.

Motivationen bakom landsfiltret var att tillåta dels större företag, som sträcker sig över landsgränser, att visa landsspecifik information på sin webbsida, samt mindre företag att välja att visa mera specifik information för inhemska besökare, och en mera generell information för internationella besökare.

5.2.4 Datumfiltret

Datumfiltret använder sig av dagens datum för att tillåta användaren att specificera när ett innehåll skall visas. Filtret kan användas med de olika jämförelse-villkoren. Användaren kan specificera ett datum efter vilket innehåll skall visas eller ett datum före vilket det skall visas. Med lika med villkoret kan man specificera en enda dag då innehållet skall visas och lika så en specifik dag då det inte skall visas.

Motivationen bakom datumfiltret var att ge användare möjligheten att visa kampanjinnehåll under en specifik tid genom att kombinera två datum filter, ett med en nedre gräns och ett med en övre.

5.2.5 Dagsfiltret

Dagsfiltret använder sig även av datumet för filtrering. Till skillnad från datumfiltret så tar dagsfiltret emot en samling av dagar då ett innehåll skall visas. Filtret kan antingen inkludera dagar eller exkludera dagar.

Användningsfallet som ledde till dagsfiltret var att en användare vill visa en tabell med öppettider under veckan men ett anpassat element under veckoslutet som föreslår att besökaren gör en beställning genom en webb-butik.

5.2.6 Besökstidsfiltret

Besökstidsfiltret använder sig av tiden en besökare spenderat på webbsidan för att filtrera innehåll. Filtret använder de olika jämförande villkoren för att tillåta användaren att visa innehåll för användare som spenderat en specifik tid på sidan. Man kan ställa in en nedre

gräns, en övre gräns och även en exakt tid då innehåll skall visas eller döljas.

Användningsfallen för besökstidsfiltret var att användaren vill skapa ett element som visar ett specialerbjudande för användare som har spenderat en längre tid på webbsidan men inte beställt något samt att användaren vill visa nyheter för användare som nyss anlänt.

5.2.7 Nyckelordsfiltret

Nyckelordsfiltret använder sig av hänvisningsinformationen för att filtrera ut vilka nyckelord som använts i en sök-motor om besökaren nått sidan via en sådan. Filtret tillåter användaren att specificera grupper av nyckelord som antingen skall eller inte skall vara bland de som använts.

Användningsfallet bakom nyckelordsfiltret är en att en användare väljer att lyfta fram speciell information som passar in på den webbsökning som besökaren använt för att hitta till webbplatsen.

5.2.8 Landningssidefiltret

Landningssidefiltret använder sig av statistikinformationen för att välja ut anpassat innehåll. Funktionaliteten fungerar genom att passa den första sidan under det aktiva besöket mot en grupp som den antingen skall eller inte skall finnas bland.

Användningsfallet för landningssidefiltret är att en användare använder sig av en marknadsförings webbadress som leder till en speciell sida. Efter detta kan elementen på sidan återspegla kampanjen.

5.2.9 Sidantalsfiltret

Sidantalsfiltret använder sig av statistikinformationen för att ge användaren möjlighet att skapa filter som använder antalet av sidor som visats under det aktiva besöket för att visa

eller dölja innehåll.

Användningsfallet för sidantalsfiltret är en användare som vill visa ett element som ändrar med antalet sidor som visats och på så vis ger information som passar med besökarens bekantskap med sidan och på samma gång ger en levande bild av informationen.

5.2.10 Hänvisarfiltret

Hänvisarfiltret använder sig av hänvisningsinformationen för att tillåta användaren att skapa filter som agerar på grupper av hänvisare och på så vis tillåter innehåll att kopplas till de sidor som sänt besökaren till webbsidan.

Användningsfallet för hänvisarfiltret är att användaren vill skapa ett element som visar speciellt innehåll för besökare som når dennes webbsida genom en av deras samarbetspartners.

5.2.11 Regionsfiltret

Regionsfiltret använder sig av IP-geolokaliseringsinformation för att tillåta användaren att visa innehåll anpassat till användarens region. I Finland kan man t.ex. specificera att ett innehåll skall visas endast om besökaren befinner sig i södra Finland.

Användningsfallet är att användaren har regionalkampanjer som använder samma element på sidan för att visa relevanta information på basis av var besökaren befinner sig.

5.2.12 Tidsfiltret

Tidsfiltret använder den aktuella tiden för att välja ut vilket innehåll som skall visas. Användaren kan skapa filter utgående från villkoren större än, mindre än, lika med och icke lika med för att skapa filter som visar olika innehåll olika tider av dagen.

Användningsfallet för tidsfiltret är att användaren vill visa speciell information när en

butik är öppen, när den håller på att stänga och när den är stängd.

5.2.13 Besöksfiltret

Besöksfiltret använder sig av statistikinformationen för att tillåta filtrering baserat på antalet besök som gjorts till webbsidan. Med hjälp av större än, mindre än, lika med och icke lika med villkoren kan användaren skapa filter som visar innehåll på basis denna statistik.

Användningsfallet för besöksfiltret är att användaren vill visa information som passar besökarens lojalitet till webbsidan. En användare som ofta återkommer kan se ett innehåll som reflekterar detta.

5.2.14 Anpassningsbart filter

Den sista filterklassen är det anpassningsbara filtret vilket tillåter användaren att skapa filter helt baserat på sina egna behov. Filtret tillåter användning av godtycklig villkorsfunktion med godtycklig data. Beroende på vilken villkorsfunktion som konfigureras kan det uppstå begränsningar på vilken typs data som kan användas för filtrering. Detta filter ger SmartElement möjligheten implementera ny filtreringsfunktionalitet genom konfiguration.

Användningsfallet för det anpassningsbara filtret är att användaren kan skapa ett filter helt på basis av sina egna behov genom att specificera vilken data som skall användas och hur den skall användas vid filtrering.

6 MJUKVARUARKITEKTUR

Arkitekturen designades utgående samma principer som den tekniska arkitekturen. Systemet har designats utgående ifrån skalbarhet. Det vill säga att systemet skall vara lätt att utveckla funktionsmässigt, det skall vara lätt att underhålla systemet och det skall vara lätt att öka dess kapacitet vid behov. (Henderson 2006 s. 203)

6.1 Horisontell skalbarhet

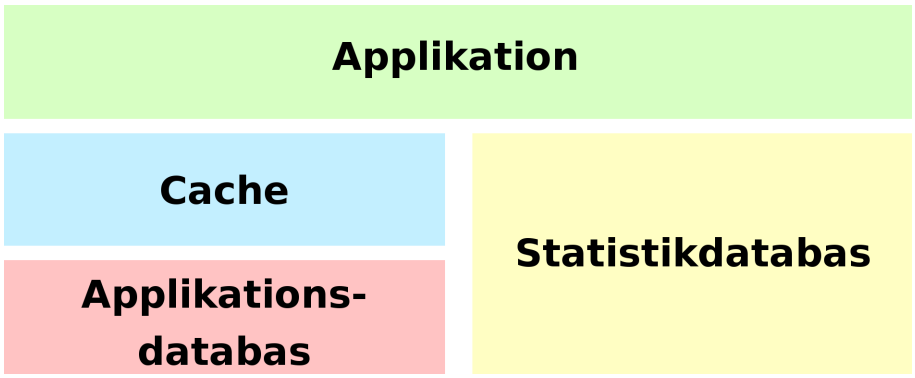
För att tillåta systemet att adapteras till växande resurskrav, har SmartElement designats för att tillåta sig skala horisontellt. Att ett system är horisontellt skalbart betyder att man kan öka systemets kapacitet genom att öka antalet av någon komponent och sprida ut belastningen.(Henderson 2006 s. 205-207)

I SmartElements fall betydde det att mjukvaruarkitekturen måste vara tillräckligt flexibel för att kunna köras på flera instanser. Detta ledde till en modulär design som försöker separera ansvarsområden inom mjukvaran. På så vis att de olika delarna kan byggas ut oberoende av varandra.

Den tyngsta delen av applikationen, vad gäller skalbarhet, är applikationsdatabasen. På grund av detta designades processer som är menade att skydda detta lager genom att spara objekt i cache minne. På så vis hålls databas antalet förfrågningarna lägre och applikationsdatabasen behöver inte skala lika mycket för att systemet skall klara större mängder förfrågningar.

6.2 Servern

Back-end systemet, eller serverkomponeneten, i SmartElement är den centrala delen av systemet var all data lagras och all filtrering sker. Systemet har två gränssnitt, ett öppet som används av tagen för att hämta innehåll vid sidvisningar, och ett stängt som används av en frontend vid konfiguration.



Figur 6. Delarna av serverkomponenten

Hela back-endsystemet är designat med en fokus på att vara tillståndslöst (jmfr. engelskans *"stateless"*) vad gäller hanteringen av förfrågningar. I praktiken betyder detta att man inte skall behöva en session med back-enden för att kunna genomföra förfrågningar, utan all information som behövs förses antingen i förfrågan, eller så kan den generaras från databasen. Detta betyder att t.ex. autentikationen är baserad på en nyckel som används för att generera en signatur för varje förfrågan. Orsaken till detta ligger i att det är lättare att skala systemet horisontellt, genom att lägga till applikationsservrar, då sessioner inte behöver synkroniseras mellan de olika servrarna.

Tekniskt sett så finns det fyra huvudsakliga komponenter i back-endsystemet som kan skiljas åt: applikationen, caching-lagret, applikations-databasen och statistik-databasen. Denna uppdelning är gjord för att försöka tillåta för oberoende horisontell skalning av resurserna för de olika ansvarsområdena genom att lägga till eller ta bort instanser av de olika komponenterna i takt med behov.

6.3 PHP-ramverket Laravel

För att snabba upp utvecklingen av systemet användes ett utvecklingsramverk. Ett utvecklingsramverk är en samling av kodpaket som används som bas vid utvecklingen av en applikation och förser utvecklaren med funktionalitet som inte varierar mycket mellan applikationer.

Ramverket som används för att utveckla SmartElement är PHP-ramverket Laravel. Det är

ett yngre ramverk som utvecklats med komponenter från Symfony-ramverket, ett väletablerat PHP-ramverk som dock kan vara rätt tungt. Laravel drar även full nytta av Composer pakethanteraren, vilket tillåter lätt importering av kodpaket i projektet. Utöver detta är hela ramverket uppbyggt för att vara modulärt och tillåter således modifikation och hel ersättning av komponenter.

Laravel-ramverket bygger på ett Model-View-Controller (MVC) mönster i sin arkitektur. Det vill säga att ramverket delar upp ansvar i olika komponenter: datalager (Model), presentationslager (View) och kontrolllager (Controller). Datalagret ansvarar för applikationslogik och data, presentationslagret ansvarar för att presentera data och kontrolllagret ansvarar för att binda ihop presentations- och datalagret.(Gamma et al. 1994 s. 14-16)

6.4 Komponenter i systemet

SmartElement består egentligen av endast några få centrala komponenter som tillsammans skapar delarna i identifikation av användare och filtrering av innehåll. Här listat dessa centrala komponenter och deras egenskaper.

6.4.1 Sidvisningshanteraren

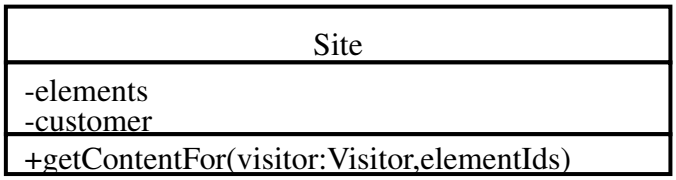
Den centrala delen av hanteringen av en sidvisning sker i en hanterare som tar emot information om besökaren och söker fram det innehåll som skall skickas tillbaka. Rent tekniskt så tar denna komponent emot förfrågan från besökarens webbläsare, synkroniserar informationen med statistikdatabasen, söker fram sidobjektet ur cachinglagret (alternativt bygger det från databasen), använder sidbjektet för att generera det innehåll som skall sändas tillbaka och formaterar ett svar som skickas tillbaka till klienten.

6.4.2 Besökarobjektet

Besökarobjektet representerar en besökare på webbsidan och innehåller all information som systemet kunnat samla ihop. Besökarobjektet är det enda objekt i filtreringsprocessen som är designat att röra databasen.

När en förfrågan kommer in, kallas en metod som försöker hitta besökaren i statistikdatabasen. Om ett resultat hittas, uppdaterars det befintliga objektet med den nya informationen från förfrågningen. Om besökaren inte är igenkänd så skapas ett nytt besökarobjekt med den försedda informationen.

6.4.3 Siteobjektet

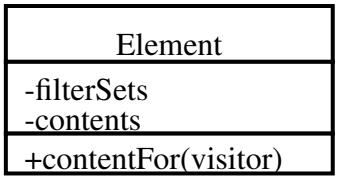


Sidobjektet är det högsta objektet i hierarkin som används för att filtrera ut innehåll. det beskriver användarens webbplats i sin helhet. Från sidobjektet går alla relationer enda ner till filtren som används för att välja ut innehåll.

När matchern samlar ihop data för att skicka tillbaka till klienten så anropar den på siteobjektet med ett besökarobjekt och därefter sköter siteobjektet om attanropapå de olika elementen som sparats under sidan och ber dessa filtrera ut lämpligt innehåll innan det lämnar tillbaka kontrollen till matchern.

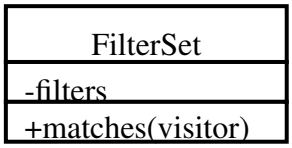
Objektet är designat så att det kan kompileras till en entitet som går att spara i ett cachinglager för kunna hålla objekten i minne vid operation. Detta tillåter snabb åtkomst till data under en sidvisning och sparar på databasförfrågningar.

6.4.4 Elementobjekten



Element objekten motsvarar ett HTML element på användarens webbplats, men de är ej bundna till någon specifik webbadress eller sida på webbplatsen. Detta betyder att man kan använda samma element på olika sidor under samma webbplats om man vill, t.ex. i en side-bar eller som del av en navigation.

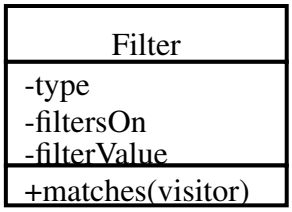
6.4.5 FilterSetobjekten



Ett element innehåller filterset, innehåll och ett status. Filterseten är samlingar av filter med en länk till ett innehållsobjekt samt en prioritet inom elementet. När sidobjektet anropar ett element för att få tillbaka det passliga innehållet kör elementet igenom sin samling filter set och ser om de passar besökarobjektet. Om fler en ett filterset passar så väljs det med högsta prioritet. När ett filterset valts ut så returnerar elementet det innehåll som filtersetet är länkat till. Om inget filterset passar så returneras ett tomt resultat.

Filterseten bygger tillsammans med filtren på en design princip som kallas för ”Strategy pattern”. Idén är att man genom att definiera ett bra gränssnitt för en process, lämnar systemet öppet för enkel vidareutveckling i ett senare skede. (Gamma et al. 1994 s. 349) FilterSet objektet i samband med Besökar objektet utgör tillsammans det kontext i vilket strategierna (Filtren) appliceras.

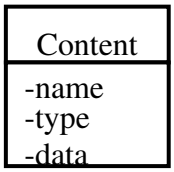
6.4.6 Filterobjekten



Filterobjekten är i slutändan de objekt som utför test mot besökarobjektet. De består av en filtertyp som definierar hur de jämför data, ett fält-id som de använder för att läsa ut data ur besökarobjektet, ett värde som de testar mot och ett villkor som de använder för att utföra testet.

Filterobjekten implementerar strategin i filtreringen som utförs i samband med filterseten. Filter tar endast ställning till en sak och returnerar ett booleanskt värde beroende på om kriteriet möts eller ej. Filtersetet vet endast att det kan ställa frågan av godtyckligt filter och filtret i sig vet att frågan alltid kommer att ställas på samma vis med likadan input.

6.4.7 Contentobjekten



Content objektet är den data som returneras till klienten efter att filtreringen skett. Systemet sätter ingen restriktion på datan annan än att det måste kunna sparas som en textsträng. Detta betyder att man kan producera avancerad funktionalitet genom att t.ex. spara javascript-kod eller JSON-data som innehåll och på så vis använda SmartElement för att styra exekvering på klienten. Det betyder också att systemet inte sätter så stora krav på anroparen eftersom det enda som är viktigt är att klienten kan tolka datan som returneras. Detta betyder att man i princip kunde skapa godtycklig klient till back-enden genom att implementera protokollet som tagen använder.

bilaga med protokoll?

6.5 JavaScript-tagen

JavaScript tagen är en kort kod som laddas i samband med sidan och samlar ihop information om besökaren och sedan skickar denna information till back-endsystemet för processering.

Tagen försöker hitta en kaka i besökarens webbläsare, om ingen finns så skapas en ny unik id och sparas i en ny kaka. Efter att tagen fått en id, läser den ut information från besökarens webbläsare. Den läser bl.a. hänvisarinformationen och extraherar sökord om hänvisaren varit en sökmotor. Tagen kan även räkna ut tiden för besöket med hjälp av en tid som sparas i kakan då den skapas. Om sidvisningen är den första i besöket läser tagen addressfältet och använder värdet som landningssida för besöket. Efter att tagen samlat ihop sin information och en kaka lästs eller skapats, skickar den iväg informationen till back-enden.

Efter att back-enden processerat förfrågan och returnerat data för de element som begärts så är standardbeteendet att back-enden genom ett JSONP-svar anropar tagen som uppdaterar dokumentet med det returnerade innehållet. Vilken funktion som kallas kan dock ändras och man kan genom att specificera en callback ändra på funktionskallet i JSONP-svaret och på så vis använda egen logik för uppdateringen av innehållet.

6.6 Begränsningar

Systemet utvecklades med relativt låg budget vad gäller både tid och pengar. Därigenom finns en del begränsningar på tekniken och hårdvaran som fanns tillgänglig. Detta återspeglar sig i en del begränsningar i själva systemet samt i funktionalitet som inte implementerats.

För tillfället sparas statistik endast som en lista av adresser som besökts. En mer avancerad modell skulle kunna aggregera data från denna lista och ge flere filtreringsmöjligheter. Statistikdatan behandlas endast i samband med besökarobjektet som systemet fungerar nu, genom att aggregera statistik för hela sidan skulle man kunna skapa rekommendatio-

ner för upprätthållaren vad gäller olika besökarkretsar.

Gränssnittet för leverering av innehåll är i dagens läge ennu ganska hårt bundet till tagen. Ett mer generellt API-skulle gagna systemet genom att tillåta enklare integrering i utomstående system.

7 SAMMANFATTNING OCH DISKUSSION

SmartElements arkitektur har så här långt uppfyllt de krav som sattes upp för systemt. Den tillåter filtrering av innehåll på basis av information som samlas in vid ett besök, vilket var det mest grundläggande målet för systemet. Den tekniska arkitekturen är lätt och flexibel, den tillåter systemet att enkelt utvidgas upp och ner i enlighet med behov. Mjukvaruarkitekturen möter behoven på mjukvaran i det skede som den är och systemet är öppet för vidareutveckling.

7.1 Idéer för vidare utveckling

SmartElement har genom projektet utvecklats till ett MVP-stadium (Minimum-Viable-Produkt). Det stöder den mest grundläggande funktionaliteten som krävs. Det finns dock en del områden var systemet skulle kunna utvecklas och tekniker som skulle kunna användas för förbättring av systemet.

7.1.1 Poängsättning av material

Ett system för poängsättning av material som besökaren ser har planerats men inte implementerats. Genom att använda anpassad information och de anpassade filtren kan man dock i dagens läge skapa en rudimentär implementation av ett poängsättningssystem. Med dedikerat stöd för poängsättning skulle användare kunna definiera kundsegment baserat på olika mätare och systemet skulle kunna använda statistikinformationen för att klassificera besökare som del av dessa.

7.1.2 Förutsägning

I dagens läge är SmartElements filtreringssystem reaktivt. Det reagerar på historisk information. Genom att koppla in ett system för förutsägning skulle man kunna förutse relevans. Systemet samlar redan in en del data som skulle kunna användas för förutsägning av relevans och genom att träna ett system med information specifik för en webbsida, kunde

man potentiellt förutsäga vilka innehåll en ny besökare potentiellt är intresserad av.

TODO LIST

KÄLLOR

Albanese, Massimiliano; Picariello, Antonio; Sansone, Carlo & Sansone, Lucio. 2004, Web Personalization Based on Static Information and Dynamic User Behavior, I: *Proceedings of the 6th Annual ACM International Workshop on Web Information and Data Management*, WIDM '04, New York, NY, USA: ACM, s. 80–87. Tillgänglig: http://doi.acm.org/10.1145/1031453.1031469.

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. & Stein, Clifford. 1990, *Introduction to Algorithms*, Massachusetts: MIT press.

2014, *DB-Engines Ranking*. Tillgänglig: http://db-engines.com/en/ranking, Hämtad: 2.5.2014.

Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P. & Berners-Lee, T. 1999, *RFC 2616, Hypertext Transfer Protocol – HTTP/1.1*. Tillgänglig: http://www.rfc.net/rfc2616.html.

Gamma, Erich; Helm, Richard; Johnson, Ralph & Vlissides, John. 1994, *Design Patterns: Elements of Reusable Object-Oriented Software*, Massachusetts: Addison-Wesley Professional.

Guy, Ido; Zwerdling, Naama; Ronen, Inbal; Carmel, David & Uziel, Erel. 2010, Social Media Recommendation based on People and Tags, I: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, New York: ACM, s. 194–201.

Henderson, Cal. 2006, *Building Scalable Web Sites: Building, Scaling, and Optimixing the Next Generation of Wb Applications*, Sebastopol: O'Reilly Media, Ltd.

Hunt, Andrew & Thomas, David. 1999, *The Pragmatic Programmer*, Massachusetts: Addison-Wesley Professional.

2014, *Memcached Wiki*. Tillgänglig: https://code.google.com/p/memcached/wiki/NewOverview, Hämtad: 2.5.2014.

MongoDB Manual - Querying Documents. Tillgänglig: http://docs.mongodb.org/manual/tutorial/query-documents, Hämtad: 6.5.2014.

Mulvenna, Maurice D.; Anand, Sarabjot S. & Büchner, Alex G. 2000, Personalization on the Net using Web mining: introduction, *Communications of the ACM*, årg. 43, , s. 123–125.

2014, *Nosto Website*. Tillgänglig: http://www.nosto.com, Hämtad: 23.3.2014.

2013, *PHP Manual*. Tillgänglig: http://fi2.php.net/manual/en/, Hämtad: 2.3.2014.

Pimcore - On-site Behavioral Targeting and Personalization Platform. Tillgänglig: http://www.pimcore.org/en/product/targeting-personalization, Hämtad: 6.5.2014.

Schwartz, Baron; Zaitsev, Peter; Vadim Tkachenko, Jeremy D. Zawodny; Lentz, Arjen & Balling, Derek J. 2008, *High performance MySql*, 2 uppl., Sebastopol: O'Reilly Media, Ltd.

Soltani, Ashkan. 2011, *Flash Cookies and Privacy II*. Tillgänglig: http://ashkansoltani.org/2011/08/11/respawn-redux-flash-cookies/, Hämtad: 2.5.2014.

2014, *TIOBE Programming community index*. Tillgänglig: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, Hämtad: 26.4.2014.