# 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 arbete är att redogöra för implementationen av ett system för automatiserad anpassning av webbinnehåll.
Arbetet presenterar i korthet anpassning av webbinnehåll och presenterar sedan systemet SmartElement som utvecklats för att tillåta webbsideupprätthållare att enkelt kunna implementera anpassat innehåll på sin sida.
Olika sätt att göra bedömningar om besökare på en webbsida studeras. Genom att redogöra vilken sorts information systemet kan samla in om en besökare definieras filtertyper som kan användas för att filtrera innehåll på basis av informationen.
Utöver detta redogörs det för den tekniska arkitektur som valts ut för systemet, samt hur de val som gjorts vid planeringen stöder systemets funktionalitet.
Slutligen diskuteras några områden var systemet skulle kunna vidareutvecklas.



| 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.
The thesis gives a brief introduction to the concept of website personalization and presents the SmartElement system, a tool that has been developed to allow website administrators to easily utilize personalization of their content.
The thesis takes a look at what kind of deductions can be made about a visitor. By defining a list of information that the system can collect about visitors, a list of filters is defined, which can be used for filtering the content on a website.
The technical architecture behind the system is also presented. Some of the choices made during the design of the architecture, and the impact of these choices, are also presented.
Finally a list of proposed areas of furher development is made, documenting areas where the system could be improved.

| Keywords: | Personalization, content-delivery, analytics, websites |
|---|---|
| Number of pages: | TBD |
| Language: | Swedish |
| Date of acceptance: | TBD |

# INNEHÅLL

# TABELLER



# FIGURER

# FÖRKORTNINGAR

**API** Application Programming Interface. 7, 16, 39

**DOM** Document Object Model. 7, 11

**JSON** JavaScript Object Notation. 7, 11, 13, 15, 16, 37

**JSONP** Json-with-padding. 7, 11, 38

**MVC** Model-View-Controller. 7, 34

**SaaS** Software-as-a-Service. 7, 9

**SQL** Structured Query Language. 7, 19, 20

# 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 innehållsfiltrering och leverans. Systemet är även lätt att vidareutveckla 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 informationens pålitligheten beaktas. Utgående från den information som samlas in definieras 14 filtertyper som implementerats i systemet. Dessutom beskrivs hur filtren 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 arbete 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 prestandaanalyser av systemet.

# 2 BAKGRUND

Projektet började hösten 2012 som ett internt verktyg på Developer's Helsinki efter att behov uppstått för ett system som skulle tillåta snabb implementation av anpassat innehåll på kunders webbsidor. Det bestämdes att det skulle utvecklas en mjukvara först för internt bruk men med målsättningen att lanseras som en SaaS-produkt. Mjukvaran skulle utvecklas för att vara möjligast anpassningsbar och tillåta flera olika sorters anpassningssätt. Utvecklingen genomfördes med ganska låg budget mätt både i tid och pengar, med fokus på att få ut en Minimum-Viable-Product innan företaget binder sig till den.

## 2.1 Developer's Helsinki

Developer's Helsinki Oy är ett finskt företag som utvecklar såväl applikationstjänster (jmfr. engelskans Software-as-a-Service (SaaS)) som webbsidor. Företaget grundades 2009 för att vidareutveckla SaaS-tjänsten Netmonitor, en produktfamilj bestående 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 använts inom nätbutiker för att visa reklam anpassad för kunden på basis av dennes beställningshistorik. Figur 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)

Anpassning av innehåll kan anses ha börjat med de första webbsidorna som tillåtit inloggning och på så vis anpassat webbsidan för en återvändande användare. Man kan anse att så lite som en användarinformationsbox för inloggade användare är anpassning av innehåll. I dagens läge menar man oftast mera avancerad anpassning baserad på segmentering av användare och klassificering av innehåll.

En för det flesta bekant form av anpassning av webbinnehåll är anpassade sökresultat.

Söktjänsten Google har sedan 2004 använt sig av anpassade sökresultat. Sedan 2009 anpassas sökresultat dessutom för icke autentiserade besökare (Hannak et al. 2013). Googles anpassade sökresultat kan ses som en föregångare för anpassat webbinnehåll, de har presenterat konceptet för en bred publik.

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.

# 3 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 kommersiella produkter som open-

sourceprojekt som erbjuder olika typer av anpassningssystem. På open-sourcefronten handlar det oftast om webbinnehållsplattformer som innehåller en anpassningsmodul som en del av helheten. På den kommersiella sidan finns det många lösningar för optimering av webbutiker genom anpassning av innehållet på sidan.

## 3.1 Kommersiella lösningar

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 utformat 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.

Den kommersiella lösning som närmast liknar SmartElement i funktionalitet är Personyze. Systemet bygger på en liknande lösning som SmartElement var upprätthållaren av en webbsida skapar regler för vilka besökarsegment som skall se specifik information och lägger till en kort JavaScript-kod i sin sida vilken sedan hämtar innehåll som passar användaren. (Personyze 2014)

## 3.2 Lösningar baserade på öppen källkod

På open-sourcefronten finns det bl.a. webbinnehållsplattformen 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 2014) Till skillnad från SmartElement är Pimcore en hel plattform för hantering av webbinnehåll. För att använda plattformen 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.

Ett mera tekniskt krävande system för utveckling av anpassat innehåll är PredictionIO. Det är en maskinlärningssystem baserat på öppen källkod som kan användas för att automatiskt generera anpassade rekommendationer för besökare. Systemet bygger på att man skapar en motor som sedan tränas genom att mata in beteende data om användare i relation till olika ting. (PredictionIO - Concepts 2014)

# 4 SMARTELEMENT

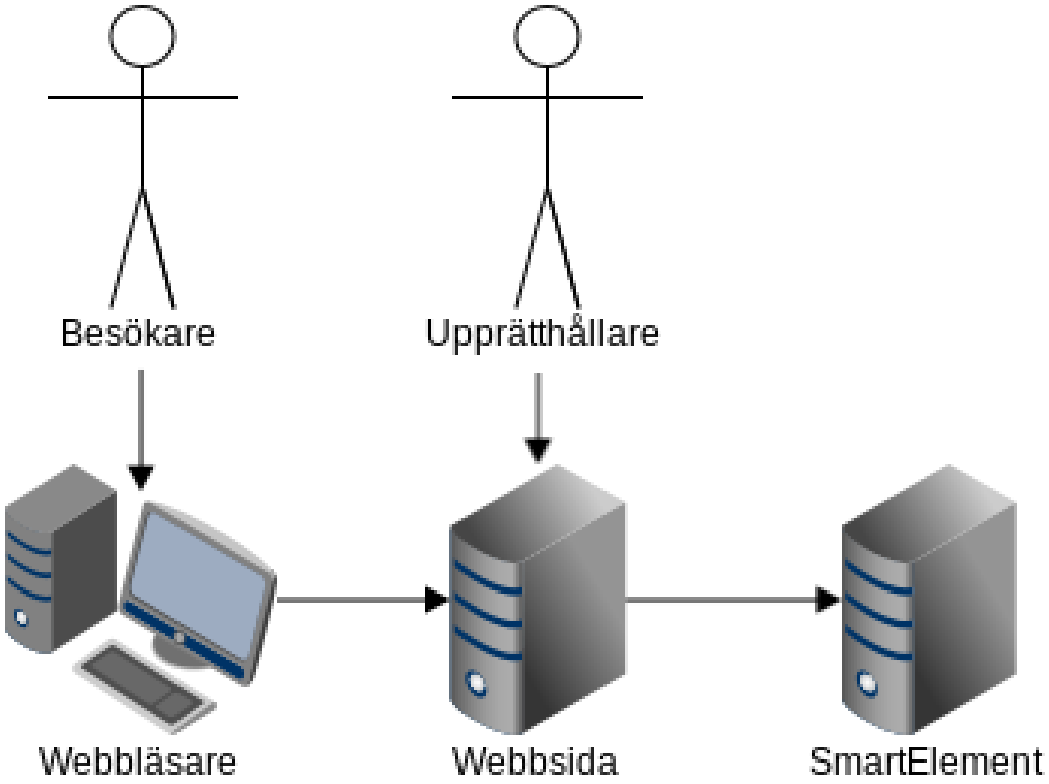
SmartElement är ett back-endsystem som sköter filtrering av innehåll på basis av information som samlas in om besökare på en webbsida. Upprätthållaren av en webbsida kan genom att integrera SmartElements JavaScript-tag i sin sida låta back-endsystemet fylla i specificerade element med anpassat innehåll.

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.

## 4.1 Aktörer

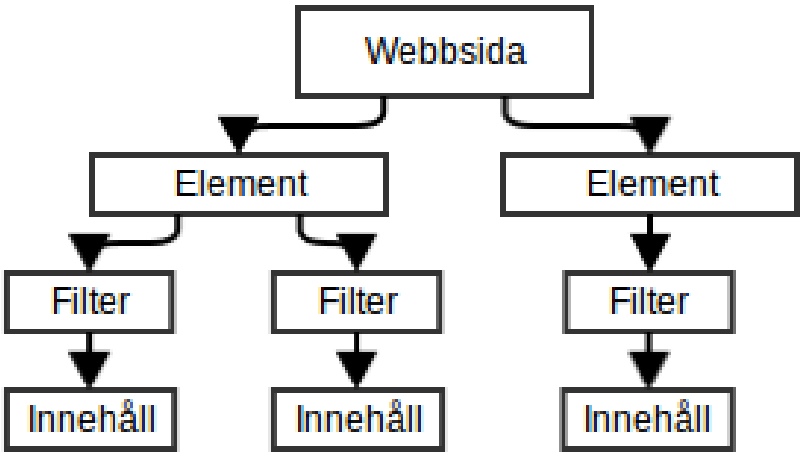
Genom detta arbete används några aktörer, visualiserade i Figur 2, för att förklara sammanhanget kring SmartElement. Närmast handlar det om besökaren, besökarens webbläsare, en webbsida, webbsidans upprätthållare samt SmartElement systemet.

Besökaren är den person som med sin webbläsare besöker webbsidan. Upprätthållaren är den person som upprätthåller en webbsida, på vilken denne installerat SmartElement-tagen. Med SmartElement ämnas, om ej annat nämns, back-endsystemet av SmartElement.




*Figur 2. Aktörer*

## 4.2 Beståndsdelar




*Figur 3. 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 3, som back-endsystemet använder för att filtrera och leverera innehåll.

### 4.2.1 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.

### 4.2.2 Element

Elementen är hållare för den data som returneras från back-endsystemet, de knyter innehållet till webbsidan. Elementet har i sig flere 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-endsystemet.

### 4.2.3 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 passar. För att ett filter skall passa krävs det att besökaren passar för alla regler i filtret. För att hantera fall var två eller flera filter passar för en besökare, har filter inom elementet en prioritetsordning som användaren själv kan definiera.

### 4.2.4 Innehåll

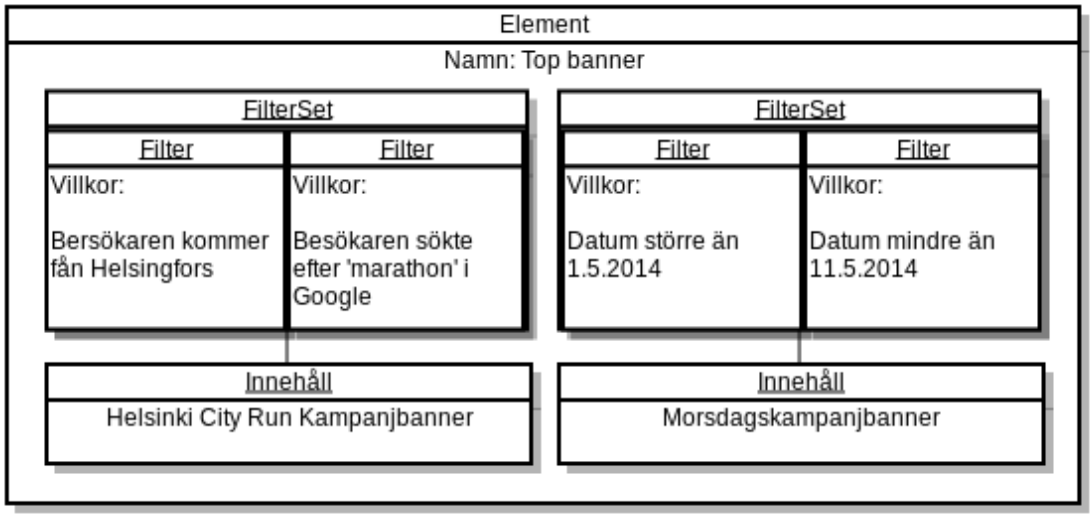
Innehåll är den konkreta data som levereras för ett element efter att ett filter valts som vinnare. Innehållet kan bestå av text, HTML-kod eller JSON-data, hurudan som helst

data som kan sparas som en textsträng. Det är upp till användaren att definiera vad för data som sparas.

## 4.3 Funktion

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

### 4.3.1 Konfiguration




*Figur 4. Exempel på en elementkonfiguration*

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. En färdig elementkonfiguration visualiseras i Figur 4.

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

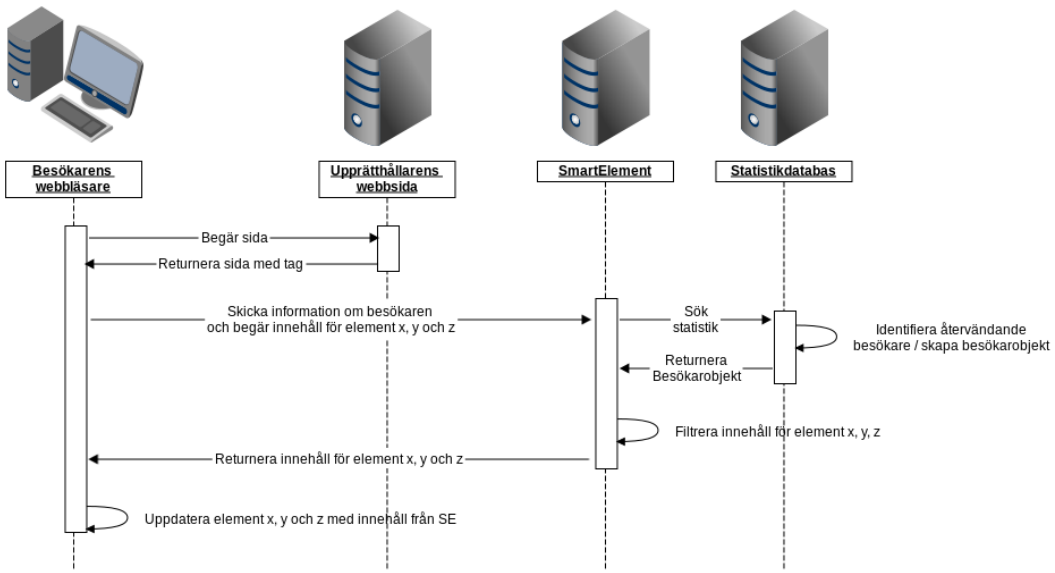
Konfigurationsprocessen börjar med att registrera en webbsida i systemet. En användare

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 nycklar som används för att representera objekt som kan se ut på olika sätt (ha olika innehåll) beroende på de filter som associeras med dem.

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 passa för att ett innehåll skall visas. Genom dessa filterkombinationer kan avancerade villkor implementeras.

### 4.3.2 Sidvisning




*Figur 5. Processen vid en sidvisning*

Figur 5 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.

På back-enden kombineras informationen från tagen med eventuell statistikdata från en statistikdatabas för att skapa ett besökarobjekt.

Backenden tar emot besökarobjektet 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 datan i besökarobjektet. 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.

# 5 TEKNISK ARKITEKTUR

SmartElement bygger i stort sett på fem centrala tekniker: programmeringsspråket PHP som används för att skriva systemets kod, MySQL som databaslager för användar- och systemdata, Memcache som cache-lager, vilket tillåter att information som är tung att beräkna kan sparas mellan förfrågningar, dokumentdatabasen 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 göra systemet skalbart för hantering av 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.

## 5.1 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). Det finns en mängd information och färdiga kodpaket till förfogande för programmerare, vilket underlättar utvecklingen och gör den snabbare. Genom sin popularitet har språket även den fördelen att det är lätt att hitta utvecklare som kan fortsätta utvecklingen av projekt skrivna i PHP.

## 5.2 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 dennes sida fungerar, så finns det två tydliga data-set som båda har egna krav.

På grund av denna indelning var det möjligt att använda två skilda databaser, var med sitt ansvarsområde. Databaslagret 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.

### 5.2.1 MySQL

MySQL är en relationsdatabashanterare som sköter om lagring av data samt modifikation och sökning genom språket Structured Query Language (SQL). MySQL utvecklades som ett öppen-källkodsprojekt 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 datastrukturen 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.

### 5.2.2 MongoDB

MongoDB är en dokumentdatabas som utvecklas av företaget MongoDB, Inc. Databasen är en av de s.k. NoSQL-databaserna, vilket är en term som används för att beteckna databaser som inte använder sig av SQL-språket för datamanipulering. 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 2014)

För lagring av extern data, relaterad till besök på upprätthållarens 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 tillåta gammal data att existera i databasen tillsammans med ny data tills den gamla uppdateras och kompletteras då en besökare återvänder till webbsidan.

I fallet av den externa datan 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 besökarobjekt, som är den snabbast växande samlingen inom systemet.

## 5.3 Memcached

Memcached är ett objektcachesystem, en mjukvara som tillåter program att spara objekt i minnet och senare läsa tillbaka dem utan att behöva bygga upp datastrukturer igen. Systemet fungerar som ett nyckel-värdesystem, dvs. att information sparas i en nivå med unika nycklar som identitetsvärden, 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 databasförfrågan för varje element i en samling som man itererar över), så bestämdes det att ett cachelager skulle användas för att lagra dessa datamodeller, och på så vis låta mjukvaran gå direkt till filtreringslogiken 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 skalbarhet skulle tillåta systemet att snabbt reagera på en växande kundskara.

## 5.4 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 exekvera på besökarens dator i samband med en sidvisning.

# 6 PROCESSER

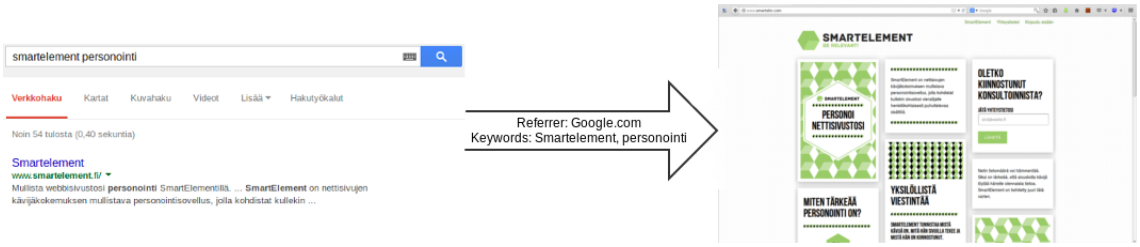
För att kunna leverera anpassat webbinehåll till besökaren av en webbsida, måste systemet ha ett sätt att kunna identifiera information om denne. SmartElement använder sig av några olika tekniker för att samla in information om besökare då de anländer på en webbsida, dels teknisk information och dels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

## 6.1 Information om besökaren

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 dokumentdatabas under ett besökarobjekt. På detta vis kan systemet komma ihåg besökare mellan sessioner och på så vis skapa historik om dennes beteende och vanor, vilket i sin tur är till nytta för att hitta det bästa innehållet att presentera.

### 6.1.1 Hänvisningsinformation




*Figur 6. Hänvisningsinformationen berättar varifrån beökaren kommer*

Den första informationen som registreras om besökaren är varifrån denne anländer till sidan. Webbläsare skickar oftast en s.k. referer-variabel när man navigerar till en webbsida genom att klicka på en länk, processen illustreras i Figur 6. Denna variabel innehåller adressen på sidan som visade länken.

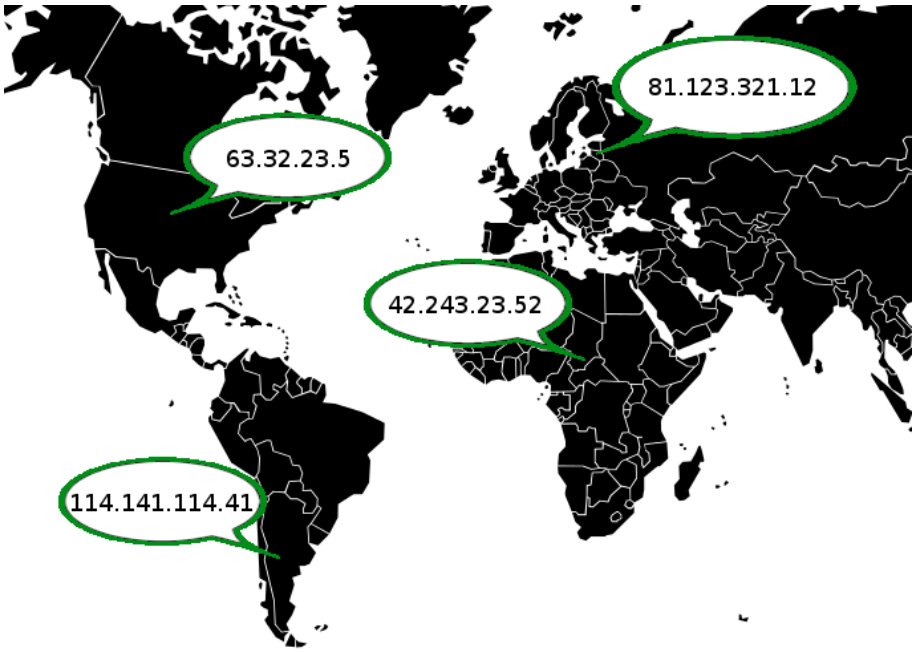
Genom att spara denna information får man dels en indikation av varifrån en besökare hittar till sidan, men även en indikation av vilken typs sidor den aktuella besökaren ofta besöker.

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

### 6.1.2 IP Geolokalisering

Nästa 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 på att upprätthålla 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 besökaren




*Figur 7. IP-geolokalisering handlar om att associera plats med IP-address*

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 lita 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 vidareförmedlar 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-operatör rapporterar. Även om IP-adressen är besökarens egna så kan leverantören ha registrerat adressen på en annan plats än den var besökaren befinner sig.

### 6.1.3 Besökarstatistik

För varje sidvisning på en webbsida med SmartElement sparas det statistik. Besökarobjektet 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 besökaren kan man generera information om besökarens beteende då denne använder webbsidan. Man kan räkna hur ofta denne besöker specifika delar av sidan, om denne inte sett en viss sida på länge, om denne 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 systemet tappar besökaren 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)

### 6.1.4 Tid

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

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

### 6.1.5 Användardefinierad information

För att tillåta så flexibel användning som möjligt, tillåter SmartElement även att upprätthållaren av en webbsida själv definierar information som skickas till servern för processering. Detta sker genom att upprätthållaren 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 som SmartElement inte i sig kan ta reda på, information som besökaren matar in på webbsidan eller information som upprätthållaren 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 upprätthållaren 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 kampanjer 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.

## 6.2 Filtrering

För att välja ut vilket innehållselement som skall skickas till besökaren, använder sig systemet av informationen som samlats in och ett eller flera filter som upprätthållaren av webbsidan 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 besökare.

### 6.2.1 Villkor

I grunden bygger SmartElements filtersystem på 12 enkla villkorsfunktioner som utför jämförelser mellan den information tagen skickat och den information som upprätthållaren sparat i filtret.

De tolv villkoren som kan användas samt de datatyper de kan hantera visas i Tabell 1.

Genom att kombinera dessa enkla villkor med den data som samlas in vid sidvisningar

| 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*

skapas filter som kan göra meningsfulla beslut om besökaren.

Under utvecklingen av SmartElement utvaldes några färdiga filtreringsmetoder för implementering i systemet som en grund för upprätthållare 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.

### 6.2.2 Stadsfiltret

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

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

### 6.2.3 Landfiltret

Landsfiltret använder sig av IP-geolokaliseringsinformationen för att tillåta upprätthållaren att avgränsa innehåll på basis av vilket land besök 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.

### 6.2.4 Datumfiltret

Datumfiltret använder sig av dagens datum för att tillåta upprätthållaren att specificera när ett innehåll skall visas. Filtret kan användas med de olika jämförelsevillkoren. Upprätthållaren 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 upprätthållaren möjligheten att visa kampanjinnehåll under en specifik tid genom att kombinera två datum filter, ett med en nedre gräns och ett med en övre.

### 6.2.5 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 upprätthållare 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.

### 6.2.6 Besökstidsfiltret

Besökstidsfiltret använder sig av tiden en besökare spenderat på webbsidan för att filtrera innehåll. Filtret använder de olika jämförande villkoren för att tillåta upprätthållare att visa innehåll för besökare 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 upprätthållaren vill skapa ett element som visar ett specialerbjudande för besökare som har spenderat en längre tid på webbsidan men inte beställt något samt att upprätthållaren vill visa nyheter för besökare som nyss anlänt.

### 6.2.7 Nyckelordsfiltret

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

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

### 6.2.8 Landningssidefiltret

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

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

### 6.2.9 Sidantalsfiltret

Sidantalsfiltret använder sig av statistikinformationen för att ge upprätthållaren 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 upprätthållare 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.

### 6.2.10 Hänvisarfiltret

Hänvisarfiltret använder sig av hänvisningsinformationen för att tillåta upprätthållare 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 upprätthållaren vill skapa ett element som visar speciellt innehåll för besökare som når dennes webbsida genom en av deras samarbetspartners.

### 6.2.11 Regionsfiltret

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

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

### 6.2.12 Tidsfiltret

Tidsfiltret använder den aktuella tiden för att välja ut vilket innehåll som skall visas. Upprätthållaren 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 upprätthållaren vill visa speciell information när en butik är öppen, när den håller på att stänga och när den är stängd.

### 6.2.13 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 upprätthållaren skapa filter som visar innehåll på basis denna statistik.

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

### 6.2.14 Anpassningsbart filter

Den sista filterklassen är det anpassningsbara filtret vilket tillåter upprätthållaren 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 upprätthållaren 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.

# 7 MJUKVARUARKITEKTUR

Arkitekturen utformades utgående från samma principer som den tekniska arkitekturen. Systemet har planerats 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)

## 7.1 Horisontell skalbarhet

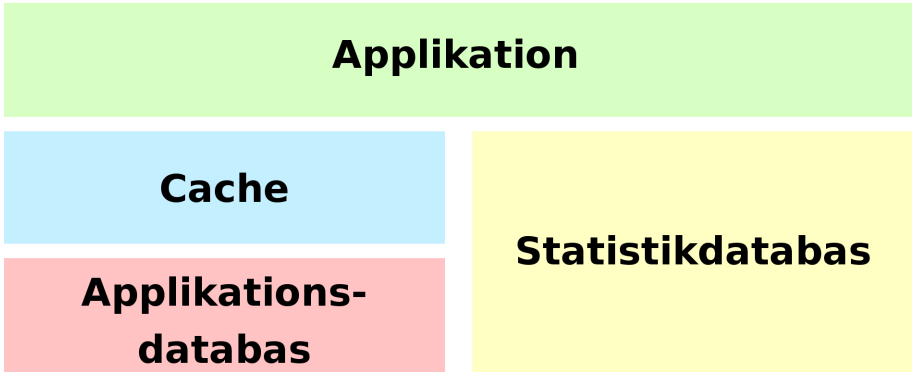
För att tillåta systemet att anpassas till växande resurskrav, har SmartElement utformats för att tillåta horisontell skalbarhet. Att ett system är horisontellt skalbart betyder att man kan öka systemets kapacitet genom att öka antalet av någon komponent och sprida ut belastningen. (Henderson 2006 s.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

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

Den tyngsta delen av applikationen, vad gäller skalbarhet, är applikationsdatabasen. På grund av detta skapades processer som är menade att skydda detta lager genom att spara objekt i cache-minnet. På så vis hålls databasförfrågningarnas antal lägre och applikationsdatabasen behöver inte vara lika skalbar för att systemet skall klara större mängder förfrågningar.

## 7.2 Servern

Back-endsystemet, dvs. serverkomponenten, i SmartElement är den centrala delen av systemet var all data lagras och all filtrering sker. De delar som bygger upp serverkomponenten illustreras i Figur 8. 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




*Figur 8. Serverkomponentens delar*

konfiguration.

Hela back-endsystemet är realiserat 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 förfrågningen, eller så kan den genereras från databasen. Detta betyder att t.ex. autentiseringen är baserad på en nyckel som används för att generera en signatur för varje förfrågning. Orsaken till detta ligger i att det är lättare att realisera horisontellt skalbarhet i systemet 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: applikationen, cache-lagret, applikations-databasen och statistik-databasen. Denna uppdelning är gjord för att försöka tillåta för oberoende horisontell skalbarhet i de olika ansvarsområdena genom att lägga till eller ta bort resurser av de olika komponenterna i takt med behov.

## 7.3 PHP-ramverket Laravel

För att göra utvecklingen av systemet snabbare 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 pakethanteraren Composer, vilket tillåter lätt importering av kodpaket i projektet. Utöver detta är hela ramverket uppbyggt för att vara modulärt och tillåter således modifikation eller full ersättning av komponenter.

Laravel-ramverket bygger på mönstret Model-View-Controller (MVC) i sin arkitektur. Dvs. 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)

## 7.4 Komponenter i systemet

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

### 7.4.1 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 en förfrågning från besökarens webbläsare, synkroniserar informationen med statistikdatabasen, söker fram sidobjektet ur cachinglagret (alternativt bygger upp det från databasen), använder siteobjektet för att generera det innehåll som skall sändas tillbaka och formaterar ett svar som skickas tillbaka till besökaren.


*Figur 9. Besökarobjektet*

### 7.4.2 Besökarobjektet

Besökarobjektet, illustrerat i Figur 9, representerar en besökare på webbsidan och innehåller all information som systemet kunnat samla ihop. Besökarobjektet är det enda objekt i filtreringsprocessen som är realiserat att använda databasen.

När en förfrågan kommer in, anropas 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.

### 7.4.3 Siteobjektet


*Figur 10. Siteobjektet*

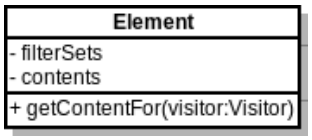
Sitebjektet, illustrerat i Figur 10, är det högsta objektet i hierarkin som används för att filtrera ut innehåll. Det beskriver upprätthållarens webbplats i sin helhet. Från sitebjektet

går alla relationer hela vägen ner till filtren som används för att välja ut innehåll.

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

Objektet är realiserat så att det kan kompileras till en entitet som går att spara i ett cache-lager för kunna hålla objektet i minne vid operationer. Detta tillåter snabb åtkomst till data under en sidvisning och sparar på databasförfrågningar.

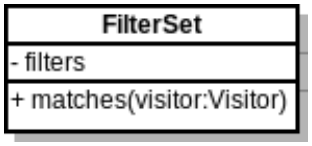
### 7.4.4 Elementobjektet




*Figur 11. Elementobjektet*

Elementobjektet, illustrerat i Figur 11, motsvarar ett HTML-element på användarens webbplats, men det är ej bundet 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 banner eller som del av en navigation.

### 7.4.5 FilterSetobjektet



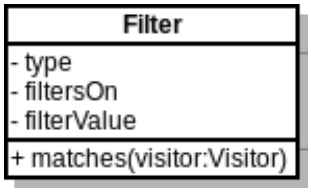

*Figur 12. FilterSetobjektet*

Ett element innehåller filterset, innehåll och ett status. Figur 12 representerar ett FilterSetobjekt. Filterseten är filtersamlingar med en länk till ett innehållsobjekt samt en prioritet inom elementet. När sidobjektet anropar ett element för att få tillbaka det lämpliga innehållet kör elementet igenom sin filtersetsamling och ser om filtren 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 ett utvecklingsmönster 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) FilterSetobjektet i samband med besökarobjektet utgör tillsammans det sammanhang i vilket strategierna (Filtren) appliceras.

### 7.4.6 Filterobjektet



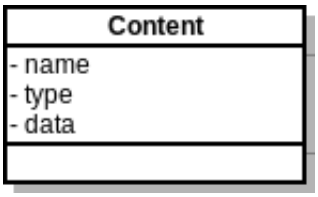

*Figur 13. Filterobjektet*

Filterobjekten är i slutändan de objekt som utför tester mot besökarobjekten. Som Figur 13 illustrerar består de av en filtertyp som definierar hur de jämför data, en fältidentitet 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. Ett filter tar endast ställning till en sak och returnerar ett logiskt värde beroende på om kriteriet möts eller ej. Filtersetet vet endast att det kan innehålla godtyckligt filter och filtret i sig vet att frågan alltid kommer att ställas på samma vis med likadan input.

### 7.4.7 Contentobjekten

Contentobjektet, illustrerat i Figur 14, är den data som returneras till besökaren efter att filtreringen skett. Systemet sätter ingen annan begränsning på datan än att den måste kunna sparas som en textsträng. Detta betyder att man kan producera avancerad funktionalitet




*Figur 14. Contentobjektet*

genom att t.ex. spara Javascript-kod eller JSON-data som innehåll och på så vis använda SmartElement för att styra exekvering i besökarens webbläsare. Det betyder också att systemet inte sätter så stora krav på anroparen eftersom det enda som är viktigt är att denne 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.

## 7.5 JavaScript-tagen

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

Tagen försöker hitta en s.k. kaka i besökarens webbläsare. Om ingen finns så skapas en ny unik id som 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 adressfältet 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 förfrågningen 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 anropas kan dock ändras och man kan genom att specificera en callback ändra på funktionsanropet i JSONP-svaret och på så vis använda egen logik för uppdateringen av innehållet.

## 7.6 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 rekommendationer för upprätthållaren vad gäller olika besökarkretsar.

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

# 8 DISKUSSION OCH SLUTSATSER

SmartElements arkitektur har så här långt uppfyllt de krav som sattes upp för systemet. Den tillåter filtrering av innehåll på basis av information som samlas in vid 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.

Projektet gav mig som utvecklare en mängd erfarenhet av genomförandet av ett mjukvaruutvecklingsprojekt. Jag hade innan projektet jobbat med såväl utveckling som planering i viss mån men SmartElement var det första projektet var jag själv lett den tekniska planeringen och utvecklingen. Att ta rollen som arkitekt och ledande utvecklare gav mig väldigt värdefull kunskap för framtiden. Projektet i sig var intressant då det handlade om att bygga en helt ny produkt för företaget. Developer's Helsinki hade tidigare jobbat med

webbanalys, men detta var första kontakten till anpassning av webbinnehåll.

## 8.1 Riktlinjer 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.

### 8.1.1 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.

### 8.1.2 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. Genom att träna ett system med information specifik för en webbsida kunde man potentiellt förutsäga vilka innehåll en ny besökare potentiellt är intresserad av. Till exempel genom att integrera PredictIO kunde man skapa rätt effektiv anpassning.

### 8.1.3 Integration med tredjepartstjänster

Ett potentiellt område för utveckling av informationen som kan användas för filtrering skulle vara att tillåta integration mot tredjepartstjänster. Genom en integration mot ett e-

postkampanjverktyg kunde man visa en lista över aktiva kampanjer att anpassa efter, utan att upprätthållaren måste komma ihåg kampanjkoder och URL-adresser. Genom integration med en dedikerad webbanalysplatform kunde man drastiskt öka den mängd data som finns tillgänglig för besökaren. Möjligheterna är många och man kunde ge upprätthållare av webbsidor bättre stöd för deras egna processer.

# 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.

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.

Hannak, Aniko; Sapiezynski, Piotr; Molavi Kakhki, Arash; Krishnamurthy, Balachander; Lazer, David; Mislove, Alan & Wilson, Christo. 2013, Measuring Personalization of Web Search, I: *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, s. 527–538. Tillgänglig: http://dl.acm.org/citation.cfm?id=2488388.2488435.

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

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

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.

2014, *Personyze*. Tillgänglig: http://personyze.com/categories/Personyze+Profiler, Hämtad: 8.5.2014.

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

2014, *Pimcore - On-site Behavioral Targeting and Personalization Platform*. Tillgänglig: http://www.pimcore.org/en/product/targeting-personalization, Hämtad: 6.5.2014.

2014, *PredictionIO - Concepts*. Tillgänglig: http://docs.prediction.io/current/concepts/index.html, Hämtad: 8.5.2014.

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.