Utveckling av webbrobot för informationsöverföring

Fredrik Englund

Examensarbete för ingenjörs (YH)-examen

Utbildningsprogrammet för informationsteknik

Vasa 2012

EXAMENSARBETE

Författare:	Fredrik Englund
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Susanne Österholm

Titel: *Utveckling av webbrobot för informationsöverföring*

Datum 11.4.2012 Sidantal 33

Abstrakt

Detta arbete behandlar planering och utveckling av en webbaserad robot. Robotens syfte är informationsinsamling samt ifyllande och skickande av formulär till externa webbsidor. Roboten utvecklades åt Centret för Livslångt Lärande med syftet att samla in kursinformation från befintliga sidor och visa den på en nyutvecklad webbplats, samt att hämta och visa ansökningsformulären från de befintliga sidorna på den nya.

Roboten utvecklades i PHP med ett gränssnittslager utvecklat som en modul till innehållshanteringssystemet WordPress, på vilket den nya webbplatsen körs. Roboten använder PHP/CURL för formulärhantering och tillåter specificering av informationen som skall hämtas via en inställningsmeny i WordPress administrationsgränssnitt.

Resultatet av arbetet var en prototypversion av roboten som kan hämta kurser och enkelt lagra deras information. Prototypen kunde också visa kursernas ansökningsformulär i en webbsida och skicka ifylld data till ett formulär på en extern webbplats.

Språk: svenska Nyckelord: webbrobot, PHP, cURL, WordPress

Arkiveras: Theseus.fi

OPINNÄYTETYÖ

Tekijä:	Fredrik Englund
Koulutusohjelma ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaaja:	Susanne Österholm

Nimike: *Tietoa siirtävän web-robotin kehittäminen*

11.4.2012 33 sivua

Tiivistelmä

Tässä opinnäytetyössä käsitellään web-pohjaisen robotin suunnittelua ja kehittämistä. Robotin tarkoitus on tiedon kerääminen sekä lomakkeiden hakeminen ja postittaminen ulkoisille verkkosivustoille. Robotti kehitettiin Centret för Livslångt Lärande:lle tarkoituksena kerätä kurssitietoja nykyisiltä verkkosivuilta ja näyttää ne uudella sivustolla.

Robotti kehitettiin PHP:llä, ja sen liittymäkerros kehitettiin moduulina WordPress sisällönhallintajärjestelmään, johon uusi verkkosivusto perustettiin. Robotti käyttää PHP/CURLia hallitsemaan lomakkeita ja sallii ominaisuuksien määrittämisen WordPressin hallintaliitymän kautta.

Opinnäytetyön tulos oli robotin prototyyppi, joka voi hakea kursseja ja tallentaa niiden tietoa sekä tulostaa niiden hakemuslomakkeita verkkosivulla ja postittaa tietoa ulkoiselle verkkosivustolle.

Kieli: ruotsi Avainsanat: web-robotti, PHP, cURL, WordPress

BACHELOR'S THESIS

Author:	Fredrik Englund
Degree Programme:	Information Technology, Vasa
Supervisor:	Susanne Österholm

Title: *Development of web robot for information transfer*

Date 11.4.2012 Number of pages 33

Abstract

This thesis deals with the planning and development of a web based robot. The purpose of the robot is to gather information as well as to fetch forms and post information to forms on an external web site. The robot was developed for the Centre for Lifelong Learning with the purpose of aggregating course information from existing web pages to show on a newly developed web site. Lastly, application forms from the existing pages should be fetched and showed on the new web site.

The robot was developed with PHP with an interface layer developed as a plugin for the content management system WordPress, on which the new web site was based. The robot uses PHP/CURL for handling forms and allows the specification of information to be fetched via a settings menu in the WordPress administration interface.

The result of this thesis was a prototype of the robot, able to fetch courses and store their information as well as show the course applications on a web page and send information to an external web page.

Language: Swedish Key words: web bot, PHP, cURL, WordPress

Innehållsförteckning

Ordförklaringar

API	Application Programming Interface, en samling funktioner som fungerar som gränssnitt till ett specifikt system.
Cookies	En fil lagrad lokalt på en användares dator, där sessionsinformation om besökta webbplatser lagras.
CSS	Cascading Style Sheets, ett stilmallsspråk för att definiera utseendet av olika markeringselement.
cURL	client URL, bland annat ett kodbibliotek för överföring av data med olika webbprotokoll.
Hook	Ett i WordPress fördefinierat skede i en sidas exekvering, till vilken man kan koppla funktionsanrop.
HTML	Hypertext Markup Language, det huvudsakliga dokumentmarkeringsspråket för webbsidor.
HTTP	Hypertext Transfer Protocol, ett protokoll för dubbelriktad informationsöverföring på Internet. Det mest använda protokollet för webbsidor.
Inlägg	Inom kontexten av detta examensarbete hänvisar ordet inlägg till behållaren i vilken WordPress lagrar information. Sidor, inlägg och kommentarer är generiska typer av inlägg i WordPress.
jQuery	Ett ramverk som bygger på skriptsspråket JavaScript med ett antal förändringar av kodsyntax. Används som huvudsakligt skriptsspråk av WordPress.
MySQL	Ett hanteringssystem för relationsdatabaser, baserat på språket SQL (Structured Query Language).
Perl	Ett allmänt programmerings- och skriptspråk som är i bred användning för webbprogrammering.
POSIX	Portable Operating System Interface, en samling standarder för att underlätta kompabilitet mellan operativsystem.
SSL	Krypteringsprotokoll för säkrad överföring av information. Kombinationen med HTTP har gett upphov till webbprotokollet HTTPS.

URL	Uniform Resource Locator, ett sätt att definiera adresser på webben, följer syntaxen *schema://domän/sökväg?förfrågningssträng*.
Utbildningar	Inom kontexten av detta examensarbete hänvisar ordet utbildningar till CLL:s utbud av fortbildning oberoende av typ (kurs, seminarie, etc.)
Webbplats	En webbplats är en samling av sammankopplade webbsidor med adresser relativa till en gemensam rot.
Webbsida	En enskild sida som blivit serverad av en webbserver till en webbläsare.
WordPress	Ett populärt open-source ramverk för bloggar eller innehållshanteringssystem.
WordPress tema	En fullständig webbplatsstruktur med sid- och stilmallar samt temaspecifik funktionalitet. Definierar en grafisk layout och design av webbplatsen.
WordPress modul	Ett tillägg till WordPress som innehåller ytterligare funktionalitet. Fungerar oberoende av tema.
XML	Extensible Markup Language, ett dokumentmarkeringsspråk för allmänna dokument.

1. Inledning

Arbetet gick ut på att planera och utveckla en webbrobot för informationsaggregation och sändning av information till formulär. Roboten skulle söka genom fördefinierade sidor och hämta specifik information för att lagra på den lokala webbservern. Roboten skulle också vara kapabel att hämta, visa och sända information till formulär på utomstående webbplatser över krypterade anslutningar.

1.1 Uppdragsgivare

Arbetet utfördes åt Centret för Livslångt Lärande vid Yrkeshögskolan Novia och Åbo Akademi (CLL). CLL erbjuder fortbildning, dvs vidare utbildning och vuxenutbildning, på orter runtom i landet med hjälp av Novias och Åbo Akademis enheter på respektive ort. CLL delas upp i tre huvudsakliga enheter: Novia-, Åbo- och Vasaenheten. Åbo- och Vasaenheterna finns huvudsakligen i ÅA:s utrymmen och Noviaenheten i Novias utrymmen på Brändö i Vasa.

CLL grundades 2009 genom en sammanslagning av fortbildningsenheterna vid Åbo Akademi, Österbottens Högskola och Yrkeshögskolan Novia, och har ca 60 anställda i bl.a. Åbo, Vasa och Jakobstad. CLL anordnar ca 200 utbildningar samt ca 25 projekt per år. (Om CLL, 2012)

1.2 Bakgrund

När detta uppdrag gavs hade CLL olika webbplatser och interna system för administration på de olika enheterna. Samtidigt utvecklades en ny webbplats för användning av alla enheter, med syftet att förenhetliga CLL:s ansikte utåt. Diskussioner fördes om ibruktagning av ett nytt administrationssystem som kunde kopplas till den nya webbplatsen, men utan kunskap om vilket system som skulle tas i bruk kunde ingen programvara för överföring av existerande information planeras.

Att ta i beaktande under planeringen av detta arbete var att den nya webbplatsen utvecklades med innehållshanteringssystemet WordPress som grund. Detta medförde en del givna teknologier, en fullständig uppsättning med webbplatsfunktionalitet (befintligt

databassystem och administrationsgränssnitt, m.m.) med möjligheten att underlätta utvecklingsarbetet. En utredning av om det skulle vara ändamålsenligt att använda dessa teknologier eller andra motsvarande sker i kapitel 3 och 4.

Innan den nya webbplatsen skulle tas i bruk hade de enskilda enheterna sina egna webbplatser, utvecklade och upprätthållna enskilt, där utbildningar skapades och ansökningsprocesser hanterades. I och med den nya webbplatsen behövde planerare från varje enhet skapa utbildningarna också på den nya webbplatsen och länka dem till de gamla webbplatserna för fullständig information och ansökningsformulär.

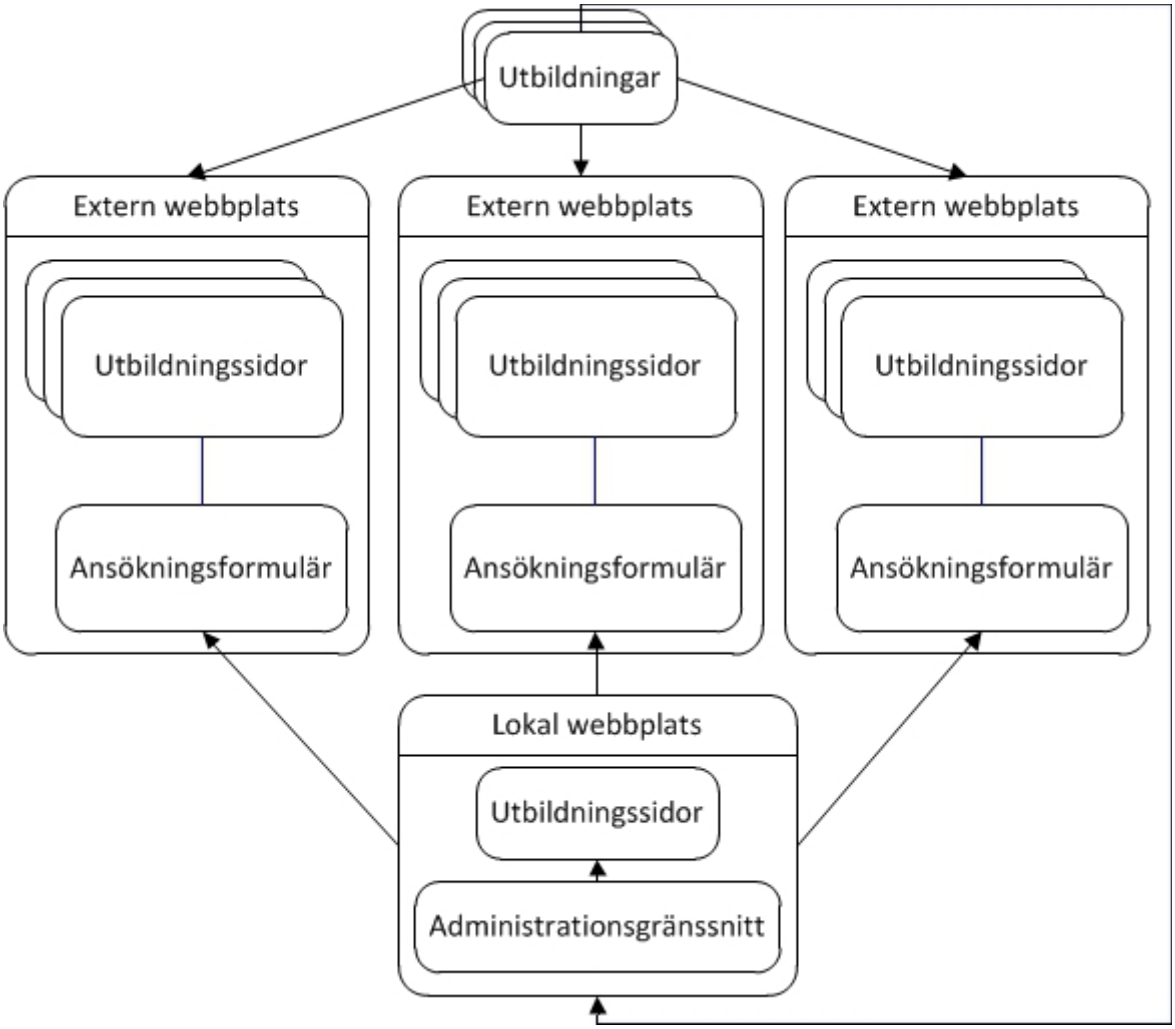
1.3 Uppdrag

Uppdraget gick ut på planering och utveckling av en webbrobot för dubbelriktad överföring av information mellan webbsidor. Ett av kraven på roboten var att kunna hämta information från externa webbsidor, hantera och lagra den samt visa den på den lokala webbplatsen. Det andra behovet var att roboten skulle kunna lagra information om formulär från externa webbsidor, duplicera formulären på den lokala och hantera överföring av formulären från den lokala sidan till den externa. Roboten skulle kunna administreras från ett gränssnitt på den lokala webbplatsen så enkelt som möjligt, medan den ändå skulle vara flexibel nog att underlätta hantering av information av olika format. Roboten skulle utvecklas som en fristående produkt som kunde implementeras på webbplatser i allmänhet för en större mängd olika syften.

Idén för uppdraget kom upp under utvecklingen av CLL:s nya webbplats och sökandet av nya system för att hantera utbildningar, kunder och processer kopplade till utbildningsansökningar och marknadsföring. Eftersom den nya webbplatsen skulle tas i användning innan detta eventuella administrationssystem, konstaterades det att ett sätt för aggregation av information skulle behövas. Tanken bakom detta var att undvika behovet av manuellt kopieringsarbete för planerarna vad kursansökningar och utbildningslistor beträffar.

Utan en automatisering av informationssamling skulle det praktiska arbetet innebära någon form av manuell kopiering av utbildningslistorna. Det vill säga att eftersom användare skulle upprätthålla utbildningslistorna på de befintliga webbplatserna, medan dessa ännu var i bruk, skulle de vara tvungna att fylla i samma information helt eller delvis också till den nya webbplatsen. Då en utbildning skapas på en befintlig sida skulle den också behöva

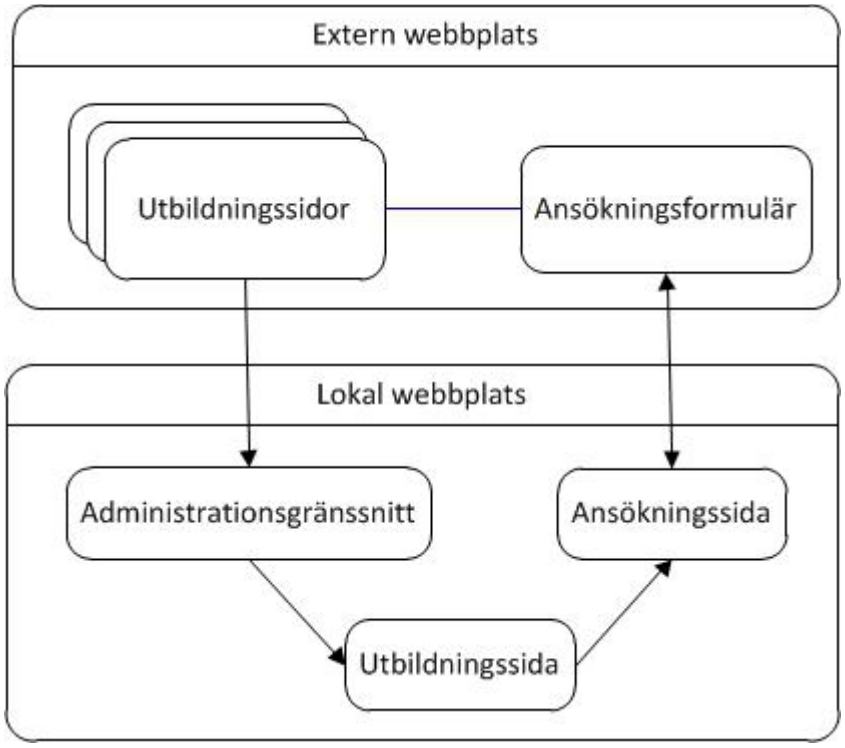
skapas i den nya (lokala) webbplatsens innehållshanteringssystem med en länk till utbildningens ansökningsformulär på den gamla (externa) webbplatsen, som förevisas i figur 1. Detta skulle innebära duplicering av utbildningsinformationen samt en hänvisning till gamla webbsidor, något som man ville undvika så långt som möjligt. Eftersom det egentliga arbetet som man ville undvika var kopiering av information, konstaterades det att den lämpligaste lösningen skulle vara utvecklingen av en webbrobot för informationsinsamling.



Figur 1. Från utbildningsutbudet fylls utbildningar in i respektive webbplats system med ansökningsformulär. I och med den nya webbplatsen behöver varje utbildning också lokalt skapas där med länkar till de befintliga sidornas ansökningsformulär.

I praktiken skulle robotens syfte vara att hämta tabellinnehåll från externa webbsidor, hantera och strukturera detta för lagring samt skapa kopior av externa formulär för ifyllning. Vid ifyllning av formulären på den lokala webbsidan skulle den ifyllda

informationen skickas till den externa sidan och fyllas in i dennes formulär. Roboten skulle alltså sköta ett arbete automatiskt, vilket en användare annars skulle vara tvungen att göra manuellt. Slutresultatet av webbrobotens utveckling förevisas i figur 2.



Figur 2. Roboten skulle hämta utbildningar från externa sidor och lagra dem i det lokala systemet för visning samt hantera en dubbelriktad överföring av formulär till de externa sidorna.

2. Användaragenter

En användaragent är en mjukvara som agerar å en användares vägnar över Internet. Exempel på användaragenter är webbläsare och e-postklienter. Användaragenter fungerar över alla protokoll som används för informationsöverföring och kan beroende på tillämpningsområde agera på all typ av information som existerar på Internet.

2.1 Skillnader mellan olika användaragenter

I sin enklaste form fungerar en webbläsare genom att hämta webbsidor med HTTP-protokollet, medan en e-postklient hämtar e-post med t.ex. POP-protokollet. Den hämtade informationen visas åt användaren och tillåts att hanteras på ett sätt som är intuitivt för typen av information; webbsidor visas grafiskt och kan manipuleras med muspekaren,

medan e-post ofta visas som text i en ordbehandlare. Dessa användaragenter fungerar väl för sina ändamål, men på grund av att de är skräddarsydda för en specifik typ av användning är de begränsade i funktionalitet.

Webbrobotar är en bred gruppering av användaragenter utvecklade för användningsområden inte lämpade för t.ex. en mänsklig användare vid en webbläsare. Jämfört med en webbläsare, som hanterar Internet i form av ”sidor”, agerar en webbrobot oftare på Internet i form av filer, helt automatiserat utan användargränssnitt. Eftersom en robot hanterar dessa filer som rena dokument av en definierad struktur är de väl lämpade till att agera på information som finns inom dessa sidor. De stora användningsområdena är aggregationsrobotar som samlar information från ett mängd källor och möjligtvis filtrerar och analyserar den, samt webbrobotar som aktivt företräder en användare, t.ex. genom att fylla i beställningsblanketter på en leverantörs webbsida när användarens lager av någon produkt faller under en bestämd nivå. En tredje gruppering utgörs av passiva robotar, som t.ex. fungerar genom att placeras mellan en användare och en webbserver, och antingen modifiera innehåll som visas åt användaren eller anonymisera användaren från webbplatser denne besöker. (Schrenk 2009, k. 2. Inspiration from Browser Limitations)

2.2 Implementationer av webbrobotar

Av vikt för detta arbete är både aggregationsrobotar samt aktivt företrädande robotar, eftersom grundprinciperna av dessa två passar in mycket väl med de två huvudsakliga funktionerna som den utvecklade webbroboten skall utföra.

2.2.1 Aggregationsrobotar

Den mest kända och allmänt nyttiga typen av webbrobot är den som används som den centrala delen av sökmotorer och sociala webbsidor, med uppgift att samla information från specificerade källor och visa den enhetligt åt en användare. Sökmotorer använder aggregationsrobotar för att hämta resultat av en sökning från en indexerad lista. Denna lista har t.ex. blivit uppbyggd av en webbspindel, en annan typ av användaragent vars funktion beskrivs i kapitel 2.3. (Schrenk 2009, k. 2. Webbots That Aggregate and Filter Information for Relevance)

2.2.2 Företrädande robotar

Robotar behöver inte endast samla in information, de kan också agera på Internet som en företrädare åt en användare. I de flesta fall handlar deras handlingar om den enklaste formen av användarinmatning på Internet: formulär. Roboten besöker en specifik webbsida innehållande ett formulär för att fylla i specifik information, så som i exemplet i kapitel 2.1. (Schrenk 2009, k. 2. Webbots That Act on Your Behalf)

2.3 Webbspindlar

En webbspindel skiljer sig inte från de två föregående typerna av webbrobotar i grundläggande funktionalitet, utan i sättet den arbetar. Webbspindlar anses inte vara en underkategori av webbrobotar utan en helt skild typ av användaragent, på grund av att de ofta arbetar självständigt och sprider sig genom Internet mer eller mindre utan användarinverkan.

En webbspindel vandrar genom webbsidor automatiskt och strukturerat, på jakt efter en specifik typ av information. I allmänhet börjar spindeln med en uppsättning av webbadresser från vilka den hämtar informationen den söker samt länkar. Till följande går spindeln in i sidorna dessa länkar leder till och upprepar insamlingen. Spindlar kan hämta länkar beroende på olika kriterier, t.ex. tills ett visst djup (antal länkningar sedan begynnelseadressen) blivit nått eller uteslutande specifika domäner. (Schrenk 2009, k. 18. How Spiders Work)

En webbspindel kan fungera för aggregation eller inmatning, och ofta används dessa två tillsammans för skadliga ändamål. En aggregationsspindel kunde söka upp sidor som använder populär forum-mjukvara (t.ex. phpBB, Invision) vars struktur och registreringsformulär är kända. En annan spindeln kan då vandra genom alla dessa forum, registrera användare och skriva inlägg, i allmänhet innehållande någon typ av reklam.

Ett mer godartat exempel på en webbspindel är Googles sökmotors webbspindel, som vandrar genom Internet och lagrar statistik för hur ofta olika ord används på olika sidor för att indexera webbsidors lämplighet för vissa sökord. Det finns många sådana webbspindlar, som aggregerar vissa typer av innehåll och visar dem på ett centraliserat sätt. (Schrenk 2009, k. 18)

3. Tekniker

Medan programvarans utveckling var för ett specifikt ändamål, skulle den ändå ske med målet att göra slutresultatet så portabelt och lättunderhållet som möjligt. Med detta som grundprincip skulle roboten utgöras av två huvudsakliga lager. Det första lagret skulle innehålla robotens processer: navigering och hantering av webbsidors innehåll och formulär samt definiering av regler för detta. I detta lager skulle inte definieras hur eller vart information lagras eller hur konfigurering av roboten sker. Det andra lagret, som skulle utvecklas för mer specifikt bruk i en webbplats, skulle innehålla ett gränssnitt för konfigurering av roboten samt lagring av informationen den hämtat.

I detta kapitel undersöks det vilka tekniker som kunde användas, hur de kunde tillämpas och i vilken grad de är lämpliga för detta arbete.

3.1 PHP

PHP är ett skriptspråk som började utvecklas 1995 och är under fortsatt öppen utveckling av PHP Group. Eftersom språket är öppet har det moduler som på ett eller annat sätt implementerar stöd för de flesta tekniker som används i samband med webbutveckling. Detta tillsammans med det faktum att de flesta webbservrar kan köra PHP-kod som standard, gör PHP till ett ypperligt val för applikationer som strävar efter högsta möjliga portabilitet. Eftersom webbplatsen som roboten skulle komma att köras på drivs av PHP är det också ändamålsenligt att använda detta språk för utveckling av robotens processer. Standardinstallationer av PHP på de flesta operativsystem innehåller funktioner för enkel hantering av dokument över webben, vilket i sig vore tillräckligt för att utveckla en enkel aggregationsrobot. (Lerdorf&Tatroe 2002, s. 2)

3.1.1 PHP/CURL

PHP/CURL är en implementation av cURL (**c**lient **URL**) som underlättar hantering av webbsidor med PHP. Med hjälp av cURL kan programvara hämta filer från webben via en mängd olika protokoll som vore omöjligt med endast PHP. PHP/CURL följer med de flesta installationer av PHP som en standardkomponent.

PHP/CURL tillåter en webbrobot att bete sig som om den vore en användare, eller rättare sagt kommunicera med en webbserver som om den vore en webbläsare med alla de

verktyg en sådan har för autentisering och lagring av tillfälligt data. En webbrobot kan t.ex. instrueras att känna igen och acceptera säkerhetscertifikat, vilket i sin tur låter roboten komma åt filer som skickas över HTTPS-anslutningar. En webbrobot kan också instrueras att ge inloggningsuppgifter till en webbserver för att komma åt filer som kräver autentisering, samt identifiera sig själv på ett korrekt sätt. Med PHP/CURL kan en webbrobot alltså skapas för att automatisera de åtgärder en användare för det mesta utför manuellt upprepade gånger och på så vis eliminera tidskrävande simpla uppgifter. (Schrenk 2009, k. 3. Introducing PHP/CURL)

3.1.2 Reguljära uttryck

Reguljära uttryck (regular expressions, förkortas till regex) är ett system för att hitta mönster av tecken i textsträngar; i allmänhet med syftet att modifiera, ta bort eller hitta förekomster av specifik text. Det är det snabbaste och kraftigaste verktyget för att genomsöka stora mängder ren text, men är också mycket svårt att använda väl på grund av dess komplicerade syntax. För genomsökning av textsträngar i PHP finns som standard två implementationer av regex, POSIX eller Perl-kompatibla, och medan det finns skillnader mellan dessa vad effektivitet och svårighetsgrad beträffar finns det inte några större fördelar med regex för utvecklingen av denna typ av webbrobot. Reguljära uttryck hanterar ren text och skulle därför kräva att en funktionalitet för sökning av olika textmönster skulle behöva utvecklas. Problemet ligger i att HTML-standarder gått genom många stora förändringar för striktare syntax sedan markeringsspråkets början, fast webbläsare ännu behöver kunna visa webbsidor skrivna enligt äldre, icke-strikt syntax. Detta ger upphov till att HTML-dokument kan se ut på många olika sätt (elementnamn skrivna i gemener eller versaler, attributvärden omgivna av ”-tecken eller inte, tvetydig struktur, m.m.) Dessa tvetydigheter skulle textgenomsökning med reguljära uttryck behöva ta i beaktande, vilket skulle innebära ett utvecklingsarbete utanför ramarna för detta uppdrag. (Lerdorf&Tatroe 2002, s. 95)

3.1.3 Rekursiva funktioner

Strukturerade programmeringsspråk som PHP har stöd för rekursiva funktioner, som utför funktionsanrop på sig själva inuti sin egen kod. Detta är ett kraftigt sätt att återanvända kod.

I praktiken fungerar rekursivitet genom att en funktion skapas som kontrollerar de givna argumenten och beroende på dess värden utför olika operationer, av vilka det finns åtminstone två och det ena kallas för basfallet. Basfallet innehåller i allmänhet ett specifikt returvärde, t.ex. ett siffervärde som uträknas och returneras inom dess kodblock, medan de andra fallen innehåller ytterliga anrop till samma funktion. Dessa rekursiva funktionsanrop görs med en del av det gamla argumentet som argument. Detta skapar en kedja av funktionsanrop där storleken av argumenten minskar tills de inte längre kan delas upp, i vilket skede funktionens basfall antas vara nått. Då basfallet nås returnerar den ifrågavarande funktionen sitt värde till funktionen som anropade den, upp genom kedjan tills den först anropade funktionen returnerar det slutgiltiga resultatet. I figur 3 förevisas hur ett tals fakultet kunde räknas ut med en rekursiv funktion.

För denna webbrobot kunde rekursivitet tillämpas eftersom varje genomsökning av en webbsida kodmässigt kunde se likadan ut medan endast argumenten (sidans adress och specifikationen av element som skall sökas efter) behöver förändras. (Recursion in C and C++ - Cprogramming.com, 2012)

Figur 3. Ett flödesschema över en rekursiv funktion som räknar ut ett tals fakultet. Funktionen returnerar sitt argument multiplicerat med returvärdet av samma funktion där argumentet minskat med 1. Basfallet inträffar då argumentet är 1, vars returvärde är 1.

3.2 JavaScript

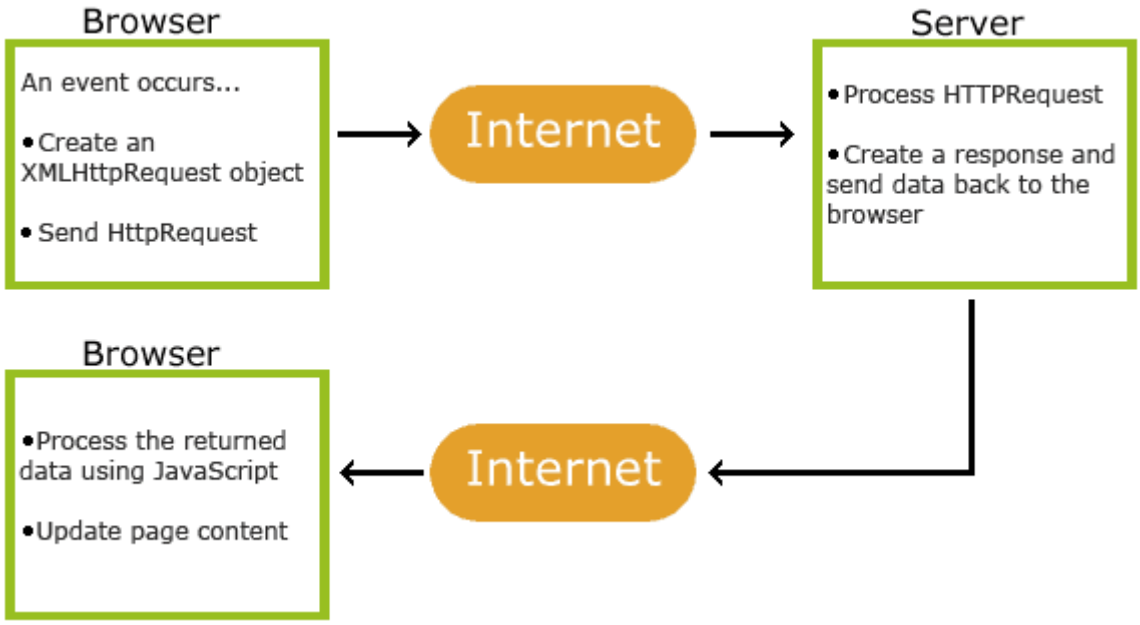
JavaScript är ett skriptsspråk som skickas med webbsidor för att inkludera dynamisk funktionalitet i en webbsida som redan blivit inladdad i en webbläsare. Medan ett skriptsspråk som PHP exekveras på webbservern för att generera HTML och skicka det till klienten, skickas JavaScript till klienten direkt för att exekveras vid behov, t.ex. för att utföra en funktion då muspekaren svävar över ett visst element av en webbsida. Detta används för att förändra HTML-koden efter att den blivit skickad till klienten, t.ex. för att

skapa grafiska effekter eller göra mer komplicerade layoutförändringar. (JavaScript Introduction, 2012)

3.3 AJAX

AJAX (Asynchronous JavaScript and XML) är en teknik för att fortgående skicka information mellan en klient och webbserver efter att en sida blivit skickad från servern. Tekniken fungerar genom att bädda in JavaScript i en webbsida, i vilken finns ett anrop till webbservern för att hämta ny information, som sedan skrivs ut av skriptet i sidan. Detta möjliggör t.ex. uppdatering av listor i realtid eller visning av resultat av en sökning utan att ladda in en ny sida. I figur 4 beskrivs hur en händelse i en webbsida anropar en JavaScript funktion som i sin tur gör en förfrågning till webbservern. Webbservern svarar på förfrågningen och JavaScript-funktionen hanterar svaret för att sedan uppdatera sidan som visas, utan att sidan behöver laddas in på nytt.

För denna webbrobot skulle AJAX användas för att möjliggöra inladdning av formulär från externa webbsidor dynamiskt. (AJAX Introduction, 2012)



Figur 4. Ett flödesschema över hur AJAX fungerar. (AJAX Introduction, 2012)

3.4 Tidy HTML

Tidy HTML är ett bibliotek för de flesta webbskriptspråk som är ämnat att gå genom HTML-kod och korrigera felaktigheter i markeringsspråket enligt nyare standarder, som

t.ex. icke-avslutade element eller felaktig användning av versaler. Detta har ett syfte eftersom HTML genom åren har gått genom många revisioner med olika standarder för syntax och struktur, vilket innebär att många webbsidor fortsättningsvis är skrivna enligt gamla standarder. Medan webbläsare för det mesta kan tyda HTML-kod som inte följer nyare standarder, kan det vara ändamålsenligt att omforma koden till ett mer standardenligt format för att eventuella genomsökningar skall fungera mer konsekvent.

Fastän Tidy HTML oftast fungerar automatiskt behövs det ändå en viss mån av konfiguration för vissa typer av problem, t.ex. där det programmatiskt inte kan bestämmas vad som bör göras på grund av tvetydigheter eller där felaktiga elementnamn har använts. Detta innebär att det inte är ett ypperligt alternativ för en webbrobot som behöver kunna arbeta fullständigt autonomt, eftersom olika dokument kunde behöva olika konfigurationer. (Schrenk 2009, k. 4. Parsing Poorly Written HTML)

3.5 Extensible Markup Language

XML är ett dokumentmarkeringsspråk, likt HTML till syntaxen. Den stora skillnaden är att XML är ett meta-markeringsspråk, vilket betyder att element inte är specifika med fördefinierade egenskaper, utan är generiska och helt och hållet definierade av användaren då dokumentet skrivs. Inom ett XML-element kan också definieras attribut på samma sätt, till skillnad från HTML, där olika element har olika specifika attribut och därmed specifika syften. HTML-elements attribut specificerar hur en webbläsare hanterar attributet, medan XML-element och deras attribut endast är till för sortering och sökning av information. XML lägger vissa krav på formatering, bland annat att elementnamnen skall skrivas i gemener och att alla element som påbörjas också skall avslutas. Dessa krav är inte nödvändiga för att HTML-kod skall kunna läsas av en webbläsare.

Eftersom XML är ett markeringsspråk som fungerar med generiska element kan också ett HTML-dokument ses som ett XML-dokument, förutsatt att HTML-koden är rätt formaterad, vilket inte bör förväntas av någon webbsida. Att kunna navigera en webbsida som ett XML-dokument ger dock stora fördelar för informationshantering.

XML lämpar sig ytterst väl för lagring och hantering av strukturerat data av den form som denna robot skulle hantera, eftersom all väsentlig information är ren text och uppställd i tabellform (eller ett motsvarande format) i utgångsläget, medan detta skulle innebära att

roboten också skulle behöva skräddarsydda funktioner för att hämta informationen ur ett XML-dokument. Alternativet vore lagring av hämtat data direkt i WordPress egna databas, där det kan hanteras med hjälp av WordPress inbyggda funktioner. (Harold&Means 2001, s. 3)

3.5.1 XPath

XPath började som en del av XML, men är nu ett fristående språk som används för att genomsöka XML-dokument. XPath kan användas också för att genomsöka DOM-dokument, vilket behandlas i kapitel 3.6.

Medan användning av XML med XPath för att hitta och manipulera data är en resurssnål metod kräver det att användaren som utför genomsökningen har kunskap om dokumentets struktur och vilka element som har definierats, eftersom XML inte innehåller element av fördefinierade namn. Eftersom XML följer en förhållandevis strikt standard krävs det av XML-dokumentet felfri syntax för att XPath överhuvudtaget skall fungera.

Eftersom både XML och HTML är markeringsspråk kan man hantera ett HTML-dokument som om det vore XML, antagande att HTML-dokumentet är skrivet enligt en standard som är jämförbar med XML. Eftersom man inte kan utgå ifrån att alla webbsidor man samlar information från följer den standard som krävs av XML, kan man inte heller utgå ifrån att XPath fungerar konsekvent mot ett HTML-dokument. (Harold&Means 2001, s.147)

3.5.2 XSL Transformations

Extensible Stylesheet Language (XSL) innehåller metoder för omvandling av XML-dokument till andra typer av dokument. Dessa XSL Transformations (XSLT) fungerar genom att en dokumentprocessor går genom ett XML-dokument tillsammans med en XSLT-stilmall, där mallar för hur olika element skall transformeras till det nya formatet lagras. Då processorn hittar dessa element i XML-dokumentet skriver den ut deras transformationer i en separat trädstruktur. Då dokumentets transformation blivit fullständigt utskriven kan den sedan lagras som ett dokument av givet format (t.ex. ren text, HTML eller XML). XSLT använder sig av XPath syntax för att hitta element som passar ihop med mallarna.

XSLT kunde användas för att t.ex. omforma ett tidigare lagrat XML-dokument med en utbildnings information till HTML, för att sedan kunna visas i webbplatsens utbildningslista. (Harold&Means 2001, s. 129)

3.6 Document Object Model

Document Object Model (DOM) beskriver ett system för att lagra information om hierarkiska dokument i ett skilt objekt. I standarden DOM Level 2 ingår stöd för HTML, XML och CSS. Denna standard har en implementation i PHP, vilket möjliggör navigering av HTML-dokument som hämtats från en webbserver.

Medan XML och XPath möjliggör navigering av det egentliga dokumentet kräver DOM att information extraherad från dokumentet lagras skilt i ett objekt för att sedan användas i navigeringen. En nackdel av detta är att mer utrymme krävs på grund av att dokumentets innehåll kopieras, men fördelen är den att man inte behöver vara medveten om den specifika formateringen eller strukturen av ett dokument för att navigera det. DOM inkluderar funktionalitet för att hämta element utgående ifrån elementets namn, specifika attribut eller ordning i trädet. Detta innebär att man genom att ha information om ett enda element (t.ex. ett tabell-element som innehåller all väsentlig data) kunde iterera genom dess struktur och besöka alla underställda element, utan att behöva känna till någon specifik information om dem.

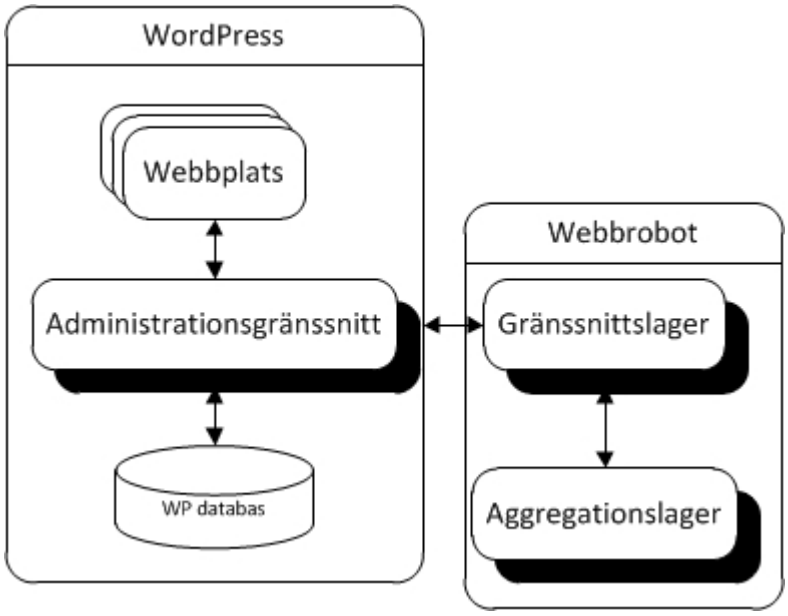
Av stor vikt för webbutvecklare är att DOM inte kräver perfekt formatering av HTML-dokument. Medan XML-kod behöver vara felfritt formaterad för att några XML-specifika operationer överhuvudtaget skall kunna utföras är DOM mycket mer tolerant för fel och är kapabelt att skapa korrekt struktur fastän det finns felaktigheter i HTML-kod (upp till en viss punkt). DOM innehåller också stöd för XPath för navigation av dokumentträd, vilket är ett av de smidigaste alternativen för navigationen och fungerar i detta fall även om originaldokumentet inte är välformaterad XML. (Harold&Means 2001, s. 236)

3.7 WordPress

WordPress (förkortat till WP) är ett ramverk för enkelt skapande av bloggar eller innehållshanteringssystem som kan användas för att driva fullständiga webbplatser med dynamiskt innehåll. WP är skrivet med skriptspråket PHP och tillhandahåller sitt eget system för lagring av data i sin databas samt visning av denna data på en webbplats. WP är

ett öppet utvecklat ramverk vars största tillgång är enkel modifiering av funktionalitet med hjälp av moduler, som kan utvecklas och underhållas skilt från kärninstallationen. (Stern, Damstra & Williams 2010, k. 1. What Is WordPress?)

För detta projekt skulle WP ha en roll som gränssnitt mellan robotens aggregationslager och användaren. Informationen som samlas av roboten skulle sorteras och lagras i WP innehållshanteringssystem. WP kunde då tillhandahålla den grafiska layouten samt sortering och visning av innehållet å en användares vägnar. I figur 5 visas ett översiktsschema över hur informationsflödet skulle fungera. Aggregationslagret skulle skicka information via gränssnittslagret till WP administrationsgränssnittet, som i sin tur skulle lagra informationen i sin dedikerade databas.



Figur 5. Översikt över hur webbroboten gränsar till WordPress.

WordPress innehåller funktioner för att lagra information i sin databas som inlägg, vilket innebär att informationen visas i administrationsgränssnittet där den kan redigeras. Detta innebär också att informationen direkt kan visas på webbplatsens sidor som visar dessa inlägg. Förutom detta kan en modulutvecklare bestämma att skapa egna tabeller i databasen. För hantering av dessa skräddarsydda tabeller finns funktioner som genererar databaskommandon utgående ifrån argument, så att en modulutvecklare inte behöver skriva databaskommandon direkt i modulens kod. Detta är ändamålsenligt eftersom det minimerar chansen att inkorrekta handlingar görs i databasen samt tillåter mer flexibilitet. (Stern, Damstra & Williams 2010, k. 6. Table Details)

3.7.1 Inlägg

WordPress lagrar huvudsakligen innehåll i datasamlingar kallade inlägg (kommer från användningen av WordPress som bloggverktyg, där man skapar blogginlägg). I en standardinstallation av WordPress existerar inlägg, sidor och kommentarer. Olika inläggstyper kan ha stöd för olika typer av innehåll, bl.a. textinnehåll, media, metadata och sammanfattningar. Inlägg kan kategoriseras och tilldelas etiketter, i huvudsak för att beskriva inläggens innehåll.

WordPress har stöd för skapandet av skräddarsydda inläggstyper (Custom Post Types). Dessa tilldelas samma svit av funktionalitet som vanliga inlägg eller sidor, med möjligheten att tilldela dem metadata, bifoga media, sortera och arkivera etc. En viktig egenskap för dessa inläggstyper är taxonomi, eller metoden att klassificera inlägg enligt olika egenskaper samt namnge och gruppera dessa klasser i en hierarki. I kodexempel 2 och 3 förevisas definieringen av en taxonomi samt termer för att kunna kategorisera information enligt geografiska platser. En taxonomi är inte gjord för att innehålla specifik information om ett unikt inlägg och är på så vis inte lämpat för att lagra data, men om man hittar gemensamma egenskaper hos flera inlägg på en mindre specifik nivå är det i en taxonomi man kan ange detta för att senare kunna sortera inläggen intelligent. (Williams, Richard & Tadlock 2011, k. 6)

Kodexempel 2. Koden för att skapa en taxonomi vid namn 'places', samt definitionen att funktionen anropas i en sidas initieringsskede . Att taxonomin är hierarkisk betyder att dess termer kan ordnas i en trädstruktur.

```
<?php
add_action ( 'init', 'create_taxonomies', 0 );

function create_taxonomies () {
	register_taxonomy ( 'places', 'post', array (
		'hierarchical' => true,
		'label' => 'Places',
			)
	);
}
?>
```

Kodexempel 3. Koden för att skapa två termer till taxonomin 'places' vid namn 'Österbotten' och 'Vasa'. Termen 'Vasa' är hierarkiskt underordnad 'Österbotten'.

```
<?php
wp_insert_term (
        'Österbotten',
        'places',
        array (
                'slug' => 'osterbotten',
        );

wp_insert_term (
        'Vasa',
        'places',
        array (
                'slug' => 'vasa',
                'parent' => 'osterbotten'
        );
?>
```

3.7.2 Plugin API

WordPress sidor genereras enligt ett specifikt händelseförlopp. WP innehåller också en API för att koppla funktionsanrop till specifika händelser i detta händelseförlopp. Detta görs genom att hitta en passande ”hook”, som beskriver i vilket skede av en sidas generering man vill anropa funktionen. I kodexempel 1 deklareras en funktion att köras efter att sidans grafiska tema blivit laddat in i systemet. Funktionen som då körs kan t.ex. innehålla kod för att göra förändringar i temats inställningar. Beroende på vilken ”hook” som används har man tillgång till olika globala variabler för att kunna åstadkomma den behövda funktionaliteten.

Kodexempel 1. En funktion kopplas för att bli anropad efter att det grafiska temat blivit laddat in i systemet.

```
<?php
add_action ( 'after_setup_theme', 'init_theme_settings', 0 );
?>
```

Detta system används för att t.ex. spara data som skrivits in i skräddarsydda textfält då ett inlägg sparas eller ladda in stillmallar och skriptfiler då specifika sidmallar körs. På liknande vis kan man definiera filter att köras vid specifika tillfällen, t.ex. för att gå genom en textsträng och radera farliga teckenkombinationer som HTML-kod innan texten körs genom systemet.

Plugin API kunde komma till nytta om man vill förändra funktionaliteten av redan existerande element i administrationsgränssnittet. Utvecklingen av en modul till WordPress förenklas avsevärt av möjligheten att kunna skapa funktionalitet utan att behöva skapa fullständigt nya gränssnitt. (Williams, Richard & Tadlock 2011, k. 3)

3.7.3 Schemalagda uppgifter

En central del av WordPress funktionalitet, som ett system självständigt från användarinverkan, är cron jobs, dvs. tidsbaserade schemalagda händelser. WP använder sig av cron jobs för allt som behöver ske med jämna mellanrum. Fastän detta betyder att man kan specificera händelser att ske under vilka specifika tider eller tidsintervall som helst, betyder det inte i sig att händelserna blir utförda precis de tiderna. Skälet för detta är att en webbplats, närmare sagt filerna som uppgör en webbplats, inte körs konstant och inte därför kan moderera sig själv enligt exakta tider. I WP kontrolleras systemtiden varje gång en sida laddas in, och ifall det konstateras att ett cron job behöver utföras utgående ifrån systemtiden görs det då. Detta kan innebära problem ifall jobbet tar en lång tid att utföra, eftersom det innebär att personen som gjort besöket till sidan är tvungen att vänta på att jobbet utförts innan sidan genereras färdigt. (Williams, Richard & Tadlock 2011, k. 13)

3.7.4 Internationalisering

För ett webbprojekt som utvecklas för att kunna användas i en mängd olika situationer är internationalisering ett viktigt tillägg. Internationalisering möjliggör att all text som visas i det grafiska gränssnittet kan lokaliseras, dvs. köras genom funktioner som kontrollerar vilket språk sidan använder och ersätter texten med korrekt översättning, förutsatt att översättningen blivit gjord och sparad på förhand. I praktiken görs detta genom att i stället för att använda en vanlig utskriftsoperator, använda en motsvarande funktion som har två argument: det första argumentet textsträngen som skall skrivas ut och det andra argumentet namnrymden inom vilken översättningen skall lagras. Sedan kan man med specialiserad mjukvara söka genom filer för att hitta dessa funktionsanrop för specifika namnrymder och

lista textsträngarna. Utvecklaren kan då själv skriva översättningar till ett nytt språk och lagra dem i en språkfil. Detta är välanvänt i samarbetssyften för mer populära projekt, där projektets slutanvändare själva skriver översättningar till olika språk ifall översättningen saknades från tidigare.

I många fall kan en kortare fras vara tvetydig då en översättning skall göras, och därför finns också funktioner som tillför möjligheten att som ett tredje argument ange ett sammanhang för översättningen, t.ex. definiera om ett ord används som verb eller substantiv, för att eliminera eventuella tvetydigheter. (Williams, Richard & Tadlock 2011, k. 5)

3.8 MySQL

MySQL är ett av världens mest använda system för hantering av relationsdatabaser. Eftersom MySQL erbjuds med fullständig funktionalitet utan kostnad, används MySQL ofta som databashanterare i icke-kommersiellt distribuerade projekt.

Den nya webbplatsen som utvecklas använder innehållshanteringssystemet WordPress, som i sin tur lagrar data i en MySQL-databas. På grund av detta kommer MySQL att vara det första valet, om informationen som roboten hämtat behöver lagras i en databas. WordPress innehåller många funktioner för att underlätta och förenkla genererande av databasförfrågningar samt öka säkerheten i databasen, vilket är av vikt för skapandet av en modul till en existerande webbplats.

4. Planering

Standardiseringen av strukturen för HTML-dokument är ett omöjligt uppdrag. De första standarderna för HTML-dokumenttypen gav friare händer åt webbutvecklare genom att tolerera tvetydigheter i struktur och syntax. Webbläsare, i sin tur, behövde utvecklas för att kunna hantera detta. Medan nya, mer strikta standarder för HTML-kod fortsätter att definieras, kommer det alltid att finnas sidor skrivna enligt gamla standarder. Detta innebär att en produkt som utvecklas för att tillförlitligt kunna navigera och analysera webbsidor behöver tillgodose den minsta gemensamma nämnaren, dvs. felformaterad dokumentkod. Med detta i åtanke behövde det planeras vilka tekniker som skulle användas för att försäkra att roboten kunde fungera konsekvent. Hur administrationsgränssnittet skulle se ut och fungera behövde också planeras innan arbetet fortskred.

4.1 Planering av roboten

4.1.1 Informationsaggregator

Planeringen påbörjades med att kartlägga om informationsaggregationen kunde verkställas med webbsidorna som DOM-dokument. DOM tillgodoser att webbsidan importeras med strukturen intakt och med möjligheten att söka efter element. Sökning kan ske utgående ifrån elementnamn, attributvärden eller ordning i DOM-strukturen. Från ett DOM-dokument kunde hämtas element för att lagras i räckor, som i sin tur lätt kunde skickas vidare från roboten för att bli lagrade i gränssnittet.

För att kunna söka efter element i ett DOM-dokument konstaterades XPath vara det bästa sättet. PHP/CURL skulle komma att behövas för att möjliggöra att webbroboten kunde hämta data över krypterade anslutningar, lagra sessionsdata samt skicka formulär.

Roboten skulle hämta informationen vid schemalagda händelser för att lagra den lokalt. Detta tillvägagångssätt skiljer sig från vanliga aggregationsrobotar, som i allmänhet skulle visa den samlade informationen direkt hämtad från källorna. Detta ansågs vara nödvändigt för att efter hämtning kunna ändra den lokala informationen till ett format eller layout som skulle lämpa sig mer för den lokala webbplatsen.

I kärnan av själva robotens funktionalitet skulle ligga en programklass, i vilken all logik kunde kapslas in. Objekt av klassen skulle instansieras med en textsträng innehållande en webbsidas URL samt en räcka med sökvägar i XPath syntax. Klassen skulle innehålla en funktion i vilken en sökning utförs utgående ifrån dessa två medlemmar och från vilken en räcka av resultat skulle returneras.

Idén bakom räckan med sökvägar var den att varje sökväg skulle representera en nivå i en trädstruktur. Den första sidan att genomsökas skulle vara trädets översta nivå, eller rot. Denna första sökning skulle ge upphov till en uppsättning av resultat på vilka en sökning med nästa sökväg skulle köras. Varje uppsättning av resultat skulle representera en nivå i det resulterande trädet.

För denna robot konsterades att varje sökväg förutom den sista endast skulle returnera ett länk-elements referensattribut, som anger en absolut eller relativ webbadress. Detta skulle innebära att varje ny uppsättning av resultat också skulle innebära en ny adress för nästa sökväg. Den sista sökvägen skulle då kunna söka efter den specifika informationen roboten var ämnad att hämta från dessa sidor. Alternativ till detta skulle kräva någon form av

intelligent urskiljning av olika resultattyper och möjligheten att fortsätta sökningen på olika vis beroende på typen av resultat som returnerades. Denna form av informationshantering bestämdes att vara onödig för detta arbete, eftersom all information som skulle sökas efter skulle finnas på samma nivå i en webbplats sidstruktur.

4.1.2 Formulärhantering

Den andra delen av robotens funktionalitet, sändning av information till ett externt formulär, var en helt skild del av planeringsarbetet. Eftersom en sida där ett formulär ifylls kunde antas använda sig av krypterad informationsöverföring, skulle roboten behöva vara kapabel att hantera användningen av certifikat och protokollet HTTPS. I och med version 5 av PHP finns stöd för HTTPS för hämtning av information, men på grund av att det inte finns stöd för själva sändningen av information via formulär skulle PHP/CURL ändå komma att vara ett krav.

Beroende på webbservern är en webbplats mer eller mindre noga med hur information blir skickad via formulär. Vissa webbservrar godkänner endast information skickat av en webbläsare, medan mer säkerhetsinriktade sidor kräver att informationen är uppställd i en viss ordning och ett visst format. Vissa webbservrar skickar också sessionsvariabler tillsammans med formuläret. PHP kan inte konfigureras för att möta dessa krav utan användning av PHP/CURL. (Schrenk 2009, k. 3. Introducing PHP/CURL)

Eftersom utbildningarna på CLL:s befintliga sidor var kopplade till ett ansökningsformulär behövde denna koppling överföras till den nya webbplatsen. Eftersom länken till ansökningsformuläret hade samma position på webbsidan oberoende av utbildningstyp, kunde denna länk också hämtas av aggregatorn och lagras som WordPress metadata för senare användning.

Då formulären visas stöter man på ett problem i hur WordPress särskiljer moduler från teman, konceptuellt om inte tekniskt. Medan en modul har samma förutsättningar som ett tema att innehålla grafiska stilmallar eller till och med helt skilda sidmallar är det inte menat att moduler skall förändra utseendet av en webbplats. Moduler är endast menade som tillägg för funktionalitet menade att kunna tillämpas på vilket tema som helst (Williams, Richard, Tadlock 2011, k. l. Advantages of Plugins). Om roboten skall kunna skriva ut ett formulär som det huvudsakliga innehållet på en sida, behöver roboten kunna ta ställning till vilket element av sidan som är menat att ha det centrala innehållet.

WordPress Plugin API innehåller information om när ett enskild inlägg visas. En funktion kunde skrivas för att anropas då ett inlägg skrivs ut, för att kontrollera om inlägget är en utbildning. Funktionen kunde då hämta ett skript för att ändra beteendet länken till ansökningssidan. Skriptet kunde använda en AJAX-funktion för att hämta formuläret från den befintliga ansökningssidan och visa det i stället för att hänvisa användaren till sidan.

4.2 Planering av gränssnittet

WordPress lagrar information för inlägg i form av metadata som kan definieras av användaren. Denna information kan sedan hämtas ur databasen med hjälp av inläggets ID samt metadatans ”nyckel”, eller namn. Eftersom informationen som samlas av roboten skulle skickas till gränssnittet som en räcka, skulle man i gränssnittet kunna lagra räckans olika element som metadata. Ett problem med detta var att olika sidor naturligtvis skulle ha olika struktur och olika namn på informationen som hämtas och gränssnittet skulle behöva ta detta i beaktande.

4.2.1 Regeluppdelning

Ett sätt att hantera skillnaden mellan hur data är uppställt på olika sidor vore att skapa skilda regler i gränssnittet. En regel skulle vara kopplad till en instans av webbroboten för en specifik extern webbplats. I en regel kunde sparas information om hur data skulle hanteras efter hämtning från denna sida, så att en användare kunde ändra vilka metadata olika element kunde sparas under. I gränssnittet skulle också kunna definieras ett tidsintervall eller specifika tider då genomsökning av de lagrade reglerna skulle utföras, med hjälp av WordPress cron jobs. Eftersom en genomsökning, beroende på regelmängden och informationen som hämtas lätt kunde räcka närmare en halv minut, är det en viktig urskiljning att inte köra dessa cron jobs för en ny sidohämtning som en kund kunde göra, utan hålla dem till sidohämtningar som användare gör i administrationsgränssnittet.

4.2.2 Datalagring

Eftersom hämtning av information för lagring sker vid olika tidsintervall och inte alltid som ett resultat av användarinverkan, behövdes ett sätt att lagra den hämtade informationen tills en användare kan gå genom den och föra in den till innehållshanteringssystemet.

Medan WordPress har funktioner för att underlätta skapande och underhåll av tabeller i databasen, konstaterades det att design och implementering av skräddarsydda tabeller skulle göra uppdraget mer komplicerat än nödvändigt. Med tanke på att hela innehållshanteringssystemet blivit uppbyggt för att hantera innehåll av denna typ, med färdiga gränssnitt för redigering och lagring av data, bestämdes det att det smidigaste sättet vore att lagra hämtad data som WordPress inlägg.

Eftersom information skulle hämtas från ett antal olika sidor och varje sida skulle innehålla flera element, skulle också det WordPress system som används kunna stöda denna uppsättning med information. Informationen kunde lagras på flera sätt, men det som konstaterades ha mest potential för integrering med användargränssnittet var Custom Post Types. Att lagra undan de hämtade sidornas information som inlägg skulle innebära att informationen sedan direkt kunde redigeras vid behov.

4.2.3 Grafiskt användargränssnitt

Den grafiska layouten av menyerna hade egentligen få begränsningar, men i och med användningen av WordPress som ramverk, konstaterades det vara ändamålsenligt att ta i beaktande de regler som lagts upp för utveckling och design av moduler, för att säkerställa att modulen skulle fortsätta att fungera långt framöver utan att behöva uppdateras.

5. Utförande

Utveckling av den fullständiga programvaran utfördes med början från roboten och fortsatte till den omgivande gränssnittsdelen. För roboten skulle utvecklas metoder att hitta specifik information från en webbplats och lagra den. I gränssnittet skulle utvecklas metoder för att klassificera och lagra informationen i en databas. I gränssnittet skulle också finnas funktionalitet för att visa de hämtade formulären dynamiskt, samt användargränssnitt för inställning av webbroboten och lagringsprocesser.

5.1 Utveckling av roboten

Roboten utvecklades och testades med ett enkelt webbskript i vilken en instans av roboten skapades och gavs parametrar till CLL:s Novia-enhets gamla webbplats. Resultatet av en testkörning visas i figur 6, där den hämtade informationen ställdes upp i ett tabellformat. I filen skapades roboten och dess sökningsfunktion påkallades, efter vilket den resulterande

räckan av DOM-element itererades genom och innehållet skrevs ut i tabellformat på webbsidan.

Figur 6. En av utbildningarna som resulterades av en testkörning av robotens aggregationsfunktion. Första elementet är titeln på den hämtade webbsidan, sedan utbildningsinformation i tabellformat och till sist länken till utbildningens ansökningsformulär.

5.1.1 Informationsaggregation

I den första iterationen av robotens sökningsfunktion fungerade den endast med två diskreta söksträngar; den första för att hitta länkar på rotsidan och den andra för att hämta de specifika elementen med innehållet man är intresserad av. Dessa returneras sedan av funktionen. Medan denna metod skulle räcka för den största delen av fallen konstaterades att funktionalitet behövdes för att inkludera så många nivåer av sökvägar som möjligt. Detta resulterade i utvecklingen av en motsvarande sökningsfunktion som fungerade med rekursivitet. Funktionen körs en gång, med den första angivna URL-adressen och räckan med sökvägar som parametrar. En sökning utförs från den första sökvägen i räckan och dess resultat lagras i en resultatvariabel. Den första sökvägen tas bort ur räckan och funktionen kallas på nytt för varje element i resultatvariabeln. Detta fortgår tills räckan endast innehåller en sökväg, i vilket fall sökningen körs och dess resultat returneras.

I WordPress modulen skapades en inläggstyp vid namn 'collected-data', vars uppgift skulle vara att hålla de hämtade sidornas information tills en användare kunde gå genom dem, redigera dem vid behov, och spara förändringarna. Via WordPress Plugin API definierades att vid publicering av ett inlägg av typen 'collected-data' skulle också ett inlägg av typen 'education' skapas, med identisk information. Vid därpå följande exekveringar av robotens informationshämtning lagras inga dubletter av redan existerande inlägg.

5.1.2 Formulärhantering

För utveckling av formulärhämtningen krävdes implementering av filhämtning med PHP/CURL. Klassen omformades för att initiera en cURL session och ange argument för hämtning och lagring av cookies, sändning av formulär samt vilken metod som skulle användas för sändningen; GET eller POST. Dessa cURL-inställningar skulle komma att vara annorlunda beroende på vilken webbplats roboten skulle gränsa till, och av den anledningen skulle de behöva förändras i instansieringen, dvs. i gränssnittet.

Då formulären hämtas behöver olika detaljer om den externa webbsidan tas i beaktande. Först och främst behöver roboten ta ställning till vare sig formulärsidan använder protokollet HTTP eller HTTPS. Detta kan bestämmas från formulärlänken. Om länken är absolut, dvs. innehåller den fullständiga adressen med protokoll, fås protokollet därifrån. Om länken är relativ till webbplatsens rot-adress och inte bestämmer ett protokoll skall den hänvisande sidans protokoll användas.

Eftersom formulär innefattar en del förändringar i hur roboten behöver hantera data behövde PHP/CURL konfigureras med grundläggande inställningar, innan något formulär kunde hämtas. Detta förevisas i kodexempel 4.

Kodexempel 4. Initiering och inställning av cURL-objektet. Användaragentens namn ändras för att lätt kunna kännas igen och verifiering av certifikat definieras att inte krävas. CURLOPT_URL och CURLOPT_REFERER definieras skilt för varje hämtning av formulär.

```
$ch = curl_init ();

/* Inställningar för robotens namn och sidan att hämtas. */
curl_setopt ( $ch, CURLOPT_USERAGENT, 'Aggregator Webbot' );
curl_setopt ( $ch, CURLOPT_URL, 'https://www.example.com/form.html' );

/* Tiden roboten väntar innan den överger en förfrågan. */
curl_setopt ( $ch, CURLOPT_CONNECTTIMEOUT, 5 );

/* Filen där cookies lagras och hämtas från, respektive. */
curl_setopt ( $ch, CURLOPT_COOKIEJAR, 'c:\cookies.txt' );
curl_setopt ( $ch, CURLOPT_COOKIEFILE, 'c:\cookies.txt' );

/* Vare sig exekveringsfunktionen skall returnera resultatet eller inte. */
curl_setopt ( $ch, CURLOPT_RETURNTRANSFER, 1 );

/* Inställningar för säker överföring. */
curl_setopt ( $ch, CURLOPT_SSL_VERIFYHOST, 0 );
curl_setopt ( $ch, CURLOPT_SSL_VERIFYPEER, 0 );

/* Sidan som roboten påstår sig refereras från. */
curl_setopt ( $ch, CURLOPT_REFERER, 'http://www.example.com/' );
```

Många av dessa inställningar är allmänna och behöver inte förändras mer specifikt för olika fall. Sökvägen till filen innehållande cookies beror på filstrukturen webbservern använder, i praktiken vare sig den ligger på en Windows, Mac eller Linux-maskin. Namnet på roboten kunde ändras till namnet på en populär webbläsare för att maskera robotens identitet, men eftersom detta uppdrag innebär hämtning av information inom organisationen behövs ingen anonymitet. Dessutom skulle det rentav hjälpa att lätt kunna följa robotens rörelser. Referensvärdet är viktigt ifall en webbserver kräver att åtkomst till

vissa sidor endast tillåts från andra sidor på den webbplatsen. Om en förfrågning görs av roboten men inget svar erhålls av webbservern, kan roboten vänta för evigt på ett svar om man inte konfigurerat timeoutintervallet CURLOPT_CONNECTTIMEOUT.

Efter att cURL initieras och inställningarna är gjorda kan sidan hämtas genom att köra exekveringsfunktionen med funktionsanropet i kodexempel 5.

Kodexempel 5. Eftersom CURLOPT_RETURNTRANSFER definierades vara TRUE returnerar funktionen resultatet av exekveringen, dvs. sidan som hämtats.

```
$data = curl_exec ( $ch );
```

Då sidan har blivit hämtad används aggregationsdelens sökfunktioner för att ta ut formulärelementet ur sidan och skicka det vidare till gränssnittsdelen för visning.

5.2 Utveckling av gränssnittet

Det påbörjande arbetet med användargränssnittet gjordes med inbyggda WordPress funktioner för skapande av menyer. WP implementerar metoder för att underlätta skapandet av gränssnitt med samma funktionalitet och utseende som WP's existerande gränssnitt. På http://www.wordpress.org finns också väldokumenterade riktlinjer för att följa så kallade ”best practices” för utveckling av moduler.

5.2.1 Skapande av menyer

Fördelarna med att använda WordPress inbyggda funktioner för utökad funktionalitet kommer fram i systemets hantering av administrationsgränssnittets sidor. Med några korta funktionsanrop, som visas i kodexempel 6, kan en ny menystruktur skapas, och innehållet i dessa menyer kan lätt skrivas med dedikerade funktioner.

Kodexempel 6. Funktionen som genererar sidorna anropas då administrationsgränssnittets menypanel laddas in. Som parametrar till funktionerna anges namnen på menyerna och funktionerna i vilka dess innehåll genereras.

```
<?php

add_action ( 'admin_menu', 'aggbot_add_menus' );

function aggbot_add_menus () {

	add_menu_page ( 'Aggregation Bot', 'Aggregation Bot', 'manage_options', 'aggbot-main-menu', 'aggbot_main_menu' );

	add_submenu_page ( 'aggbot-main-menu', 'Add New Aggregation Rule', 'New Rule', 'manage_options', 'aggbot-new-rule', 'aggbot_new_rule' );

	add_submenu_page ( 'aggbot-main-menu', 'Manage Aggregation Rules', 'Manage Rules', 'manage_options', 'aggbot-manage-rules', 'aggbot_manage_rules' );
}

?>
```

5.2.2 Visning av formulär

Medan roboten har all funktionalitet för att söka efter och hämta formulärelementet från en specificerad sida, behöver den veta när detta skall ske och hur formuläret skall visas. Visningen skulle behöva hantera WordPress-systemet så allmänt som möjligt för att möjliggöra kompatibilitet med en WordPress-sida oberoende av det grafiska temat i användning. I WordPress plugin API finns en ”hook” vid namn ”the_post”, som passeras varje gång ett nytt inlägg laddas in i systemet för hantering, t.ex. då inlägget skall visas på webbsidan. I detta skede, då ett inlägg av typen ”education” skall visas, körs ett skript för att ändra funktionaliteten av länken till ansökningssidan. I kodexempel 7 visas handlingen som körs vid ”the_post” och dess kopplade funktion.

Kodexempel 7. Skriptfilen laddas in med parametrar för AJAX-funktionalitet då ett inlägg av typen ”education” påträffas.

```
<?php
add_action ( 'the_post', 'aggbot_add_js' );
```

```
function aggbot_add_js () {
	global $post;
	if ( get_post_type ( $post ) == 'education' ) {
		wp_enqueue_script ( 'aggbot_script', plugin_dir_url ( __FILE__ ) . 'js/aggbog-script.js',
array ( 'jquery' ), AGGBOT_VERSION, true );
		$protocol = isset ( $_SERVER[ "HTTPS" ] ) ? 'https://' : 'http://';
		//output admin-ajax.php url with right protocol
		$params = array (
			'ajaxurl' => admin_url ( 'admin-ajax.php', $protocol )
		);
		wp_localize_script ( 'aggbot_script', 'aggbot_script', $params );
	}
}
?>
```

Eftersom ett inlägg av typen ”education” antas ha metadata som innehåller adressen till ansökningssidan med formuläret kan skriptet, som visas i kodexempel 8, söka efter denna adress i sidan som skrivs ut. Om adressen hittas ändras länkens beteende till att göra ett anrop till roboten. Detta innebär att användaren inte ser den egentliga sidan med formuläret, utan formuläret skrivs ut till sidan användaren redan är på och formuläret kan därifrån skickas. Om adressen inte överensstämmer med den som blivit specificerad som ansökningssidan, eller om användarens webbläsare inte tillåter JavaScript, tas ingen handling och länken fungerar som vanligt. Detta innebär att i värsta fall blir en användare hänvisad till en befintlig sida med ett fungerande ansökningsformulär.

Kodexempel 8. Länkens standardbeteende nekas och i stället laddas data in från en AJAX-funktion och sänds tillbaka till klienten för att adderas till slutet av innehållselementet.

```
( function ( $ ) {
	$ ( document ).ready ( function () {
		$ ( 'a[href="' + education-link + '"]' ).click ( function () {
			$ ( this ).preventDefault ();
			$.get ( aggbot_script.ajaxurl, data, function ( data ) {
				$ ( '#content' ).append ( data );
			}, 'html');
		});
```

```
});
}) (jQuery);
```

Likaså då formuläret är ifyllt och postas utförs en liknande operation. Sidan töms på formuläret och i stället hämtas den följande sidan som skulle visas, antingen formulärsidan på nytt med meddelanden om att sändningen misslyckades eller en text om att den lyckades.

6. Användning av roboten

Roboten är menad för användning av en planerare med tillgång till administrationsgränssnittet. Eftersom sökvägarna görs med XPath syntax är det troligt att allmänna användare av webbplatsen inte kommer att kunna skräddarsy sökvägarna själva. Eftersom webbsidan roboten påbörjar sökningen från inte antas förändras nämnvärt efter att roboten blivit konfigurerad, krävs det inte heller desto mera arbete att upprätthålla dess funktionalitet. För detta ändamål kan konfigurationssidan döljas för användare som inte har administratörsrättigheter.

6.1 Skapande av regler

Då en regel skapas kan man ange namnet för regeln, adressen till sidan från vilken sökningen skall påbörjas samt räckan med sökvägar som regeln använder. Sidan för skapandet av en ny regel visas i figur 7.

Figur 7. Ett exempel på en regel. Den första sökvägen hämtar länk-attributen från den specificerade sökvägen och den andra hämtar sidans titel, tabell och ett specifikt länk-attribut till sidans ansökningsformulär.

6.2 Tilldelning av metadata

Efter att en regel blivit konfigurerad kan man utföra en genomsökning, efter vilken en resultatlogg visar vilka nya inlägg som skapats och vilka inlägg som existerat sedan tidigare, vilket visas i figur 8. Alla hämtade inlägg kan redigeras direkt via administrationsgränssnittet.

Figur 8. Resultatlogg av exekvering av aggregationsfunktionen.

Utbildningarna sparas som en inläggstyp skapad enbart för denna modul och kommer på så vis inte i användning av andra delar av WordPress. Inläggen fungerar enbart som behållare för informationen så att en planerare kan gå genom vilka utbildningar som inte blivit införda till sidan efter att de blivit hämtade av roboten. Enligt funktionen som förklarades i kapitel 5.1.1 skapas ett inlägg av typen ”education”, vilket är i direkt användning av webbplatsen, då man väljer att publicera ett inlägg av typen ”collected-data”. Detta innebär att utbildningar inte direkt publiceras till webbplatsen efter hämtning, så att en användare kan kontrollera innehållet av en utbildning innan den publiceras.

7. Resultat

Resultatet av uppdraget är en prototyp av roboten som fungerar mot en av CLL:s existerande webbplatser, nämligen Noviaenhetens. Detta eftersom Noviaenheten ställer upp utbildningslistan och de enskilda utbildningssidorna på ett enhetligt vis, väl lämpat för

en automatiserad hämtning. Åbo- och Vasaenheternas webbplatser listar kurser mindre enhetligt, med länkar icke-konsekvent placerade i HTML-dokumentets struktur, vilket förhindrar robotens förmåga att fungera konsekvent.

Roboten var kapabel att hämta utbildningarna från sidor och spara dem som WordPress inlägg. Roboten innefattade också funktionaliteten för att hämta formulär och skicka tillbaka deras information över krypterade anslutningar, men utvecklingsarbetet för att anpassa detta till ett grafiskt tema blev inte fullbordat innan detta arbete skrevs.

8. Diskussion

Då idén för webbroboten framkom var den en behövlig del för vad som skulle komma att bli CLL:s nya webbplats. Tanken var att man kunde hantera utbildningslistorna från var och en webbplats utan att behöva manuellt kopiera innehållet från de gamla sidorna till de nya, och att kunden kunde fylla i formulär och se utbildningsinformationen utan att behöva lämna den nya webbplatsen för att bli överförd till de gamla, föråldrade sidorna. Under arbetets gång blev det dock klart att roboten inte skulle kunna fungera konsekvent på alla webbplatser, eftersom utbildningslistornas information och länkar inte var konsekvent strukturerade.

Vad robotens funktionalitet beträffar var det lättare än väntat att skapa sökvägar för element och hämta deras innehåll samt visa formulär från externa sidor och skicka den ifyllda informationen. Hanteringen av de hämtade sidorna med DOM gav all flexibilitet som skulle behövas. Eftersom utbildningslistorna inte var alltför stora led hanteringen inte av DOM:s något större krav på arbetsminne. Skapandet av ett fullständigt gränssnitt mellan roboten och WordPress fullbordades inte, detta på grund av att utvecklingen av ett gränssnitt med bred funktionalitet i slutändan konstaterades vara utanför tidsramen för det skrivna arbetet. I slutändan fick uppdraget fungera som en ren prototyp för hur roboten kunde implementeras för en webbplats så långt som att importera utbildningar och deras information. Prototypen visade också att hämtningen av formulär över krypterade anslutningar fungerade, samt att denna information kunde skickas tillbaka utan att användaren behövde besöka den externa sidan.

Vidare utveckling av roboten skulle i första hand innefatta implementering av den grafiska delen av att kunna bädda in formulären i en WordPress sidmall på ett dynamiskt vis.

Källförteckning

AJAX Introduction (2012). http://www.w3schools.com/ajax/ajax_intro.asp (hämtad: 30.3.2012)

Harold, R.E. & Means, W.S. (2001) *XML In a Nutshell* Sebastopol: O'Reilly and Associates

JavaScript Introduction (2012) http://www.w3schools.com/js/js_intro.asp (hämtad: 30.3.2012)

Lerdorf, R. & Tatroe, K. (2002) *Programming PHP* Sebastopol: O'Reilly and Associates

Om CLL (2012). http://abonovia.abo.fi/cll/om-cll/ (hämtad: 30.3.2012)

Recursion in C and C++ - Cprogramming.com (2012)
http://www.cprogramming.com/tutorial/lesson16.html (hämtad: 30.3.2012)

Schrenk, Michael (2009). *Webbots, Spiders and Screen Scrapers (*2. uppl.) San Fransisco: No Starch Press

Stern, H., Damstra, D. & Williams, B. (2010) *Professional WordPress Design and Development* Indianapolis: Wiley Publishing

Williams, B., Richard, O. & Tadlock, J. (2011) *Professional WordPress Plugin Development* Indianapolis: Wiley Publishing