Att utveckla en Portlet enligt JSR-168

Fredrik Tuomas

1 juli 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Jan-Erik Moström
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Sammanfattning

Denna rapport beskriver ett examensarbete utfört på UMDAC vid Umeå Universitet. Syftet med examensarbetet har varit att erhålla erfarenheter kring och utvärdera *Java(TM) Portlet Specification*. En köp- och säljportlet har utvecklats i detta arbete. Rapporten innehåller även en mindre studie över tillgängliga system för att bygga en webbapplikation och lite generell information om portaler.

Abstract

This master thesis report describes the development of a JSR-168 based portlet at UMDAC, Umeå University. The goal with the master thesis has been to acquire experience and information about portlet development with the Java(TM) Portlet Specification. A "Buy and Sell" portlet has been implemented and is part of the work. The first chapters in this report contains a rather simple comparison of different methods to create a web application.

Innehåll

1 Inledning och syfte **1**

1.1 Bakgrund . . . 2
1.2 Problembeskrivning . . . 2
1.3 Rapportens uppbyggnad . . . 2

2 Att generera dynamiskt innehåll på world wide web **3**

2.1 Viktiga punkter . . . 4
2.2 Server Side Scripting . . . 4
2.2.1 CGI - Common Gateway Interface . . . 4
2.2.2 FastCGI . . . 7
2.2.3 mod_perl . . . 8
2.2.4 PHP: Hypertext Preprocessor . . . 8
2.2.5 Java 2 platform, Enterprise Edition (J2EE) . . . 9
2.2.6 Microsoft: Active Server Pages och ASP.NET . . . 11
2.2.7 RoXen Macro Language, Cold Fusion mm . . . 11
2.2.8 Prestandajämförelse Server side scripting . . . 12
2.3 Client scripting . . . 12
2.3.1 JavaScript . . . 13
2.3.2 VBScript . . . 13
2.4 Ajax . . . 13
2.4.1 Vad differentierar AJAX från en vanlig webbapplikation? . . . 14
2.4.2 AJAX i praktiken . . . 14
2.4.3 Utveckla tjänster enligt AJAX . . . 14
2.4.4 Säkerhet . . . 16

3 Portaler och portlets **17**

3.1 Vad är en portal? . . . 17
3.2 uPortal . . . 19
3.3 Umeå universitets studentportal . . . 19
3.4 Umeå universitets val av uPortal . . . 19
3.5 JSR-168 - Java™ Portlet Specification 1.0 . . . 20

4 Köp- och säljportlet **21**
4.1 Planering . . . 21
4.2 Verktyg . . . 21
4.3 Implementation . . . 22
4.3.1 Logik . . . 22
4.3.2 Skapa en ny annons . . . 23
4.3.3 Utföra en sökning . . . 23
4.3.4 Webbgränssnitt . . . 24
4.3.5 Databas . . . 24
4.4 Övrigt . . . 24
4.5 Problem . . . 25
4.6 Utseende och funktionalitet . . . 25
4.7 Uppfyllelse av kravspecifikation . . . 25

5 Begränsningar och framtida arbete **31**
5.1 Begränsningar . . . 31
5.2 Framtida arbete . . . 31

6 Tack **33**

Referenser **35**

A Ordlista **37**

B Kravspecifikation **39**

Kapitel 1

Inledning och syfte

Fler och fler tjänster görs tillgängliga på webben i dagens samhälle. Idag betalar de flesta sina räkningar på nätet och vissa deklarerar också på det sättet. Nätet utvecklas och varje månad dyker det upp nya tjänster. Spel, telefoni och dating är bara några av alla de nya tjänster vi ser.

Umeå universitet har planer på att införa en studentpotal. En studentportal är i detta fall en webbplats för att samla de webbtjänster Umeå universitet erbjuder sina studenter. Idag finns bland annat en webbtjänst för e-post och en där studenter kan se sina studiemeriter. Ideén med en studentportal är att få alla webbtjänster samlade på ett ställe med en gemensam grafisk profil. Ytterligare en fördel med en studentportal är att bara behöva logga in en gång och sedan kunna nyttja alla tjänster, utan att behöva autentisera sig på nytt. Idag krävs en separat inloggning för varje tjänst.

UMDAC är Umeå universitets datorcentral och är underställt rektor som fastställer föreskrifter för UMDACs verksamhet. UMDAC är formellt en serviceenhet vid Umeå universitet. UMDAC har till uppgift att på uppdrag och mot ersättning tillhandahålla IT-tjänster för forskning, utbildning, administration och studenter vid Umeå universitet. UMDAC får även åta sig uppdrag från externa uppdragsgivare.

Denna rapport är en del i ett examensarbete utfört vid institutionen för datavetenskap vid Umeå universitet. Examensarbetet har utförts på uppdrag av UMDAC. Examensarbetets syfte var att kunna leverera en köp- och säljfunktion till en blivande studentportal vid Umeå universitet. Delsyften med examensarbetet har varit att identifiera möjligheter och problem vid utvecklingsarbete av portlets baserade på JSR-168 i uPortal. För att underlätta den framtida utvecklingen av portlets vid UMDAC har det varit ett krav att modularisera den utvecklade portleten.

Webbplats, webbsida mfl stavas konsekvent med två b i denna rapport enligt Svenska Språknämndens rekommendation.

I denna rapport beskrivs examensarbetet i sin helhet.

1.1 Bakgrund

Umeå universitet har idag två officiella installationer av portalramverket *uPortal*[1]. En för informationsspridning till universitetsförvaltningen och en för att exponera ett antal studierelaterade tjänster för studenter.

De installationer av uPortal som finns vid universitetet idag är baserade på en föråldrad version av portalramverket. uPortal utvecklas kontinuerligt och den teknik som används i de nuvarande portalerna för att tillhandahålla information och tjänster kommer att fasas ut successivt till förmån för *portlets*.

1.2 Problembeskrivning

Portlets enligt JSR-168 har endast varit en standard sedan drygt ett år vilket innebär att det inte finns många referensimplementationer. Detta examensarbete går därför ut på att erhålla erfarenheter av utvecklingsarbete enligt denna standard samt en implementation av en köp- och säljfunktion.

Köp- och säljfunktionen skall bland annat ha stöd för olika typer av annonser (tex Säljes, Köpes, Bytes) och olika kategorier (tex Cyklar, Kurslitteratur, Fordon). Namnen på dessa olika kategorier och typer av annonser skall vara redigerbara i efterhand utan förändringar av befintliga annonser.

Vidare skall funktionaliteten modulariseras så att den går att återanvända i andra projekt, speciellt vid utvecklingen av andra portlets i en kommande studentportal.

För en fullständig kravspecifikation se appendix B.

1.3 Rapportens uppbyggnad

Kapitel 2 behandlar olika metoder för att bygga dynamiska webbplatser, vilka metoder det finns, vilken prestanda de levererar och lite om framtidens webbtjänster.

Kapitel 3 ger en närmare presentation av portaler och beskriver också portletstandarden och uPortal.

Kapitel 4 behandlar den portlet som implementerades i examensarbetet.

Kapitel 5 innehåller slutsatser och diskussion.

[1] En närmare presentation av uPortal ges i sektion 3.2

Kapitel 2

Att generera dynamiskt innehåll på world wide web

För att kunna skapa interaktiva högkvalitativa webbtjänster behövs en lösning för att generera dynamiskt innehåll på webbplatsen. Detta kan till exempel innebära att en webbsida sätts samman utifrån information hämtad från en databasserver tillsammans med information användaren har matat in via sin webbläsare.

Den vanligaste metoden för att generera dynamiska webbsidor kallas ”Server side scripting” (sv ung ”skriptning på servern”). Skriptning på servern innebär att servern (i detta fall webbservern) bygger ihop den webbsida den serverar klienten (i de flesta fall en webbläsare) utifrån data den hämtar på olika håll. Normalt sker detta genom att en mall för sidan hämtas och fylls ut med data från till exempel ett databasanrop eller en fil nåbar av webbservern. Dessa data kan bestå av produktinformation och/eller inställningar användaren har gjort.

I motsats till ”Server side scripting” finns ”Client scripting” eller skriptning på klientsidan. Exempel på sådan skriptning är JavaScript. Detta innebär att klienten kör ett skript den läser från webbsidan. Detta skript kan sedan påverka utseendet eller beteendet hos webbsidan.

Denna metod kan vara att föredra när en enklare form av dynamik önskas, användningsområdet för ”Server side scripting” och ”Client scripting” är något skilda och kan inte direkt jämföras.

I detta kapitel presenteras kortfattat några lösningar för att generera dynamiska sidor. Studien baseras på artiklar och kortare intervjuer med webbutvecklare på UMDAC.

2.1 Viktiga punkter

I denna studie av plattformar för webbutveckling läggs större vikt vid några punkter. Dessa är:

- ⋆ Möjlighet att skilja på logik och utseende
- ⋆ Prestanda
- ⋆ Tillgängliga utvecklingsmiljöer
- ⋆ Möjligheter till felsökning (debugging)

Detta urval baseras på den kunskap studier av artiklar i ämnet har frambringat samt tips och information erhållen från webbutvecklare vid UMDAC.

Möjlighet att skilja på logik och presentation är mycket viktig då dessa bitar ofta görs på olika håll en i organisation. Görs detta på rätt sätt kan det också vara möjligt att återanvända logiken i andra applikationer (detta behöver inte bara inkludera webbapplikationer). Att göra på detta sätt gör också att det blir lätt att ändra utseendet. Dessutom blir det lättare att underhålla kodbasen.

Den prestanda ett visst system kan prestera är viktig. Olika system skalar olika bra och det är viktigt att välja ett system som kan växa i takt med trafiken till den aktuella webbapplikationen.

En gedigen utvecklingsmiljö underlättar utvecklingen och tillhandahåller stöd för utvecklaren i form av goda möjligheter till felsökning och hjälp.

När en webbutvecklingsplattform skall väljas finns det ytterligare två viktiga punkter att ta hänsyn till:

- ⋆ Kompetensen hos de befintliga utvecklarna i organisationen
- ⋆ Licenskostnader

Har en organisation redan kompetens inom en viss plattform, till exempel *J2EE* är det naturligtvis en stor fördel att välja J2EE även för webbutveckling.

2.2 Server Side Scripting

2.2.1 CGI - Common Gateway Interface

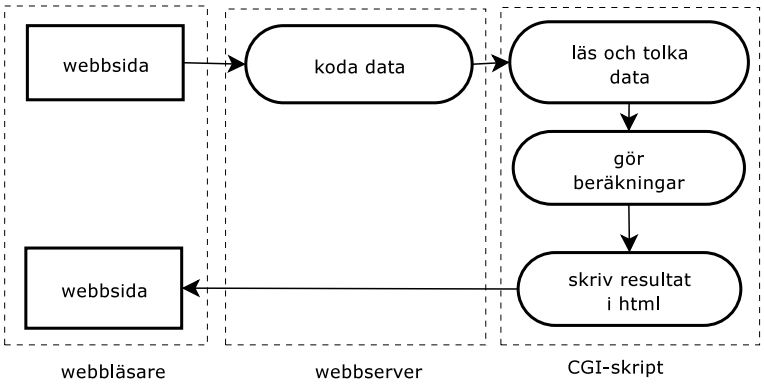
Common Gateway Interface eller CGI är den första metoden som utvecklades för att skapa dynamiskt webbinnehåll [7]. CGI fungerar helt enkelt genom att låta webbservern

exekvera ett program eller skript och därigenom skapa den webbsida som efterfrågats. CGI-programmet eller CGI-skriptet för att generera en webbsida kan implementeras i många olika programspråk/skriptspråk, detta begränsas egentligen bara av vad som är tillgängligt i systemet. Exempel på programspråk/skriptspråk är [1]:

- C/C++
- Fortran
- PERL
- TCL
- Bourne shell
- C shell
- Visual Basic
- AppleScript
- Python

CGI är egentligen endast ett protokoll på hur webbservern skickar data till och från ett externt program. Kortfattat fungerar det så att när webbservern får en förfrågan till ett skript via en URL (exempelvis http://www.cs.umd.edu/cgi-bin/search.pl) startar webbservern programmet/skriptet och överför datan från anropet på en bestämd form.

Hur webbservern överför data är lite beroende på operativsystem. Under UNIX-kompatibla operativsystem sker dataöverföringen genom en kombination av miljövariabler och standard in/ut.



Figur 2.1: Exekvering av ett CGI-skript.

Prestanda

När webbservern får ett anrop till ett CGI-skript startar den en ny process och sätter upp miljön (med rätt miljövariabler) för att exekvera skriptet. Webbservern tar sedan hand om utdata från den nya processen och vidarebefordrar detta till den anropande klienten.

Att starta en ny process varje gång ett skript ska exekveras är bra. Detta medför att ett felaktigt skript inte automatiskt krashar webbservern. Om webbservern och skriptet hade körts i samma process finns det en risk att hela processen hängt sig och webbservern slutat att fungera.

Att starta en ny process varje gång en webbsida skall genereras blir snabbt ohållbart när webbserverns belastning, dvs antal förfrågningar per sekund, ökar. Detta beror på att kostnaden, operativsystemets sk "overhead", är stor när det gäller att starta en helt ny process. När exekveringen sker på detta sätt innebär det också att de resurser ett skript använder måste allokeras gång på gång istället för att återanvändas. Exempel på en sådan resurs kan vara en databaskoppling eller inläsning av data från en inställningsfil.

Sammantaget innebär detta att CGI i denna form inte lämpar sig för webbplatser med väldigt hög belastning.

Utvecklingsverktyg

Då CGI fungerar med nästan alla programspråk/skriptspråk finns det många utvecklingsmiljöer och verktyg tillgängliga.

Möjligheterna till felsökning (debugging) beror mycket på vilken webbserver och vilka utvecklingsverktyg som används.

Portabilitet

CGI-skript fungerar på nästan alla webbservrar och därigenom också på väldigt många plattformar. Det kan vara ett problem att flytta ett CGI-skript mellan två olika plattformar om skriptet använder sig av en operativsystemsspecifik funktion. För att vara säker på att kunna flytta ett CGI-skript mellan olika plattformar måste detta beaktas noga under utvecklingen av skriptet.

Sidgenerering

CGI-standarden behandlar inte någon speciell funktion för att skilja på logik och presentation. Oftast kodas presentationen direkt i CGI-skriptet. Det finns dock paket och tillägg fritt tillgängligt på nätet, t ex Template Tookit[1].

[1] http://www.template-toolkit.org/

2.2.2 FastCGI

FastCGI är en öppen och gratis teknologi utvecklad av företaget *Open Market Inc.* FastCGI är väldigt likt CGI med några viktiga skillnader [9]:

1. FastCGI håller skripten/programmen levande mellan varje anrop och måste således inte starta en ny process mellan varje anrop
2. FastCGI kommunicerar med webbservern via en pipa med full duplex istället för via miljövariabler
3. Ett FastCGI skript/program behöver inte köras på samma maskin som webbservern

När ett FastCGI-skript körs på samma maskin som webbservern sker kommunikationen via en dubbelriktad pipa. Körs FastCGI på en annan maskin sker kommunikationen över en TCP-koppling.

Prestanda

Prestandan för FastCGI är som namnet antyder, bättre än den vanliga metoden för att exekvera CGI-skript. De avgörande skillnaderna ligger i att FastCGI låter skripten leva mellan anropen och på så sätt behöver inte onödiga resursallokeringar ske för varje anrop.

Ett FastCGI-skript kan skapa bestående resurser, detta kan till exempel vara en databaskoppling. Denna databaskoppling kan sedan vara öppen hela tiden och återanvändas varje gång FastCGI-skriptet anropas. Detta medför att databaskopplingen inte behöver öppnas varje gång ett sidanrop sker och genom detta ökas effektiviteten.

Sammantaget innebär dessa förbättringar att prestandan för FastCGI-skript vida överskrider den för ordinära CGI-skript.

Utvecklingsverktyg

FastCGI skrivs antingen i skriptspråket Perl eller programmeringsspråket C. Båda dessa är väldigt spridda och har använts i många år. Det finns därför mycket bra utvecklingsverktyg och många färdiga bibliotek att använda.

Portabilitet

FastCGI finns tillgängligt för en mängd olika webbservrar från olika tillverkare. Bland dessa återfinns Apache och Internet Information Server.

Portabiliteten hos ett enskilt FastCGI-skript beror endast på dess innehåll. Används en operativsystemsspecifik funktion blir det svårare att portera skriptet till en annan plattform. Undviks dessa funktioner går det relativt enkelt att flytta skripten till en annan plattform.

2.2.3 mod_perl

mod_perl är en modul (dynamisk utökning) av webbservern Apache. Den gör att Perl-skript och program skrivna i C (de vanligaste formerna av CGI-programmering) går att köra på ett effektivt sätt i en av de mest använda webbservrarna idag (Apache). Genom att använda en modul för att exekvera dessa skript/program istället för den klassiska CGI-metoden undviks de största prestandaproblemen. [7]

Prestanda

Prestandan för C och Perl-baserade CGI-skript vid användandet av mod_perl är mycket högre än den för motsvarande skript med den konventionella exekveringsmetoden. Detta gör att användandet av Apache tillsammans med mod_perl är väldigt vanligt.

Då en modul används körs skripten/programmen i webbserverns process. Detta innebär att de resurskrävande processinitialiseringarna och miljövariabeluppsättningarna undviks. Det arbete som krävs vid varje anrop till ett skript kan minimeras och kommunikationen mellan skript och webbserver blir också effektivare.

Utvecklingsverktyg

Se sektion 2.2.1

Portabilitet

Se sektion 2.2.1

Sidgenerering

Se sektion 2.2.1

2.2.4 PHP: Hypertext Preprocessor

PHP är ett skriptspråk utvecklat speciellt för att användas på webben till att generera dynamiska webbsidor. Det har en rätt brokig historia och har i omgångar skrivits om

helt från grunden. Den senaste stabila versionen är 5.0 och har ett utökat stöd för objektorientering och felhantering.

PHP är i grunden ett html-inbäddat språk där koden skrivs direkt i html-sidorna blandat med den vanliga html-koden. Idag finns dock många verktyg och paket för att separera logik och presentation.

PHP är helt fritt att använda och det finns väldigt mycket funktionalitet och stöd tillgängligt gratis på www.

Prestanda

PHP körs normalt antingen som CGI-skript på en kompatibel webbserver eller genom modulen *mod_php* till webbserven Apache. Den vanligaste metoden är att använda mod_php då det ger bäst prestanda. När PHP körs i form av ett CGI-skript exekverar en PHP-interpreterare det aktuella PHP-skriptet.

Utvecklingsverktyg

Till PHP finns det många utvecklingsverktyg tillgängliga för olika plattformar. Möjligheter till felsökning beror mycket på i vilken miljö utvecklingen sker.

Portabilitet

Ett PHP-skript är fullt portabelt så länge inga operativssystemspecifika metoder har använts. Det vanligaste är förmodligen att PHP körs på en Linuxplattform tillsammans med Apache, men det går lika bra att köra PHP och Apache på Windows. Det går även bra att köra PHP på Internet Information Server.

Sidgenerering

Till PHP finns flera olika färdiga paket för att hantera separation av logik och presentation. Ett av de mest populära heter *Smarty*[2]. Smarty erbjuder väldigt kraftfulla funktioner för sidgenerering. Smarty har en inbyggd felsökningsfunktion för att underlätta utveckling.

2.2.5 Java 2 platform, Enterprise Edition (J2EE)

Suns teknologi *JAVA* innehåller två komponenter, så kallade *Standard JAVA Extension API*, för att utveckla dynamiska webbsidor och bygga stora webbplatser/webbtjänster.

[2]http://smarty.php.net/

Dessa är: *The Java Servlet API* och *JavaServer Pages(TM).*

Servlets är en mycket omfattande teknologi för att skapa allt från renodlade webbtjänster och stora webbplatser till att skapa enklare webbsidor för det mindre företaget eller hemmamarknaden. Servlets gör hela JAVA-plattformen tillgänglig för webbutvecklare med alla de fördelar det innebär (stark typning och felhantering osv) [5].

JavaServer Pages använder en form av html-inbäddade taggar för att på ett enkelt sätt generera sidor utifrån mallar men också för att kunna göra väldigt komplexa operationer utan att behöva skriva en enda rad JAVA-kod. De speciella taggarna specificeras genom så kallade *Tag libraries*. Förutom de standardiserade taggbiblioteken tillgängliga går det att skapa egna för att tillföra ytterligare funktion till JavaServer Pages-tekniken [6]

Om dessa två tekniker (Java Servlet och JavaServer Pages) kombineras fås ett mycket kraftfullt verktyg för att skapa funktionella webbplatser.

Prestanda och säkerhet

En Java Servlet körs i en server med stöd för JAVA Servlets. En servlet laddas alltid dynamiskt och körs i serverns process i en egen tråd. Detta innebär att de kostsamma skapandet av en ny process vid varje klientanrop kan undvaras och resultatet blir möjligheter till riktigt bra prestanda. En servlet har också möjlighet att skapa bestående uppkopplingar mot till exempel databaser eller annan resurs.

Varje servlet körs i en speciell container vilket innebär att en eventuellt dåligt programmerad servlet inte påverkar säkerheten hos hela systemet.

Utvecklingsverktyg

Till Java finns det en mängd utvecklingsverktyg där Eclipse och IBM WebSphere är de största. Dessa integrerade utvecklingsmiljöer har mognat under en längre tid och innehåller idag det mesta en utvecklare kan önska sig.

I Eclipse finns goda möjligheter till debuggning, paketering och allt annat en utvecklare kan önska sig av en utvecklingsmiljö. Integrationen med webbservrar/applikationsservrar vid webbutveckling är tillfredställande men ej fulländad. Bland annat saknas ett smidigt sätt att testköra webbapplikationen direkt från Eclipse.

Portabilitet

Javamiljön är mycket portabel och en webbtjänst utvecklad i Java kan driftsättas på ett brett urval av plattformar. Alltifrån den fria Tomcat till IBM's kommersiella IBM WebSphere® Application Server. I stort sett kan en webbtjänst driftsättas på vilken plattform som helst som har stöd för Java.

2.2.6 Microsoft: Active Server Pages och ASP.NET

Active Server Pages lanserades i december 1996 av Microsoft för att möta efterfrågan på ett enkelt språk för att skapa dynamiska webbsidor. I denna första version av Active Server Pages användes Visual Basic Script inbäddat i webbsidan. Denna version av ASP utökades under några år för att sedan övergå till att bli ASP.NET.

ASP.NET är mycket flexibelt och tillåter att webbsidor skapas utifrån en mängd olika programspråk, bland annat C# och VS.NET. Den nya versionen av Active Server Pages är ett mycket kraftfullt verktyg för att utveckla avancerade webbtjänster och webbplatser.

Prestanda

ASP.NET skapar tillsammans med Internet Information Server en mycket flexibel miljö där mycket god prestanda kan uppnås. Databaskopplingar kan hållas öppna mellan sidanrop och skripten exekveras utan onödig resursförbrukning.

Utvecklingsverktyg

Microsoft erbjuder mycket bra utvecklingsverktyg. Främst genom Microsoft Developer, deras senaste integrerade utvecklingsmiljö med möjlighet att utveckla i väldigt många olika programspråk och skriptspråk både för webb och normal applikationsutveckling.

Portabilitet

Den största svagheten med att använda Microsofts system för att generera dynamiska webbsidor är att det endast är möjligt att använda Internet Information Server som webbserver och Microsoft Windows som plattform. Utvecklar du en tjänst i ASP.NET är du tvungen att köra den på en Microsoftwebbserver (Internet Information Server).

2.2.7 RoXen Macro Language, Cold Fusion mm

Utöver de tekniker omnämnda tidigare i denna rapport existerar flera olika mer eller mindre spridda alternativ. Bland dessa återfinns bland annat *RoXen Macro Language* (RXML) vilket är ett html-inbäddat server-sido skriptspråk utvecklat för att användas tillsammans med den svenska fria webbservern *Roxen*[3].

ColdFusion är en teknik utvecklad från början av Allaire men ägs och utvecklas idag av Macromedia (numera uppköpta av Adobe). ColdFusion har funnits länge och fått

[3]http://www.roxen.com

många anhängare genom åren. Det är ett html-inbäddat taggbaserat språk som kompilerar till bytekod (mellanliggande kod) och har därför potential att bli snabbare än ett interpreterande språk som t ex PHP.

2.2.8 Prestandajämförelse Server side scripting

Lance Titchkosky, Martin Arlitt och Carey Williamson presenterar i sin artikel "A Performance Comparison of Dynamic Web Technologies" [3] en studie om prestandan hos några av de vanligaste metoderna för server side scripting. De metoder de jämför är Perl (via mod_perl på Apache 1.3), JAVA (med tre olika servletcontainrar: Tomcat, Jetty och Resin) och PHP (med mod_php på Apache 1.3 och 2.0).

Testerna utfördes i ett speciellt nätverk där webbservern kördes på en egen maskin och en andra maskin genererade förfrågningar på olika sätt för att efterlikna en tungt belastad webbplats. Resultaten visar på att av de testade teknikerna har PHP sämst prestanda och de JAVA-baserade alternativen har den bästa prestandan.

Ett av testerna gick ut på att låta webbserven servera en dynamiskt ihopsatt sida på 64Kb. I detta test klarade PHP-lösningen (Apache 2.0) att leverera ca 150 sidor per sekund respektive 250 sidor per sekund för PHP på Apache 1.3. JAVA-baserade Resin klarade av 550 förfrågningar per sekund. Perl på Apache 1.3 uppnådde ca 400 förfrågningar per sekund.

Microsoft uppger att ASP.NET är upp till fyra gånger snabbare än en lösning baserad på JAVA [4], notera dock att detta är tillverkarens egna uppgifter.

2.3 Client scripting

Skriptning på klientsidan växte fram då behovet att kunna skapa dynamiska sidor växte och det ibland uppstod problem med kapaciteten på serversidan. Dåtidens CGI-skript tog helt enkelt för mycket CPU-tid och det genererades en hel del nätverkstrafik när varje klient skulle hämta ny data hela tiden.

För att råda bot på detta problem utvecklades olika metoder för att låta klienten (webbläsaren) utföra vissa operationer på den sida den fått levererad. Själva skriptkoden finns inbäddad i html-sidan eller hämtas separat via en länk i html-sidan.

Det första klientskriptspråket var *LiveScript*. LiveScript uppfanns av Netscape Communications (egentligen Brendan Eich och hette från början Mocha). LiveScript introducerades under namnet JavaScript i samband med lanseringen av Netscape Communications webbläsare Netscape 2. JavaScript blev snabbt väldigt populärt och en utökad version lanserades med Netscape 3. I samband med att Netscape 3 lanserades svarade Microsoft med att lansera en egen version av skriptning på klientsidan: JScript. JScript var delvis kompatibelt med JavaScript.

För att minska förvirringen och för att skapa en stabil kärna till skriptspråk på klientsidan samarbetade Sun Microsystems och Netscape med ECMA[4] och skapade en standard.

Idag implementerar alla moderna webbläsare ECMAs standard på ett eller annat sätt. Microsofts Internet Explorer implementerar standarden genom sitt JScript och de Mozillabaserade webbläsarna (Mozilla, Firefox) implementerar JavaScript. Både JScript och JavaScript innehåller utökningar till standarden på olika sätt (ej kompatibla med varandra).

Utöver JScipt innehåller Internet Explorer en implementation av VBScript. Det går alltså även att använda VBScript inbäddat i en html-sida men det fungerar då endast med MSIE[5].

2.3.1 JavaScript

LiveScript bytte namn till JavaScript ungefär vid lanseringen av Netscape 2. Själva namnet JavaScript har lett till rätt mycket förvirring då JavaScript egentligen inte har något att göra med programspråket Java. JavaScript och programspråket Java har vissa likheter i syntax men har stora skillnader i semantiken. JavaScript är till syntaxen influerat av *C* och *Perl* [8]

2.3.2 VBScript

VBScript är en delmängd av Visual Basic och kan användas till webb, styrning av Windowsapplikationer och för att ersätta så kallade ”batchfiler” på Windowsplattformen. På webben kan det speciellt användas i webbsidor men då det krävs att webbläsaren Internet Explorer används på klientsidan föredrar de flesta utvecklare att använda någon form av JavaScript istället.

2.4 Ajax

AJAX är en term myntad av Jesse James Garrett[2] för att beskriva det nyaste och för tillfället hetaste trenden för att utveckla webbapplikationer. Termen AJAX är bildad av Asynchronous JavaScript och XML, några av de tekniker en webbapplikation uppbyggd kring AJAX använder sig av. Följande tekniker och standarder brukar räknas in när AJAX beskrivs:

- ⋆ Standardbaserad presentation med XHTML och CSS
- ⋆ Dynamiskt utseende och interaktion med Document Object Model

[4]European Computer Manufacturers Association, http://www.ecma-international.org
[5]Microsoft Internet Explorer

- ⋆ Datautväxling och manipulation med XML och XSLT
- ⋆ Asynkron datahämtning med dataobjeket XMLHttpRequest
- ⋆ JavaScript

Att kombinera dessa tekniker ger möjligheter att skapa funktionella webbapplikationer ett helt nytt sätt. Användaren av en sådan webbapplikation upplever det mer likt att använda en riktig applikation än en webbapplikation - känslan av snabb respons och dynamik går ej att jämföra den gamla tidens webbtjänster.

2.4.1 Vad differentierar AJAX från en vanlig webbapplikation?

En standardwebbtjänst idag låter en användare fylla i en webbsida och skicka den till en webbserver. Webbservern behandlar den data den mottagit och skapar en ny sida utifrån de val användaren gjort. Användaren mottar en ny sida och kan göra nya val eller bara läsa/se den hämtade informationen. Denna metod kräver att användaren väntar på en nya sida från webbservern efter att data har sänts iväg och under tiden blir det en del väntan. Den returnerade sidan innehåller ofta mycket redundant information.

AJAX använder sig istället av ett gränssnitt konstruerat till stora delar med hjälp av JavaScript och DM[6]. När sedan användaren interagerar med detta gränssnitt vilket går att bygga mer levande än ett rent html/xhtml gränssnitt kommunicerar JavaScriptmotorn i gränssnittet själv med webbservern via en metod kallad XMLHttpRequest. På detta sätt kringgås standardmodellen för en webbapplikation (klient/server, fråga/svara), se även bild 2.2, 2.3 och 2.4. Användaren av en webbapplikation uppbyggd med denna metod märker ofta inte när ny data hämtas och själva webbsidan behöver sällan laddas om. På detta sätt sparas också en hel del bandbredd då bara den efterfrågade informationen hämtas, och inte en helt ny webbsida vilket är fallet i en klassisk webbtjänst.

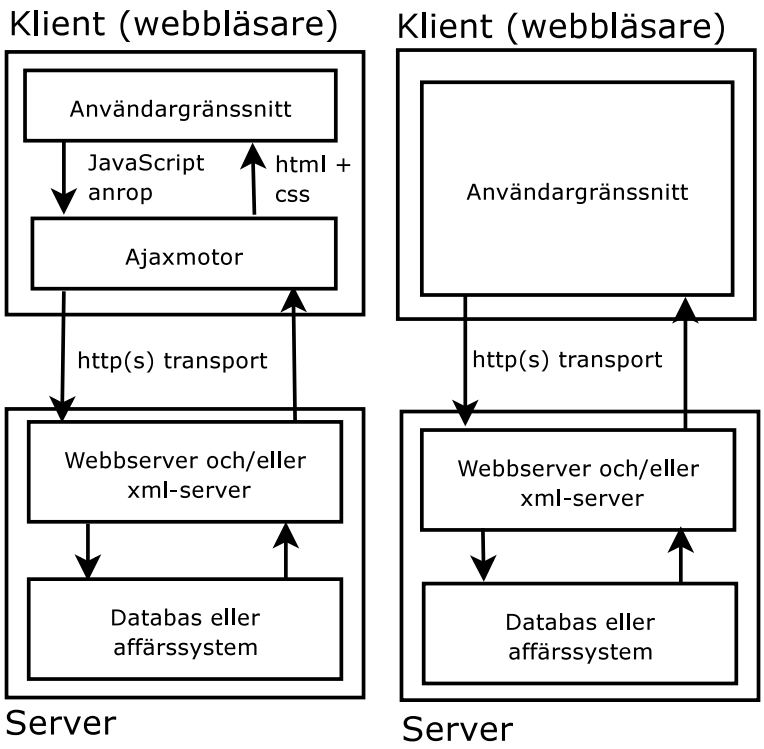
2.4.2 AJAX i praktiken

Google (företaget med sökmotorn med samma namn, *http://www.google.com*) har på kort tid lanserat flera tjänster byggda kring AJAX-tekniken. *Google Suggest* är en söktjänst där Google lämnar förslag på möjliga sökord eftersom tecken matas in i sökrutan. Det hela fungerar väldigt bra och känslan av att använda en webbapplikation infinner sig inte.

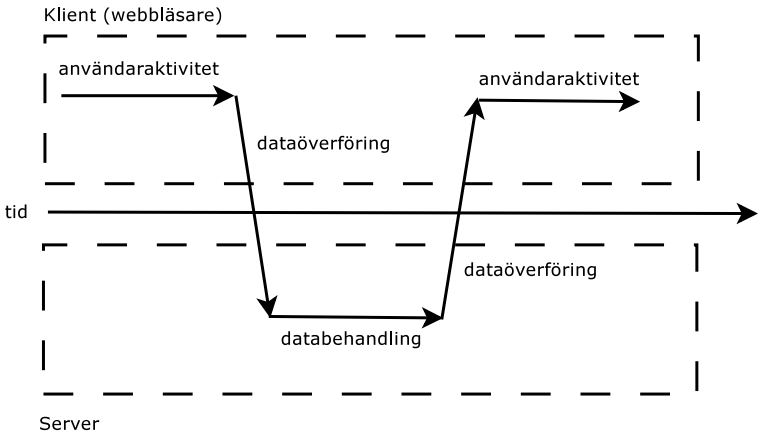
2.4.3 Utveckla tjänster enligt AJAX

Att utveckla moderna webbapplikationer enligt AJAX kräver mycket av utvecklarna; både kunskap om de enskilda teknologierna och kunskap om hur de ska kombineras för att uppnå den kvalité och funktion tjänster baserade på AJAX karaktäriseras av.

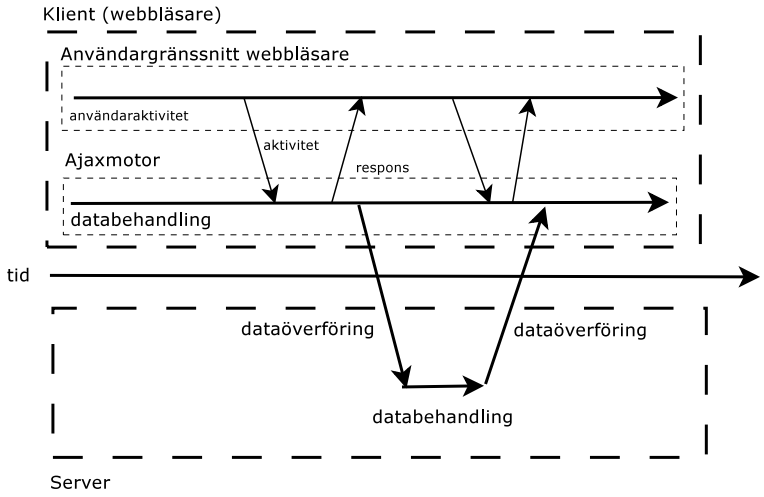
[6]Document Object Model, http://www.w3.org/DOM/



Figur 2.2: En webbtjänst baserad på ajax (till vänster) jämfört med en webbtjänst uppbyggd på ett vanligt sätt.



Figur 2.3: Den klassiska modellen över hur en webbtjänst fungerar



Figur 2.4: AJAX-modellen på hur en webbtjänst fungerar. AJAX-motorn interagerar med användaren och hämtar själv data från webbservern.

Microsoft Develop tillsammans med ASP.NET har ett visst stöd för att bygga webbapplikationer med JavaScript på ett enkelt sätt. Det vore dock fel att kalla detta en fullständig utvecklingsmiljö för webbtjänster utvecklade kring AJAX-idéerna. Det saknas alltså idag en fullständig integrerad utvecklingsmiljö där det går att skapa tjänster enligt AJAX-modellen.

2.4.4 Säkerhet

Då AJAX-modellen är helt ny och det finns ett begränsat antal implementationer i skarp drift är det svårt att säga hurvida säkerheten kan bli ett problem eller inte. Det är alltså värt att lägga extra vikt på säkerhet vid utveckling av webbapplikationer enligt AJAX-modellen.

Kapitel 3

Portaler och portlets

3.1 Vad är en portal?

En portal är en webbplats där flera tjänster samlas och en användare på ett enkelt sätt kan nyttja olika tjänster utan att behöva besöka flera olika webbplatser. En önskvärd egenskap hos en Portal är att en användare endast behöver logga in en gång för att sedan kunna nyttja allt det som erbjuds utan att behöva logga in på nytt för att tillgodogöra sig nästa tjänst.

Det går särskilt att profilera några olika huvudtyper av portaler:

- ⋆ Länkportal
- ⋆ Universitetsportal
- ⋆ Startsida

En länkportal är kort och gott en webplats med väldigt många länkar inom ett område eller kategoriserat i flera olika ämnesområden. En länkportal erbjuder sällan extra tjänster i form av epost eller annat. På senare tid har dykt upp allt fler länkportaler med väldigt tunt innehåll, dessa länkportaler återfinns i stor utsträckning under domännamn med felstavade varianter på stora kända webbplatser. En sådan länkportal existerar endast för att ta trafik då användare skriver fel adress och hamnar fel. Exempel:

- ⋆ http://www.catweb.nu/ (svensk länkportal med länkar efter ämnesområde)
- ⋆ http://www.mns.com/ (enkel länkportal för att fånga trafik när användare skriver www.msn.com fel)

De flesta av Sveriges stora internetleverantörer har en egen portal där de hoppas att deras kunder ska börja sin resa ut på nätet. Exempel:

- http://www.tele2internet.se.everyday.com/
- http://www.startsidan.telia.se/

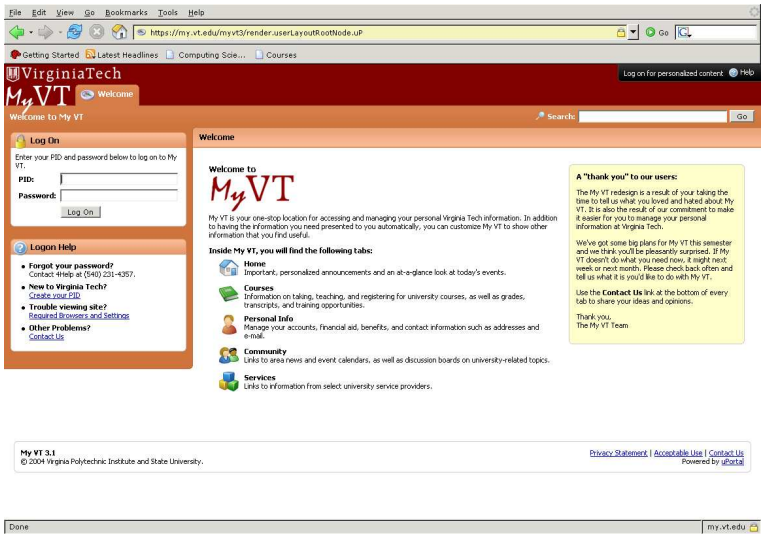
Internetleverantörerna har dessa portaler för att exponera sina tjänster och stärka sitt varumärke hos sina kunder.

En universitetsportal är helt enkelt en portal där skaparen är ett universitet med syftet att underlätta för studenterna/anställa vid universitet. Ofta kan dessa portaler innehålla tjänster för att underlätta användarens vardag. Exempel på tjänster som kan finnas i en universitetsportal:

- Epost
- Kalender
- Betygsinformation (studenter)
- Lönespecifikation (anställda)
- Väderinformation

Exempel:

- https://my.vt.edu (Virginia Polytechnic Institute and State University)



Figur 3.1: Virginia Polytechnic Institute and State University portal MyVT byggd på uPortal

3.2 uPortal

uPortal är en fritt portalramverk under utveckling av högskolor och universitet världen över (mestadels i USA). uPortal bygger bland annat på standarderna JAVA, XML, JavaServer Pages och JAVA 2 Enterprise Edition.

uPortal utvecklas av JA-SIG[1]. Fokus ligger på universitetsvärlden och funktionerna i uPortal speglar detta. uPortal är inte en färdig lösning utan är ett ramverk och det krävs utveckling av tjänster att fylla ramverket med. Dessa tjänster implementeras antingen i form av en *iChannel* eller en *Portlet*. IChannel kommer att fasas ut till förmån för Portlets.

3.3 Umeå universitets studentportal

Det sker för närvarande ett förankringsarbete inom Umeå universitet om att så snart som möjligt införa en studentportal. Vad som skall återfinnas i denna studentportal är ännu inte fastställt, men ett flertal tjänster har identifierats som synnerligen viktiga för att studentportalen skall uppskattas av studenterna.

De studierelaterade tjänsterna som återfinns i Studenttjänster på webb kommer vid ett införande att implementeras i studentportalen, eftersom målsättningen med dessa hela tiden varit att de skall kunna införas med marginella anpassningar i en studentportal och därför utvecklades från första början i uPortal.

3.4 Umeå universitets val av uPortal

Under 2001 och 2002 genomfördes en förstudie[2] på uppdrag av IT-chefen vid Umeå universitet. Denna förstudie syftade till att testa och utvärdera olika portallösningar. I förstudien testades olika portallösningar och dess för- och nackdelar vägdes mot varandra.

I förstudiens slutrapport rekommenderas universitetet att använda sig av portalramverket uPortal för en eventuell framtida portallösning. uPortal förordades då bruket av öppna standarder och den flexibilitet uPortals ramverk har ger långsiktiga fördelar.

[1] Java in Administration Special Interest Group, http://www.ja-sig.org

[2] Förstudien och de olika rapporterna förknippade med den går att finna på http://www.umu.se/it/umu_internt/portaler/index.html

3.5 JSR-168 - Java™ Portlet Specification 1.0

En portlet är en javabaserad teknologi för att utveckla komponenter i en portal. En komponent i en portal kan sägas vara en tjänst som användarna av portalen kan använda sig av. Exempel på en sådan tjänst är en kalender eller en adressbok. En tjänst i en portal kan också ses som en applikation som erbjuds användaren. Användaren kan sedan personifiera sin portalsida utifrån de applikationer som finns tillgängliga.

Java™ Portlet Specification 1.0 är en standard för hur en portlet skall vara uppbyggd och hur den skall fungera. Standarden specificerar också hur en portletcontainer ska implementeras och fungera. Här är några viktiga bitar ur standarden:

- En portlet exekveras i en portletcontainer i webbservern och portleten serverar dynamiskt innehåll till användaren av portalen.
- En portlet producerar bara bitar eller fragment av den hela webbsida som utgör portalen.
- En portlet är på många sätt väldigt lik en servlet, det är samma koncept med en webbserver och en container vari den exekveras. Båda laddas dynamiskt i webbservern. En servlet generar dock nästan alltid en hel sida själv, inte bara fragment av en webbsida som en portlet gör. En portlet är inte bunden till en speciell url som en servlet är, utan existerar som en del av en portal och kan till och med finnas flera gånger på samma sida i en portal. Genom portalen har en portlet möjlighet att spara inställningar och tillgång till information om användaren den för tillfället hanterar. En portlet har också speciella funktioner för att generera url:er till sig själv i portalen.
- Som en konsekvens av konceptet med portlets kan en portlet inte fritt välja innehållstypen (eng ”content type”) den genererar eller att sätta en egen teckenkodning, utan den måste fråga portalen om vilka de tillgängliga innehållstyperna är och sedan om den vill sätta en av dessa (detta behöver dock inte specifieras då det normalt sätts den innehållstyp övriga portalen använder, portleten är en del av hela portalen, och portalen sätter dessa).
- Portletstandarden specificerar hur portletens olika webbsideelement skall märkas med stilmallar enligt standarden CSS[3].

[3]Cascading Style Sheets

Kapitel 4

Köp- och säljportlet

4.1 Planering

Under förarbetet till att utveckla den Köp- och säljportlet som ingick i specifikationen av examensarbetet studerades Blocket[1] noggrant. Kravspecifikationen hänvisar till Blocket och Blocket anses av många vara en branschstandard.

I den kravspecifikation jag erhållit från UMDAC var den önskade portletens funktionalitet rangordnad i ett antal punkter utöver de allmänna kraven. Målet var att implementera alla dessa om tiden räckte till.

Vidare ingick det i examensarbetet att sätta upp en databas med hjälp av databashanteraren *PostgreSQL*. Denna databas används för att lagra annonserna.

I förarbetet ingick också att testa de tekniker portleten skulle använda sig av. Detta inkluderar bland annat *PostgreSQL JDBC Driver*[2], *log4j*[3] och *JAVA Graphics*[4]

4.2 Verktyg

Allt utvecklingsarbete har skett i Eclipse 3.1 med Exo Portlet Plugin. uPortal 2.4.2 har körts på Tomcat med HSQL för uPortals databas. PostgreSQL 8.0 har använts för annonsdatabasen.

[1] Köp- och säljmarknad på nätet, http://www.blocket.se
[2] En JDBC-kompatibel PostgreSQL databasdrivrutin till JAVA, http://jdbc.postgresql.org/
[3] Ett system för att hantera loggning, http://logging.apache.org/log4j/docs/
[4] Ett bildmanipuleringsbibliotek, http://java.sun.com/j2se/1.4.2/docs/api/java/awt/Graphics.html

4.3 Implementation

Att implementera en portlet enligt JSR-168 har varit intressant och lärorikt. För att komma igång snabbt införskaffades boken *Building Portals with the Java Portlet API*[5]. Boken har varit nyttig under arbetet och är en av få källor till information kring utvecklingen av portlets.

Köp- och säljwebbapplikationen kan sägas bestå av tre huvuddelar:

- ⋆ Logik (Portlet)
- ⋆ Webbgränssnitt (JSP)
- ⋆ Databas (PostgreSQL)

4.3.1 Logik

Köp- och Säljportletens huvudklass *BuyAndSellPortlet* ärver från den abstrakta klassen *GenericPortlet*. Detta görs för att få normal portletfunktionalitet. Detta innebär bland annat metoder för att hantera minimering och maximering av portleten när detta begärs av användaren (se bild 4.1)

Figur 4.1: Maximeringsknapp (längst till vänster) och minimeringsknapp (näst längst till vänster). Dessa knappar sätter portalen (i detta fall uPortal) i ramen kring en portlet.

De viktigaste metoderna i portleten är doView(RenderRequest request, RenderResponse response) och processAction(ActionRequest request, ActionResponse response). Den första av dessa metoder anropas varje gång en av portletens sidor skall visas[6]. Den andra, processAction, anropas bland annat när en användare skapar en ny annons eller gör en sökning[7].

I doView undersöks den medföljande request för att avgöra vilken sida som skall visas. Möjliga sidor är bland annat:

[5] Jeff Linwood, Dave Minter Apress 2004
[6] När klienten (webbläsaren) gör ett GET-anrop enligt HTTP 1.1 standarden
[7] När klienten (webbläsaren) gör ett POST-anrop enligt HTTP 1.1 standarden

- ⋆ Startsidan (se bild 4.2)
- ⋆ Visa annons (se bild 4.3)
- ⋆ Ny annons (se bild 4.4)
- ⋆ Mina annonser

Om ingen information hittas i request visas startsidan. Beroende på vilken sida som ska visas hämtas olika information från databasen. Denna information bifogas sedan request och anropet skickas till rätt JSP-sida.

I processAction sker en liknande process, request undersöks och beroende på dess innehåll utförs korrekt åtgärd. Är det en enkel åtgärd görs detta direkt i processAction, är det en mer avancerad omfattande åtgärd (tex skapandet av en ny annons) hanteras det av en egen klass. Denna klass instansieras då och tar hand om förfrågan. De aktuella fallen är bland annat:

- ⋆ Skapa en ny annons
- ⋆ Svara på en annons
- ⋆ Utföra en sökning
- ⋆ Ta bort en annons

4.3.2 Skapa en ny annons

När en ny annons skapas kontrolleras att tillräcklig data har angivits (inte tom rubrik osv). Efter detta lagras informationen i databasen. Annonskategori och annonstyp sparas i form av en nyckel till dessa tabeller, detta gör det möjligt att byta namn på en kategori eller annonstyp utan att de befintliga annonserna blir påverkas.

Har användaren laddat upp en bild till annonsen undersöks denna. Visar det sig att bilden överskrider den tillåtna storleken skalas bilden om till en mindre storlek. Kravspecifikationen säger att portleten får uppta maximalt 50 procent av skärmbredden vid en upplösning med 1024 pixlar på bredden. Efter praktiska försök bestämdes att en bild får vara max 480 pixlar bred för att klara detta krav. Är bilden bredare än detta skalas den om till denna bredd. Bildens höjd kontrolleras också. Inte heller denna får överskrida 480 pixlar.

4.3.3 Utföra en sökning

En sökning görs genom en fritextsökning i databasen med hjälp av operatorn ILIKE i PostgreSQL. ILIKE är en PostgreSQL specifik operator för att söka utan att ta hänsyn till stora och små bokstäver. Att operatorn är specifik för PostgreSQL är en nackdel då det blir svårare att byta databashanterare.

Att byta ut användningen av ILIKE mot en kombination av Java och operatorn LIKE[8] hoppas kunna slutföras inom ramen för examensarbetet.

Sökningen med hjälp av ILIKE sker i rubrik- och textfälten i varje annons.

4.3.4 Webbgränssnitt

Portletens webbgränssnitt är uppbyggt kring ett 10-tal JSP-sidor. Dessa sidor är mallar och dess slutgiltiga utseende skapas dynamiskt vid varje sidanrop. JSP-sidan tar den data portleten hämtade och fyller i sig själv med datan. JSP-sidorna kan innehålla Javakod men helst används speciella taggar från något av de fördefinierade taggbiblioteken. Köp- och Säljportleten använder bland annat portlettaggbiblioteket. I detta bibliotek finns metoder för att skapa två olika typer av länkar: *actionURL* och *renderURL*. Dessa länkar används för att länka till andra sidor i portleten och för att använda som *action*[9] till html-formulär i portleten. Det är nödvändigt med metoder för att skapa länkar då det inte i förväg går att veta var portleten kommer att driftsättas. När en länk skapas går det att sätta ytterligare attribut på den. Detta nyttjas för att skapa länkar till olika sidor i portleten - det är dessa attribut doView och processAction letar efter.

4.3.5 Databas

Databasen där annonserna lagras är inte speciellt avancerad. Den består av fyra tabeller. En för själva annonserna och en för bilder, därutöver två tabeller för att hålla reda på annonstyper och kategorier för annonser. För tillfället används ingen inbyggd logik i databasen i form av *triggers*[10] eller liknande konstruktioner. Detta har inte ansetts nödvändigt då databasen är väldigt enkel.

Bilderna lagras binärt i databasen och serveras via en servlet till portleten vid anrop. Denna konstruktion med en servlet för att servera bilder från databasen är nödvändig då en portlet inte kan leverera en bild själv. Givetvis går det att länka till en bild, men det går inte att generera en bild utifrån en databas, detta beror på att en portlet inte fritt kan välja vilken innehållstyp den ska leverera (se 3.5).

4.4 Övrigt

Paketet *log4j* används av portleten för loggning. Inställningar för denna loggning går att göra i samma inställningsfil där övriga portletinställningar görs. Loggning sker bland annat när en annons skapas och när epost skickas.

[8] En standardiserad SQL-operator för fritextsökning, tar dock hänsyn till små och stora bokstäver

[9] I ett html-formulär med metoden POST anges en sida för att ta hand om den data formuläret innehåller

[10] En form av inbyggd logik i databasen, skulle i detta fall kunna vara en funktion för att notera en viss användare när en speciell annons sätts in, en form av bevakning. Se även sista punkten i stycke 4.7

4.5 Problem

Under utvecklingen har en mängd mindre problem funnits, de flesta av dessa har gått att lösa genom att söka information på nätet eller i den tidigare nämnda portletboken.

Uppladdningen av bilder vid skapandet av en ny annons är något som ställt till med stora problem, dessa problem visade sig sedan härröra till en bugg i uPortal 2.4.0, efter en uppgradering till uPortal 2.4.2 försvann dessa problem.

Den javakod JSP-sidorna innehåller kompileras dynamiskt av Tomcat vid anrop. Detta förfarande medför att felsökning blir svårt. Att felsöka JSP-sidorna är det som varit svårast under hela utvecklingsarbetet.

4.6 Utseende och funktionalitet

På följande sidor följer några bilder för att illustrera utseendet och funktionaliteten hos den utvecklade portleten.

När en användare först ser portleten i portalen visas de senast inlagda annonserna och en sökruta, se bild 4.2. I detta läge är det möjligt att klicka på en av de visade annonserna eller att ange ett sökord i sökrutan. Bredvid sökrutan går det även att välja en speciell annonstyp och/eller annonskategori.

Vid en sökning genomsöks databasen efter annonser där sökordet förekommer i rubrik eller annonstext.

När en annons väljs visas en sida med annnonsen, se bild 4.3. På denna sida visas även annan information, till exempel telefonnummer till annonsören och vem som har annonserat. Det ges även möjlighet att direkt svara på annonsen genom att klicka på en länk.

Längst upp i portleten finns det hela tiden tre länkar. Dessa används för att navigera till portletens startsida och för att visa användarens egna annonser. Den sista länken används för att skapa en ny annons. Sidan för en ny annons visas på bild 4.4.

4.7 Uppfyllelse av kravspecifikation

I kravspecifikationen finns en rad punkter att uppfylla:

- ⋆ *Det tilltänkta systemet skall som minimum erbjuda en färdig köp- och säljfunktion*

  Detta krav är uppfyllt. Portleten fungerar tillfredställande och erbjuder ett fungerande köp- och säljsystem.

Figur 4.2: Detta är portletens första sida. De fem senast inlagda annonserna listas och sökrutan visas

⋆ *Sortering skall kunna ske på samtliga kolumner som presenteras*

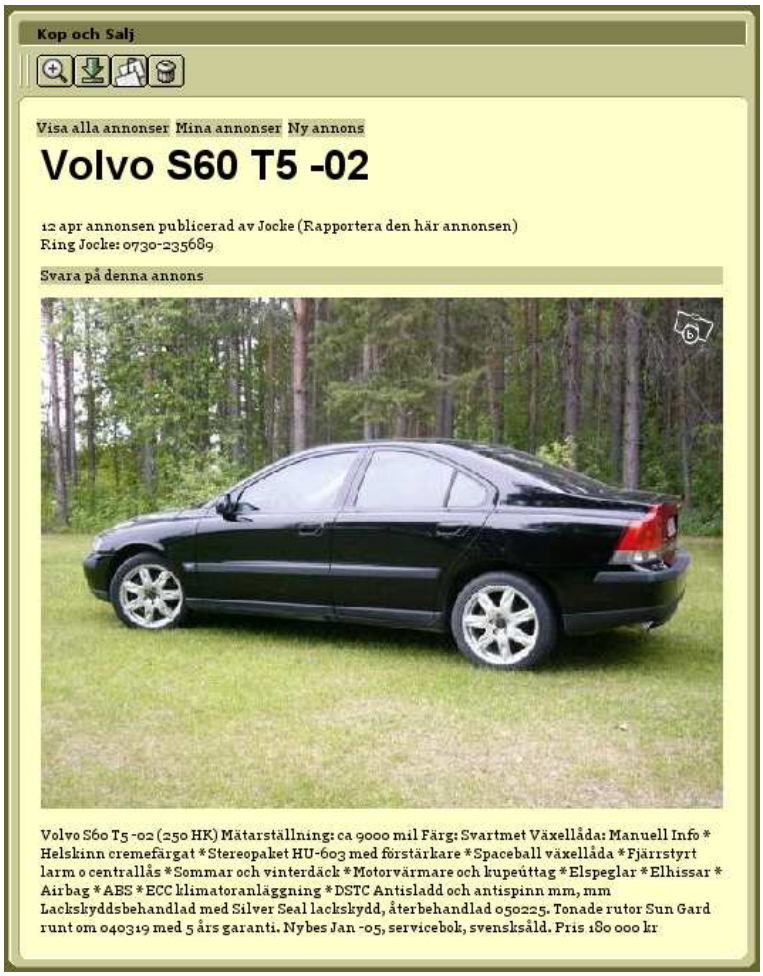
I portleten går det att sortera på de kolumner som presenteras. Dessa är datum, annonsrubrik och annonskategori.

⋆ *Då användaren väljer* Mina annonser *skall en lista presenteras på vilka annonser en användare är ägare av. Kopplat till detta skall möjligheten finnas att uppdatera eller ta bort en annons.*
En annons skall innehålla en rubrik och och ett fritextfält som anges av annonsören. Annonsören skall ange ett namn eller alias. Frivilliga uppgifter är telefonnummer och e-postadress. I anknytning till inmatningsfältet för e-post skall en markering visa att e-postadressen döljs om inte avmarkering görs av annonsören. En inställningsfil skall göra det möjligt att ange om e-postadressen skall hämtas via LDAP[11] *och presenteras som default-värde. Inställlningsfilen skall innehålla uppgifter som behövs för för LDAP-sökning, samt en flagga för e-postinställningen för att uppnå följande funktioner:*

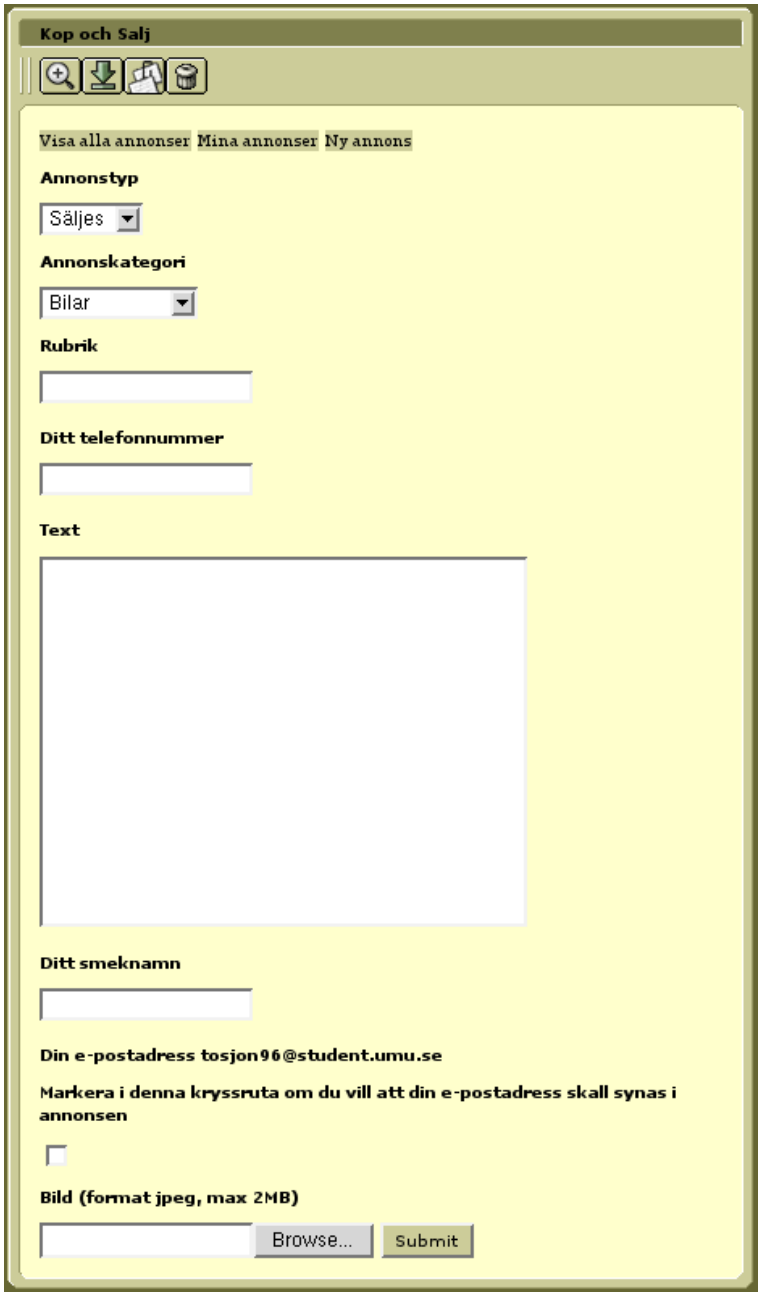
- *Ingen LDAP-sökning*
- *Sökning, men ändringsbar e-postadress i applikationen*
- *Sökning men ej ändringsbar e-postadress i applikationen*

Portleten använder sig av en inställningsfil där diverse olika inställningar kan göras, bland annat den ovan beskrivna funktionen för att hämta en epostadress från en LDAP-källa. I övrigt innehåller en annons de fält kravspecifikationen efterfrågar.

[11] Lightweight Directory Access Protocol, ett protokoll för att överföra bl a namn och adressuppgifter



Figur 4.3: Här visas en annons



Figur 4.4: Detta är sidan för att skapa en ny annons

- ⋆ *Vid svar på en annons skall ett webbformulär göra det möjligt att skicka ett e-postmeddelande till annonsören. Avsändarens e-postadress skall vara konfigurerbar på samma sätt som då en annons skapas (via inställningsfil)*

- ⋆ *Kategorier skall finnas i ett centralt register. Namnet på kategorier skall vara ändringsbart i efterhand utan att annonser påverkas. Möjliga kategorier är t ex Kurslitteratur, Bostad, Hemelektronik.*

  Namnet på de möjliga kategorierna lagras separat i en tabell i databasen. Det medgör att namnet på en kategori förändras utan att de befintliga annonsera måste förändras på något sätt.

- ⋆ *Annonstyper skall finnas i ett centralt register. Möjliga annonstyper är t ex Köpes, Säljes, Bytes. Namnet på annonstyper skall vara ändringsbart i efterhand utan att annonser påverkas.*

  Även annonstyperna lagras i en egen tabell för att möjliggöra ändring av dessa utan att befintliga annonser påverkas.

- ⋆ *Det skall vara möjligt att söka information både i rubrik och annonstext. Omfattningen av sökfunktionen skall vara begränsad till annonstyp. Träffresultatet skall presenteras i normalvyn för aktuell annonstyp, dvs egentligen en filterfunktion.*

  Här följer implementationen inte riktigt kravspecifikationen. Sökfunktionen tillåter sökning där både annonstyp och kategori specificeras. Efter samtal med beställaren anses detta inte vara någon brist utan snarare en förbättring av sökfunktionen.

- ⋆ *Applikationen skall kunna visa ett driftmeddelande om underhåll sker på applikationen. I detta fall skall applikationen inte vara användar.*

  Detta går att göra via inställningsfilen. Hurvida detta är en accepterad lösning har diskuterats med beställaren Niklas Lundgren och denne har godkänt denna metod.

- ⋆ *Möjlighet att ladda upp filer till applikationen. Modulen skall kunna begränsa filstorlek och vilka bildformat som är tillåtna. Filerna ska lagras på en central yta.*

  Portleten tillåter uppladdning av en bild till en annons. Bilderna lagras i databas (detta motsvarar en ”central yta”). Denna del är ännu ej helt modulariserad på det sätt kravspecifikationen kräver, men detta kommer förhoppningsvis att hinna åtgärdas inom tidsramen för examensarbetet.

  Förutom ovan nämnda punkter finns det några generella riktlinjer och ytterligare några krav uppradade i kravspecifikationen. Av dessa är det endast fyra stycken som ej implementerats (språkstöd, auktion, nyttiga länkar och prenumeration).

Kapitel 5

Begränsningar och framtida arbete

5.1 Begränsningar

Köp- och säljportleten har ännu inte testkörts i den vidd det skulle behövas innan den går att sätta i skarp drift. Förmodligen finns det fortfarande kvar ett antal mindre fel. Den kanske största begränsningen för tillfället är dess språkstöd, idag stöder den endast svenska.

Utseendet hos portletens webbgränssnitt har endast tagits fram för teständamål. Före en framtida driftsättning bör en enkel utvärdering av gränssnittet ske. Utformning av webbgränssnittet har ej innefattats i examesarbetet.

5.2 Framtida arbete

Det finns en del arbete att utföra vid ett eventuellt fortsatt utveckling av denna portlet. Det kanske viktigaste arbetet är att skapa stöd för flera olika språk på ett enkelt sätt.

Att implementera de sista fyra modulerna i kravspecifikationen är också en återstående del. Hur viktigt detta arbete måste beställaren avgöra.

Att utförligt testa och finputsa portleten är det sista arbetet som måste göras. Förhoppningsvis kan sedan den implementerade portleten användas på ett eller annat sätt i en kommande studentportal vid Umeå universitet.

Kapitel 6

Tack

Dessa personer har på ett eller annat sätt bidragit till att mitt examensarbete och jag vill därför tacka dessa:

- ⋆ Niklas Lundgren, UMDAC
- ⋆ Tommy Larsson, UMDAC
- ⋆ Jan-Erik Moström, Datavetenskap
- ⋆ Kurskamrater och vänner som stöttat under våren

Referenser

[1] cgi@ncsa.uiuc.edu. The common gateway interface. Hemsida, 29 mars 2005. http://hoohoo.ncsa.uiuc.edu/cgi/.

[2] Jesse James Garret. Ajax: A new approach to web applications. Hemsida, 18 februari 2005. URL: http://www.adaptivepath.com/publications/essays/archives/000385.php.

[3] Martin Arlitt och Carey Williamson Lance Titchkosky. A performance comparison of dynamic web technologies. *ACM SIGMETRICS Performance Evaluation Review*, 31(3), 2003.

[4] Microsoft. What is microsoft .net? Hemsida, 2 maj 2005. URL: http://msdn.microsoft.com/netframework/technologyinfo/overview/default.aspx.

[5] Sun Microsystems. The java servlet api white paper. Hemsida, 23 april 2005. URL: http://java.sun.com/products/servlet/whitepaper.html.

[6] Sun Microsystems. Javaserver pages[tm] technology - white paper. Hemsida, 23 april 2005. URL: http://java.sun.com/products/jsp/whitepaper.html.

[7] Alexandros Labrinidis och Nick Roussopoulos. Generating dynamic content at database-backed web servers: cgi-bin vs mod_perl. *ACM SIGMOD Record*, 29(1):26–31, Mars 2000.

[8] Kailash Chandra och Sapana Suhani Chandra och Shyamal Suhana Chandra. A comparison of vbscript, javascipt and jscript. *Journal of Computing Sciences in Colleges*, 19(1):323–335, 2003.

[9] Inc Open Market. Fastcgi: A high-performance web server interface. Webpage, 2 april 1996. URL: http://www.fastcgi.com/devkit/doc/fastcgi-whitepaper/fastcgi.htm.

Bilaga A

Ordlista

webbapplikation En tjänst eller datorprogram tillgängligt via en webbläsare

webbtjänst En samling protokoll och standarder för att utbyta data mellan olika datorprogram

webbplats En samling av webbsidor

webbsida En enskild sida på en webbplats

portal En plats på internet där många tjänster och länkar är samlade på ett och samma ställe

J2EE JAVA Platform 2 Enterprise Edition, en version av JAVA

JAVA Ett programmeringsspråk utvecklat av Sun Microsystems

Bilaga B

Kravspecifikation

UMEÅ UNIVERSITET
UMDAC

Kravspecifikation
Examensarbete – Köp & sälj

Innehållsförteckning

1 INLEDNING 2

1.1 HANDLEDNING 2
1.2 SYFTET MED EXAMENSARBETET 2
1.3 ÖVERGRIPANDE INFORMATION 2

2 SYSTEMBESKRIVNING 3

2.1 ALLMÄNNA FUNKTIONER 3
2.1.1 Sortering 3
2.1.2 Annonsering 3
2.1.3 Svar på annons 3
2.1.4 Kategorier 3
2.1.5 Annonstyp 4
2.1.6 Användarmoderering 4
2.1.7 Sökfunktion 4
2.1.8 Driftmeddelande 4
2.2 MODULER 4
2.2.1 E-post 4
2.2.2 Administratörsmoderering 4
2.2.3 Filuppladdning 5
2.2.4 Bildhantering 5
2.2.5 Språkstöd 5
2.2.6 Auktion 5
2.2.7 Nyttiga länkar 5
2.2.8 Prenumeration 5

3 KRAV PÅ APPLIKATIONEN 6

3.1 GENERELLA KRAV 6
3.2 ALLMÄNT 6
3.3 ANVÄNDARMILJÖ 6
3.4 SÄKERHET 6
3.5 SPECIFIKA KRAV 6
3.6 DATABASHANTERARE 6
3.7 DOKUMENTATION 6

1 Inledning

1.1 Handledning

Niklas Lundgren fungerar som beställare av examensarbetet.
Tommy Larsson fungerar som teknikstöd.

1.2 Syftet med examensarbetet

Det övergripande syftet med examensarbetet är att kunna leverera en köp- och säljfunktion till en blivande studentportal vid Umeå universitet.

Delsyften med examensarbetet är att

- Identifiera möjligheter och problem med portlets baserade på JSR-168 i uPortal.
- Utveckla ett antal moduler som kan användas vid annat utvecklingsarbete.

1.3 Övergripande information

Umeå universitet har i dagsläget två officiella installationer av portalramverket uPortal. En för informationsspridning till universitetsförvaltningen och en som exponerar ett antal studierelaterade tjänster för studenter.

Vid Umeå universitet bedrivs för närvarande en förstudie om hur ett införande av en studentportal kan och bör ske. En av de tjänster som efterfrågas är en köp och säljfunktion. Detta är också den mest populära tjänsten i amerikanska studentportaler.

De installationer av uPortal som finns vid universitetet idag är baserade på en gammal version av portalramverket. uPortal utvecklas kontinuerligt och den teknik som används i de nuvarande portalerna kommer att fasas ut successivt till förmån för portlets.

Portlets enligt JSR-168 är en standard med ett drygt år på nacken, vilket medför att det inte finns så många referensimplentationer. Vi vill därför erhålla erfarenheter av utvecklingsarbete enligt denna standard, tillsammans med en implementation i uPortal.

2 Systembeskrivning

Det tilltänkta systemet skall som minimum erbjuda en färdig köp- och säljfunktion. Kopplat till detta finns ett antal kompletterande moduler som skall implementeras enligt angiven prioritetsordning.

Anledningen till att en uppdelning i moduler görs är två. Dels för att det skall finnas tydliga gränser för vad en modul är och vad den gör, dels för att erhålla en produkt som fungerar även om inte all önskvärd funktionalitet finns med. Systemet skall dock designas på ett sådant sätt att samtliga moduler skall vara möjliga att implementera i en förlängning.

Blocket bör fungera som inspirationskälla under arbetet, då Blocket i det närmaste har blivit en ”branschstandard” som de flesta kan relatera till.

2.1 Allmänna funktioner

Följande grundfunktioner skall ingå i systemet. Generellt gäller att applikationen skall hämta nödvändiga inställningar från en inställningsfil. Inställningar skall vara ändringsbara utan att portalen måste startas om.

2.1.1 Sortering

Sortering skall kunna ske på samtliga kolumner som presenteras.

2.1.2 Annonsering

Då användaren väljer ”Mina annonser” skall en lista presenteras på vilka annonser som användaren är ägare av. Kopplat till detta skall möjligheten finnas att uppdatera eller ta bort en annons.

En annons skall innehålla en rubrik och ett fritextfält som anges av annonsören. Annonsören skall ange ett namn eller alias. Frivilliga uppgifter är telefonnummer och e-postadress. I anknytning till inmatningsfältet för e-post skall en markering visa att e-postadressen döljs om inte en avmarkering görs av annonsören. En inställningsfil skall göra det möjligt att ange om e-postadressen skall hämtas via LDAP och presenteras som default-värde. Inställningsfilen skall innehålla uppgifter som behövs för LDAP-sökning, samt en flagga för e-postinställningen för att uppnå följande funktioner:

- Ingen LDAP-sökning
- Sökning, men ändringsbar e-postadress i applikationen.
- Sökning med ej ändringsbar e-postadress i applikationen.

2.1.3 Svar på annons

Vid svar på en annons skall ett webbformulär göra det möjligt att skicka ett e-postmeddelande till annonsören. Avsändarens e-postadress skall vara konfigurerbar på samma sätt som då en annons skapas. (se kapitel 2.1.2)

2.1.4 Kategorier

Kategorier skall finnas i ett centralt register. Namnet på kategorier skall vara ändringsbart i efterhand utan att annonser påverkas. Möjliga kategorier är t ex Kurslitteratur, Bostad, Hemelektronik.

2.1.5 Annonstyp

Annonstyper skall finnas i ett centralt register. Möjliga annonstyper är t ex Köpes, Säljes, Bytes. Namnet på annonstyper skall vara ändringsbart i efterhand utan att annonser påverkas.

2.1.6 Användarmoderering

Användare skall kunna begära att en annons skall tas bort om användaren finner annonsen oseriös. Denna funktion skall konfigureras i en inställningsfil för följande funktionalitet:

- Ej tillgänglig
- Anmälan, vilket innebär att en administratör erhåller ett meddelande om att annonsen anses som oseriös.
- Aktiv moderering, vilket innebär att annonsen tas bort tillfälligt.

Vid anmälan eller aktiv modering skall orsak anges via ett webbformulär. Vid aktiv moderering skall användare, annons och tidpunkt loggas, samt att annonsägaren får ett meddelande om den orsak som användaren angivit.

2.1.7 Sökfunktion

Det skall vara möjligt att söka information både i rubrik och i annonstext. Omfattningen av sökfunktionen skall vara begränsad till annonstyp. Träffresultatet skall presenteras i ”normalvyn” för aktuell annonstyp, dvs egentligen en filterfunktion.

2.1.8 Driftmeddelande

Applikationen skall kunna visa ett driftmeddelande om underhåll sker på applikationen. I detta fall skall applikationen inte vara användbar.

2.2 Moduler

De fristående modulerna skall utvecklas i följande prioriteringsordning:

2.2.1 E-post

En besökare skall kunna kontakta annonsägaren utan att annonsägaren behöver visa sin e-postadress genom ett formulär i applikationen. Denna funktion skall logga vilken användare som skickade brevet, mottagande e-postadress samt aktuell tid.

2.2.2 Administratörsmoderering

En funktion som möjliggör att annonser kan:

- Tas bort tillfälligt
- Publiceras på nytt
- Tas bort helt ur systemet, både en specifik annons och en ”bäst-före” funktion som kan ta bort annonser äldre än ett visst antal dagar.

Administrationen skall göras i applikationen och endast vara tillgänglig för de som tilldelats administratörsbehörighet. Administratörer skall tilldelas dessa rättigheter genom en grupptillhörighet som erhålls via uPortal och/eller via en inställningsfil.

Loggning av användningen skall ske då någon åtgärd utförs.

2.2.3 Filuppladdning

Möjlighet att ladda upp filer till applikationen. Modulen skall kunna begränsa filstorlek och vilka bildformat som är tillåtna. Filerna skall lagras på en central yta.

2.2.4 Bildhantering

En funktion som visar en bild i applikationen och anpassar denna till lämplig storlek oavsett format eller filtyp.

2.2.5 Språkstöd

Informationstexter i applikationen skall kunna hämtas på flera olika språk. De olika språken skall vara redigerbara via förslagsvis en XML-fil. Aktuellt språk i applikationen skall avgöras av användarens personliga inställningar i uPortal.

2.2.6 Auktion

En funktion som möjliggör auktioner. Auktioner skall vara tidsstyrda där kvarvarande auktionstid skall vara identisk i samtliga instanser av applikationen.

2.2.7 Nyttiga länkar

En funktion som gör det möjligt att tipsa användarna om saker med anknytning till aktuell annonskategori. T ex kan man tänka sig en länklista på bostadsbolag då man använder sig av annonskategorin ”Bostad”.

2.2.8 Prenumeration

En funktion som gör det möjligt för användare att prenumerera på nya annonser i en eller flera annontyper eller kategorier.

3 Krav på applikationen

3.1 Generella krav

Applikationen skall vara modulärt uppbyggd för att modulerna skall gå att återanvända i andra utvecklingsprojekt. Samtliga moduler skall dokumenteras så att andra utvecklare enkelt kan sätta sig in i hur de fungerar.

3.2 Allmänt

Applikationen skall hantera multipla instanser av både applikationen i sig, samt av användare som har flera sessioner mot en eller flera instanser av applikationen.

Detta innebär mer konkret att applikationen skall kunna köras samtidigt i flera portaler mot en gemensam databas där användaren är inloggad i en eller flera portaler, med en eller flera sessioner mot varje portal.

3.3 Användarmiljö

Användargränssnittet till applikationen skall vara intuitivt och rymmas i en 50% spalt i uPortal vid skärmupplösningen 1024 x 768.

3.4 Säkerhet

Applikationen skall integreras i en portal som använder SSL. Databasanrop och annan eventuell extern kommunikation, t ex via LDAP, skall vara krypterad där så stöds av det externa systemet. Om kryptering skall användas eller inte skall anges vid installation av applikationen. Applikationen skall använda sig av den inbyggda autentiseringsfunktionen i uPortal.

3.5 Specifika krav

Applikationen skall exponeras via portletstandarden JSR-168 och inrymmas i portalramverket uPortal version 2.4 eller senare.

3.6 Databashanterare

Applikationen skall använda sig av databasen PostgreSQL, men vara konstruerad på ett sådant sätt att den på ett enkelt sätt skall kunna användas tillsammans med andra databashanterare.

3.7 Dokumentation

Kommentarer i källkod och installationsanvisningar skall skrivas på engelska.