Handelshögskolan vid Göteborgs universitet
Institutionen för informatik

Att navigera på webbplatser

- En dynamisk lösning för decentraliserat ansvar

Mattias Lindström

Kurs: Examensarbete II (10 poäng)
Nivå: D
Handledare: Wera Tegner Johansson
Termin: VT 1999

Sammanfattning

I detta arbete presenteras en Java-baserad navigeringsfunktion som möjliggör decentralisering av ansvaret för organisationens webbplats. Genom att dela upp internet bläddraren i två frames, och placera navigeringsfunktionen i den ena, skapas basen för en tydlig webbplats. Navigeringsfunktionen har ett gränssnitt och en funktionalitet som liknar filhanteraren i Windows med den viktiga skillnaden att "filerna" har ersatts med länkar till HTML-dokument vilka visas i motstående frame. Navigeringsfunktionen stöder därmed att länkobjekt (filer) struktureras i olika kataloger utifrån ämnesområde eller dylikt. En sökfunktion har också implementerats vilken möjliggör sökning efter HTML-dokument som matchar användarens sökkriterium. Den stora nyheten med lösningen är att användare direkt, via navigeringsfunktionens gränssnitt, kan lägga till, ändra eller ta bort katalog- och länkobjekt. Grundtanken i lösningen är sålunda att användare direkt skall kunna uppdatera navigeringsfunktionen samt därtill kopplade dokument vilket minskar behovet av en central webbavdelning (eller webbadministratör). Då lösningen inte stöds av gamla internet bläddrare är den främst lämpad för intranät även om den under vissa förhållanden kan vara intressant på internet.

Innehållsförteckning

Inledning

Utvecklingen på internet har varit explosionsartad de senaste åren. Ständigt lanseras ny teknologi vilken syftar till att förbättra mediets funktionalitet och utseende. Även för den professionella utvecklaren börjar det bli svårt att hålla reda på alla begrepp. För bara några år sedan handlade webbutveckling uteslutande om HTML-design. Idag har ett antal programmeringsspråk, komponentstandarder och tillägg gjort möjligheterna för webbutvecklare enorma. Java, JavaBeans, ActiveX, CGI, Active Server Pages (ASP), dynamiskt HTML (DHTML) och JavaScript är bara ett urval att modeord bakom vilka olika teknologier döljer sig. Trots en explosionsartad utveckling på internet ser vi idag en rad eftersatta behov. Vardagsproblem som döda länkar, inaktuell information, och dåligt strukturerad information är måhända inte så

spännande men de skapar otvetydigt användarmissnöje och leder till minskat förtroende för webbplatser. Istället för att koncentrera sig på tekniken måste helheten analyseras och förstås om en väl fungerande webbplats skall kunna skapas.

I detta arbete presenteras en Javabaserad lösning ämnad att skapa bättre webbplatser. Lösningen är främst ämnad för intranät och bygger på tanken att arbetet med webbplatsen skall decentraliseras ner i organisationen. En webbplats som underhålls direkt av medarbetarna kan bli mer levande samtidigt som informationen blir mer aktuell. Samtidigt spar även organisationen pengar då stora delar av den annars nödvändiga webbavdelningen kan frigöras till andra göromål. Den färdiga lösningen kan beskådas på www.emlin.se och gäster kan – efter att de online erhållit ett användarkonto – testa lösningen fullt ut. Nödvändig programvara för att lösningen skall fungera är Internet Explorer 4.0 eller Netscape Communicator 4.04 med uppdaterad Java-tolk.

Syfte

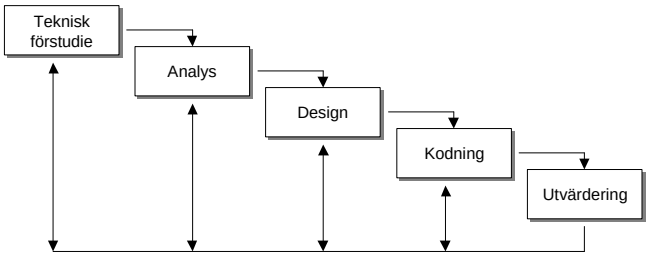
Syftet med detta arbete är att utveckla en attraktiv och tydlig navigeringsfunktion för webbplatser. Användare skall kunna uppdatera navigeringsfunktionen dynamiskt i realtid vilket möjliggör att arbetet med webbplatsen kan decentraliseras ner i organisationen. Arbetet förutsätter att läsaren har grundläggande kunskaper om internet/intranät samt objektorienterad programmering.

Metod

Detta är ett praktiskt arbete ämnat att ta fram en fullt fungerande programvara i enlighet med syftet. För att åstadkomma detta krävs god kunskap inom problemdomänen samt relevanta utvecklingsverktyg och arbetsmetoder. Kunskap inom problemdomänen har jag erhållit sedan jag 1997 skrev ett examensarbete på IBM om möjligheter att använda programmeringsspråket Java i databaslösningar. Kort tid därefter startade jag även (det numera inaktiva) företaget Emlin AB vilket sålde enklare Java- och CGI-lösningar till företag. Nuvarande examensarbete kan ses som en utveckling av de enklare program som skapats under denna tid.

Vad gäller utvecklingsverktyg och programmeringsspråk har jag huvudsakligen använt mig av programmeringsspråket Java för att skapa eftersträvad lösning. I Java har utvecklingsmiljön Java Development Kit 1.1 (JDK 1.1) använts. JDK 1.1 är utvecklad av Sun och den funktionalitet som har utnyttjats i detta paket består i huvudsak av ingående klasspaket, kompilator samt program för att exekvera och testa skrivna program (appletviewer). Då lösningen även har krävt några enklare (CGI) program som exekveras på HTTP servern (Webb-server) har jag, förutom Java, även varit tvungen att programmera i C. I detta programmeringsspråk har utvecklingsmiljön Borland Turbo C++ använts. I denna utvecklingsmiljö har endast ANSI C kompatibla funktioner utnyttjats då aktuell Solaris (Unix) HTTP server bara stöder sådana.

Utöver relevanta kunskaper inom problemdomänen och goda utvecklingsverktyg krävs även en arbetsmodell för att skapa struktur i arbetet. Som arbetsmodell valdes den i systemeringskretsar välkända vattenfallsmodellen. Vattenfallsmodellen erbjuder förutom struktur även ett bra ramverk för uppsatsen. Som synes i figur 1 (nedan) består vattenfallsmodellen av de fem arbetsfaserna teknisk förstudie, analys, design, kodning och utvärdering. (Pressman, 1994)



Figur 1: Vattenfallsmodellen (Pressman, 1994)

Eftersom en mjukvarulösning alltid är en del i ett större system fastställs i den tekniska förstudien vilka krav omgivningen – t.ex. användare, hårdvara och operativsystem – ställer på navigeringsfunktionen. Analysfasen syftar till att fastställa de grafiska och funktionella kraven på lösningen, d.v.s. vilket gränssnitt som skall användas samt vad lösningen skall klara av. I designfasen fastställs en relativt detaljerad systemlösning vilken huvudsakligen illustreras grafiskt. När designarbetet är avslutat konstrueras, i kodningsfasen, ett fungerande program i enlighet med tidigare designbeslut. Avslutningsvis utvärderas lösningen och jämförs med de krav som fastställdes i analysfasen. (Pressman, 1994) Rent strukturellt kommer utvecklingsfaserna att redogöras för i kapitel tre t.o.m. sju. För att arbetet skall vara lättläst inleds det dock med en kort bakgrundsdiskussion i kapitel två samt avslutas med en slutdiskussion i kapitel åtta.

Bakgrund

Många organisationer har idag webbplatser där informationen är dåligt strukturerad. Användare tvingas allt som oftast klicka på ett antal blindminor innan de hittar eftersökt information. (Noren, 1998) Enligt Jakob Nielsen, en känd expert på användargränssnitt, är några av de vanligaste orsakerna till dåliga webbplatser ologiska URL:er (Uniform Resource Locator), döda länkar, för långa sidor, dåliga navigerings- och sökfunktioner, inaktuell information samt långa väntetider. (Nielsen, 1999-02-15) Vid undersökningar med simulerade webbplatser har det framkommit att ungefär hälften av användarna någon gång känner sig "lost in hyperspace". (Lim & Paynter, 1998) Vidare visas att bara 10 procent av läsarna "scrollar ner" i en lång Web-sida om de inte letar efter något speciellt. (Geijer, 1996)

Ett vanligt sätt att lösa problemen med långa sidor och dåligt navigeringsstöd är frames kombinerade med navigeringsfunktioner. (Spool, Scanlon, Schroeder, Snyder, DeAngelo, 1997) I figur 2 (nedan) syns t.ex. Dagens Industris (DI:s) webbplats med en navigeringsfunktion i vänster frame och information i höger frame. Användare hittar eftersökt information i menyn genom att expandera katalogobjekt vartefter eftersökta länkobjekt väljs. Navigeringen förenklas i dylika lösningar genom att alla länkar samlas i en dynamisk och expanderbar navigeringsfunktion vilken är separerad från informationen. Undersökningar har visat att dynamiska navigeringsfunktioner minskar användares navigeringstid jämfört med statiska funktioner som det inte går att bläddra i. (Nation, Plaisant, Marchionini, Komlodi, 1999-02-20)

Figur 2: Dynamisk navigeringsfunktion (Dagens Industris webbplats, 1999-03-03)

Problemet med flertalet befintliga navigeringsfunktioner är att de är skapad m.h.a. HTML (Hypertext Markup Language). När en användare klickar på ett katalogobjekt måste en HTTP server anropas och hela navigeringsfunktionen laddas om. Att ladda om navigeringsfunktionen tar tid vilket skapar irritation, särskilt när belastningen på servern är hög. Nielsen (1999-02-15) nämner också långa väntetider som ett av de vanligaste problemen med webbplatser. För att minska detta problem vore det fördelaktigt att skapa en klientbaserad lösning där användare kan navigera i navigeringsfunktionen utan att någon HTTP server behöver anropas. Fördelen med en klientbaserad ansats vore alltså att nätverkstrafiken skulle minska vilket ger bättre prestanda. Om navigeringsfunktionen dessutom tillåter att kataloger nästlas i andra kataloger kan navigeringsfunktionen i teorin rymma ett oändligt antal länkar.

Utöver problemet med väntetider nämner Nielsen (1999-02-15) döda länkar och inaktuell information som två stora problem. För att hålla informationen på intranätet aktuell krävs att organisationen avsätter tillräckliga resurser till den avdelning inom organisationen som ansvarar för intranätet. Om denna avdelning, eller person, inte får tillräckligt med resurser riskerar den att bli en flaskhals som hindrar intranätet från att bli riktigt levande. Med otillräckliga resurser kommer inte dokument från de anställda upp på nätet tillräckligt snabbt samtidigt som problematiken med inaktuell information och döda länkar tilltar. För att möjliggöra levande intranät för organisationer med begränsade resurser vore det bra om beroendet av webbavdelningen minskade. (IDG News, 1999) Ett sätt att åstadkomma detta är att de anställda på egen hand uppdaterar navigeringsfunktionen och informationen på webbplatsen.

För att ett sådant arbetssätt skall fungera rent praktiskt krävs att navigeringsfunktionen går att uppdatera online. Detta innebär att det är möjligt för användare att direkt via internet bläddraren lägga till, uppdatera eller ta bort katalog- och länkobjekt i navigeringsfunktionen. Vid efterforskningar på internet har jag inte funnit något material om navigeringsfunktioner som tillåter att användare uppdaterar dem direkt, via deras eget gränssnitt. Möjligtvis kan bristen på dylika lösningar förklaras av att sådan funktionalitet inte är intressant på internet och att internet har drivit den tekniska utvecklingen på området. På internet skulle främmande användare kunna förstöra webbplatser genom att t.ex. lägga in länkobjekt till suspekta sidor eller på annat sätt manipulera webbplatsen. På ett mindre företags interna nätverk är dock sådan funk-

tionalitet intressant. Om användare har möjlighet att enkelt uppdatera navigeringsfunktionen samt därtill kopplade dokument, kan beroendet av en central webbavdelning minimeras[1]. Som en följdverkning kan även problemen med döda länkar och inaktuell information minska eftersom personalen direkt kan administrera navigeringsfunktionen och därtill kopplade dokument.

Nielsen (1999-02-15) nämner även dåliga eller icke existerande sökfunktioner som ett stort problem på webbplatser. Vid undersökningar har det visat sig att ca en tredjedel av alla användare försöker navigera på webbplatser m.h.a. platsens sökfunktion. (Spool, Scanlon, Schroeder, Snyder, DeAngelo, 1997) Eftersom navigeringsfunktionen, i enlighet med syftet, skall hjälpa användare att navigera på webbplatser så måste en sökfunktion inkluderas. Sammanfattningsvis är målet med navigeringsfunktionen att den skall eliminera eller förbättra följande punkter i Nielsens lista:

- Problemet med ologiska URL:er minskar då dessa adresser ej syns i en lösning som baseras på frames. När information om en sidas URL ändå behövs skall det vara lätt att, i navigeringsfunktionen, ta fram den.
- Problemet med döda länkar och inaktuell information minskar då ansvaret för intranätet decentraliseras och det blir enklare för informationsägare att uppdatera länkar (och dokument).
- Problemet med dåligt stöd för navigering och sökning minskar genom att länkar centraliseras och struktureras samtidigt som en speciell sökfunktion byggs.
- Problemet med långa väntetider minskar genom att användaren snabbare hittar rätt information samtidigt som den javabaserade navigeringsfunktionen erbjuder bättre prestanda än motsvarande HTML-lösningar.

För att problemet med döda länkar och inaktuell information skall lösas förutsätts att organisationen stöder ett decentraliserat arbetssätt. Om användare inte tar personligt ansvar för länkar i navigeringsfunktionen, samt därtill kopplade dokument, kommer naturligtvis problemet med länkar och inaktuell information snarare att bli värre. Härav är det viktigt att en analys görs av organisationens möjligheter att stöda ett decentraliserat arbetssätt innan navigeringsfunktionen installeras.

Teknisk förstudie

Den tekniska förstudien är den första arbetsfasen i vattenfallsmodellen och syftar till att fastställa vilka krav som ställs på lösningen i relation till den omgivande miljön. Då det är svårt att uttala sig om slutanvändare eller systemmiljö kommer huvudsakligen valet av programmeringsspråk att diskuteras. Kritiska framgångsfaktorer vad gäller valet av programmeringsspråk är god prestanda, maximal portabilitet mellan olika typer av operativsystem samt gott stöd för grafik- och nätverksprogrammering. Eftersom jag har tidigare erfarenhet av Java är det min bedömning att detta programmeringsspråk har de egenskaper som krävs för att skapa en väl fungerande lösning i enlighet med syftet. Utöver valet att använda Java för att implementera lösningen krävs även att en specifik version av programmeringsspråket väljs. Generellt sett är senare versioner av Java mer avancerade men stöds i gengäld av färre internet bläddrare. Då det är viktigt att vald version av Java stöder popup-menyer – vilket inte de första versionerna

[1] Att uppdatera HTML-dokument är idag relativt enkelt då de flesta nya ordbehandlare, t.ex. MS Word och Lotus Word Pro, stöder att dokument sparas som HTML.

av programmeringsspråket gjorde – måste version 1.1 användas. (JDK 1.1 Documentation, 1999-03-10) Denna version av Java är idag (1999) cirka två år gammal och stöds i nuläget av (minst) Internet Explorer 4.0 och Netscape Communicator 4.04 med uppdaterad Java-tolk (Java Virtual Machine)[2].

Java är ett objektorienterat programmeringsspråk som bygger på klasser och klassinstanser, s.k. objekt. (Walsh, 1997) Klasser används som mallar för hur instanserna, objekten, ska se ut. I klasserna inkapslas de variabler och metoder som objektet tillhandahåller. Metoderna används för att manipulera variablerna som är objektets egenskaper. Objektorientering tillåter arv. Med arv menas att en klass kan använda andra klassers metoder och variabler genom att ärva dessa. Java stöder inte multipelt arv, d.v.s. en klass kan inte ärva från fler än en annan klass. En ärvande klass kallas för subklass och klassen den ärver från kallas basklass. En subklass är i de flesta fall mer specialiserad än sin basklass eftersom den innehåller både basklassens egenskaper plus egna, mer specialiserade egenskaper. (Budd, 1991)

Problem löses i ett objektorienterat språk genom att objekt (klassinstanser) kommunicerar med varandra. Ett objekt behöver m.a.o. inte lösa alla uppgifter själv utan kan skicka en begäran om hjälp till ett annat objekt. Objektorientering har blivit väldigt populär och många gamla programmeringsspråk förnyas och blir allt mer objektorienterade. Exempel på detta är t.ex. det objektorienterade programmeringsspråket C++ som uppkommit från språket C. (Budd, 1991). Javas syntax är väldigt lik den syntax som används i C++ men många av de mer krångliga passagerna har tagits bort varför begrepp som pekare, överlagring av operatorer och multipelt arv inte finns i Java. I Java sker även så kallad skräphantering (garbage collection) automatiskt. Med detta menas att objekt som har skapats förstörs automatiskt när dessa inte används mer. Detta förenklar programmeringen avsevärt gentemot C++ där avallokering av minne måste göras explicit av programmeraren. (Cornell & Horstmann, 1997)

Java har bra stöd för nätverks protokoll som HTTP, Telnet och FTP vilket varit något av en förutsättning för detta examensarbete. Trots att vissa fel numera har upptäckts i Javas säkerhetsmodell måste språket ändå anses som relativt säkert. För närvarande har olika typer av Java-program olika säkerhetsnivåer. Sålunda har ett Java program på Internet (applet) betydligt mindre rättigheter att utforska och manipulera sin omgivning än ett Java-program som exekveras lokalt från hårddisken. (Cornell & Horstmann, 1997) Vad gäller applets pratar Sun ofta om den sandlåda i vilken den nedladdade appleten kan manipulera sin omgivning. I dessa termer är appleten fast i sandlådan och kan inte manipulera eller hämta information utanför sandlådan. Sandlådan är i liknelsen den bit av primärminnet eller hårddisken som appleten tilldelats av operativsystemet och internet bläddraren. (Ek, 1996)

Den mest banbrytande egenskapen hos programmeringsspråket Java är att det är plattformsoberoende. Sun marknadsför språket med devisen "Write once, run anywhere" vilket innebär att programmerare bara behöver skriva ett program vilket sedan kan användas på ett flertal olika plattformar utan besvär med konvertering eller omkompilering. (100 Percent Pure Java Program, 1999-02-10) Rent tekniskt implementeras plattformsoberoendet genom att javakompilatorn genererar en arkitektoniskt neutral bytekod som är exekverbar på ett flertal olika processorer. Bytekoden består av instruktioner som är generella och lätta att översätta till maskinkod på det specifika systemet. (Walsh, 1997)

---

[2] För att uppdatera Java-tolken på Netscape Communicator 4.04 kan man gå till Netscapes hemsida: www.netscape.com. Om man har senare versioner av Netscapes internet bläddrare behöver man förmodligen inte uppdatera Java-tolken.

Arbetet med att översätta bytekod till maskinkod kallas interpretering och Java är därmed ett interpreterande programmeringsspråk. Denna konstruktion gör Java-program något långsammare än andra program eftersom bytekoden inte kan optimeras för någon specifik processor. Vinsten är naturligtvis att Java program utan omskrivning kan exekveras på ett flertal olika operativsystem. Java är även designat för trådar vilket gör det möjligt att utnyttja multipla processorer, alternativt att skriva bättre och effektivare kod för system med en processor. Med trådar kan olika kodavsnitt köras i separata trådar vilka exekveras i parallell. Exempelvis kan en tidskrävande filläsning ske i en tråd samtidigt som användaren interagerar med gränssnittet vilket exekveras i en annan tråd. Utan trådar hade användaren varit tvungen att vänta på filläsningen för att sedan interagera med gränssnittet. (Cornell & Horstmann, 1997)

Analys

I följande avsnitt redogörs för de krav som ställs på navigeringsfunktionens funktionalitet och gränssnitt. Sist i avsnittet beskrivs den övergripande systemlösningen.

Funktionella krav

Under rubrik två (bakgrund) avhandlades kort ett antal problem som navigeringsfunktionen skulle lösa. I följande avsnitt redogörs mer detaljerat för de funktionella krav som ställs på lösningen. Det första viktiga kravet är att navigeringsfunktionen skall underlätta för användare att hitta rätt information på webbplatser. Detta krav mötes genom att navigeringsfunktionen separeras från informationen m.h.a. frames samt att alla dokument på webbplatsen kan nås via nämnda navigeringsfunktion. Undersökningar visar att om frames används på "rätt sätt" så förbättras webbplatsens struktur, vilket gör det enklare att navigera[3]. (Spool, Scanlon, Schroeder, Snyder, DeAngelo, 1997)

Det andra viktiga kravet på navigeringsfunktionen är att den skall vara möjlig att uppdatera i realtid. För att informationen i navigeringsfunktionen skall vara möjlig att förändra i realtid krävs att strukturen inte är hårdkodad utan läses in från en fil. Navigeringsfunktionen skall alltså läsa in strukturbeskrivningen från en fil och rita upp ett gränssnitt i enlighet med denna beskrivning. För att förändra gränssnittet modifieras strukturbeskrivningen i filen vartefter appleten läser in den nya strukturbeskrivningen och ritar upp gränssnittet. För att förhindra anarki på hemsidan måste någon form av användarregistrering ske så att inte strukturen kan ändras av personer utan behörighet. För att kontrollera användaridentitet måste användaruppgifter registreras i ytterligare en fil.

För att göra det möjligt att snabbt hitta rätt information krävs en sökfunktion. Sökningen bör automatiskt inkludera all tillgänglig information på webbplatsen på samma sätt som sökfunktionen i de flesta operativsystem inkluderar all information på en dators hårddiskar. Sammanfattningsvis fastställs att navigeringsfunktionen bör tillhandahålla följande funktionalitet:

- strukturera information i expanderbara katalog- och länkobjekt.
- möjliggöra sökning i HTML-dokument.

[3] Användning av frames är en het potatis vad gäller webb design. Vissa undersökningar visar på positiva samband mellan frames och användbarhet medan andra visar på negativa samband. De motstridiga resultaten beror troligen på hur frames har implementeras.

- möjliggöra att användare lägger till katalog- och länkobjekt.
- möjliggöra att användare uppdaterar katalog- och länkobjekt.
- möjliggöra att användare tar bort katalog- och länkobjekt.
- ge allmän information om katalog- och länkobjekt.
- skapa möjligheter att skydda ömtåliga katalog- och länkobjekt.

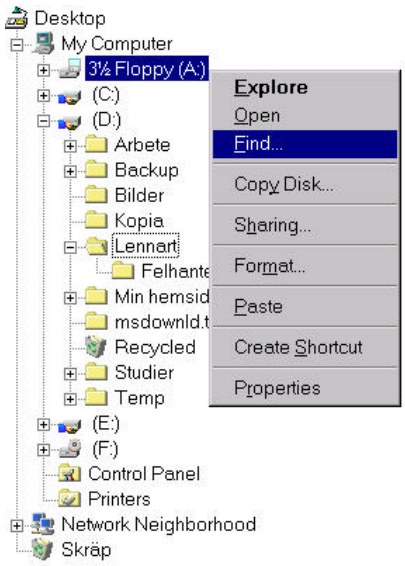
Krav på gränssnittet

Insikt i användarens uppfattning av hur gränssnittet borde fungera är avgörande för om ett nytt system verkligen kommer att hjälpa användaren. Ett intuitivt och lättanvänt gränssnitt syns ej - det är inte en belastning i arbetet. (Sandbäck & Göransson, 1999-03-08) Centralt är att användarens mentala modell av det nya systemet är besläktad med programmerarens mentala modell. (Preece, 1994) I linje med detta måste valt gränssnitt för navigeringsfunktionen vara attraktivt, robust och lättförståeligt.

Alla åtgärder bör ge någon form av återkoppling. Om inte användaren omedelbart får återkoppling på ett arbetsmoment tenderar han att upprepa åtgärden, vilket kan leda till oönskade konsekvenser. Även vid fel är det nödvändigt med återkoppling eftersom det viktigt att användaren på ett tidigt stadium får reda på eventuella felinmatningar etc. (Preece, 1994) Särskilt känsliga operationer vad gäller återkoppling är tiden som går när appleten laddas hem till klienten och tiden som går när sökningar genomförs. För att minimera användarosäkerhet krävs någon form av feedback. När appleten laddas hem kan en statusladdare visas vilken minskar risken för att användaren avbryter nedladdningen. När sökningar genomförs kan återkoppling ske på så sätt att markören byter form till ett timglas.

Flera undersökningar visar att det finns klara fördelar med att använda välkända metaforer för gränssnittet. (Preece, 1994) Dialogen ska överensstämma med användarens förväntningar samt med dennes kunskap och erfarenheter. (Sandbäck & Göransson, 1999-03-08) En välkänd metafor, som stöder det funktionella kravet på expanderbara katalog- och länkobjekt, är filhanteraren i Windows (se figur 3)[4]. Eftersom filhanteraren är välkänd kommer den att ligga till grund för arbetet med det slutliga gränssnittet. Valet att använda filhanteraren i just Windows kan förbrylla något men förklaras enklast av att Windows är det vanligaste klientbaserade operativsystemet idag. Windows är alltså en maximalt välkänd metafor som bör gå snabbt att lära sig för de flesta användare. Eftersom bakgrundsfärg, textfärg och ikoner går att byta är det enkelt att förändra gränssnittet för de organisationer som inte gillar vald metafor[5].

[4] Filhanterarna i Windows –95, Windows –98 och Windows NT 4.0 är snarlika.
[5] Ikonerna i lösningen är transparenta GIF-bilder (Graphics Interchange Format).



Figur 3: Filhanterare i Microsoft Windows

I navigeringsfunktionen kommer filerna i filhanteraren att ersättas med länkobjekt till HTML-dokument. När en användare klickar på en fil visas alltså det kopplade dokumentet i internet bläddraren. För att användare skall veta vilka URL:er de tidigare besökt är det lämpligt att besökta länkobjekt markeras på något sätt. Precis som i filhanteraren i Windows skall en po-pup-meny visas när användare högerklickar på katalog- och länkobjekt. I popup-menyn skall det finnas kommandon för att öppna och stänga enheter; söka dokument; lägga till, ta bort och uppdatera enheter samt få allmän information.

Vad gäller sökfunktionen, så skall även denna dialog i så hög grad som möjligt kännas famil-jär för maximalt många användare. De flesta sökfunktioner inkluderar ett fält där söksträngen fylls i, en sökknapp samt en textarea där resultatet visas. I aktuell sökfunktionen är det lämp-ligt att dessa element kombineras med en knapp för att bläddra till de HTML-dokument som matchar sökkriteriet. Språket i sökfunktionen, och alla andra dialoger, blir engelska. Skälet är att navigeringsfunktionen med detta språk kan användas av flera användare än bara svenskar. Ett vanligt problem med sökfunktioner på internet/intranät är att användare lätt missförstår vilken del av webb platsen som "sökmotorn" går igenom. (Spool, Scanlon, Schroeder, Sny-der, DeAngelo, 1997) För att undvika missförstånd är det därför viktigt att användaren ges information om var sökningen sker. Denna information kan enkelt ges med en statustext i sökdialogens nederkant.

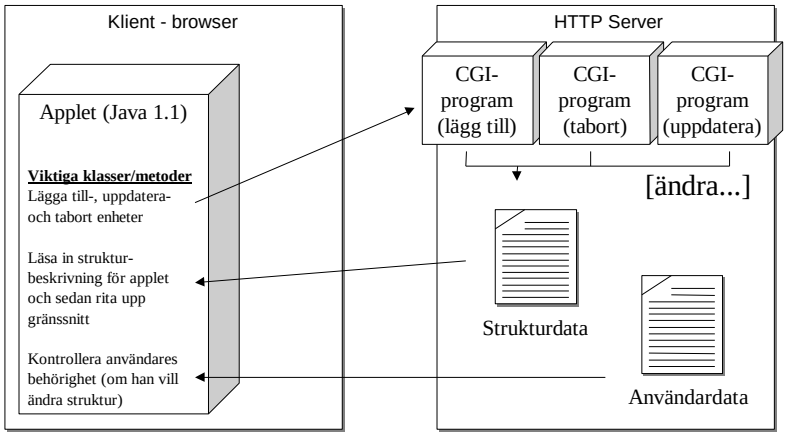
Ett problem med sökfunktionen är att den skall visas i ett fönster som är skilt från internet bläddraren. Många användare har klagomål på webbsidor som startar nya fönster huller om buller. (Spool, Scanlon, Schroeder, Snyder, DeAngelo, 1997) Tyvärr får det nog ändå anses som bättre att sökfunktionen visas i ett separat fönster jämfört med att samma funktion poppar upp mitt i internet bläddraren. Samma lösning, med ett fristående fönster kommer även att behöva användas när 1) användare skall modifiera katalog- och länkobjekt samt 2) visa att de äger rättigheter att göra så.

Grundläggande systemlösning

För att implementera navigeringsfunktionen krävs en Klient/Server lösning. Klient/Server är en teknik där två program utbyter instruktioner och data med varandra. Användaren arbetar

mot klientprogrammet och behöver inte se bakomliggande serverprogram. (Blume, 1999-03-09) I den aktuella lösningen ansvarar klienten (appleten) för användargränssnittet samt input- och output till användare. HTTP servern lagrar data om navigeringsfunktionens struktur i en strukturdatafil vilken läses in varje gång klienten startar. Med tre CGI (server) program som ansvarar för att lägga till, uppdatera och ta bort rader i strukturdatafilen kan navigeringsfunktionens katalog- och länkobjekt modifieras. Om en användare t.ex. vill lägga till ett katalog- eller länkobjekt anropar klienten CGI-programmet för att lägga till enheter vilket modifierar filen med strukturdata genom att skjuta in en ny rad med den begärda enhetens egenskaper. Därefter läser klienten återigen in den förändrade strukturdatafilen och ritar upp det nya gränssnittet.

För att kontrollera användaridentitet måste en fil med användardata skapas som lagrar data om giltiga användare. Innan en användare tillåts uppdatera navigeringsfunktionen måste han eller hon skriva in användarnamn och lösenord vilket sedan kontrolleras mot filen med användardata. Endast om användaren återfinns i filen med användardata och har skrivit in rätt användarnamn och lösenord tillåts han/hon att uppdatera navigeringsfunktionen. Nyskapade enheter skall även märkas med användarens namn vilket är viktigt för att förhindra missbruk, t.ex. att användare lägger in tvivelaktiga länkar i tron att de är osynliga. I figur 4 (nedan) visas en grafisk illustration av den övergripande systemlösningen.



Figur 4: Grundläggande principer för systemlösningen

På klientsidan kommer navigeringsfunktionen att vara uppbyggd av ett antal Java-klasser med olika funktionalitet. Instanser av dessa klasser (objekt) kommunicerar sedan internt med varandra när programmet exekveras. Några klasser som på ett tidigt stadium går att urskilja är en klass som ansvarar för användargränssnittet; en klass som representerar katalog- och länkobjekt; en klass som är ansvarig för att läsa in information från strukturdatafilen; en klass som hanterar sökfunktionen samt en klass som kontrollerar användaridentitet.

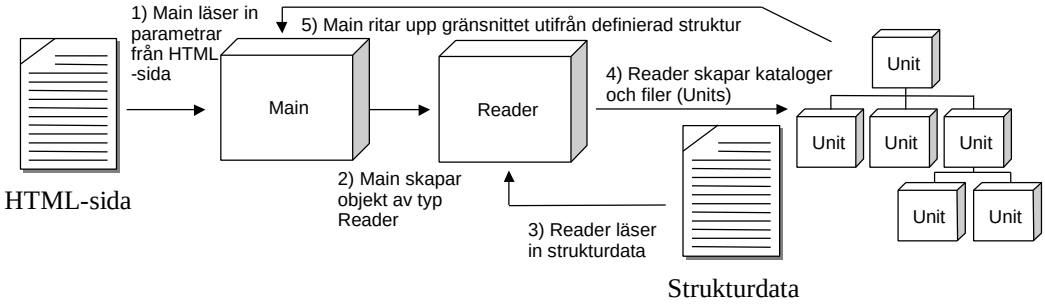
Design

Som nämnts tidigare skall huvuddelen av lösningen skapas med en objektorienterad ansats i programmeringsspråket Java. I kommande avsnitt följer en mer detaljerad beskrivning av de klasser som krävs för att skapa appleten, C-programmen på servern samt nödvändiga filer. De ingående klasserna i appleten kan till viss del delas upp efter funktionalitet varför jag börjar med att redogöra för de Java-klasser som har till uppgift att skapa det grafiska gränssnittet.

Det grafiska gränssnittet

För att i appleten skapa ett användargränssnitt som liknar filhanteraren i Windows krävs de tre klasserna Main, Reader och Unit. Instanser av dessa klasser (objekt) anropar sedan varandra flera gånger under programmets exekvering. Förloppet startar med ett inbäddat anrop i HTML-kod till klassen Main, vilken ärver klassen applet. Ett objekt av typen Main instansieras vilket läser in de parametrar som har bifogats i HTML-anropet. Parametrarna specificerar egenskaper som sökvägar till ikoner och indatafiler; bakgrunds- och textfärg m.m. (se appendix B). Efter detta skapar objektet Main ett objekt av typen Reader. Detta objekt läser in strukturdatafilen vilken innehåller information om navigeringsfunktionens struktur. Strukturdatafilen har en syntax där varje rad innehåller information om ett katalog- eller länkobjekt.

Utifrån strukturdatafilen skapar objektet Reader katalog- och länkobjekt samt fastställer dessa enheters inbördes relation. De katalog- och länkobjekt som Reader skapar är instanser av klassen Unit (enhet) och de mest centrala egenskaperna hos dessa objekt är: typ (katalogobjekt eller länkobjekt), namn, ikon, URL och barn. Enheterna ordnas i en trädstruktur där varje enhet vet vilka barn den har. Denna kunskap är viktigt då det alltid går att nysta ut enheternas inbördes relationer genom att fråga den ultimata föräldern; roten. I figur 5 (nedan) visas en förenkling av hur objekt av typen Main, Reader och Unit interagerar för att skapa användargränssnittet.



Figur 5: Hur klasserna Main, Reader och Unit interagerar.

Efter att objektet Reader har skapat katalog- och länkobjekt förstörs det. Objektet Main återtar kontrollen och nu används en programmeringsteknik kallad rekursion för att rita upp de katalog- och länkobjekt som Reader skapat. Rekursion är en vanlig och effektiv programmeringsteknik för att hantera trädstrukturer. (Budd, 1991) Nedan visas en rekursiv funktion i pseudokod, vilken utmärks av att den anropar sig själv:

```
funktionSkrivUtEnhet (Enhet e)                          // funktionshuvud
{
   skrivTillSkärmen (e.returneraNamn());                // skriv ut enhets namn på skärmen
   funktionSkrivUtEnhet (e.returneraBarn());            // upprepa förfarande med barn (rekursion)
}
```

Om vi tänker oss en trädstruktur och anropar denna funktion med roten som argument kommer rotens namn att skrivas ut vartefter funktionen anropar sig själv med rotens barn. Barnens namn skrivs ut och sedan anropas funktionen sig själv igen fast nu med rotens barnbarn. Namnet på barnbarnen skrivs ut och så fortsätter proceduren till dess att inga fler barnnoder finns. När inga fler noder finns har också namnet på alla enheter i trädstrukturen skrivits ut. Principerna i funktionen ovan är exakt de samma som används av Main; först ritas roten ut och sedan barnen och sedan barnbarnen o.s.v.. Förutom enheternas namn ritas även ikoner

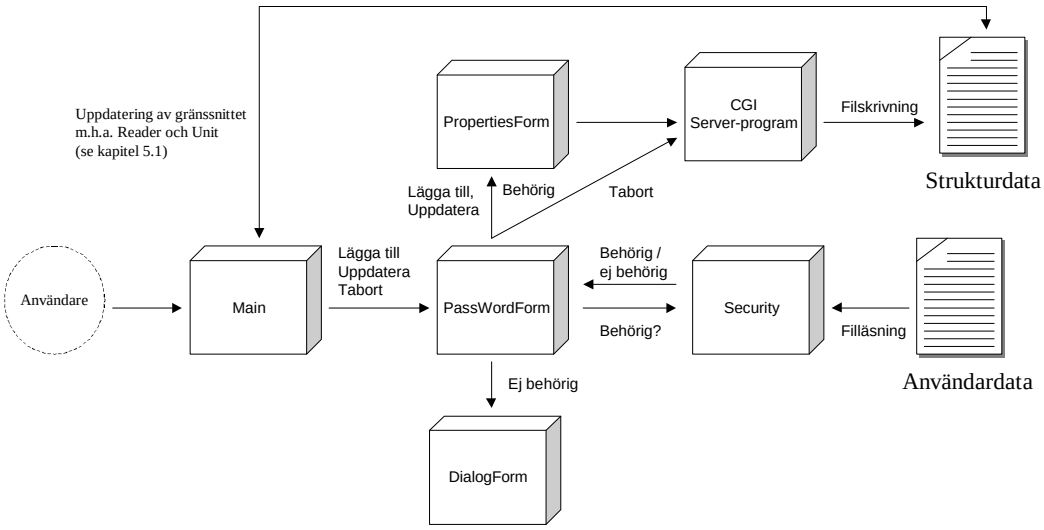
och linjer ut. Enheterna ritas också ut på ett attraktivt sätt vilket sker m.h.a. diverse variabler som håller reda på X- och Y-koordinater. Förutom i Main används ett rekursivt förfarande även i klassen SearchForm när den söker igenom länkade HTML-dokument i jakt på sidor som matchar inmatad söksträng (se rubrik 5.3).

Lägga till, uppdatera och ta bort enheter

Om användare vill lägga till, uppdatera eller ta bort katalog- och länkobjekt skall de kunna högerklicka på dem varvid de får upp en popup-meny med motsvarande val. Precis som i filhanteraren i Windows skall popup-menyn bara vara tillämplig för den enhet som användaren högerklickade på. Programmeringstekniskt används de fyra klasserna PassWordForm, PropertiesForm, DialogForm och Security för att implementera funktionalitet för att lägga till, ändra eller ta bort enheter. Ändelsen "Form" (fönster) i tre av klasserna har valts då klasserna inte ärver klassen Applet utan klassen Frame vilken används för att skapa fristående fönster.

PassWordForm är ett fristående fönster som används för att kontrollera användares behörighet att lägga till, uppdatera eller ta bort enheter. Fönstret skall ha två textfält; ett för användarnamn och ett för lösenord. I fönstret kommer även två knappar, ok och avbryt, att finnas. PassWordForm använder sig av klassen Security för att kontrollera om användare är behöriga att ändra i navigeringsfunktionen. Security kontrollerar behörighet genom att jämföra inmatade användarnamn och lösenord med information som finns i en fil på HTTP servern. Då filer på HTTP servern kan läsas av alla användare måste uppgifterna i denna fil vara förvrängda så att inte obehöriga användare kan bryta sig in i navigeringsfunktionen.

I de fall en användare är giltig och vill lägga till eller uppdatera en enhet, visas ett nytt fönster, PropertiesForm, där uppgifter kan skrivas in eller ändras. I de fall en användare är behörig och vill ta bort en enhet visas inga fler dialog utan vald enhet tas bort. Oavsett om användaren vill lägga till, uppdatera eller ta bort en enhet anropas sist i förfarandet ett CGI-program på HTTP servern som modifierar filen med strukturdata. Om en användare fyller i fel användarnamn eller lösenord skall en dialog (DialogForm) visas med ett enkelt felmeddelande. I figur 6 (nedan) visas det ovan beskrivna förloppet.



Figur 6: Tekniskt tillvägagångssätt för att lägga till, uppdatera och ta bort enheter i navigeringsfunktionen

Det kan förefalla konstigt att det krävs ett särskilt program på HTTP servern för att modifiera filen med strukturdata. Kunde inte appleten modifiera denna fil själv? Tyvärr är det omöjligt för appleten att göra så då de säkerhetsrestriktioner som gäller för applets inte tillåter dem att skriva på HTTP serverns hårddisk[6]. Restriktionerna för applets är befogade då det annars vore möjligt för dem att nedladda virus eller ändra viktiga parametrar på servern. Genom att kräva att applets går omvägen via serverbaserade program ökar möjligheten att kontrollera att de inte gör något olämpligt; program som exekveras på en server kan vanligtvis inte ändra i filer eller variabler utanför den av systemadministratören tillåtna miljön. (Cornell & Horstmann, 1997)

De program som anropas på HTTP servern är s.k. CGI-program. CGI står för Common Gateway Interface och är den vanligaste standarden för att sända information från en internet bläddrare till en HTTP server. Vanliga programmeringsspråk för att skapa CGI-program är C, Perl och Unix script. I en typisk CGI-lösning fyller användaren i ett formulär som är inbäddat i en HTML-sida och klickar på en knapp för att skicka iväg informationen. Internet bläddraren anropar sedan en HTTP server vilken vidarebefordrar anropet till det CGI-program som skall ta emot informationen. CGI-programmet utför sin uppgift och skapar sedan en svarssida vilket HTTP servern vidarebefordrar tillbaka till klienten. (Common Gateway Interface, 1999-03-08) I aktuell lösning kommer, till skillnad från ovan beskrivna förlopp, inte internet bläddraren att vara ansvarig för att skicka data till HTTP servern. Istället kommer appleten att skicka information direkt till CGI-programmet. Informationen som skickas till CGI-programmen varierar något beroende på om en enhet skall läggas till, uppdateras eller tas bort, men inkluderar följande:

- Namn på katalog- eller länkobjekt som står före aktuell enhet i strukturdatafilen. Används när CGI-programmet söker efter rätt rad i strukturdatafilen.
- Aktuell enhets namn i navigeringsfunktionen.
- Om aktuell enhet är ett katalog- eller ett länkobjekt.
- Om aktuell enhet initialt skall vara öppen (vald) eller stängd (icke vald).
- Aktuell enhets ägare, d.v.s. den person som skapat objektet.
- Om aktuellt länkobjekt skall visas i en frame eller hela internet bläddraren.
- Aktuellt länkobjekts ikon.
- Aktuellt länkobjekts URL.

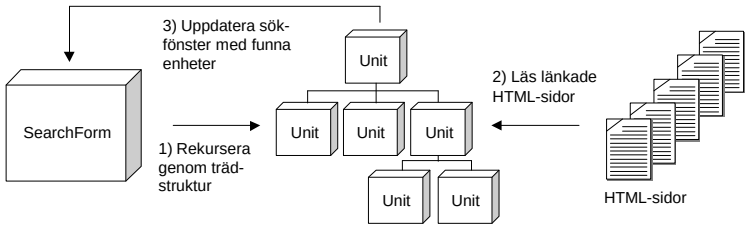
CGI-programmet måste i filen med strukturdata leta efter rätt rad för att lägga till, ändra eller ta bort rader. För att lägga till, uppdatera och ta bort katalog- och länkobjekt används två filer, en originalfil med strukturdata och en temporär tom fil. Från originalfilen kopieras sedan rad efter rad till den temporära filen. Efter att eftersökt "förälderobjekt" har påträffats skjuts en rad in med uppgifter som stämmer med de data som har skickats från appleten. Om ett länkobjekt skall tas bort hoppar CGI-programmet bara över att kopiera aktuell enhet. Om ett katalogobjekt skall tas bort blir CGI-programmet mer komplext eftersom det också måste ta bort alla underkataloger och filer[7]. Efter att aktuell enhet har lagts till eller tagits bort fortsätter CGI-programmet att kopiera resterande rader från originalet till den temporära filen. Sist tas originalfilen bort och ersätts med den temporära nyskapade filen. Förfarandet är i stort sett detsamma när en enhet skall läggas till eller uppdateras – skillnaden är bara vilken rad som

[6] En applet kan inte heller skriva på klientens hårddisk om inte användaren ger uttryckligt tillstånd till detta.
[7] För att se hur CGI-programmet tar bort katalogobjekt, samt subenheterna däri, hänvisas den intresserade läsaren till www.emlin.se/cgi-bin/delete.c

manipuleras. När ett objekt skall läggas till infogar CGI-programmet en ny rad medan det vid uppdatering skriver över en befintlig rad med nya uppgifter.

Sökning i navigeringsfunktionen

Den tidigare påtalade sökfunktionen kommer att implementeras m.h.a. klassen SearchForm. Liksom tidigare nämnda klasser med ändelsen "Form" är SearchForm ett fristående fönster där de viktigaste beståndsdelarna är ett textfält i vilket söksträngen fylls i; två knappar benämnda sök och gå-till-sida samt en textarea där resultatet visas. Sökdialogen kommer att nås genom att användare i popup-menyn ger kommandot sök. Därefter skall sökning ske i subkataloger och länkobjekt till den enhet som användaren högerklickade på. Som synes i figur 7 (nedan) används, liksom i klassen Main, ett rekursivt förfarande för att söka igenom trädstrukturen.



Figur 7: Hur sökning sker i länkade HTML-sidor

SearchForm rekurserar genom alla underkataloger och länkobjekt. I de fall enheten är ett länkobjekt sökes det kopplade dokumentet igenom. SearchForm söker igenom det kopplade dokumentet till dess att användarens söksträng hittas eller dokumentet tar slut. I de fall söksträngen hittas läggs länkobjektets namn och URL till i textarean med matchande enheter.

Rent programmeringstekniskt ställer klassen SearchForm ett intressant krav. För att det skall vara möjligt för användaren att avbryta tidskrävande sökningar måste trådar utnyttjas. Programmering med trådar innebär att man låter olika funktioner exekveras i parallell på processorn. På så sätt behöver inte en funktion vänta på att en annan skall slutföras utan kan istället utföras samtidigt och oberoende av den andra. (Cornell & Horstmann, 1997) För att en avvändare skall kunna avbryta sökningen placeras alltså denna funktionalitet i en separat tråd vilken en annan tråd, kopplad till ett kommando för att stoppa sökningen, kan avbryta. På så sätt behöver inte användaren vänta på att sökningen slutförs innan han t.ex. kan modifiera sin sökning eller stänga fönstret.

Syntax på indatafiler

Som framgått tidigare krävs två stycken filer i lösningen; en fil som innehåller strukturdata och en som innehåller data om behöriga användare. Som nämnts läser objektet Reader filen med strukturdata medan objektet Security läser filen med användardata. För att dessa två objekt skall kunna genomföra felfria läsningar är det viktigt att bägge dessa filer har en i förväg väl genomtänkt syntax. Vad gäller filen med strukturdata är syntaxen speciellt viktig då raderna måste läsas in i samma ordning som det rekurserande objektet Reader bygger upp den interna trädstrukturen (m.h.a. klassen Unit). Nedan visas den syntax som kommer att användas för filen med strukturdata:

```
M1            Kalle     Vårt intranät
M0            Kalle     Nyheter
F0F4          Mattias   Ledningen informerar  http://www.foretaget.se/info.htm
F0T4          Mattias   Till salu             http://www.foretaget.se/salu.htm
E
E
```

I den första kolumnen ges kort övergripande information om enheten. Det första av de maximalt fyra argumenten i första kolumnen anger om enheten är en katalog (M) eller ett länkobjekt (F); det andra argumentet anger om enheten är öppen (1) eller stängd (0); det tredje argumentet anger om länkobjekts kopplade dokument skall visas i en frame (F) eller i hela internet bläddraren (T) och det sista argumentet anger vilken av maximalt sex möjliga ikoner länkobjekt skall visas med[8]. Namnen i den andra kolumnen anger vem som äger den aktuella enheten. Detta namn sammanfaller med användarnamnet på personen som skapade enheten. Texten i den tredje kolumnen anger vilket namn enheten skall ha i navigeringsfunktionen medan den fjärde kolumnen anger vilken URL länkobjekt har. Sist i ovanstående exempel följer två nästan tomma rader med bokstaven E. Detta E skall tolkas som slut (End) och anger att räckvidden för närmast ovanstående katalog tar slut. Eftersom det är möjligt att uppdatera navigeringsfunktionen dynamiskt (via gränssnittet) behöver användare inte kunna något om strukturdatafilens syntax.

Filen med användardata rymmer användarnamn och lösenord separerade med tecknet "|". För att användare skall vara behöriga att ändra i menyn måste de registreras i denna fil vilket kräver att en systemadministratör skriver in uppgifterna manuellt[9]. Då alla personer kan läsa filen med användardata (liksom alla andra filer på HTTP servern) måste åtminstone lösenorden förvrängas. I detta arbete har ingen avancerad lösenordsalgoritm använts utan lösenordet vänds bara baklänges. Om navigeringsfunktionen i framtiden skall användas skarpt är detta skydd inte acceptabelt utan en mer avancerad algoritm måste införas. Nedan visas ett exempel på hur filen med användardata kan se ut. Notera att användarnamnen och de "förvrängda" lösenorden i detta exempel sammanfaller:

```
Mattias|saittaM|
Kalle|ellaK|
Mia|aiM|
Lisa|asiL|
```

För en mer avancerad version av navigeringsfunktionen vore det intressant att inkludera mer information i användardatafilen än bara användarnamn och lösenord. Till exempel kunde E-mail adresser lagras och göras tillgängliga när användare vill se egenskaper för katalog- och länkobjekt. E-mail adressen, tillsammans med funktionalitet för att starta en E-mail klient, skulle då förenkla om användare vill skicka synpunkter till de personer som ansvarar för katalog- och länkobjekt samt därtill kopplade dokument.

Kodning

I följande avsnitt visas några centrala kodavsnitt från navigeringsfunktionen. Då menyn är skapad m.h.a. ca 1800 rader Java- och C-kod har bara några intressanta och förkortade avsnitt

[8] Användare kan inte välja ikon för katalogobjekt. Dessa representeras av en ikon när objektet är öppet och en annan när det är stängt.
[9] På www.emlin.se skapar ett CGI program automatiskt användarkonton till alla som vill testa lösningen.

infogats i detta arbete. Texten som förklarar kodavsnitten hålles relativt kort men detta bör uppvägas med rikligt kommenterad kod. Det första kodavsnittet visar hur statusladdaren – som skall visas innan appleten har hämtat hem alla bilder och parametrar – har implementerats. Kodavsnittet är hämtat från klassen Main (Java-kod):

```
public void paint(Graphics g)                                   // (fördefinierade) metod som används för att rita på
skärmen
 {
  if (Loaded == false)                                                    // variabel som anger om paramet-
rar och bilder är laddade
  {
    g.drawString("Loading menu... Please wait", 12, 60);          // skriv text på skärmen
    g.drawString("Developed by M Lindström", 12, 160);// skriv text på skärmen
    g.drawString("lindstroem@altavista.net", 12, 180);   // skriv text på skärmen
    g.drawRect(30, 68, 117, 32);                                  // rita ut (ihålig) fyrkant i svart
    g.setColor(Color.gray);                                       // sätt färg till grå
    g.fillRect(32, 70, 114, 29);                                  // rita ut fylld fyrkant i grått
    g.setColor(Color.blue);                                       // sätt färg till blå
    g.fillRect(35, 72, 20, 25);                                   // rita ut fylld fyrkant i blått
    loadParameters();                                             // börja ladda parametrar, t.ex. färger, sökvägar mm
(metod-anrop)
    g.fillRect(57, 72, 20, 25);                                   // rita ut fylld fyrkant i blått
    waitForImagesOne();                                           // vänta på att gif-bilder (ikoner) hämtas över nätet
(metod-anrop)
    g.fillRect(79, 72, 20, 25);                                   // rita ut fylld fyrkant i blått
    waitForImagesTwo();                                           // vänta på att gif-bilder (ikoner) hämtas över nätet
(metod-anrop)
    g.fillRect(101, 72, 20, 25);                                  // rita ut fylld fyrkant i blått
    loadMenu();                                                   // läs in filstruktur m.h.a. objektet Reader (metod-
anrop)
    g.fillRect(123, 72, 20, 25);                  // rita ut fylld fyrkant i blått
    Loaded = true;                                                // ange att alla variabler är laddade
    repaint();                                                    // rita om gränssnittet
  }
  else                                                                    // alla parametrar och bilder är
laddade
  {
    ;                                                                     // kod för att rita upp meny
  }
 }
```

Kodavsnittet är förhoppningsvis relativt självförklarande. Värt att notera är metoden loadMenu (se nedan) i vilken objektet Reader skapas. Reader läser in strukturdatafilen och bygger upp en trädstruktur av katalog- och länkobjekt. När Reader har läst in hela strukturdatafilen returnerar det rotkatalogen till Main vilken använder denna enhet för att rita upp hela trädstrukturen. Metoden loadMenu, i klassen Main, visas nedan (Java-kod):

```
public void loadMenu()                                                  // metod som skapar objekt av typ
Reader
{
  Reader r = null;                                                      // nollställ objekt Reader
  root = null;                                                          // nollställ rotkatalog
  r = new Reader(this.getFont());                                       // skapa objekt av typ Reader
  root = r.ReadFile(this.getParameter("Menu"));          // 1) objekt Reader tar som argument sökvägen till strukturdatafilen
                                                         // 2) sökväg återfinns i HTML-parametern "Menu" (se appendix B).
                                                                        // 3) objektet Reader returnerar rotkatalogobjekt till
                                        Main
}
```

Som märkts både i analysfasen och i ovanstående kod har objektet Reader en synnerligen central funktion att fylla för att navigeringsfunktionen skall gå att förändra dynamiskt. Om Reader inte kan genomföra felfria läsningar av strukturdatafilen klickar hela lösningen. Nedan visas den viktigaste metoden i Reader vilken läser rad för rad i strukturdatafilen och skapar katalog- och länkobjekt av typen Unit (Java-kod):

```
 public void ReadRow(Unit p)                                    // metod som läser rader i strukturdatafilen och skapar
Units
 {
  try                                                                    // försök...
  {
    int i = 0;                                                  // index-variabel
    row = in.readLine();                                        // läs en rad i filen (filobjektet "in" är definierat i en
annan metod)
    while(row.substring(0, 1).equals("S") == false)         // läs till slut (S) i filen
    {
     if(row.substring(0, 1).equals("M"))                        // detta är ett katalogobjekt (M)
     {
       name = row.substring(30, 60).trim();                     // läs in enhetens namn
       open = row.substring(1, 2);                              // skall enheten vara öppen?
       owner = row.substring(10, 30).trim();                    // läs in enhetens ägare
       name.trim();                                             // ta bort eventuella blanksteg
       owner.trim();                                            // ...
       textwidth = fm.stringWidth(name);                        // bredd på text i vald font på spec. system
       if (open.equals("1"))                                    // om katalogobjekt skall vara öppet...
        child[i] = new Unit(name, true, textwidth, owner);  // skapa öppen barnnod
       else                                                     // annars...
        child[i] = new Unit(name, false, textwidth, owner); // skapa stängd barnnod
       p.setChild(child[i]);                                    // sätt rot som förälder
       ReadRow(child[i]);                                       // REKURSERA (med subenheter som argument)
     }
     else if (row.substring(0, 1).equals("F"))                  // detta är ett länkobjekt (F)
     {
       name = row.substring(30, 60).trim();                     // läs in enhetens namn
       open = row.substring(1, 2);                              // skall enheten vara öppen?
       owner = row.substring(10, 30).trim();                    // läs in enhetens ägares
       url = row.substring(60, 139).trim();                     // läs in enhetens URL
       ImageNr = new Integer(row.substring(3, 4)).intValue(); // läs in enhetens ikon
       textwidth = fm.stringWidth(name);                        // bredd på text i vald font på spec. system
       if(row.substring(2, 3).equals("F"))                      // skapa barnnod (som visas i frame)
        child[i] = new Unit(name, url, textwidth, true, ImageNr, owner);
       else                                                                   // skapa barnnod (som visas i hela
internet bläddraren)
        child[i] = new Unit(name, url, textwidth, false, ImageNr, owner);
       p.setChild(child[i]);                                    // sätt rot som förälder
      else
       {throw (new IOException());}                             // felaktig syntax i filen
     i++;                                                                   // öka på index i arrayen
     row = in.readLine();                                       // läs in ny rad
    }
  }
  catch (IOException e)                                                  // fel...
    {System.out.println("Error: " + e);}                        // visa felmeddelande
 }
```

Efter att objektet Reader har skapat katalogobjekt och länkobjekt så förstörs det. Main återtar återigen kontrollen och använder den av Reader returnerade rotkatalogen för att rita upp det grafiska gränssnittet. I nedanstående kodavsnitt visas förenklat hur trädstrukturen ritas ut av Main. Notera att första gången nedanstående metod anropas är dess argument ett objekt av typ Graphics; rotkatalogen samt X- och Y-koordinater (med värdet noll). I takt med att metoden rekurserar kommer rotkatalogen att ersättas med barnnoderna och Y-koordinaten att öka efter-som enheterna skall ritas upp under varandra. X-koordinaten är, för att hålla nedanstående exempel enkelt, oförändrat (Java-kod):

```
public int draw (Graphics g, Unit[] u, int y, int x)
{
  for(int i = 0; u[i] != null; i++)                             // sök igenom alla noder på
  {                                                                      // denna nivå
   g.drawImage(u[i].getIcon(), x, y, this);                     // rita upp ikon (katalogobjekt/länkobjekt)
   g.drawString(u[i].getName(), x+20, y+10);                    // skriv ut text (katalogobjekt/länkobjekt)
   y = y + 20;                                                  // rita nästa enhet under denna
   if (u[i].getFolder() == True)                                // om enheten är ett katalogobjekt
     if (u[i].getOpen() == True)                                // om enheten är öppen
      draw (g, u[i].getChildren(), y, x);                       // REKURSERA (med subenheter som argument)
  }
  return y;                                                              // returnera Y-värde
}
```

För att det skall vara möjligt för användare att öppna och stänga katalogobjekt eller välja länkobjekt krävs att navigeringsfunktionen kan ta emot input från användare. För att Main skall kunna kontrollera vilken enhet användaren klickade på måste varje enhet veta om sina X- och Y-koordinater. Även här används ett rekursivt förfarande där rekursion sker till dess att en enhets X- och Y-koordinater stämmer med de koordinater som systemet angivit för användarens musklick. När enheten påträffas öppnas eller stängs den vartefter den modifierade trädstrukturen återigen ritas upp (Java-kod):

```
mouse_click(int userX, int userY, Unit[] u)                    // userX och UserY är koordinater där anv. klickade...
{
 for(int i = 0; u[i] != null; i++)                             // sök igenom alla noder på denna nivå
 {
  if (u[i].getX() == userX && u[i].getY() == userY)        // om anv. klickat på enhet
  {
    if (u[i].getFolder() == true)                               // enhet är ett katalogobjekt
    {
      if (u[i].getOpen()                                                    // om enhet är öppen
        u[i].setClosed();                                                   // stäng enhet
      else                                                                  // om enhet är stängd
        u[i].setOpen();                                                     // öppna enhet
    }
    else                                                                    // enhet är ett länkobjekt
      browse(u[i].getURL());                                    // metodanrop - visa URL...
    repaint();                                                  // måla om skärmbild
  }
  else                                                                      // anv. ej klickat på enhet
    if (u[i].getFolder() == True)                               // om enhet är ett katalogobjekt
      if (u[i].getOpen() == True)                               // om katalogobjekt är öppet
        mouse_click(userX, userY, u[i].getChildren());     // REKURSERA (med subenheter som argument)
 }
}
```

Förhoppningsvis har ovanstående kodavsnitt visat de mest centrala aspekterna i användargränssnittets tekniska design. I nedanstående två kodavsnitt visas hur sökningen startar i klassen SearchForm genom att en användare klickar på knappen sök (Find Now). Som nämnts tidigare skall det vara möjligt för användare att avbryta tidskrävande sökningar varför hela sökfunktionen måste exekveras i en separat tråd. I koden nedan syns därför också hur en separat tråd startas när användaren trycker på sökknappen och hur denna tråd stoppas när användaren trycker på stop (Java-kod):

```
public void actionPerformed(ActionEvent evt)                   // metod som anropas när en händelse sker
{
 Object source = evt.getSource();                              // hämta information om var händelsen skedde
 if (source == button && button.getLabel().equals("Find Now"))// om händelsen var knappen "button" med texten "Find Now"
 {
   button.setLabel("Stop");                                     // byt label på knappen
   runner = new Thread(this);                                   // skapa separat tråd (att exekvera sökning i)
   runner.setPriority(Thread.NORM_PRIORITY);        // ange att tråden skall ha normal prioritet vid exekvering
   runner.start();                                              // starta tråd (sökning)
 }
 else if (source == button && button.getLabel().equals("Stop"))// om händelsen var knappen button med texten "Stop"
 {
   button.setLabel("Find Now");                                 // byt label på knappen
   runner.stop();                                               // stoppa tråd (sökning)
 }
}

public void run()                                              // metod som anropas när en tråd startas - runner.start()
{
 if (root.getFolder() == true)                                 // om rotobjekt är en katalog
   if (root.hasChildren())                                     // om rotobjekt har barn
     searchForChildren(root, root.getChildren());              // sök i barnnoder (se kodavsnitt nedan)
}

private Thread runner;                                                  // klassvariabel: tråd
```

```
private Button button = new Button("Find Now");          // klassvariabel: knapp
}
```

I kodavsnittet ovan syns i metoden run ett anrop till metoden searchForChildren i vilken själva sökningen sker. Som visades i figur 7 söks alla katalog- och länkobjekt i trädstrukturen igenom. Om enheten är ett länkobjekt startar en filläsning där appleten läser rad för rad i det kopplade HTML-dokumentet till dess att en matchande sträng hittas eller dokumentet tar slut. Om enheten är ett katalogobjekt kan ingen sökning ske och sökfunktionen fortsätter istället att leta vidare i objektets barn. I kodavsnittet nedan visas de två metoder som används för att 1) gå igenom alla enheter i trädstrukturen samt 2) söka igenom länkobjekts kopplade HTML-dokument (Java-kod):

```
public void searchForChildren(Unit Parent, Unit[] m)     // metod som söker igenom alla enheter i trädstrukturen
{
  for(int i = 0; m[i] != null; i++)                                  // gå igenom alla enheter på denna (träd) nivå
  {
    if (m[i].getFolder() == true)                                    // detta är ett katalogobjekt
    {
     if(m[i].hasChildren())                                          // om katalogobjekt har barn...
       searchForChildren(m[i], m[i].getChildren());          // REKURSERA (med subenheter som argument)
    }
    else                                                                          // detta är ett länkobjekt
     connectToURL(m[i]);                                             // anslut till länkobjektets URL
  }
}

public void connectToURL(Unit u)                                     // metod som ansluter till URL:er
{
  boolean notFound = true;                                           // variabel som anger om vi hittat en enhet
  try
  {
    Socket s = new Socket(domain, 80);                               // skapa en länk till HTTP server
                                                                                  // skapa output-ström till HTTP
server
    DataOutputStream os = new DataOutputStream(s.getOutputStream());
                                                                                  // skapa input-ström till HTTP ser-
ver
    BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
    os.writeBytes("GET " + u.getURL() + "\n\n");                     // be HTTP-server skicka angivet HTML-dokument
    while((row = br.readLine()) != null && notFound == true)         // så länge vi kan läsa rader och inte hittat en enhet...
    {
     if (row.toLowerCase().indexOf(searchString) >= 0)  // om sökträng är inkluderad i denna rad
     {
       ta1.addItem(u.getName());                                     // lägg till enheten i sökfönstret
       notFound = false;                                             // vi har hittat en enhet (d.v.s. stoppa loopen)
     }
    }
    os.close();                                                      // stäng IO-ström till servern
  }
  catch(Exception e)                                                              // något har gått fel...
    {System.out.println("Error: " + e);}                            // skriv ut fel
}
```

För att lägga till, uppdatera och ta bort enheter krävs som redan nämnts att appleten samarbetar med tre CGI-program på HTTP servern. Innan denna lösning implementerades skapades två enklare program för att testa att det tekniska förfarandet fungerade. I nedanstående kodavsnitt inkluderas dessa två testprogram vilka har många likheter med den slutgiltiga lösningen. I testet skulle appleten skicka över de fem bokstäverna "abcde" till CGI-programmet vilket skulle skriva ut dem på en fil. Nedan visas den Java-kod som krävdes för att skapa testappleten (Java-kod):

```
import java.applet.*;                                                         // paket: applet
import java.net.*;                                                   // paket: nätverksprogrammering
import java.io.*;                                                    // paket: filhantering

public class NetworkTest extends Applet                              // definiera klass
```

```
{
  public void init()                                          // metoden init är alltid den första som exekveras i en
applet
  {
   try                                                                    // försök...
   {
     Socket s = new Socket("www.bahnhof.se", 80);       // anslut till port 80 på server
                                                                          // skapa output-ström till HTTP
server
     DataOutputStream os = new DataOutputStream(s.getOutputStream());
                                                                          // skapa input-ström till HTTP ser-
ver
     DataInputStream is = new DataInputStream(s.getInputStream());
                                                                          // skicka data till HTTP server
     os.writeBytes("POST http://www.bahnhof.se/~mattiasl/cgi-bin/test.cgi\n\n");
     os.writeBytes("Content-type: text/plain\n");                 // formatet på informationen är "text/plain" (HTTP
servern vill veta)
     os.writeBytes("Content-length: 5\n\n");                      // informationen består av fem tecken (HTTP servern
vill veta)
     os.writeBytes("abcde");                                      // de fem bokstäverna "abcde" skickas

     while((rdata = is.readLine()) != null)                       // ta emot data från HTTP server
      System.out.println(rdata);                                  // skriv ut data från HTTP server
     is.close();                                                  // stäng input-ström
     os.close();                                                            // stäng output-ström
   }
   catch(Exception e)                                                       // om fel...
    {System.out.println("Error: " + e);}                         // skriv ut felmeddelande
  }
  private String rdata;                                                     // skyddad klassvariabel
}
```

Värt att notera i ovanstående kod är att appleten ansluter till port 80 på HTTP servern vilket är den port som används för internet/intranät trafik. Vad gäller de fyra raderna som börjar med os.writeBytes är det viktigt att formatet på informationen som skickas till servern har rätt syntax. Först anges POST, vilket kan översättas med posta, samt sökvägen till det CGI-program som skall ta emot informationen. Därefter följer två radbrytningar samt parametern "Content-type" vilken beskriver informationens format. Efter en ny radbrytning följer en parameter som anger hur många tecken som skall skickas (Content-length) vilken efter två ytterligare radbrytningar följs av informationen som skall sändas; i detta fall bokstäverna "abcde". CGI-programmet får denna information via HTTP servern. För att ta emot de fem bokstäverna och skriva ut dem på en fil skapades, som nämnts, ett enklare CGI-program. CGI-programmet skapades i programmeringsspråket C vilket förklaras av viss tidigare erfarenhet av programmering i detta språk. Nedan visas koden för CGI-programmet (C-kod):

```
#include <stdio.h>                                                // paket: standard IO
#include <stdlib.h>                                                       // paket: standard Library

int main()                                                                // funktionshuvud
{
  FILE *write;                                                    // definiera en filvariabel
  const char *str_len=getenv("CONTENT_LENGTH");   // ta emot information om längd på indata (5 tecken)
  char DATA[10];                                                  // variabel för indata
  double length;                                                  // variabel för längd på indata
  length = strtod(str_len, NULL);                                 // konvertera str_len (char) till length (double)
  for(int i = 0; i < length; i++)                                 // läsa in indata - "abcde"
    fscanf(stdin, "%c", &DATA[i]);                                // ...
  if ((write=fopen("test.txt", "w")) == NULL)                     // öppna filen "test.txt" för skrivning (write)
  {
    printf("Error when opening file.");                           // om fel – felmeddelande till applet
    return(1);
  }
  fprintf(write, "%s", DATA);                                     // skriv ut indata (abcde) på fil
  fclose(write);                                                  // stäng fil
  printf("Content-type: text/html\n\n");                  // returnera budskap i HTML-format
  printf("Task completed without errors.");                       // returnera info. till applet
  return(0);
}
```

När CGI-programmet exekverats färdigt skickar det, om allt går bra, meddelandet "Task completed without errors." till klienten vilken tar emot det och skriver ut det (se kod för applet). Skillnaden mellan de ovan beskrivna testprogrammen och de program som användes i den slutgiltiga lösningen är att i de senare skickas, istället för några bokstäver, specifik data om den enhet som skall modifieras. På servern krävdes också komplexare CGI-program eftersom de skulle kunna hitta rätt rad i strukturdatafilen innan filen modifierades (se kapitel 5.2).

Utvärdering

I denna arbetsfas skall den slutliga lösningen utvärderas. Användarbarhetsutvärdering kan göras på många olika sätt. Vanliga tillvägagångssätt är expertutvärdering, scenariobaserad utvärdering och fältstudier. Expertutvärdering - utvärdering baserad på erfarenhet - utförs utan direkt användarmedverkan. Experten nyttjar de kunskaper om användarens arbetssituation, bakgrund, kunskaper etc. som erhållits från analyserna i tidigare skeden. Scenariobaserad utvärdering passar utmärkt vid nyutveckling och ihop med en iterativ prototyping metodik. De tilltänkta användarna får ett antal scenarier att utföra med hjälp av systemet vartefter experter studerar hur användarna utför arbetsuppgifterna. Fältstudier, eller observationsintervjuer, utförs på system som redan är satta i drift och bedrivs hos användaren i dennes dagliga arbetsmiljö. (Sandbäck & Göransson, 1999-03-08)

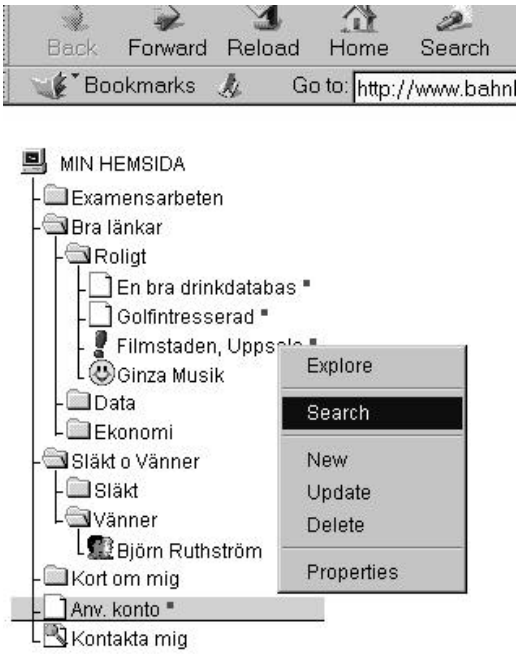
Då det inte funnits tid för användning av någon mer komplex utvärderingsmetod har expertutvärdering använts. Då det är svårt att ha någon djupare kunskap om slutanvändarna eller deras systemmiljö kommer navigeringsfunktionen i huvudsak att utvärderas gentemot kraven från analysfasen. Kraven var i korthet att det skulle vara möjligt att strukturera information i expanderbara katalog- och länkobjekt; att det skulle gå att söka information på webbplatsen; att det skulle vara möjligt för användare att lägga till, uppdatera och ta bort katalog- och länkobjekt; att det skulle vara möjligt att få allmän information om katalog- och länkobjekt; att det skulle vara möjligt att skydda ömtåliga katalog- och länkobjekt samt, avslutningsvis, att lösningen skulle vara stabil, grafiskt attraktiv och enkel att använda.

Relativt tidigt fastställdes att kraven skulle mötas m.h.a. ett gränssnitt som liknar filhanteraren i Windows. Även sökfunktionen skulle baseras på ett gränssnitt som känns familjärt för användare av grafiska operativsystem. Det underströks även att väntetider och användarosäkerhet skulle minimeras vilket bl.a. ställde krav på att en statusladdare visades medan appleten läste in parametrar och ikoner. I föregående avsnitt (Kodning) visades hur statusladdaren implementerades rent programmeringstekniskt. I figur 8 (nedan) visas hur statusladdaren ser ut i praktiken.

Figur 8: Statusladdare som visas medan parametrar och ikoner hämtas över nätet.

Efter att appleten har laddats visas navigeringsfunktionen vilken, enligt fastställda krav, skulle likna filhanteraren i Windows. I figur 9 (nedan) visas appletens huvudsakliga användargränssnitt vilket har tydliga likheter med det eftersträvade gränssnittet. I appleten kan information ordnas i expanderbara katalogobjekt precis som i förebilden. När en användare vänsterklickar på ett länkobjekt markeras det med blå färg samtidigt som kopplad HTML-sida visas i internet bläddraren. Om en användare vänsterklickar på ett katalogobjekt öppnas eller stängs det beroende på initial status. Om en användare högerklickar på ett katalog- eller länkobjekt visas en popup-meny med valen utforska (Explore); söka (Search); ny enhet (New); uppdatera enhet (Update); ta bort enhet (Delete) och egenskaper (Properties).

Alla valen i popup-menyn är tillämpliga för både katalogobjekt och länkobjekt. Efterföljande dialoger kan dock variera något eftersom inte alla gränssnittskomponenter är tillämpliga i alla situationer. Vid gränssnittsdesign brukar man rekommendera att användare bara skall kunna manipulera de gränssnittskomponenter som är relevanta i den specifika situationen medan resterande komponenter bör vara otillgängliga. (Preece, 1994) I linje med detta kan t.ex. en användare inte uppdatera ett katalogobjekts URL eller byta dess ikon eftersom det bara går att ändra dessa egenskaper på länkobjekt.



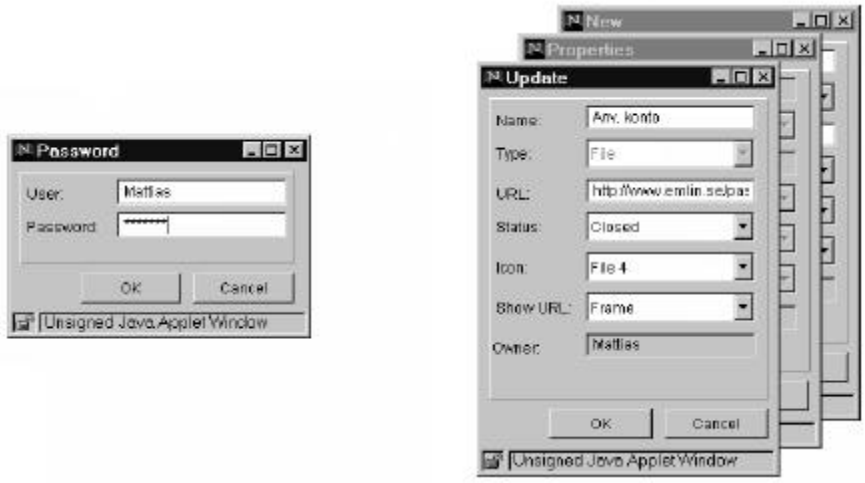
Figur 9: Den färdiga navigeringsfunktionen

För att användare skall veta vilka URL:er de har besökt markeras besöka länkobjekt med en röd fyrkant bakom namnet. I och med att både aktuellt länkobjekt och tidigare besökta länkobjekt markeras vet användare alltid var de är samt vart de varit. Denna kunskap är viktig då användare på bra webbplatser alltid skall kunna svara på de tre frågorna: Var är jag? Var har jag varit? Vart kan jag gå? (Mountford, 1999-03-09) I aktuell lösning kan katalogobjekt bara ha två ikoner medan användare kan välja mellan upp till sex ikoner för länkobjekt. Denna valfrihet för länkobjekt har ansetts viktig då användare genom ikonerna får ledtrådar till vilken typ av dokument länkobjekten är kopplade till. Ikonerna är transparenta vilket möjliggör att navigeringsfunktionens bakgrundsfärg kan ändras utan att ikonerna behöver ändras.

För att lägga till, uppdatera eller ta bort enheter måste användare högerklicka på enheter vart-

efter de i en popup-meny väljer eftersträvat alternativ. När användare har valt alternativ får de upp ett lösenordsfönster där användarnamn och lösenord fylls i. Förutsatt att användaren är behörig är grundregeln att bara ägare till enheter kan ta bort eller uppdatera dem. Det är dock möjligt för användare att lägga till enheter i kataloger som skapats av andra användare[10]. Denna rättighet kan skapa problem i de fall en katalogägare bestämmer sig för att ta bort "sin katalog" med tillhörande subenheter som skapats av andra användare. I nuvarande (enkla) version av navigeringsfunktionen verkar den bästa lösningen på problemet vara att systemadministratören skapar grundstrukturen i navigeringsfunktionen och garanterar vissa kritiska katalogobjekt.

Förfarandet för att lägga till, alternativt uppdatera enheter, är i stort sett detsamma. Om en användare t.ex. vill uppdatera en enhet högerklickar han på denna enhet och väljer alternativet uppdatera (Update) i popup-menyn. Med hjälp av ett lösenordsfönster kontrolleras sedan att användaren är behörig. Om inmatade användaruppgifter är korrekta visas fönstret uppdatera (Update). I detta fönster återges alla egenskaper för den valda enheten vilka sedan kan modifieras av användaren. Om användaren t.ex. ändrar enhetens namn och trycker på knappen ok så uppdateras enheten i enlighet med önskemål. I figur 10 (nedan) visas lösenordsfönstret och fönstret för att uppdatera en enhet.



Figur 10: Fönster för användarregistrering (vänster) samt fönster för att uppdatera enheter (höger).

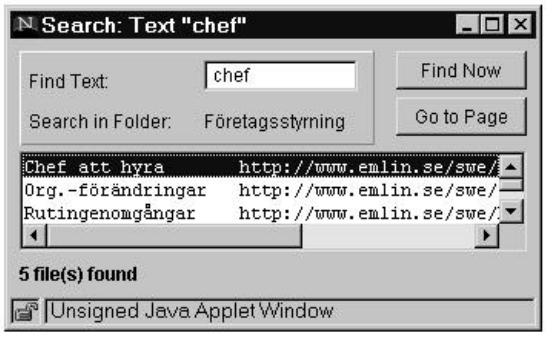
Klassen PropertiesForm som används för att skapa fönstret uppdatera (Update) används också för att skapa fönstrena ny enhet (New) och egenskaper (Properties). I figur 10 (ovan) syns dessa fönster delvis skymda bakom fönstret för att uppdatera en enhet. Skillnaden mellan de tre fönstrena är att de ingående textkomponenterna har olika status beroende på i vilket läge fönstret visas. När fönstret används för att lägga till en enhet är fönstret tomt så att användaren kan fylla i ny information för den nya enheten; när fönstret används för att se egenskaper för en vald enhet behöver inte användaren vara behörig men han kan i gengäld inte heller förändra någon data, d.v.s. alla textkomponenter är låsta; när fönstret används för att uppdatera en enhet återges den valda enhetens egenskaper vilka också kan modifieras av användaren.

En irriterande detalj i alla fristående fönster som visas i applets är texten "Unsigned Java Applet Window" i fönstrets nederkant. Denna text är tyvärr omöjlig att tabort för en

[10] Om det inte vore tillåtet för användare att lägga till enheter i kataloger som de själva inte är ägare till kan navigeringsfunktionen i praktiken bara uppdateras av en användare; ägaren av rotnoden.

programmerare. Skälet till textens existens är för att minska risken att elaka applets använder fönster som härmar interna applikationer och lurar användare att fylla i, och skicka iväg, känslig information. (Cornell & Horstmann, 1997)

Den sista viktiga funktionen i navigeringsfunktionen är sökning. Sökfunktionen nås liksom kommandona för att modifiera enheter, via högerklick och därtill kopplad popup-meny. Om användaren i popup-menyn väljer alternativet sök (Search) visas en speciell sökdialog. När användaren har fyllt i sin söksträng söker sökfunktionen igenom alla underenheter till den enhet som användaren högerklickade på. I kravfasen fastställdes att sökfönstret skulle ha ett enkelt och intuitivt gränssnitt. I figur 11 (nedan) visas detta gränssnitt.



Figur 11: Sökfönster

De huvudsakliga elementen i sökdialogen är ett textfält där eftersträvad söksträng fylls i, en sökknapp (Find Now) samt en textarea där de länkobjekt som matchade sökkriteriet skall visas. Om användaren dubberklickar på funna länkobjekt visas kopplad HTML-sida i framen. Samma sak händer också om användare markerar ett länkobjekt och trycker på knappen gå-till-sida (Go to Page). Som berättats tidigare finns möjlighet att avbryta tidskrävande sökningar genom att sökknappen (Find Now) – efter att användaren tryckt på den – får texten stop. Om användaren trycker på knappen i detta läge avbryts sökningen genom att tråden som exekverar sökfunktionen dödas. För att undvika användarosäkerhet vid sökning förvandlas markören till ett timglas samtidigt som statusfältet i dialogens nederkant upplyser användaren om vilken URL som sökfunktionen för närvarande går igenom.

En stor brist i den färdiga sökfunktionen är att den bara kan söka igenom dokument på samma HTTP server som den (appleten) själv hämtades från. Detta innebär att sökfunktionen inte klarar av att söka igenom externa länkar, d.v.s. dokument på externa HTTP servers. I installationen på www.emlin.se är denna brist uppenbar eftersom nästan alla länkar är externa. Skälet till denna brist i sökfunktionen är, återigen, säkerhetshänsyn. (Cornell & Horstmann, 1997) Om än allvarlig är ovanstående brist förmodligen mer allvarlig på internet än på intranät, där de flesta dokumenten är placerade på den interna HTTP maskinen.

Slutdiskussion

Sammanfattningsvis har de krav som ställdes på lösningen i kravfasen implementerats framgångsrikt. Lösningen har, via internet, testats på Windows NT 4.0 och Windows –95 med Internet Explorer 4.0 samt Netscape Communicator 4.04 med uppdaterad Java-tolk. I testen har en dator med Pentium Pro, 180 Mhz och 48 MB RAM använts. I alla testscenarior med

ovanstående konfiguration har navigeringsfunktionen fungerat utan problem. Enklare test har även företagits på operativsystemen Även om jag bedömer att de flesta intranät använder PC-klienter vore det naturligtvis intressant att även testa programvaran på Macintosh- och UNIX-klienter.

Den framtagna lösningen bör kunna vara intressant för mindre och medelstora företag som vill implementera intranät men saknar nödvändiga resurser för ett centraliserat arbetssätt. Med navigeringsfunktionen kan webb avdelningen frigöras till andra göromål samtidigt som organisationen inte behöver bygga upp eller köpa in avancerad kunskap. Innan navigeringsfunktionen implementeras är det dock viktigt att organisationen analyseras och att användarna uttrycker välvilja för det arbetssätt som krävs. Om användarna inte gillar arbetssättet eller saknar nödvändig kunskap för att t.ex. publicera HTML-dokument bör inte heller navigeringsfunktionen användas. Förutom på intranät är navigeringsfunktionen förmodligen även intressant som basen för projektarbete på distans. Till exempel skulle forskare i olika världsdelar, via internet, snabbt kunna göra sitt material tillgänligt för övriga forskare inom projektgruppen.

Gränssnittsmetaforen bör efter lite, eller ingen träning vara enkelt att förstå för det stora flertalet användare. Vissa användare kan dock ha svårigheter med att popup-menyn är kopplad till höger musknapp. För att åtgärda dylika problem kan en hjälpsida skapas där det finns anvisningar för hur navigeringsfunktionen skall användas. Det största problemet med navigeringsfunktionen är att den kräver viss användardisciplin. Tänkbara problem är att användare 1) endast lägger till objekt och aldrig tar bort dem samt 2) lägger till katalogobjekt på ett ologiskt sätt vilket skapar förvirring för övriga användare. Det första problemet kan möjligen åtgärdas genom att systemansvarig eller liknande med jämna mellanrum går igenom webbplatsen och påminner användare om att ta bort gammal information. För att åtgärda det andra problemet byggs lämpligen en permanent grundstruktur vilken inte kan modifieras av slutanvändarna. Om detta inte fungerar måste kanske navigeringsfunktionen vidareutvecklas så att bara ett fåtal användare har rättigheter att lägga till både katalog- och länkobjekt medan det stora flertalet användare bara kan lägga till länkobjekt.

Avslutningsvis anser jag att syftet med arbetet är uppfyllt. Navigeringsfunktionen fungerar bra även om den omgivande systemmiljön måste analyseras innan den kan installeras skarpt. Gränssnittsmetaforen har tydliga likheter med filhanteraren i Windows samtidigt som lösningen är såpass flexibel att metaforen utan svårigheter kan ändras. Uppslag inför framtiden är främst att filerna i lösningen ersätts med en databas. På så sätt kan lösningen via SQL integreras med teknologi som Microsofts Active Server Pages (ASP) och Lotus Domino. Sist vill jag tacka läsaren för visat intresse samtidigt som jag hoppas att diskussionen och det färdiga resultatet varit tankeväckande. Tack!

Källförteckning

Litteratur och artiklar

Budd, T, *Classic Data Structures In C++*, Addison-Wesley, 1991

Cornell, G. & Horstmann, C, *Core Java*, Second edition, SunSoft Press, 1997

Ek, J. *Java-programmering, En introduktion.* Pagina Förlag AB, 1996

Geijer, E, *Håll Web-sidorna enkla, överdesigna inte*, Computer Sweden, Nr 37, 1996

IDG News, *Webbavdelningen förlorar sin särställning*, Computer Sweden, nr 23, 1999

Lim, E & Paynter, J, *Design Considerations for Web Site Navigation*, Department of Management Science and Information Systems, University of Auckland, 1998

Noren, K-J, *Är din sida bra?*, Navigera@internet, Nr 4-5 Maj, Pagina Förlag, 1998

Preece, J, (medförfattare: Rogers, Y, Sharp, H, Benyon, D, Holland, S, Carey, T), *Human-Computer Interaction*, Addison-Wesley, 1994

Pressman, R, *Software Engineering – A Practitioner's Approach*, Third Edition, McGraw Hill, 1994

Spool J, Scanlon, T, Schroeder W, Snyder C, DeAngelo T, *Web Site Usability – A Designer's Guide*, User Interface Engineering, 1997

Walsh, A, *Java for Dummies*, Second Edition, IDG Books Worldwide, 1997

Elektroniska källor

Blume, C, *Internet - hur funkar det, vad skall man välja?*
http://www.fyrbodal.bengtsfors.se/text_l_is_bfs.htm, 1999-03-09

Common Gateway Interface, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html, 1999-03-08

Dagens Industris webbplats, http://www.di.se/NoFlash.asp, 1999-03-03

JDK 1.1 Documentation, http://www.javasoft.com/products/jdk/1.1/index.html , 1999-03-10

Mountford, J, *Web Interfaces Live: What's Hot, What's Not?*,
http://www.acm.org/sigchi/chi97/proceedings/panel/jn.htm, 1999-03-09

Nation, D, Plaisant, C, Marchionini, G, Komlodi, A, *Visualizing websites using a hierarchical table of contents browser: WebTOC*, http://www.uswest.com/web-conference/proceedings/nation.html, 1999-02-20

Nielsen, J, *Top Ten Mistakes in Web Design*, http://www.useit.com/alertbox/9605.html, 1999-02-15

Sandbäck, T & Göransson B, *Användargränssnitt kräver bra design*, http://www.hci.uu.se/~bengt/design.html, 1999-03-08

100 Percent Pure Java Program, http://java.sun.com/marketing/collateral/pure_brochure.html, 1999-02-10

Appendix A: Ordlista

Applet - Applet är ett Java-program som exekveras via Intranät eller Internet. Applets har ofta ytterst begränsad åtkomst till klientens hårddisk och primärminne.

CGI - CGI står för Common Gateway Interface och är en standard för att nå externa applikationer via t.ex. Web-servers. CGI är ett vanligt sätt att nå databaser, filer och annan information från WWW.

FTP – FTP står för File Transfer Protocol och är ett protokoll som används för att överföra filer mellan olika datorer på olika operativsystem.

HTML - HTML står för HyperText Markup Language och är det definitionsspråk som ligger bakom de hemsidor vi möter på WWW. Applets anropas från HTML-kod med syntaxen: *<APPLET CODE=x.class WIDTH=200>* där "x.class" är den färdigkompilerade bytekoden och 200 anger hur stort utrymme Appleten skall visas i.

HTTP server - HTTP (Web) servern är den dator i organisationens nätverk som lagrar HTML-dokument, bilder, ljud m.m.. Internet bläddrare anropar sedan HTTP servern (vanligen på port 80) och begär att få tillgång till specificerat material.

Internet bläddrare - Med internet bläddrare menas ett verktyg för att titta på hemsidor som ligger på WWW. Vanliga bläddrare idag är Netscape Communicator och Internet Explorer.

Intranät och Internet - Med Intranät och Internet menas nätverk som är baserade på protokollet TCP-IP. Skillnaden mellan Intranät och Internet är att Internet är ett globalt nätverk medan Intranät är ett lokalt nätverk. Inget hindrar dock att ett Intranät, via en brandvägg, står i förbindelse med Internet.

JDK - JDK står för Java Development Kit och är Suns rudimentära utvecklingsmiljö för Java. JDK har en speciell status i Java-världen då det sätter den standard vilken andra aktörer följer. Sålunda implementerar JDK all funktionalitet i programmeringsspråket Java utan några tillägg eller bortdragningar.

Java Virtual Machine - Java Virtual Machine är den kompilator som omvandlar bytekod till maskinkod. Man pratar ofta om att Java-program kan exekveras överallt där en JVM finns installerad. Med detta menas att Java-program kan exekveras överallt där det finns en kompilator som kan översätta bytekoden till maskinkod för det specifika operativsystemet.

SQL - SQL står för Structured Query Language och är ett logiskt databasspråk med vilket man ställer frågor och utför uppdateringar av databaser.

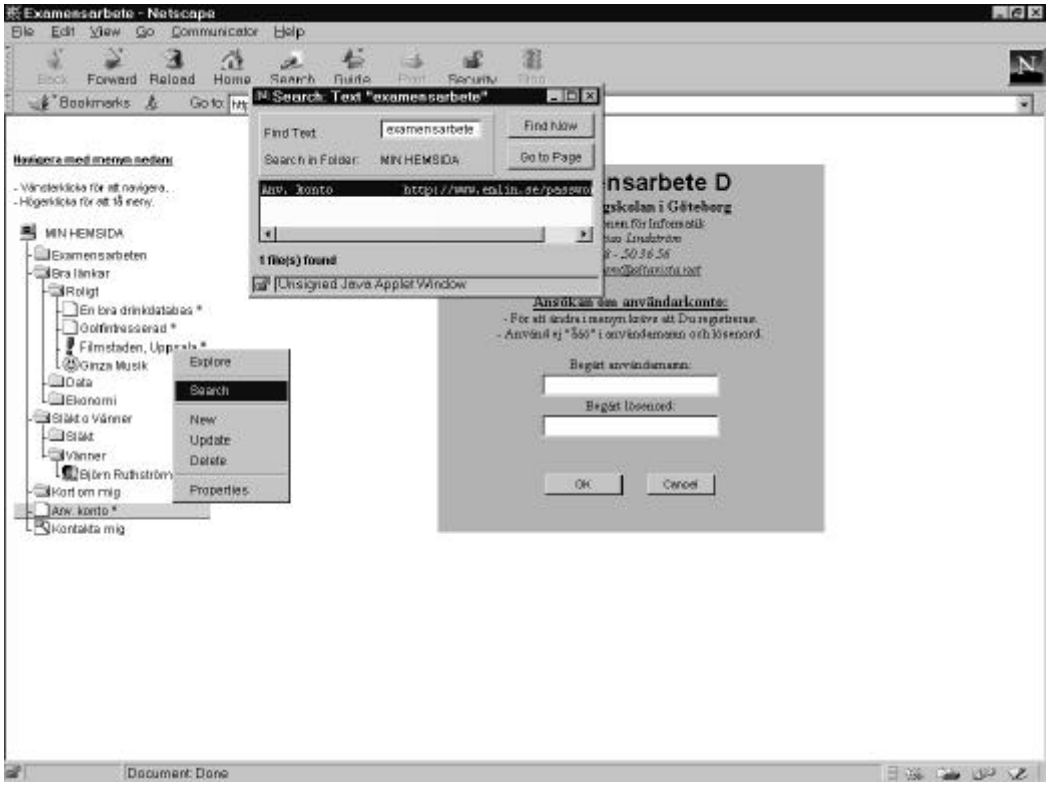
URL - URL står för Uniform Resource Locator och är det sätt på vilket man anger Internetadresser på WWW.

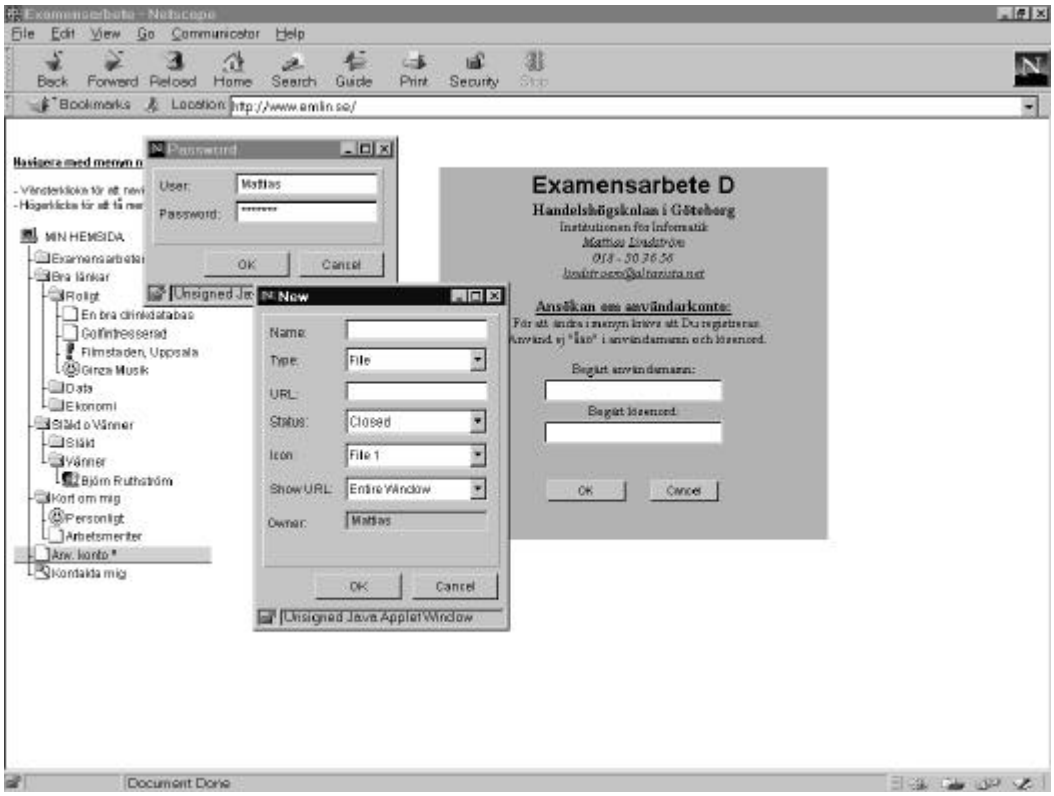
WWW - WWW står för World Wide Web och är den multimediala delen av Internet där information framställs m.h.a. HTML.

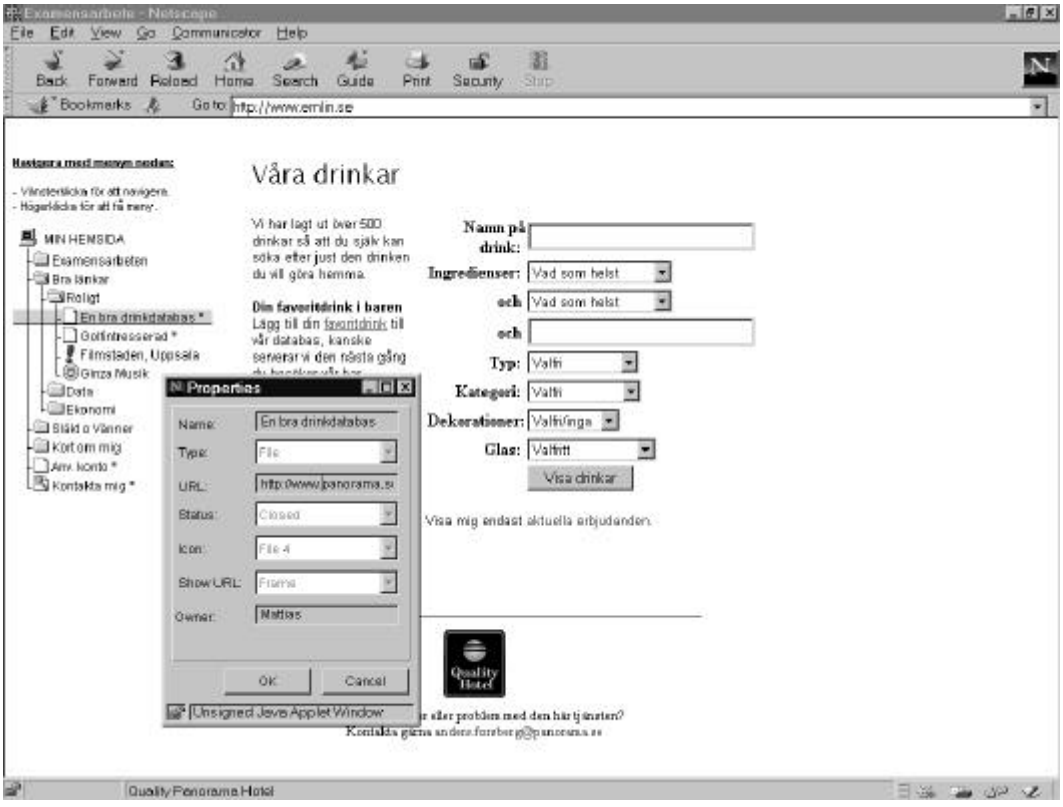
Appendix B: HTML-kod

```
<html>
<head>
</head>
<body>
<applet code="main.class" width="200" height="450"> <!—- Visa applet på 200x450 pixels -->
  <param name="root" value="computer.gif">                    <!—- Sökväg till ikon för rotenheten -->
  <param name="folder_open" value="mapp1.gif">      <!—- Sökväg till ikon för öppen katalog -->
  <param name="folder_closed" value="mapp2.gif">    <!—- Sökväg till ikon för stängd katalog -->
  <param name="file1" value="file1.gif">                      <!—- Sökväg till ikon för fil 1 -->
  <param name="file2" value="file2.gif">                      <!—- Sökväg till ikon för fil 2 -->
  <param name="file3" value="file3.gif">                      <!—- Sökväg till ikon för fil 3 -->
  <param name="file4" value="file4.gif">                      <!—- Sökväg till ikon för fil 4 -->
  <param name="file5" value="file5.gif">                      <!—- Sökväg till ikon för fil 5 -->
  <param name="file6" value="file6.gif">                      <!—- Sökväg till ikon för fil 6 -->
  <param name="bgcolor" value="255,255,255">        <!—- Bakgrundsfärg, röd-grön-blå (RGB-skala) -->
  <param name="textcolor" value="0,0,0">                      <!—- Textfärg, röd-grön-blå (RGB-skala) -->
  <param name="right_frame" value="right">                    <!—- Namn på höger frame. Denna variabel behövs
                                                    för att applet  skall kunna bläddra dokument i motstående frame-->
  <param name="domain" value="www.emlin.se">                  <!—- Domännamn. Denna variabel behövs för att applet skall kun-
                                                    na ansluta sig till HTTP server -->
  <param name="Menu" value="Swedish.txt">                     <!—- Sökväg till fil med strukturdata -->
  <param name="users" value="securityRegister.txt">  <!—- Sökväg till fil med användardata -->
</applet>                                                                   <!—- Slut på applet -->
</body>
</html>
```

Appendix C: Skärmbilder




Examensarbete - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://www.emlin.se/
Password
User: Mattias
Password:
OK Cancel
New
Name:
Type: File
URL:
Status: Closed
Icon: File 1
Show URL: Entire Window
Owner: Mattias
OK Cancel
Unsigned Java Applet Window
MIN HEMSIDA
Examensarbeten
Bra länkar
Roligt
En bra drinkdatabas
Golfintresserad
Filmstaden, Uppsala
Ginza Musik
Data
Ekonomi
Släkt o Vänner
Släkt
Vänner
Björn Ruthström
Kort om mig
Personligt
Arbetsmeriter
Anv. konto *
Kontakta mig
Examensarbete D
Handelshögskolan i Göteborg
Institutionen för Informatik
Mattias Lindström
018 - 30 36 56
lindstrom@alumni.net
Ansökan om användarkonto:
För att ändra i menyn krävs att Du registrerar.
Använd ej "Åke" i användarnamn och lösenord.
Begärt användarnamn:
Begärt lösenord:
OK Cancel
Document Done


Examensarbete - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Go to: http://www.emlin.se
Hastigera med menyn nedan:
- Vänsterklicka för att navigera.
- Högerklicka för att få meny.
MIN HEMSIDA
Examensarbeten
Bra länkar
Roligt
En bra drinkdatabas *
Golfintresserad *
Filmstaden, Uppsala
Ginza Musik
Data
Ekonomi
Släkt o Vänner
Kort om mig
Anv. konto *
Kontakta mig *
Properties
Name: En bra drinkdatabas
Type: File
URL: http://www.panorama.se
Status: Closed
Icon: File 4
Show URL: Frame
Owner: Mattias
OK Cancel
Unsigned Java Applet Window
Våra drinkar
Vi har lagt ut över 500 drinkar så att du själv kan söka efter just den drinken du vill göra hemma.
Din favoritdrink i baren
Lägg till din favoritdrink till vår databas, kanske serverar vi den nästa gång
Namn på drink:
Ingredienser: Vad som helst
och Vad som helst
och
Typ: Valfri
Kategori: Valfri
Dekorationer: Valfri/inga
Glas: Valfritt
Visa drinkar
Visa mig endast aktuella erbjudanden.
Quality Hotel
Kontakta gärna anders.forsberg@panorama.se
Quality Panorama Hotel