Datavetenskap

Patrik Jansson

Integrerat publiceringsverktyg för webben

Examensarbete, C-nivå

2002:09

Integrerat publiceringsverktyg för webben

Patrik Jansson

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Patrik Jansson

Godkänd, 2002-06-04

Handledare: Tim Heyer

Examinator: Tim Heyer

Sammanfattning

Denna artikel behandlar ett sätt att bygga upp ett system som separerar design från information på en sorts webbsida. Detta tillåter att informationen läggs i en trädstruktur, liknande den som finns i ett vanligt filsystem, och tilldelas en mängd nyckelord. Vidare ger det möjligheten att dessa webbsidor kan innehålla trädmenyer och även menyer baserade på de nyckelord som delas av informationerna.

Integrated publishing tool for web pages

Abstract

This article describes how to create a system, which separate design from the information on a specific kind of web page. This allows the information to be put into a tree structure, and assigned a number of keywords. Further, this allows the web pages to contain a tree menu and also a menu based upon those keywords shared among other pages.

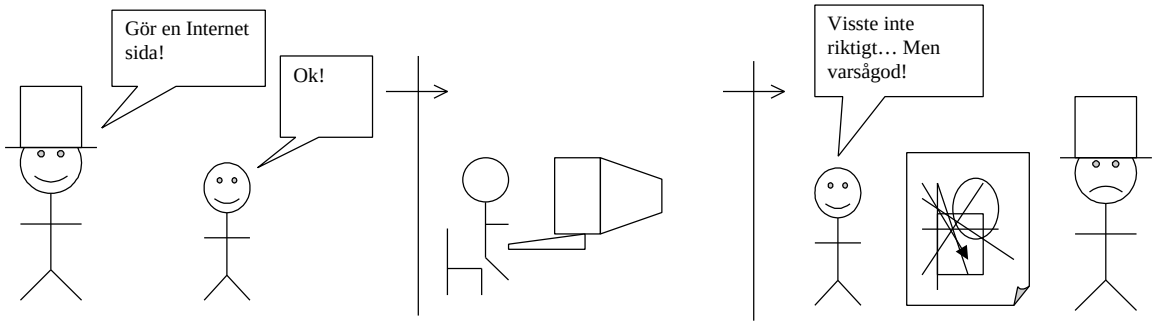
Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Inledning

Internet har idag blivit en del av vår kultur, och att inte finnas med på webben har börjat ses lika illa som att inte har någon kontakttelefon eller någon besöksadress. Företagen får en möjlighet att föra en dialog med sina kunder, och kunderna kan lätt få den information de söker om ett företag.

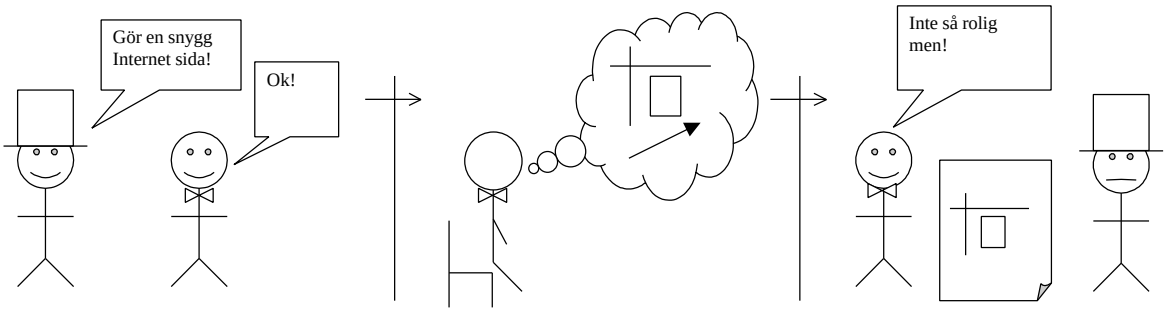


Figur 1: Så här kunde det gå till förr

När Internet blev känt utanför universitetsvärlden så började företagen få upp ögonen för vad som kunde göras här. De kunde nå sina kunder som aldrig förr. Det var inte ovanligt att företagen anställde en eller ett par unga killar för att göra deras hemsida[1], eftersom väldigt få experter då fanns på området. Men tyvärr brukade oftast dessa unga killar inte ha speciellt mycket känsla för vare sig sidans utseende eller dess interna upplägg. Därför blev sidorna ganska dåliga, och företagets kunder som nådde sidan tappade lätt intresset, detta illustreras i Figur 1.

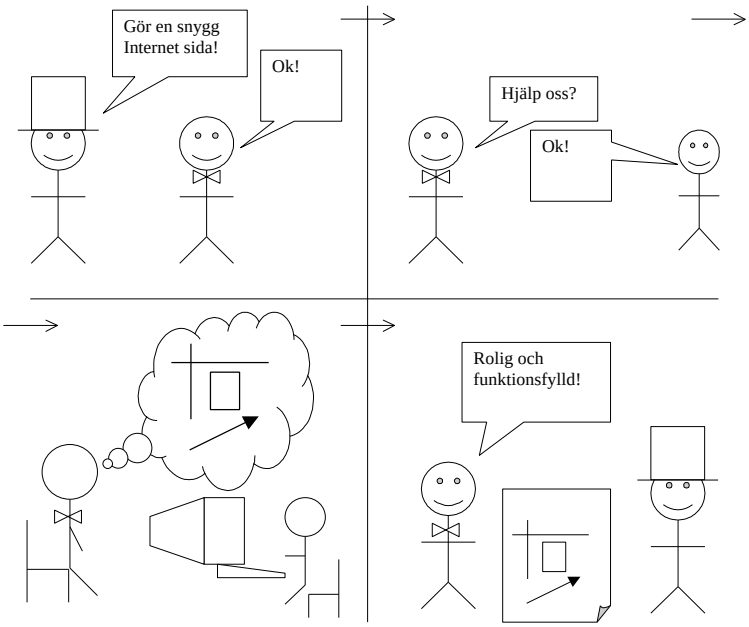
Då dröjde inte länge förrän en ny avdelning på reklambyråerna dök upp, s.k. webbyråer. Här fanns expertisen redan för att skapa tilltalande sidor. Nu gick byrån bara över till att göra sidorna på Internet. Vanligtvis har en reklambyrå inte någon programmerare som kan göra avancerade sidor, så när byråns kunder ville ha såna lösningar, så kunde de inte leverera sidan, Figur 2. Så då blev reklambyrån i sin tur åter tvungen att anlita den unga killen, som nu blivit anställd i ett programmeringsföretag. Kommunikationen mellan reklambyråer och programmeringsföretagen fungerar inte alltid bra, och reklambyrån får inte alltid det resultat som de önskar sig.

[1] Hemsida och sida kommer att användas om vartannat för att beskriva vad som vanligtvis beskrivs som sajt, internetsida, eller en hel webbplats.



Figur 2: Så här blir det nu när det ska se bra ut

Företagen som nu vill ha en trevlig sida[1] på nätet, söker alltså upp en reklambyrå och ber dom göra en sida som passar företagets profil, Figur 2. Ganska ofta så dyker det upp önskemål om att företaget själv ska kunna gå in och ändra på sidornas innehåll, men utan risk att förstöra designen. Det är då de mer avancerade lösningarna dyker upp, som här för att kunna separera designen från innehållet på sidan. Figur 3 försöker visa ett sätt för att få med även mer avancerade funktioner.



Figur 3: Ett sätt för att göra det roligt och snyggt

En ofta vald lösning är att lagra informationen i en databas och sedan länka in informationen på sidan med en mängd funktioner. Oftast har inte vanliga reklambyråer expertisen som krävs för att skapa funktionaliteten. Vad byrån då brukar göra är att köpa in tjänsten från ett externt programmeringsföretag. Ett problem med det är att

programmeringsföretagen inte är speciellt snabba eller säkra på att leverera tjänsten. Reklambyrån äger oftast inte heller den levererade tjänsten, utan det är antingen deras kund, alltså något annat företag, eller programmeringsföretaget som blir ägare till den här tjänsten. Så när nästa kund behöver en liknande sida, så måste samma process göras om, en gång till. Istället är det då vanligt att kunden nöjer sig med en enklare sida, där kunden inte på ett bra och enkelt sätt själv kan ändra sidan utan att riskera att förstöra designen.

I framtiden vill reklambyrån kunna leverera en skapligt avancerad sida med skiljd design och text. Byrån vill ha en färdig lösning, där det redan finns en databas och en uppsättning funktioner färdiga att använda direkt. Då kan byrån snabbt komma igång med sidan och istället koncentrera sig på det som reklambyråer egentligen är bra på, att göra snygga sidor.

1.1 Inledande samtal med en reklambyrå

För att få ett hum om vad ett sådant system skulle kunna göra fördes några samtal med sakkunniga på en reklambyrå. Följande är en sammanställning av önskemålen som dök upp under samtalen.

Systemet som ska skapas ska ha sin grundidé i att det ska vara enkelt att skilja på designen och informationen på en webbsida. Reklambyrån tycker om sättet en vanlig tidning är upplagd; med artiklar som har huvudrubrik, ingress och en artikelkropp. Dessutom tycker reklambyrån att det borde gå att organisera sin webbsida i någon form av menysystem.

Figur 4: Exempel på hur termerna huvudrubrik, ingress och artikelkropp används.

Reklambyrån uttryckte önskemål om att det vore bra om de kunde stänga av och sätta på artiklar och rubriker i menysystemet. Dessutom ville byrån att när de stängde av en rubrik så stängdes samtidigt alla underliggande rubriker av, så att surfaren inte längre kunde nå dessa. Åtminstone på något sätt få välja vilka rubrikerna som ska vara osynliga för den vanliga Internetsurfaren.

Byrån sa att de vore bra om de kunde ha flera nivåer av underrubriker under huvudrubrikerna. En analogi är att tänka på hur tidningen vanligtvis är upplagd, där det t.ex. finns en sportsektion och i sportsektionen finns det resultatsidor, och är i finns information om olika sporter och resultat där ifrån.

Ett önskemål var att en artikel kan ha några nyckelord, om t.ex. det var en viss sport eller ett resultat, och att de andra artiklarna som har ungefär samma nyckelord listas i en ruta någonstans. Dessutom önskade reklambyrån att kunna ordna artiklarna i valfri ordning på en sida, t.ex. att fotboll alltid kommer överst, och att hästsporterna alltid kommer nederst.

Helst, sa reklambyrån, ska det finnas funktioner som gör det allra mesta, ritar upp en meny med alla rubriker, och kanske någon underrubrik, så de slipper ens skriva en sådan liten grej själva.

De vill använda redan existerande webbservrar, vilka är bland annat Windows 2000-servrar. Systemet ska fungera även på populära webbplatser, så det borde kunna betjäna minst 1 000 000 besökare varje dag, annars är det svårt att motivera användningen av det.

Reklambyrån hade länge haft problem med att outbildade programmerare skriver mjukvaror som inte fungerar på alla byråns datorer, speciellt inte i Machintosh-datorerna. Därför är det viktigt att det här systemet fungerar på alla deras datorer. Det är okej att administrera systemet genom ett webbgränssnitt, sades det.

1.2 Mål

Menysystemet ska fungera som ett träd. Varje artikel eller rubrik i trädet ska kunna ha en text som beskriver vad som finns i den, eller kanske en kort summering av olika rekommenderade artiklar.

Administratören ska kunna stänga av en nod, då kommer inte den rubriken med alla dess underrubriker och artiklar att visas längre.

Artiklarna ska ha olika nyckelord. När Internetsurfaren visar en artikel så ska han eller hon få en lista av de andra artiklarna som har flest gemensamma nyckelord med den nuvarande artikeln. Funktionsbiblioteken som ska göras består alltså av minst två olika delar.

Den första delen av för inmatning av menyer och artiklar i någon databas. Denna del ska kunna nås från en vanlig bläddrare[2]. Sidan ska ligga på en Windows2000-server, som kör IIS 5.0 eller senare för stöd för ASP v3.0. Denna del kan komma att delas upp ytterligare så att det blir en för att skapa menyer och välja var artiklarna ska hamna och en för att skicka in nya artiklar i systemet och redigera trädstrukturen. Istället för brödtext i en artikel ska den kunna att ha en bild.

Den andra delen består i att skapa ett funktionsbibliotek för att göra det möjligt för webbdesignern att komma åt menyhuvuden, artikelrubriker, ingress och brödtext. Biblioteket ska innehålla funktioner som gör nästan allt, t.ex. skriver ut en meny i html. Det ska även finnas funktioner som är så finkorniga att de ger en rad i menyn i ren text.

1.3 Förväntat resultat

Det som ska ha skapats är alltså:

1. En datamodell för artiklarna och menyerna, databasen ska ligga på en Windows 2000-server.
2. Ett gränssnitt som fungerar i en vanlig bläddrare mot databasen som gör det möjligt att administrera menysystemet. Det ska även finnas ett gränssnitt mot databasen som gör det möjligt att lägga till artiklarna, alternativt sitter dessa två gränssnitt ihop.
    a. Gränssnittet ska vara skrivet i HTML/ASP.
3. Ett bibliotek av funktioner åt webbdesignern. Sidan kommer att ligga på samma dator som databasen. Funktionerna som behövs är att hämta menyer, artiklar, samt de artiklar som är relaterade till nuvarande artikel, baserat utifrån en mängd nyckelord.
4. Till sist en exempelsida som demonstrerar systemets funktioner.

[2] Termen bläddrare avser här program så som Internet Explorer och Netscape.

1.4 Läsanvisningar

Hela den här rapporten kan läsas från första sidan till sista utan att tappa något. Inledningen skapar en förståelse för hela uppgiften, och i bilaga A, finns en kort introduktion till ASP. Det förekommer inga lösningar i själva artikeln, utan det är i först i bilaga B som en listning av alla funktioner dyker upp och en förklaring på vad dessa gör.
Rapporten är skriven så att den ska vara lätt ska följa med i utvecklingen av systemet. Ibland hoppar den fram och tillbaka mellan olika stadier, men det är främst för att få en övergripande inblick i hur de olika delarna utvecklades. Kod, i den grad det finns, har skrivits som följande text:

```
Exempel på formateringen av programkoden
```

Överlag är det författaren själv som skapat alla bilder, och de kommer att användas flitigt för att skapa ett tydligare sammanhang för hur saker fungerar ihop.

I det första kapitlet ges en överblick på problemet, och vad som ska göras. De mål som sätts upp specificeras ytterligare i förväntat resultat. Det andra kapitlet sammanställer alla krav för systemet och är alltså det som stod till grund för systemet. Innan det tredje kapitlet läses behövs lite kunskaper om ASP och kraven för systemet, och det sakerna återfinns i bilaga A respektive åtminstone 1.1 samt 1.3 är ett måste. Det här kapitlet borde läsas från början till slut. Fjärde kapitlet ger exempelwebbsidorna. De två underkapitlen behandlar sidan för att titta på systemet och sidan för att ändra i systemet. I femte kapitlet summeras förvärvade kunskaper och en utsikt för hur systemet kan komma att utvecklas i framtiden. Det sjunde och sista kapitlet försöker dra någon form av slutsats av hela arbetet. I bilagorna finns en snabbkurs i ASP och listning av funktionerna i funktionsbiblioteken.

2 Förutsättningar och krav

Vem som helst ska kunna använda sidorna i systemet. Det ska inte finnas orimliga krav på bläddraren, inte heller ska det finnas krav att ladda ner någon speciell mjukvara, helst ska inte användaren av systemet vara medveten om att den använder något av det. Dock behövs att bläddraren är skapligt ny, hanterar bilder (för bildartiklarna), uppladdning av filer och formulär.

De som skriver webbsidorna och de som lägger in nya sidor i systemet ska inte heller behöva ha någon speciell mjukvara eller bläddrare. De ska kunna använda det de har att tillgå.

Dock måste webbservern vara en IIS-server från Microsoft av minst version 5. Databasen som ska användas måste vara av en vanlig typ som utvecklarna av systemet är bekant med, och som också helst har ett eget grafiskt gränssnitt. Mer specifikt i det här projektet så ska databasen vara Access 2000.

2.1 Sammanställning av krav för systemet

Dessa krav kommer ifrån samtalet med webbyrån, förutsättningarna som fanns för att genomföra systemet och från förväntade resultat.

1. En databas behövs för artiklarna och menyerna. Den ska ligga på en Windows 2000-server. Det ska vara en Access 2000 databas. Databasen ska lagra följande:
    a. Menysystemet; det ska finnas (minst) en rot som ska kunna ha minst 30 olika huvudmenyer under vilka det ska kunna skapas minst 30 olika undermenyer. Det ska gå att skapa minst 8 nivåer av undermenyer.
    b. Varje meny ska ha en summeringsartikel
    c. En artikel ska innehålla följande delar
        i. Huvudtitel (ska få innehålla minst 30 tecken eller fler)
        ii. Ingress (ska få innehålla minst 200 tecken eller fler)
        iii. Brödtext (ska få innehålla minst 4000 tecken eller fler) alternativt en bild som ska få vara minst 20 kb eller större (valfritt format)
        iv. Nyckelord (åtminstone 10 nyckelord à 15 bokstäver)
    d. En rubrik ska kunna tilldelas en bestämd plats i en meny
2. Ett gränssnitt som fungerar i en vanlig bläddrare mot databasen som gör det möjligt att konstruera menysystemet ska implementeras, med följande funktionalitet:
    a. Skapa menyer
        i. Skapa huvudmeny, undermenyer och så vidare.
    b. Menyerna ska kunna ha en beskrivande text för vad de gör.
3. Det ska även finnas ett gränssnitt mot databasen som gör det möjligt att lägga till artiklarna. I grunden ska det skapas ett lösenordskyddat funktionsbibliotek som kan göra följande:
    a. Lägga upp artiklar
        i. Välja var artikel ska vara
        ii. Ladda upp artikel, huvudrubrik, ingress samt bild eller brödtext

iii. Val av nyckelord för en artikel

b. Lägga upp nyckelord

i. Lägga till nyckelord

ii. Ta bort nyckelord

4. Ett funktionsbibliotek till webbdesignern som sedan ska skapa den slutliga webbplatsen. Den kommer att ligga på samma dator som databasen. Funktionerna som behövs kommer vara

a. Menyaccessfunktioner, med hänsyn till nuvarande stället i menysystemet

i. Hämta menyrubriker

ii. Välja en menyrubrik och få dess underrubriker

iii. Ytterligare funktioner som identifieras som nödvändiga

b. Artikelaccessfunktioner

i. Hämta huvudrubrik

ii. Hämta ingress

iii. Hämta info: Är innehållet en bild eller brödtext?

iv. Hämta brödtext eller bild

c. Relaterade-artiklar-funktioner

i. Hämta huvudrubriker för relaterade artiklar. Baserat på hur många nyckelord en vald artikel har gemensamt med andra artiklar.

ii. Hämta ingress för den mest relaterade artikeln, baserat på något av kraven angett ovan

5. Databasen ska klara av att tjäna 1 000 000 artiklar om dagen.

3 Beskrivning av design och implementation

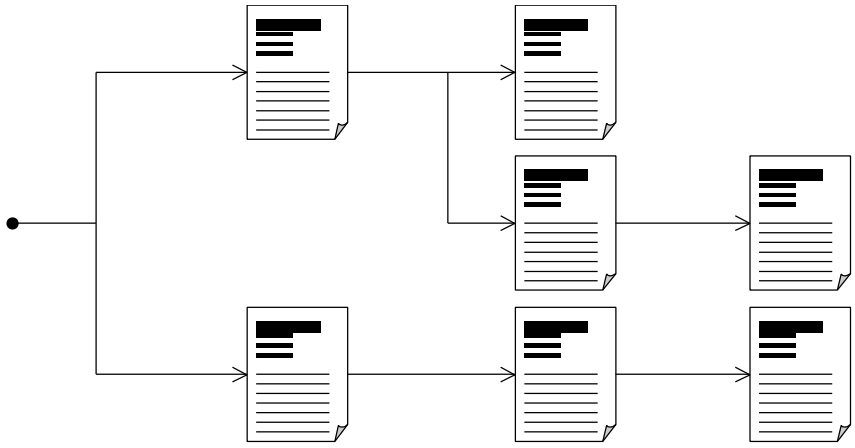
Denna del av dokumentet beskriver ingående hur de olika konstruktionerna i systemet är gjorda, varför det valdes att göra på just det här sättet och vilka alternativ som begrundades och valdes bort.

Det finns ett antal delar i det här projektet som fick utvecklas parallellt för att de skulle ha en chans att bli färdiga i tid. Delarna var konstruktionen av databasen, skapandet av två funktionsbibliotek samt göra en exempelimplementation av systemet. Uppdelningen gjordes på dessa delar eftersom det var vad som framgår ur målen för systemet. Det fattades

gränssnittsdesign för exempelwebbsidans två aspekter, dels den som de vanliga Internetanvändarna kommer att nås av, och dels den som systemadministratörer, ansvariga utgivare, författare och nyckelordshanterarna kommer ha tillgång till. Det här gör att de här sidorna och bilderna ifrån systemet har mindre bra grafisk design.

3.1 Tidiga resonemang kring systemet

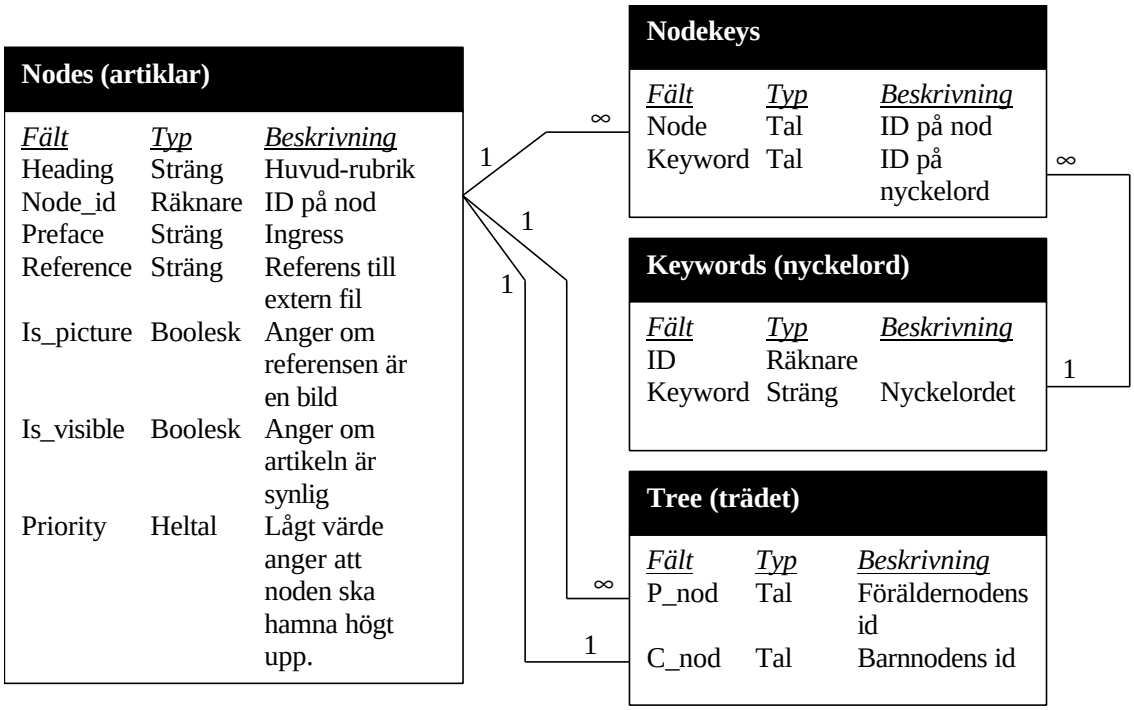
Systemet ska alltså kunna ha massor av menyer, som håller undermenyer som sedan i sin tur, någon gång, har en artikel. Det är värt att notera att alla dessa menyer, undermenyer och sedan artikeln kommer att ha en beskrivande text. Ett annat sätt är att se alla dessa menyer, undermenyer och artiklar som en och samma sak, kalla dom noder. Det rubriker och artiklar kan behandlas på samma sätt, men ändå uppfylla kraven. Detta synsätt på artiklar och rubriker som samma sak illustreras i Figur 5.



Figur 5: Synen på menyer som inget annat än artiklar med underartiklar

3.2 Databasens design och konstruktion

Den här delen av dokumentet kommer att beskriva hur databasen är konstruerad samt hur data kan hämtas ifrån den. Den kommer också att tala om hur vilka alternativa design av databasen som fanns och varför just en av dom valdes. Databasen blev att se ut som Figur 6 visar vid efter design och konstruktion:



Figur 6: Databasens struktur efter design och implementation

3.2.1 Inledande resonemang om design av databasen

Vad är det som behövs lagras i databasen, och framför allt, vad *kan* lagras i databasen. Ett av kraven på systemet var att det skulle kunna hålla bilder, och bilder har en tendens till att vara ganska stora, i dagsläget ungefär 10-300 KiByte[3] för en jpeg-bild med den allra vanligaste storleken och kvaliteten. Det betyder att föra att laga en bild i databasen behövs en datatyp som kan hantera även stora bilder. Visst, ett krav tillåter dom att aldrig bli större än 20 KiByte, men det kändes lite konstigt att begränsa sig där.

Ett sätt för att lägga hela artikelkroppen eller bilden i själva databasen, var då att ha dem som strängar om 256 tecken som kedjas ihop. Den här lösningen kändes dock lite komplicera, och det skulle bli lite långsamt för stora bilder.

Det visade sig emellertid att Access största lagringskapacitet var hos PM eller Notes[4]. Dessa kan hålla data på upp till 64 KiByte, vilket skulle vara mer än tillräckligt för krav 1.c.iii men med den stora nackdelen att inte icke-alfanumeriska tecken kan användas. Detta betydde i sin tur att för att lagra en bild i databasen behövs den kodas om till base64, eller nått

[3] Beteckningen KiByte är hämtad ifrån Svenska Datatermsgruppens rekommendationer [1]. Prefixet Ki betecknar här 1024.
[4] PM eller Notes är en och samma datatypen i Access, men bara på svenska och engelska.

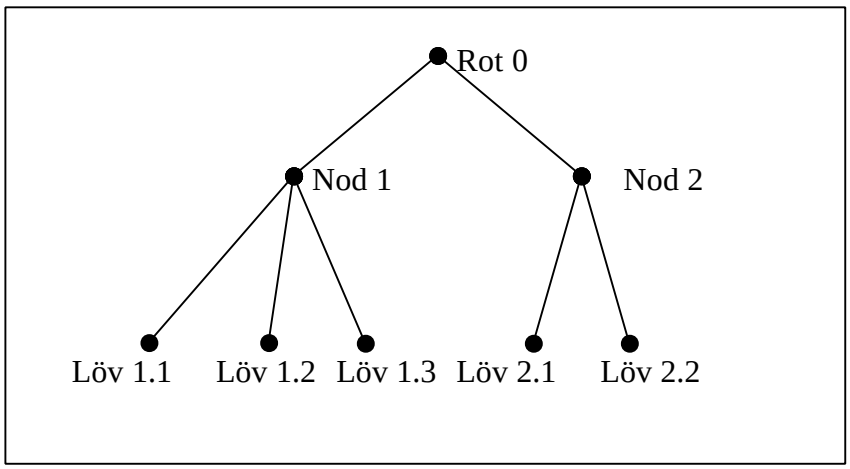
liknande. Men både erfarenheter och information om sådana omkodningar saknades så det var meningslöst att använda någon av PM eller Notes.

Det fanns ytterligare en möjlighet. Så kallade OLE-objekt. Men med bristande tid och erfarenhet även av dessa tekniker, valdes att lösa problemet genom att lägga bilderna och texterna i en extern fil och en referens lades i databasen istället.

Det fanns önskemål från reklambyrån om en trädstruktur och en mängd med nyckelord som kunde knytas till artiklarna. Eftersom artiklarna skulle hamna i en trädstruktur av något slag, så kommer artiklar härefter räven kallas för noder.

3.2.2 Trädstrukturen

Att kunna representera ett träd i en databas är inte så svårt, men det är inte så lätt att förstå problemet tillräckligt bra för att kunna lösa på ett smidigt sätt. För att lättare förstå resonemanget referera till Figur 7.



Figur 7: Exempel på ett träd

Ett träd är ett specialfall av en graf. Det består alltså av noder. Noderna i figuren representeras av svarta punkter. De förbinds av streck som kallas för kanter. Ett träd utan noder är ett giltigt träd. Har trädet mer än en nod har det alltid exakt en rot. Ifrån roten går kanter till dess barn. Då kallas roten för förälder. Nod 1 och 2 är barn till Rot 0 och Löv 1.1, Löv 1.2 o.s.v. är barn till Nod 1 resp. Nod 2. Förutom roten som inte får ha någon förälder så får varje nod bara ha exakt en förälder. Ett löv är en nod som inte har ett barn.

Trädet kan ses som att det består av två mängder, en med noder och en med kanter. Därför valdes att separera noderna från kanterna, till nodes (artiklar) och tree (träd). Ett exempel på tree (träd), finns i Tabell 1. Barnen kan bara ha en förälder, men en förälder kan ha många

barn. Det gör barnen kandidater till primärnyckeln. Om Figur 7 skulle representeras som en mängd kanter, så som trädet är beskrivet i databasen, skulle den få följande utseende på den:

Förälder	**Barn**
Rot 0	Nod 1
Rot 0	Nod 2
Nod 1	Löv 1.1
Nod 1	Löv 1.2
Nod 1	Löv 1.3
Nod 2	Löv 2.1
Nod 2	Löv 2.2

Tabell 1: Visar hur ett träd kan lagras, d.v.s. kanterna är beskrivna här

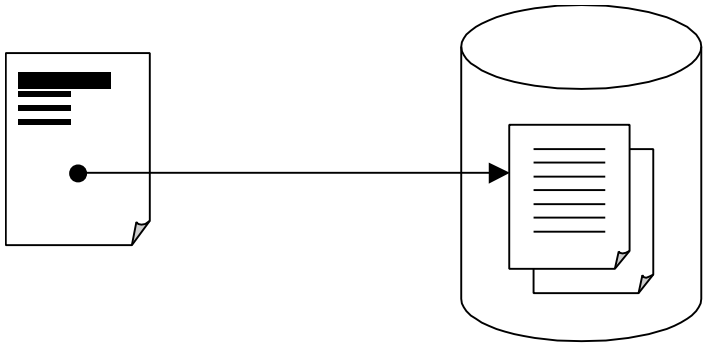
Se nästa rubrik för att veta vad som blev av tabellen som hade noderna.

3.2.3 Artiklarna (noder)

Det fanns bland målen en rad som sa att det borde gå att använda två olika sorters artiklar, antingen en bildartikel eller en textartikel. I inledande resonemang till datamodellen beskrivs vad orsaken är till att en extern referens borde vara bättre, eller åtminstone mer genomförbart, än att lagra allt i databasen[5]. Vad det än var, behövde den lagras i en extern fil. Men vad ska göras åt huvudrubriken och ingressen? En huvudrubrik kommer troligen aldrig bli speciellt lång, på sin höjd kanske 100 tecken. Den fungerade att ha som en sträng i databasen, lika så ingressen, som inte den heller får vara längre än en sträng i databasen. Och maxlängden för en sträng i Access 2000 är 256 tecken. Därför finns dessa begränsningar för längden av huvudrubrik och ingress. Huvudrubriken heter ”heading” i Figur 6 och ingressen ”preface”.

Två noder kan vara identiska, om så önskas, så ett unikt id-nummer för varje nod skapades. Detta id-nummer användes senare även i förälder/barntabellen, och överallt annars där det behövs referera till en nod. Detta nodnummer heter ”node_id” och är en räknare.

[5] Egentligen var målet att artiklarna skulle ligga i databasen. Men merarbetet var för mycket för att hinna med det.



Figur 8: Artikelkroppen (eller bilden) ligger inte i databasen, utan på hårddisken

Det sades i intervjun med reklambyrån att de vill välja plats åt en artikel i en meny. Önskemålet finns som ett krav och därför finns ett fält som heter ”priority”, vilket sorterar artiklarna med lägst värde, alltså högst prioritet överst i listan. Fältet är ett heltal. En diskussion om alternativen finns under 3.4.2.

För att kunna tillgodose ett annat önskemål som står nämnt bland målen att möjligheten att stänga av en artikel finns så finns också en flagga, ”is_visible” som kan sättas i efterhand för att markera att en artikel är synlig.

3.2.4 Nyckelorden

Utan att klia skallen av sig är det ganska lätt att förstå att de här är det mest unika med systemet. Tanken med nyckelorden är att de ska kunna knyta samman olika artiklar med varandra på de mest oväntade sätt. Det ska finnas en lista med alla nyckelord, och sedan kan artiklar få några av nyckelorden tillägnade. Villkoren är satta att det ska vara ganska många nyckelord, så lösningen fick bli en många-till-många-relation mellan nyckelorden och noderna. För att skapa en sådan många-till-många-relation så behövs en mellantabell, som i Figur 6 heter nodekeys. Tabellen som lagrar nyckelorden heter ”keywords”. Fältet id är en räknare och keyword är en vanlig textsträng. För en längre och mer detaljerad beskrivning om nyckelord samt figur, se stycke 3.3.

3.2.5 Användardatabasen

Det står nämnt i samtalet med reklambyrån att de vill ha ett lösenord för att kunna manipulera databasen ifrån webben. Tolkningen av det här önskemålet var att de egentligen ville ha en användaredatabas i systemet där alla användarna lagras. Några olika roller fanns beskrivna i bakgrunden, bland annat någon administratör för menyerna, en artikelförfattare och kanske

någon ansvarig för hela systemet. Exakt vad de olika rollerna[6] innebär var inte speciellt tydligt, men en tabell gjordes med följande fält för att markera tillgång eller inte tillgång till vissa delar av systemet, fälten är alltså ”ja/nej” eller så kallade booleska fält:

- **Nodes**: har rättighet att lägga till noder, ändra noder, radera noder eller förändra nyckelorden härrörande till noder.
- **Tree**: har rättighet att flytta noder runt om i trädet, sätta deras prioritet, välja om de är synliga, radera noder eller förändra nyckelorden härrörande till noder.
- **Users**: har rättigheten att lägga till användare, radera användare, förändra rättigheter för en användare och sätta nytt lösenord till andra användare.
- **Keys**: får lägga till och radera nyckelord.

Tabellen blir alltså att se ut så här:

Userdb (Användardatabasen)

Fält	*Typ*	*Beskrivning*
ID	Räknare	ID på användar
User	Sträng	Användarenam
Passwd	Sträng/ Lösenord	Lösenord för användaren
Nodes	Boolesk	Anger artikel-rättighet
Tree	Boolesk	Anger trädrättighet
Users	Boolesk	Anger användare-rättighet
Keys	Boolesk	Anger nyckel-rättighet

Figur 9: Användardatabasen

Fältet user är en vanlig textsträng, och ID är en räknare. Passwd fältet är ett vanligt textfält, men med tanke på att den helst inte ska kunna visas, så har den också en indatamask av typen ”Lösenord”, som gör att texten blir stjärnor istället.

[6] För en mer ytterligare beskrivning av rollerna i systemet, se 3.4.3

3.2.6 Relationer mellan tabellerna

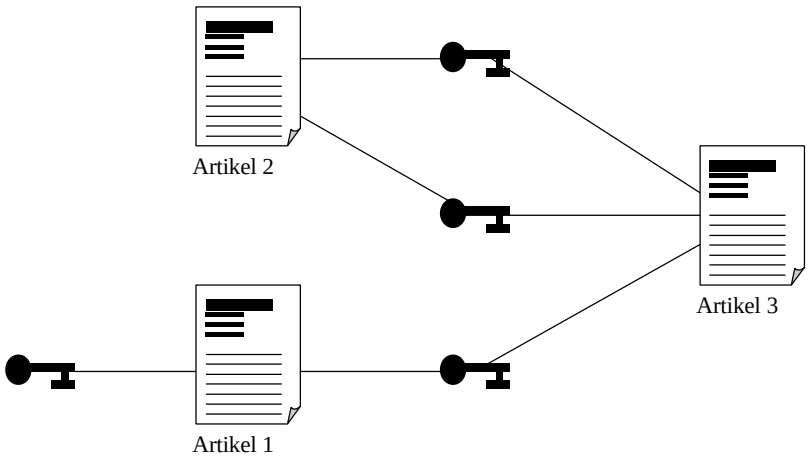
Hur hänger nu allt samman? Relationer skapas mellan tabellerna för att förhindra att databasen blir okorrekt, att dess integritet inte är satt ur spel. För att motivera kopplingarna i Figur 6, behövs lite tankearbete.

Egentligen ska en artikel kunna ha många nyckelord, och ett nyckelord ska kunna ha många artiklar. Relationen mellan dessa två skulle alltså kunna beskrivas som en ”många-till-många” relation. Men ”många-till-många” relationer kan inte existera i en databas, eller åtminstone inte i Access 2000, så standardlösningen är att lägga en mellantabell. Här heter den ”nodekeys” för att markera att det är den mellantabellen. Relationerna sattes sedan som Figur 9 visar mellan tabellerna.

Det finns en ”en-till-många” relation mellan artiklarna och sig själva. Här sitter en mellantabell, som representerar trädet. Egentligen hade trädet kunna ha varit ett fält i artikel/nodtabellen, men fördelen med att ha den trädet i en egen tabell är att det går snabbare att skapa trädmenyer.

3.3 Relaterade artiklar

Det fanns ett krav på att kunna hämta alla relaterade artiklar till en artikel. Även om det låter ganska självklart vad som ska göras, är det ganska mycket mindre självklart hur det ska gå till, att finna vilka artiklar som är relaterade. I den här texten kommer orden nycklar och nyckelord att användas för att beskriva samma sak.



Figur 10: Artiklar kan knytas till olika och gemensamma nyckelord.

Låt Figur 10 ligga till grund för ett beskrivande exempel. Bilden har tre artiklar och fyra nyckelord, som symboliseras av nycklar. En tabell som skapats för att se vilka artiklar har hur många nyckelord gemensamt med de andra, så får detta utseende:

	Artikel 1	**Artikel 2**	**Artikel 3**
Artikel 1	(2)	0	1
Artikel 2	0	(2)	2
Artikel 3	1	2	(3)

Tabell 2: Anger antalet gemensamma nycklar mellan olika artiklar. Siffrorna inom parantes ger antalet nycklar en artikel har.

Nu börjar det kanske klarna lite om hur en artikel är mer eller mindre relaterad till en annan artikel. Ur tabellen syns att artikel 1 och 2 inte är relaterade alls, medan artikel 3 och 2 har två nyckelord gemensamma.

För ett par exempel till, om en meny med relaterade artiklar till artikel 1, skulle det bli (i den här ordningen) en meny med artikel 3, och inte 2, eftersom det inte ens finns en gemensam nyckel till artikel 2. För artikel 3 hade menyn blivit artikel 1 följt av artikel 2.

3.4 Gemensamt för funktionsbiblioteken

Med en databas i ryggen var det en fråga om att lära sig tillräckligt om ASP och dataaccess för att kunna hämta data från databasen och skriva ut den i sidan. För den som känner sig obekant med ASP kan läsa bilaga A som behandlar programmiljön lite lätt, eller också en bok som t.ex. [2].

3.4.1 Menyer

Genomgående i systemet så finns behovet av att lista upp något. Är det en lista av nyckelord som finns tillgängliga, eller kanske lista alla artiklar under en annan artikel, så måste det till någon form av lista. Att kalla dom för listor hade varit ett naturligt val, men eftersom senare så väljs något ifrån dom, så var det trevligare att kalla listorna för menyer.

Det finns ett antal menyer i systemet. Utifrån kraven gers att de menyfunktioner som behövs är följande:

- Hämta alla (under) artiklar, för att kunna visa trädmenyer.
- Hämta alla relaterade artiklar, för att kunna visa relaterade-artiklar-menyer.

- Hämta alla nyckelord till en artikel, för att kunna göra ett gränssnitt åt designern där han eller hon får veta vilka nyckelord som är använda till en specifik artikel.
- Hämta alla användare, för att kunna se vilka användare som finns i systemet.
- Hämta alla nyckelord som finns tillgängliga, för att veta vilka nyckelord det går att välja mellan.

De två första menyerna ger ungefär en enkelriktad sekvens eller lista av artikelid-nummer. De tre sista menyerna ger nyckelordsid-nummer, användareid-nummer eller nyckelordid-nummer.

Liksom för sekvenser och listor, så behövs något sätt att stega genom en meny. De sätten som är viktiga i det här sammanhanget är skapa menyn, eller ett menyhandtag, stega framåt samt att sedan faktiskt kunna hämta någon form av nyckelinformation om det nuvarande elementet i menyn. Det är ett idnummer här, som sedan kan användas för att slå upp huvudrubriken till en artikel, ett nummer får att kunna slå upp ett nyckelord eller för att finna information om en användare.

För att senare kunna intressanta trädmenyer eller nyckelordsmenyer behövs även lite andra funktioner. Här följer en lista på bra funktioner för det ändamålet

- Kontrollera om en artikel har barn, för att veta om det är värt att skapa en undermeny.
- Skapa en sökväg ifrån roten till nuvarande artikeln, så att man kan utveckla lagom stor del av varje underrubrik, om så önskas.
- Hämta en artikel på angiven plats inom en sökväg, för att hitta ett en artikel i sökvägen.
- Hämta längden av en sökväg, så att man vet hur djupt nere i trädet sökvägen går.
- Hitta om en artikel finns med i en viss sökväg, för att snabbt veta göra beslut om utveckling av en meny eller inte.

Dessa är ganska fritt skapade utifrån vad som behövdes för att skapa exempelwebbsidan.

3.4.2 Prioritet för en artikel

Det fanns önskemål om att kunna ordna rubrikerna så att de kommer i en vald ordning. Alltså, så att den trädansvarige kan bestämma när surfaren expanderat en gren till en rubrik vilken underrubrik som kommer övers o.s.v. Därför finns det något som heter prioritet för en artikel. Ju lägre prioritetsvärde en artikel har, desto högre upp i en meny hamnar de. Det är bara i samband med de trädmenyer som det spelar roll.

Klart finns andra sätt att skapa ordning i menyn än att bara sortera efter ett fält. Två tänkbara alternativ är att ha ett sorteringsvärde i trädet istället, alltså det som kallas tree, trädet, i Figur 6, eller att skapa en lista som håller vilken artikel som följer efter en annan.

Egentligen är det inget som talar för att den lösningen med ett prioritetsvärde är den bästa, men den har fördelen att den senare skapar ett relativt enkelt SQL-uttryck.

3.4.3 Inledande om rättigheter för användare

Det fanns inget krav om att kunna skydda delar av systemet mot obehöriga, men eftersom testsystemet ligger på den delvis exponerad dator så lades säkerhetsfunktionerna till.

Alla som inte har loggat in sig i systemet är bara ”gäster” i systemet. Det gör att de funktioner som inte tillåter att gäster använder dom inte heller kommer att fungera. Lite senare kommer det att beskrivas vilka rättigheter en användare behöver för att använda vilka delar av systemet. Som en riktlinje har systemet skapats för att det ska finnas fem olika sorters användare, eller roller, av systemet. De är:

- Vanliga Internetsurfare, det är de som når systemet utan att ha några rättigheter alls, förutom läsrättigheter på synliga artiklar, och skapa träd- och relaterade-artiklar-menyer.
- Artikelförfattare, har rättighet att skapa en ny artikel samt att lägga till nyckelord till den. De har även rättigheten att radera en artikel. En artikelförfattare kan se dolda, tidigare kallat avstängda, artiklar.
- Nyckelordsförfattare, har rättigheten att skapa eller radera nycklar men inte att lägga till nyckelord till en artikel och inte heller att radera nyckelord från en artikel, så vida inte det inte är nyckelordet i sig som raderas, i vilket fall det försvinner ifrån alla artiklar där det används.
- Trädansvariga, har rättigheten att sätta in en artikel i trädet. Den får välja var en artikel ska ligga, sätta prioritet på artikeln och göra den senare synlig. En trädansvarig kan se dolda artiklar.
- Användaransvarig, ansvarar för användarkonton och får skapa och radera användare, får även byta lösenord på vilken användare som helst.

Vanligtvis är åtminstone trädansvarig och artikelförfattare samma person.

Funktionen för att se vilka rättigheter som den nuvarande användaren har finns i funktionsbiblioteket för att titta på systemet. Anledningen är att även obehöriga användare ska få använda den delen av systemet som är till för dom. Den andra delen av systemet ska inte

synas alls för dessa användare. Därför finns funktionen för att logga in i systemet i biblioteket för att ändra på systemet.

3.4.4 Litet om datahämtningen ifrån databasen

Tack vare valet av en kombination av ASP, VBScript och Access 2000 så finns det ett enkelt sätt att nå information nere i databasen ifrån funktionsbiblioteken. Det görs lättast genom att skriva SQL-uttryck som hämtar upp allt i en variabel, och det är precis så som det här systemet är skapat.

3.5 Funktionsbibliotek för att kunna titta på menysystemet

Den här delen av systemet är relativt enkelt att förstå sig på. Det behövs inte allt för många funktioner för att hämta artiklar, träd- och nyckelordsmenyer från ett menysystem. En snabb överblick av kraven framhäver ett behov av funktioner för att hämta en meny av artiklar, för att hämta en huvudrubrik, för att hämta en ingress, och för att hämta kroppen på artikeln. Tillsist behövs en funktion för att hämta en meny med relaterade artiklar.

3.5.1 Artikelkroppen

En designfråga var vad som skulle göras med de olika sorters artiklarna. Det ska ju finnas bildartiklar och textartiklar. Dessa skiljer sig ifrån varandra bara på vad som finns i den externa filen, och sedan om flaggan ”är bild” är satt eller inte. Det finns några alternativ när i samanhanget av att titta på en artikel. Följande är en kort summering på några av alternativen.

Ett sätt är att ha en funktion som talar om ifall artikelkroppen är en bild eller en text. Om det är en textartikel, så kan egenskriven kod kopiera in texten i sidan och om det istället var en bildartikel så skapas en referens till bilden. Bildreferensen kan antingen vara till bildens fil som ligger på hårddisken, eller kan det vara en speciell sida som bara infogar innehållet i filen och ändrar lite i huvudet till svaret till bläddraren[7].

En annan lösning till problemet är att gör en funktion som gör allt åt en, antingen ger innehållet i filen eller en färdig bildtagg med adressen till någon av varianterna på den externa filen.

För det här systemet valdes det senare alternativet, och att refererar direkt till den externa filen. Säkerhetsmässigt är det ju inte speciellt bra att adressen till filen blir exponerad, och att det ökar risken att någon som orkar gissa länge skulle kunna få tillgång till både bilder och

textfiler som de inte skulle ha fått sett i vanliga fall. I en framtida version kan en funktion som är ansvarig för att hämta den här informationen läggas in istället.

3.5.2 De andra funktionerna

Det är ganska lätt att skapa funktioner som matchar nästan helt ett till ett mot kraven för funktionsbiblioteket. Dessa har inte beskrivits i stycke 3.4.1, dock följer de kraven i avsnitt 2.1. De funktioner som behövs ytterligare är följande

- Hämta huvudrubrik
- Hämta ingress
- Kolla om den nuvarande inloggade användaren har en viss rättighet
- Kolla om en artikel är synlig (fungerar bara om användaren har rättighet för att göra det)
- Hämta prioritet för en artikel

3.6 Funktionsbibliotek för att ändra i menysystemet

För att kunna ändra på systemet så behövs en mängd funktioner. Ingen av funktionerna är speciellt avancerade, men för att abstrahera bort implementationen av databasen från systemet så blir det många ”ändra”, ”radera”, och ”lägg till” funktioner i systemet, nämligen minst en uppsättning av dessa tre typer för varje tabell, rimligtvis. Inte alla behövs, t.ex. finns inget krav för att kunna ändra nyckelord, därför behövs ju inte den funktionen. Det skulle gå bra att med bara ”lägg till” och ”radera” funktionerna i kombination med någon ”hämta” funktion för att göra en ”ändra” funktionen. Ibland fungerar inte det nämnda sättet, och det är när det finns en räknare och referenser med i bilden som inte borde ändras.

De olika sakerna som ska kunna manipuleras är i stort

- **Artiklar**; det ska gå att ändra en artikel, radera den och lägga till en ny. Det är artikelförfattaren som har rättighet att använda de här funktionerna. Artikelförfattare och trädansvarig kan se dolda artiklar.
- **Trädet**; de ska gå att flytta en nod runt trädet, kunna sätta nodens prioritet och göra den synlig eller osynlig. Det är trädansvarig som kan göra det här.
- **Nyckelorden**; Det ska gå att skapa nya nyckelord samt radera dem. Ytterligare ska det gå att knyta nya nyckelord till en artikel och det ska gå att radera alla

[7] Svaret till klienten i http-sessionen.

bindningar mellan en artikel och nyckelorden. Artikelförfattaren har rättigheten att knyta och radera nyckelorden till en artikel, och endast nyckelordsförfattaren kan skapa och radera nyckelorden som finns tillgängliga.

- **Användare**; det ska gå att skapa nya användare, ändra på en användares rättigheter, ändra sitt eget lösenord, ändra en glömsk användares lösenord och radera en användare. Funktionen för att ändra på sitt eget lösenord får vilken användare som helst använda, men de övriga funktionerna är det bara användareansvarig som har tillgång till.

3.6.1 Inloggning

I det här biblioteket ska även funktionen för att logga in finnas. Den bara kollar att användarnamnet och lösenordet matchar med vad som står i databasen. För att markera att en användare är inloggad finns några sätt. Det lättaste sättet är att låta en variabel markera användarnamnet. Just så gör det sättet använder det här systemet. I en sessionsvariabel[8] lagras det, så att användaren blir ”utslängd” ifall han eller hon stänger av bläddraren, eller om surfaren lämnar systemet för länge.

4 Implementering, test och konstruktion av exempelwebbsidorna

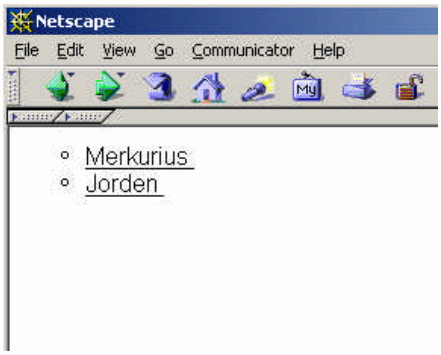
Nu fanns en otrolig massa funktioner att som kan användas för att skapa ett artikelsystem, även för att visa och ändra på det. Här finns de två exempelsidorna som skapades under resans gång, men tänk på att det inte finns någon grafisk design i grunden, så de här sidorna är mer i funktionellt syfte snarare än att visa hur vackert det blev. Det var ju en inte en reklambyrå utan snarare en ung kille som gjorde den här rapporten och systemet.

Exempelsidan som är skapad bygger på astronomi och geografi samt små förklarande texter och nyckelord om de olika platserna. Sidan innehåller artiklar med geografiska och astronomiska ting från grovkornighet av planterar ner till finkornighet på städer. En helt subjektiv bedömning av behovet av nyckelord gjordes, och sedan hur de knöts till platserna var också inte heller helt objektivt.

[8] För en närmare beskrivning av vad en sessionsvariabel är, se bilaga A.

4.1 Exempelwebbsida mot Internetsurfarna

Här följer några skärmdumpar som visar hur systemet fungerar.



Figur 11: Det första som användaren ser på exempelsidan

Figur 12: Så här ser det ut när någon har surfat ner lite genom trädet. Dessutom syns att det saknas en artikelkropp.

De här två bilderna visar hur Internetsurfaren initialt kan använda systemet. Se Figur 14 för ett exempel på hur bilder och liknande fungerar, och att skymtar underst den relaterade menyn med artiklar i Figur 12. Den HTML/VBScript-kod som behövdes för att skapa sidorna i bilderna Figur 11 och Figur 12 är följande:

```
1. <!-- #include FILE="library_get.asp" -->
2. <!--#include FILE="menu.asp" -->
3. <!-- Menyn -->
```

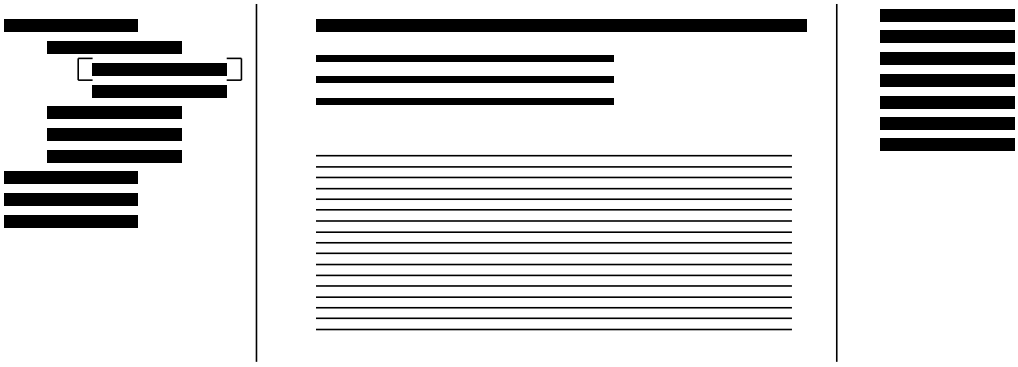
```
4. <%= make_menu(current_node, "index.asp?current_node=")%>
5. <br>
6. <%
7. if(current_node<>"") then
8. %>
9. <hr>
10.  <!-- Artikel -->
11.  <h1><%=fetch_heading(current_node)%></h1><p>
12.  <h2><%=fetch_preface(current_node)%></h2><p>
13.  <code><%=fetch_body(current_node, "")%></code><p>
14.  <hr>
15.  <ul>
16.  <%
17.   menu=related_menu_init(current_node)
18.   dim count
19.   count=0
20.   Do While count<5 and not menu_at_end(menu)
21.    Response.Write("<li> <a href=index.asp?current_node=" &_
22.     menu_get_current(menu) & ">")
23.    Response.Write(fetch_heading(menu_get_current(menu))&_
24.     "</a>")
25.    menu_go_next(menu)
26.    count=count+1
27.   Loop
28.   menu_exit(menu)
29.  %>
30.  </ul>
31.  <%end if%>
```

Liten förklarning till programkoden här ovan (numreringen tillhör inte programkoden):

Antingen taggarna `<script language=VBScript runat=SERVER> kod </server>` eller `<% kod %>` kan användas för att skriva in VBScript-kod i ASP-sidor. På raderna 1 och 2 hämtas funktionsbiblioteken för att hämta data och för att visa en meny. I det första biblioteket tas en variabel i url-strängen emot som heter ”current_node” och fås tillbaka vid namn ”current_node”. Sedan på rad 4 skapas trädmenyn, som kan se överst i figurerna 11 och 12. På raderna 10-12 skriver den ut huvudrubriken, ingressen och artikelkroppen, av vilken sort den än är. Tyvärr saknas en högre menyfunktion för att skapa relaterade-menyn, så

därför görs den ifrån grunden i raderna 17-28. Raderna 7 och 31 kontrolleras att inte en artikel och dess relaterade artiklar inte visas om artikeln inte finns eller att ännu inget valts i någon meny.

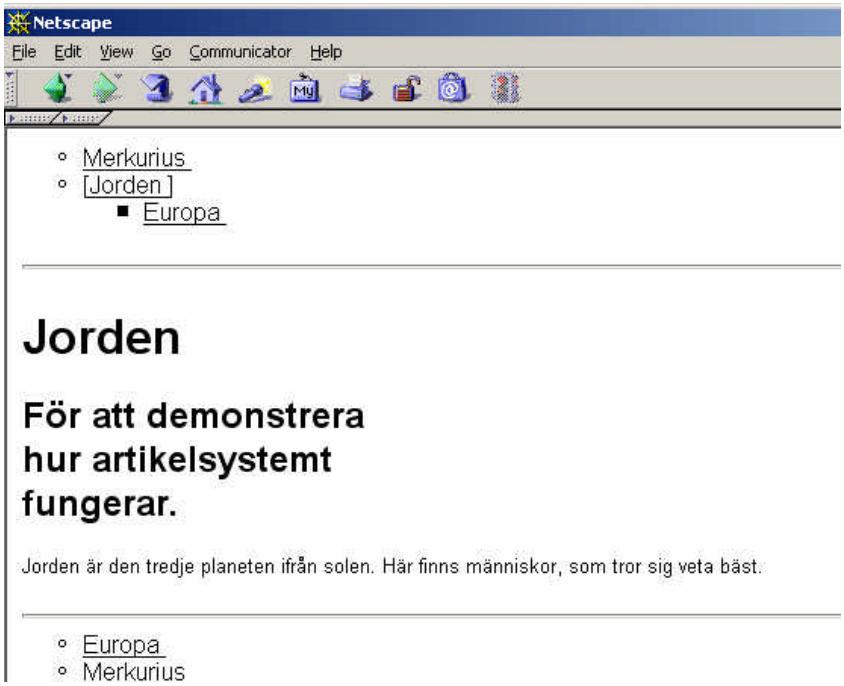
Målet var att sidan skulle ha ett mer skönare utseende. Det som var föreslaget finns beskrivet i Figur 13 är ett förslag. De tjocka linjerna är huvudrubriker eller rubriker, de mellantjocka är ingress och de smala är antingen avdelare eller artikeltext.



Figur 13: Snyggare design för exempelwebbsidan mot Internet

Tyvärr fanns det inte tid till att använda den här designen. Vidare i Figur 15 är ett exempel med lite text i artikelkroppen, och ett exempel med en bild i Figur 14.

Figur 14: Så här ser en solig himmel ut i Karlstad.



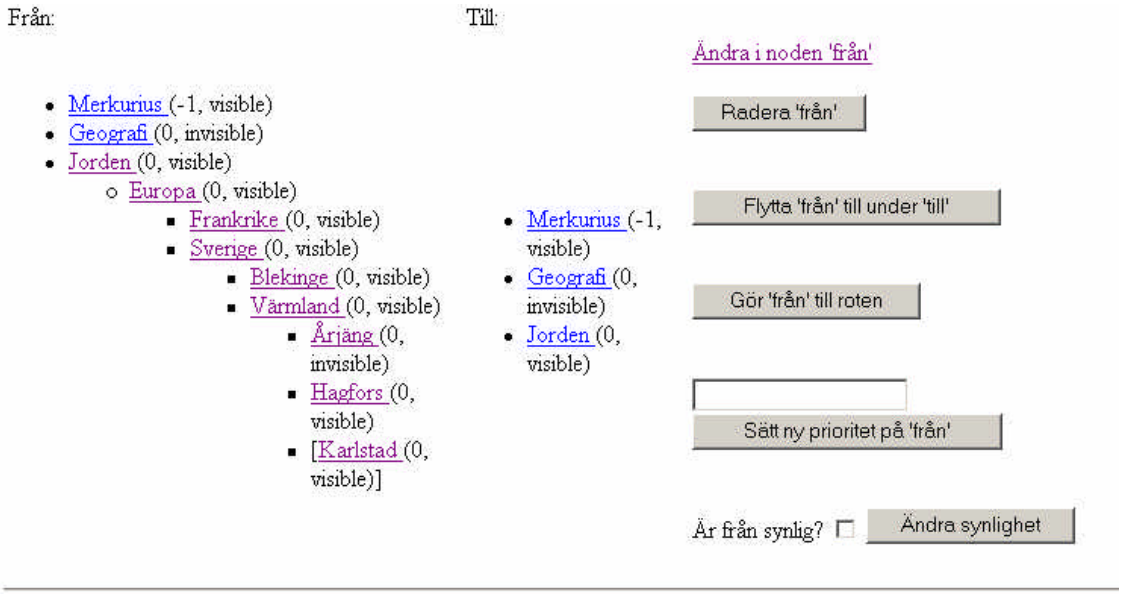
Figur 15: Exempel på hur en textartikel visas

4.2 Exempelwebbsida mot de som administrerar menysystemet

Den här webbsidan skapades också helt utan tanke på att det skulle se snyggt ut. Det enda som var intressant var att skapa ett gränssnitt mot funktionerna där nere i funktionsbiblioteken. Eftersom inte alla funktioner är implementerade på ett sådant sätt att de hanterar vilken indata som helst, utan faktiskt har förvillkor, så måste det till en del kontroller på vad som skickas ner i funktionerna även i interfacet. Dessa kontroller är gjorda i exempelgränssnittet, så senare om ett nytt, förhoppnings bättre, gränssnitt skapas så går det att återanvända den mesta av den lite mer avancerade koden som finns i exempelgränssnittet.

Samma menymodul används här för visa trädstrukturen som återfinns i den andra exempelwebbsidan, alltså den mot Internet. Exempelwebbsidan mot Internetsurfarna skapades så att när någon behörig är inloggad i systemet, och har rättigheter för att se de osynliga artiklarna, så visas de, men nu också med information om deras synlighet, om de är avstängda eller inte, och även deras prioritet.

Här följer några exempelbilder, och lite text som förklarar vad det är syns, eftersom den här delen av systemet inte förklarar sig själv så mycket.



Figur 16: Användaren här har alla rättigheter som finns

I Figur 16 visas hur ändringar på trädstrukturen kan göras på ett enkelt sätt. Den här användaren har också rättigheten att radera en nod, alltså har den rättigheter för minst både trädstrukturen och artiklarna. I bilden kallas artiklarna fortfarande för noder, precis som de gjorde i introduktionen för träd (Trädstrukturen, 3.2.2). Från vänster till höger i bilden finns två trädmenyer samt en spalt med en mängd knappar. Över trädmenyerna står det ”från” och ”till”, vilket ska vara vad som menas med till och från bland knapparna. I spalten med knappar är allt som behövs kunna göras med en artikel. Det går att ändra i artikeln, radera densamma, flytta runt artiklar i trädet, flytta en artikel till direkt under roten, ändra prioritet på artikeln samt ändra dess synlighet.

Även visas ett par artiklar som är markerade att vara osynliga, invisible, t.ex. är ”Årjäng” invisible. ”Årjäng” syns inte i menyträdet sett för en vanlig Internetsurfare, Figur 17.

Figur 17: Visar trädet sett för en vanlig Internetsurfare, här syns inte ”Årjäng”.

I funktionen för att lägga till eller radera nyckelord, finns en lista med nyckelord och sedan två knappar och en text ruta. På knapparna står det ”radera valt nyckelord” och ”Skapa nytt nyckelord”. I textrutans kan nya ord matas in.

Nedan ges en bildserie på förfarandet att lägga till och knyta nya nyckelord till en artikel.

Figur 18:”Jorden” är valt, för den var utan nyckelord

Figur 19: Här har några nyckelord som passar för Jorden valts

Figur 20: Nu har nyckelorden lagts till

Figurerna Figur 18, Figur 19 och Figur 20 visar hur funktionen för att tillägna nyckelord till en artikel fungerar. Först väljs vilka nyckelord (genom att ctrl-markera) som ska användas, och sedan trycker på knappen för ”tillägna”, så knyts nyckelorden till artikeln. Det tycks ju ha fungerat i den här bildserien åtminstone.

Hur det ser ut när ändringar på användare ska göras är mindre viktigt. Det finns allt det som behövs för en rimlig kontroll på användare kan göras, men det finns uppenbarligen ingen loggningsfunktion. Det är ju inte bra, men så fanns det heller inget krav på en loggningsfunktion.

4.2.1 Ett problem: Uppladdning av bilder

Att ladda upp filer till en IIS-server är inte helt trivialt. I vanliga formulär som kan skapas på en HTML-sida och skickar till ASP-servern, kan innehållet nås med en av Request.Form() eller Request.Querystring() funktionerna.

Exempel på hur html-koden ska se ut för att skicka ett formulär:

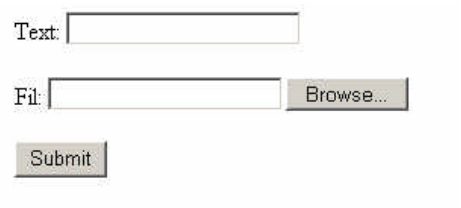
```
<HTML>...
<form method=post action=some.asp>
        Text: <INPUT id=text1 name=text1>
        <INPUT id=submit1 type=submit value=Submit name=submit1>
</form>
...</HTML>
```

För att sedan ta emot formuläret på `some.asp`-sidan, och skriva ut det på sidan, är det här ett sätt:

```
<%@ Language=VBScript %>
```

```
<HTML>...
<% Response.Write(Request.Form(”text1”)) %>
...</HTML>
```

Det ser ju enkelt ut, men att försöka göra samma sak med en fil. Genom att lägga till en filruta som Figur 21 visar, så kommer `Request.Form(”fil”)` oväntat nog inte åt själva innehållet i filen istället når den är filnamnet och dess sökväg på den dator som filen skickades ifrån.



Figur 21: Exempel formulär

Eftersom bilder var ett krav, och det var ett krav att skulle gå att administrera systemet ifrån ett webbaserat gränssnitt, var detta med uppladdning av filer, speciellt bilder, ett problem som behövde en lösning. Efter lite sökande på Internet dök flera lösningar upp som hanterade filuppladdning till ASP-sidor. En av de lösningarna som kom ifrån en programmerare på Microsoft var inte komplett[3], men den gick att ändra på så att systemet skulle kunna ta emot bilder. Så här blev ASP/VBScript-koden för att ta emot filer.

```
<!-- #include file=upload.asp --> <!--filen som gör det möjligt med uppladdningen-->

<% ’ Aspkoden som tar emot grejerna
Text=MyRequest(”Text1”)
FilNamn=MyRequestFiles(1,2)
FilKropp=MyRequestFiles(1,1)
%>
```

5 Erfarenheter och rekommendationer

Efter att ha skapat det här systemet har följande kunskaper erhållits:

- Skapa en säker plats på en IIS-server
- Programmera i ASP/VBScript

- Skriva frågeuttryck bättre i SQL
- Koppla samman ett system med databas och komplicerade ASP-sidor
- Ladda upp filer till ASP genom rena ASP/VBScript

För att verkligen kunna ha glädje av funktionsbiblioteken borde den som använder dom att ha minimala kunskaper om VBScript. Den introduktion som finns bland bilagorna är inte tillräcklig, men efter att ha läst [2] så fås en god start för att komma igång.

5.1 Vidareutveckling

Systemet kan utvecklas i många riktningar. En riktning är att lägga till en ikon till varje artikel, antingen en användaredefinierad eller en som systemet valt själv. Den här skulle kunna göra det möjligt att visa en mer grafisk trädmeny istället för den nu ganska torftiga textmenyn.

Bättre säkerhet vore bra. Systemet borde logga vem som ändrade i en artikel, vem som lade till vilket nyckelord, vem som knöt vilket nyckelord till vilken artikel och när. Allt sånt för att kunna spåra någon som avsiktlig saboterar systemet.

I en framtid kanske det vore bra att kunna lägga in annat än bara text och bild i systemet. Kanske en ren datafil istället för bilden eller artikeln, eller att lägger in en hel html-sida, med egen formatering, även om just det här skulle arbeta emot poängen med systemet. Att göra just den här förändringen är inte så långt borta, men den togs inte i avseende när vissa funktioner skapades.

Ytterligare saker som borde göras är fler kommandon som gör mer avancerade saker, som att skriva ut hela menyer, och fler färdiga menyer att välja bland.

6 Slutsatser

Det här arbetet var en aningens utsvävande till att börja med, och målen fick justeras lite undertiden, men kärnan för målet, att separera den grafiska designen från informationen lyckades.

Funktioner för att hämta information om artiklarna, menyträdet, nyckelorden och relaterade artiklar utvecklades. Ytterligare funktioner för att addera eller manipulera och hantera artiklarna, menyträdet, nyckelorden eller Användardatabasen skapades.

Hela systemet är utvecklat att fungera på en IIS 5.0 server med stöd för ASP 3.0/VBScript, vilket är standard för vilken Windows 2000 utgåva som helst. Microsoft Access 2000 användes som databas, men den kan självklart bytas ut mot en snabbare eller pålitligare databas.

Av de förväntade resultaten som fanns i stycke 1.2 har följande nåtts, numreringen här matchar emot den som finns i det stycket:

1. En datamodell för artiklarna, menyerna skapades. Den som användes fungerar under Microsoft Access 2000 och Windows 2000
2. Ett gränssnitt skapades som fungerade i en vanlig bläddrare. Gränssnittet kan användas för att lägga till och manipulera data i databasen.
    a. Gränssnittet skrevs i HTML och ASP/VBScript
3. Ett funktionsbibliotek mot designern skapades. Det här biblioteket innehåller en mängd funktioner för att hämta menyer, artiklar och en relateradartikelmeny, som baseras på olika nyckelord som har knutits till olika artiklar.
4. En exempelwebbsida skapades för att demonstrera systemets funktioner

Systemet är inte så snabbt som det hade varit med ett eget, fristående, interface som inte var beroende av att fungera genom någon bläddrare. Om det någonsin kommer att kunna hantera 1000 träffar per dag är svårt att säga. Det är helt enkelt för långsamt, även för att skapa och bygga upp trädet med artiklar och rubriker. Används dock en snabbare server kanske det går att få upp hastigheten så att hämta sidor inte längre är störande långsamt, vilket var ett problem på testutrustningen.

Det återstår att se om någon vill använda det här systemet för att skapa sig en sida. Det har redan funnits spekulanter.

Referenser

[1] Svenska datatermsgruppen har en webbsida, och den kan nås på www.nada.kth.se

[2] Stefan Arvidsson, Jesper Ek och Ulrika Eriksson. *Active Server Pages 3.0; & databaser på Internet.* Pagina Förlags AB, 2000

[3] Steven W. Disbrow, *Upload Files with HTML Forms and Pure ASP*, Microsoft, Oktober 2001. msdn.microsoft.com

A Snabbkurs i ASP

ASP är inte ett programmeringsspråk, utan ett ramverk med funktioner som kan användas för att skapa en dynamisk webbsida på en IIS-server. Det har kommit några versioner av ASP, den i skrivandets stund gäller är ASP 3.0, men ASP .NET har även börjat dyka upp nu.

För att skriva en sida så finns flera skriptspråk att välja på, tillgängliga i en av dll-erna till systemet. Vanligtvis används VBScript för att skriva sidan i, men valet finns att skriva den i t.ex. JScript, som nästan är JavaScript, eller något annat skriptspråk som det finns stöd för. I hela den här rapporten användes VBScript som programmeringsspråk.

A.1 Användbara objekt

I ASP-ramverket finns en mängd objekt för att göra livet lättare för webbprogrammeraren. Här finns Request-objektet som har hand om alla saker som kommer ifrån bläddraren just nu. Här finns Response-objektet som har allt som skickas till användaren i den här vevan. Session och Application objekten har hand om sessionen och applikationen respektive. Session är ett objekt som är knutet till varje användare, dvs. alla som når hemsidan får varsitt Session-objekt knutet till dem, och objektet dör antingen genom att det blir gammalt eller genom att användaren surfar ifrån sidan. Application-ojektet är knutet till varje program som går på servern. Flera program kan vara igång på webbservern samtidigt. Session- och Application-objekten är bland annat användbara för att hålla variabler, som lever med objektet. Alltså, sessionsvariabler är unika för varje användare och finns så länge användaren finns. Applikationsvariabler finns så länge en applikation är igång, och delas mellan sessionerna, men inte med andra applikationer. Det finns även ett server-objekt. Det här objektet håller funktioner som bland annat att konvertera en text till html-kod, så att taggar[9] syns och inte tolkas i bläddraren. Dessutom går det att lagra variabler här, och de variablerna blir tillgängliga för alla applikationer och sessioner över hela servern.

Oberoende av vilket skriptspråk som används, så har finns alltid dessa objekt tillgängliga, och deras funktioner är alltid desamma.

[9] T.ex. <bold>, <a href=blablabla> osv.

B Funktionslistning

Här är en genomgång om vilka funktioner som finns i funktionsbiblioteken. Notera att programkoden inte finns med här i.

B.1 Funktionsbiblioteket ”library_get.asp”

B.1.1 fetch_heading(node)

Hämtar huvudrubriken till artikeln med numret `node`. Det måste finnas en nod med numret `node` och `node` måste vara ett nummer. Om artikeln är gömd och den nuvarande användaren inte har access till den kommer huvudrubriken inte att returneras.

B.1.2 fetch_preface(node)

Hämtar ingressen till noden med numret `node`. Det måste finnas en nod med numret `node` och `node` måste vara ett nummer. Om artikeln är gömd och den nuvarande användaren inte har access till den kommer ingressen returneras.

B.1.3 fetch_body(node, picargs)

Samma sak här, `node` måste vara ett nummer och referera till en giltig nod. Om artikeln är gömd och den nuvarande användaren inte har access till den kommer huvudrubriken inte att visas. Är det en textartikel kommer artikelkroppen att returneras, är det en bildartikel kommer en bild-tag (`<img src=bild>`) med en giltig referens till bilden att returneras. Om artikeln är osynlig och användaren varken har artikel- eller trädrättigheter så får användaren inte något tillbaka. `picargs` är vad som skjuts in i `<img>`-taggen för att kunna skicka med argument tillbilden, t.ex. höjd och bredd förhållanden.

B.1.4 fetch_reference(node)

Hämtar referensen till den externa filen, förutsatt att `node` är en giltig siffra och referens till en existerande nod. Om artikeln är osynlig och användaren varken har artikel- eller trädrättigheter så får användaren inte se referensen.

B.1.5 node_is_picture(node)

Om `node` är en giltig siffra och dess värde representerar ett existerande artikel-id så returnerar den här funktionen `true`, sant, ifall den valda artikeln är en bildartikel. Om det inte är en bildartikel, kommer funktionen returnera `false`.

B.1.6 menu_have_children(node)

node måste vara en giltig siffra och referera till en existerande artikel. Returnerar true om den valda artikeln har underartiklar, annars returneras falskt.

B.1.7 menu_init(node)

Skapar en artikelmeny kring node. Om node är en tom sträng så skapas en meny utifrån roten av trädet, om den inte är en tom sträng så måste node referera till en gitlig nod i trädet, och dessutom vara en siffra. Returnerar en menu_handler, ett menyhandtag. Menyhandtaget är alltså en identifierare för en meny.

B.1.8 related_menu_init(node)

Skapar en meny med relaterade artiklar till den vald i node. node måste finnas och det måste vara en siffra. Den returnerar max 10 relaterade artiklar. Funktionen skickar tillbaka är ett menyhandtag.

B.1.9 menu_exit(menu_handler)

Avslutar en meny. Efter detta går det inte längre att använda menu_go_next(menu_handler), eller någon annan av menyfunktionerna på menyhandtaget. menu_handler måste vara ett giltigt menyhandtag.

B.1.10 menu_go_next(menu_handler)

Flyttar fram till nästa menyelement. menu_at_end får inte vara true före anrop till denna funktion.

B.1.11 menu_get_current(menu_handler)

Hämtar numret till nuvarande menyelement. Funktionen menu_exit får inta ha använts på menyhandtaget innan den här funktionen används. Givetvis måste menu_handler vara ett giltigt menyhandtag.

B.1.12 menu_at_end(menu_handler)

Talar om ifall slutet på listan är nådd. Om anropet returnerat true ska inte funktionen menu_go_next användas för att hämta nästa element, dock ger menu_get_current det sista elementet i menyn.

B.1.13 get_rootpath(node)

Hämtar en sökväg ifrån roten till den valda noden i `node`. Noden måste finnas och `node` måste vara en siffra. Sökvägen ser ut så här `1/2/3`, där `1` är första undermenyn, `2` andra osv sett ifrån roten.

B.1.14 get_depth_of_rootpath(rootpath)

Ger djupet av sökvägen till roten i `rootpath`. `rootpath` måste vara en giltig rotsökväg skapad av `get_rootpath`. Ex, är rotsökvägen `1/2` returneras `2`.

B.1.15 get_node_at_depth(depth, rootpath)

Hämtar noden vid djup `depth` ur `rootpath`. `depth` måste vara ett giltigt värde som inte överstiger djupet av rotsökvägen eller är mindre än `0` och `rootpath` måste vare en giltig rotsökväg.

B.1.16 is_in_path(node, rootpath)

Talar om ifall `node` finns i `rootpath`. `rootpath` måste vare en giltig rotsökväg. Returnerar sant om `node` fanns i `rootpath`.

B.1.17 access(resource)

Talar om ifall den nuvarande användaren har tillgång till `resource`. `resource` måste vara en av `"nodes"`, `"tree"`, `"users"` eller `"keys"` för att tala om ifall den inloggade har access till artiklar, trädet, användarna respektive nycklarna.

B.1.18 get_priority(node)

Om användaren har access till trädet får den veta vilken prioritet som `node` har. `node` måste vara en giltig siffra och noden måste existera.

B.1.19 is_visible(node)

Om användaren har access till trädet får den veta om `node` är synlig eller inte. `node` är ett giltigt värde och dess värde är en giltig referens till en artikels id.

B.1.20 myBool(is_on)

Returnerar sant, `true`, om `is_on` är antingen `"on"`, `"true"` eller `"checked"`. Används tillsammans med formulär.

B.2 Funktionsbiblioteket ”library_add.asp”

Det här biblioteket används för att ändra i systemet. Följande funktioner finns tillgängliga.

B.2.1 auth(user, passwd)

Loggar in en användare. Matchar `user` och `passwd` mot databasen. Om de stämmer, är användaren inloggad. Returnerar true om det gick bra, annars false.

B.2.2 node_add(heading, preface, ref, is_picture)

Lägger till en ny nod. `heading` och `preface` borde ha gått genom funktionen `header_and_ingress_encode` innan anropet till den här funktionen, för att vara säker på att det fungerar. `Ref` är en namnet på den externa referensfilen, `is_picture` markerar ifall den är en bild, och måste alltså vara en booleskvariabel. Det går bara att lägga till en nod om den inloggade användaren har artikelrättigheter (`”nodes”`). Returnerar `true` om det gick bra, annars `false`.

B.2.3 saveBody(body, appendix)

Sparar en `body` till hårddisken i en fil med ändelsen appendix. Filnamnet skapas unikt utifrån tidpunkten då anropet gjordes. Den inloggade måste ha artikelrättigheter för att kunna använda funktionen. Returnerar namnet på den sparade filen.

B.2.4 menu_nodes_with_no_keywords()

Returnerar ett menyhandtag till en meny med alla artiklar som inte har något nyckelord tilldelat. Den inloggade måste ha antingen artikel eller nyckelrättigheter för att använda den här funktionen.

B.2.5 menu_keywords()

Returnerar ett menyhandtag till en meny med alla nyckelord som finns. Det gör att `menu_get_current` inte returnerar ett nod-id, utan ett nyckel-id. Den inloggade måste ha artikel eller nyckelrättigheter för att kunna använda den här funktionen.

B.2.6 menu_keywords_to_node(node)

Returnerar ett menyhandtag till en meny med alla nycklar som finns knytna till `node`. `Node` måste vara ett värde och en giltig referens till en existerande nod. Som med `menu_keywords` så är inte den här listan av nodeids, utan nyckelordids. Den inloggade användaren måste ha artikel eller nyckelrättigheter för att få använda den här funktionen.

B.2.7 assign_key_to_node(key, node)

Tilldelar en nyckel till en nod. `Key` är ett värde som refererar till en existerande nyckel, och `node` är ett värde som refererar till en existerande artikel. Den inloggade användaren måste ha nyckel eller artikelrättigheter för att kunna använda den här funktionen Returnerar true om det gick bra, annars false.

B.2.8 get_keyword(keyid)

Hämtar ett nyckelord. `keyid` måste finnas och den inloggade användaren måste ha antingen nyckel eller artikelrättigheter för att kunna anvädna den här funktionen.

B.2.9 assign_node_to_node(node, to_node)

Knyter `node` till `to_node`. Nu kommer `node` att finnas under `to_node` i menyträdet sedan. Den inloggade måste ha trädrättigheter för den här funktionen, och som vanligt måste `node` och `to_node` vara giltiga värden och referenser till existerande noder. Returnerar `true` om det gick bra, annars `false`.

B.2.10 make_node_to_root(node)

Tar en nod och flyttar den till roten på trädet. `Node` måste existera och vara ett giltigt värde. Den inloggade måste ha trädrättigheter för den här funktionen. Returnerar `true` om det gick bra, annars `false`.

B.2.11 delete_node(node)

Raderar noden ifrån databasen och tar bort den externa filen om det inte längre finns någon som refererar till den. `node` måste givetvis finnas innan, och vara ett värde, efteråt finns den inte. Det krävs att den inloggade användaren har artikelrättigheter för att använda den här funktionen. Returnerar `true` om det gick bra, annars `false`.

B.2.12 set_priority(node, newpriority)

Sätter en ny prioritet på en artikel. `node` och `newpriority` är båda giltiga heltalsvärden, men `node` är också en referens till en existerande artikel. Den som önskar använda den här funktionen behöver trädrättigheter. Returnerar `true` om det gick bra, annars `false`.

B.2.13 set_visability(node, visability)

Ändrar synligheten av en artikel. `node` är ett giltigt värde och en referens till en existerande artikel. `visability` är antingen `true` eller `false`, eller en booleskvariabel. Den här

funktionen vill att den inloggade användaren har trädrättigheter för att tillåta dess funktion. Returnerar `true` om det gick bra, annars `false`.

B.2.14 remove_allkeys_from_node(node)

Raderar alla nycklar ifrån en artikel. `node` är ett giltigt värde och den existerar som en artikel. Användaren som är inloggad måste ha nyckel eller artikelrättigheter för den här funktionen. Returnerar `true` om det gick bra, annars `false`.

B.2.15 new_key(key)

Skapar en ny nyckel. `key` är en textsträng. Kräver nyckelrättigheter. Returnerar `true` om det gick bra, annars `false`.

B.2.16 delete_key(keyid)

Raderar en ny nyckel. `keyid` är en giltig referens till en existerande nyckel. Kräver nyckelrättigheter. Returnerar `true` om det gick bra, annars `false`.

B.2.17 add_user(username,passwd)

Skapar en ny användare med inga rättigheter, användarnamnet `username` och lösenordet `passwd`. Kräver användarrättigheter, och att `username` inte redan existerar. Returnerar `true` om det gick bra, annars `false`.

B.2.18 set_user_privilages(userid, nodes, tree, users, keys)

Sätter en användares rättigheter. `userid` är ett giltigt `userid` och `nodes`, `tree`, `users` och `keys` är alla antingen `true`, `false` eller en boolesk variabel. Kräver användarerättigheter. Returnerar `true` om det gick bra, annars `false`.

B.2.19 set_user_passwd(oldpwd, newpwd, newpwd2)

Uppdaterar den nuvarande användarens lösenord. Om `oldpwd` är det som användaren hade förut och `newpwd` och `newpwd2` är identiska, kommer användarens lösenord att ändras. Kräver att en användare är inloggad. Returnerar `true` om det gick bra, annars `false`.

B.2.20 remove_user(user_id)

Raderar en användare. `user_id` är en giltig siffra och ett giltigt användareid. Den inloggade behöver användarerättigheter för att utföra den här funktionen. En boolesk variabel returnerar som reflekterar om det gick bra eller inte.

B.2.21 force_new_user_password(userid, newpwd1, newpwd2)

Tvingar på `userid` ett nytt lösenord, om `newpwd1` och `newpwd2` är identiska. Dessutom måste `userid` existera och vara en giltig användareid. Vill att den inloggade användaren har användarerättigheter.

B.2.22 get_username(user_id)

Hämtar användarnamnet på `user_id`, som måste vara ett giltigt värde och en existerande användare. Kräver användarerättigheter.

B.2.23 menu_users()

Skapar ett menyhandtag med alla användare. Värdena som fås ifrån `menu_get_current` på det här handtaget är alltså inte referenser till artiklar, utan snarare till användare. Kräver användarerättigheter.

B.2.24 have_access(user, resource)

Talar om ifall `user`, som är en siffra och ett giltigt användarid, har tillgång till `resource`. Resource måste vara `nodes`, `tree`, `users` eller `keys`. Funktionen fungerar inte utan att den inloggade användaren har användarrättigheter.

B.2.25 logged_in()

Returnerar `true` ifall det finns en inloggad användare.

B.2.26 exists_user(user)

Talar om ifall `user`, som är ett användarenamn, existerar eller inte. Den inloggade användaren ska ha användarrättigheter för att funktionen ska fungera. Om det inte gick bra så skickar funktionen `false` tillbaka.

B.2.27 update_node(node_id, new_heading, new_preface, new_ref, new_is_picture)

Uppdaterar en artikel med `node_id`, som är en giltig `node_id`, med en ny rubrik, ingress och ny referensfil. Samma gäller här annars som för `node_add` (B.2.2)

B.2.28 header_and_ingress_encode(str)

Gör så att str går att använda som huvudrubrik och ingress. Den här funktionen bör användas till `header` och `preface` parametrarna vid anrop till `update_node` eller `node_add`.

B.3 Menybiblioteket menu.asp

Den här filen innehåller en par offentliga funktioner nämligen de här:

B.3.1 make_menu_recurse(menu, node, refstring)

Skapar en rekursiv meny utifrån `menu`, som är ett giltigt menyhandtag till en artikelmeny. Node är den nod menyn ska vikas ut kring, `refstring` är det som ska klistra in före värdet på noden i länkankaret. Alltså

```
<a href=refstring?nummer>
```

Där `nummer` är numret på den artikeln, värdet till noden, som just då länkas till.

B.3.2 make_menu(selected_node, refstring)

Skapar en meny kring `selected_node`, med `refstring` som referenssträng. Se ovan för en beskrivning på hur den fungerar.