Sammendrag for hovedprosjekt

Tittel:	Project Flash	Nr. :
		Dato : 23. Mai 2001
Deltaker(e):	Ragnhild Hammerstad	
	John Veflen	
	Geir-Olav Goksøyr	
	Stian Torp	
Veileder(e):	Øivind Kolloen	
Oppdragsgiver:	I|O Publishing Technology A/S	
Kontaktperson:	Marek Bieler	
Stikkord (4 stk)	Dynamisk Nettavis i Flash	

Antall sider: 131	Antall bilag: 13	Tilgjengelighet (åpen/konfidensiell): Åpen

Sammendrag: Se neste side.

Nødvendige opplysninger:
Prosjektets webside: http://prosjekter.hig.no/flash/
Brukernavn: ”IO” Passord: ”Flashsun”
Bruker-delen: http://prosjekter.hig.no/gei_goks/index.html
Administrator-delen: http://prosjekter.hig.no/gei_goks/admin.html
Administrator-del:
Sjef:
Brukernavn: ”okolloen@hig.no”
Passord: ”hamstere”
Ansatt:
Brukernavn: ”ing.jens@flashavisen.no”
Passord: ”hamstere”

Sammendrag

Oppdragsgiveren for dette prosjektet var I|O Publishing Technology. De ga oss i oppgave å finne ut hvordan informasjon som ligger i en database kan vises og oppdateres i en dynamisk flashside. Hvis vi fikk tid skulle vi også se på hvor godt egnet Flash er til å vise bilder i en dynamisk flashside sammenlignet med HTML.

De ønsket også at flashsiden skulle være en dynamisk nettavis og at vi drøftet denne avisen i forhold til en lignende utgave i HTML. Etter en del undersøkelser fant vi ut at det var best å bruke ASP i kommunikasjonen mellom serveren og Flash, men oppdragsgiveren så helst at vi også tok inn XML i denne delen. For å kunne gi svar på problemstillingen måtte vi da lære oss XML og ASP i tillegg til Flash.

Vi startet med å lage databasen og ble enige om strukturen i flashavisen. Der etter lagde vi et forprosjekt der deler av denne strukturen og databasen ble brukt i en dynamisk nettavis i ASP og HTML. Dette forprosjektet var en selvdefinert obligatorisk oppgave i et annet fag vi hadde i samme periode. Denne ASP-avisen delte vi i to deler, en for ansatte og en for brukere av avisen.(*se vedlegg L: statusrapport 2*). Samme inndeling som i ASP-avisen, brukte vi også i flashavisen, delen for ansatte har vi kalt for Admin-siden og brukersiden har vi gitt betegnelsen Index-siden.

I Admin-siden bruker vi ASP i kommunikasjonen mellom Flash og serveren. Det er i denne siden gitt svar på hele problemstillingen og vist hvordan data fra Flash kan lagres i en database og hentes opp igjen for redigering. Dette har vi demonstrert ved at det er mulig å lagre artikler i databasen og hente dem opp igjen for redigering. Det er også mulig å legge inn andre deler som er tilknyttet til artikler og data som er nødvendig i administrasjonen for en nettavis. Vi har også vist kommunikasjonen på flere andre måter blant annet i dynamisk dropdown meny og i en dynamisk meny.

Det er brukt XML i tillegg til ASP i Index-siden. I denne delen hentes informasjon opp fra databasen og presenteres for brukere av avisen. Her blir ingresser for de siste artiklene og linker til de tilhørende artiklene lagt ut på forsiden. Ingressene og artiklene er også sortert under kategorier som kan velges i en dynamiske menyen. Kommunikasjonen vises også i en liste over de fem siste nyhetene og ved at relaterte artikler blir vist under den som er valgt. I denne delen er det i tillegg laget en funksjon som demonstrerer kommunikasjon begge veier, ved at det er mulig å søke etter artikler i databasen.

(Flytskjema for Index.swf s... og Utførelse Admin-siden s. 43)

Vi har brukt ASP i Admin-siden og både ASP / XML i Index-siden og har derfor gitt en utmerket besvarelse og vist to metoder for hvordan kommunikasjon mellom Flash og en server kan løses. Denne kommunikasjonen har vi vist på flere måter, slik at flashavisen ligner på en vanlig dynamisk nettavis. Vi har oppfylt målet og gitt et meget god svar på problemstillingen og oppdragsgiveren er fornøyd.

På grunn av mangel på programvare og ressurser har vi ikke fått lagt inn bilder, lyd og video i Flashavisen, men vi har etter ønske fra oppdragsgiveren sett på mulighetene og funnet ut hvordan det kan gjøres. (*se Alternative løsninger og valg under veis s. 62)*

Vi har drøftet Flashavisen i forhold til ASP-avisen og har kommet frem til en rekke punkter. Når bilder blir tatt med så vil vi på grunn av de store fordelene Flash har for design, anbefale en kombinasjon der HTML brukes i Admin-siden og Flash i Index-siden.

(*se ASP-avisen kontra Flashavisen s. 61*)

Forord

Vi vil først få takke oppdragsgiver I|O Publishing Technology for en utfordrende og lærerik oppgave. Våre kontaktpersoner var i denne forbindelse Marek Bieler og Tomas Andre Jørgensen. De ga oss mange gode tips og råd, som kom til nytte under hele prosjektet.

Den gode kritikken av betaversjon fra kontaktpersonene under siste møte, ga oss flere innspill til forbedringer av Flashavisen. Vi er også veldig glade for at de fulgte oss opp med flere møter i bedriften. Disse møtene var med å motivere til raskere fremgang i prosjektet. Oppdragsgiver har også hjulpet oss mye, ved å dekke de fleste og største utgiftene vi har hatt i forbindelse med hovedprosjektet.

Til slutt vil vi få lov til takke Øyvind Kolloen, som var vår veileder på høgskolen har han bestandig stilt opp med godt humør og hjulpet oss når vi trengte det. Han var spesielt til stor hjelp da vi skrev ASP-filene, og kritikken han ga av betaversjonen, førte også til at vi fikk et bedre slutt produkt.

Innholdsfortegnelse

Innledning

Organisering av sluttrapporten

Denne delen av rapporten handler om det som ble gjort før selve gjennomføringen av prosjektet. Fasen har vi kalt for planlegging og strategi del. Her blir problemstillingen belyst og ønsker som oppdragsgiveren har i forhold til prosjektet. Det svar på oppgaven som vi har kommet frem til er å finne under konklusjon. Det er skrevet to mottakeranalyser som kommer inn under denne delen i rapporten. Vi ser også litt på organisering av arbeidet og hva vi måtte lære for å kunne løse denne oppgaven.

Andre del kommer inn under hovedkapitler. Her blir selve gjennomføringen av produktet beskrevet. Valg av utstyr, program og programmerings språk. Beskrivelse av struktur og funksjon for de forskjellige delene i nettavisen. Det er også laget oversikt over hvor de forskjellige filene hører til i nettavisen.

I den avsluttende delen står konklusjonen, der peker vi på de svarene vi har kommet frem til og analysere dette i forhold til problemstillingen. Det vi har kommet frem til blir etter oppdragsgiverens ønske, drøftet i forhold til en mer vanlig løsning. Vi kommer også med en kritikk av prosjektet og ser på hva som kunne vært gjort på en annen måte. Til slutt blir det vurdert om det er mulig å videreføre hovedprosjektet.

Siste del er evaluering av gruppens arbeide. Her blir organiseringen av prosjektet beskrevet. Det blir vurdert om den løsningen vi valgte for gjennomføringen av prosjektet var tilfredsstillende eller om ting kunne vært gjort på en annen måte. Til slutt har alle i gruppen skrevet en egen evaluering over hvordan de opplevde hovedprosjektet.

Prosjektbeskrivelse

Vår oppdragsgiver er I|O Publishing Technology. Firmaet har en lang kundeliste og produserer større websider, de har blant annet laget startsiden sol og websiden for Aftenposten. Dette er et firma som har mye å gjøre og har planer om å utvide med flere avdelinger. De har derfor liten tid til forsøk og innovasjon. Vår oppgave blir å prøve ut en ny måte å lage dynamiske sider med en database i bunnen.

Det absolutte krav de stilte til oss er at vi lager en dynamisk flashside med database i bunnen. De er mest interessert i kommunikasjonen mellom serveren og flashsiden, samt kommunikasjon mellom databasen og serveren. Problemet blir å løse dette på en god måte og drøfte det vi kommer frem til i forhold til en dynamisk htmlside.

Det andre kravet de stilte er at vi lager gode muligheter for oppdatering av denne dynamiske flashsiden. Det skal være enkelt for eieren av flashsiden å forandre innholdet i den. Hvis det for eksempel er en journalist som har ansvaret for å oppdatere siden, skal vedkommende kunne slette artikler og legge til nye i databasen på en effektiv måte.

Problemstilling

Hvordan kan informasjon som ligger i en database, vises og oppdateres i en dynamisk flashside.

Oppdragsgiveren ønsket at vi lagde en dynamisk flashside som skulle fungere som en nettavis, men det var ikke et krav. Under det første møtet med I|O kom vi frem til at dette var en grei oppgave, som vi kunne mestre. Vi ble også enige om å lage en mal for visning og innlegging av informasjon i flashsiden.

Det skal drøftes ulemper og fordeler med den flashsiden vi kommer frem til, i forhold til en lignende side i html. Det skal da funksjonelt begrunnes hvorfor den ene løsningen er bedre enn den andre. Hvis vi fikk tid skulle vi også se på hvor godt egnet Flash er til å vise bilder sammenlignet med HTML.

Målgruppe

Vi har to målgrupper for oppgaven. Den ene er oppdragsgiver og den andre er brukere av nettavisen. For sluttrapporten kan vi også se på sensor som målgruppe, uten at vi skal utdype dette nærmere. Vi har valgt å skrive en liten mottaker analyse for nettavisen vi skal lage i flash, for å få oversikt over brukerens ønsker og krav til en slik side. Den andre analysen vi har skrevet, var for å få et innblikk i hvilke forventninger oppdragsgiver kan ha for prosjektet og sluttproduktet.

Mottakeranalyse over brukere av nettavisen

Mottakergruppen er stor og variert, det vil være alle som bruker nettet for å lese avis. Det er stadig flere som tar i bruk nettet for en slik hverdags ting som å lese avisa. Vi kan tenke oss at det er funksjonalitet og brukervennlighet de setter sterkest krav til, fordi de fleste vil lese avisen raskt og lett. De vil også kreve at siden er daglig oppdatert.

Det skal være lett å finne frem på siden, derfor må menyvalgene settes opp hierarkisk og markert. På toppen av dette hierarkiet vil vi anta at de forventer å finne at sammendrag over de siste nyhetene. Deretter ville nok de fleste ønske innenriks og utenriks nyheter. Andre menyvalg som vi vil anta at avisleserne vil ønske er sport, kultur og økonomi.

(*se Vedlegg F: Flytskjema for brukerdelen*)

For at mottakerne skal ta i bruk avisen må de identifisere den med en seriøs nettavis. Det skal ikke være rom for tvil, om at dette virkelig er en avis med alle de viktige nyhetene. Nettavisen skal også sende tydelige signaler om at dette er en avis som følger med i markedet og tar i bruk flere medieelementer. Ved å bruke flere kommunikasjons måter, slik som lyd og video, kan avisen nå mange brukere og heve seg i konkurranse med andre nettaviser.

Det er viktig at avisens design er behagelig , uten for mange kontraster, bevegelser og andre forstyrrende elementer. Hvis det er for mange blikkfang blir avisen ubehagelig å lese, og da vil brukeren velge en annen avis. Elementene i siden må stå symetrisk eller asymmetrisk, slik at de skaper et balansert helhets inntrykk som er behagelig å se på.

Mottakeranalyse over oppdragsgiver

Under første møte med oppdragsgiveren, ble det fremmet en del ønsker knyttet til sluttproduktet og rapporten. Deler av denne analyse er formulert ut fra stikkord som ble gitt under dette og noen sener møter.

I forhold til koden vi skal skrive så var det viktig at den ble formulert på en forståelig måte med gode variabel navn. Det skal være mulig for en utenforstående å forstå de forskjellige kildekodene. Det er viktig at det er lett å finne fram i det vi har laget, slik at de kan bruke det som eksempel, hvis de selv skal lage en dynamisk flashside. I den forbindelse må sluttrapporten være godt strukturelt med gode forklaringer, slik at de får en forståelse for hvordan elementene i sluttproduktet henger sammen.

Det vi kan tenke oss er viktigste for oppdragsgiveren er at vi finner svar på problemstillingen. Resultatet skal også drøftes, der vi belyser problemer og forandringer som er gjort i forhold til problemstillingen. Andre deler de kan være interessert i, er at vi finner annerledes måter å presentere ting på, enn slik det blir gjort i en html-side. Vi må peke på det som er positivt og negativt med en dynamisk flashavis kontra en dynamisk nettavis i HTML. (*se ASP-avisen kontra Flashavisen s. 61*).

De er ute etter en løsning for dynamiske websider som er bedre enn den de bruker nå slik at de kan tilby kundene dette produktet og kapre et større marked. I denne forbindelsen er de mest interessert i de funksjonelle svarene vi kommer frem til. Vår utfordring blir å finne ut om det går an å lage en slik dynamisk nettavis i flash og vurdere om dette er en bedre løsning.

Valg av oppgave

Det første målet for hovedprosjektet var å få en databedrift som oppdragsgiver. Dette var fordi vi ville ha en oppgave som var tilpasset markedet. Andre motivasjoner som lå bak var å få litt flere opplysninger om slike bedrifter, som vi har tatt utdannelse for å jobbe i. Vi var interessert i å lære litt om kulturen og organiseringen i bedriften. Denne kunnskapen kan gi motivasjoner for et bedre resultat i det siste semestret, fordi vi ser om det vi lærer på skolen er ettertraktet i jobbsammenheng.

For eksempel så bruker oppdragsgiveren XML i de større dynamiske websidene de lager og sannsynligvis er det flere bedrifter som benytter denne løsningen. Dette har kanskje bidratt i at en del av oss har brukt mer energi på de to fagene XML og server side programmering, enn det de ellers hadde gjort i dette semestret. Tips som oppdragsgiveren gir oss om hva slike bedrifter ser etter og krever for ansettelse, er også nyttig kunnskap når vi skal søke jobb.

Vi mailet til flere databedrifter for å finne oppdragsgiver og noen tilbud fikk vi. Den oppgaven i valgte skulle utføres i Flash, et program som stadig flere webprodusenter tar i bruk. De fleste i gruppen hadde lyst til å lære seg Flash, fordi det virker som et spennende program med mange muligheter. Det å kunne beherske dette programmet er også en ettertraktet kunnskap blant flere data bedrifter og kan derfor bidra i at vi letter får jobb.

Flash er mest brukt for å gi en fin design med fancy animasjoner, men vi skulle prøve ut en annerledes måte å benytte programmet, ved lage en dynamisk side med en database. Vi ble motiverte for å prøve ut noe nytt og finne løsninger for hvordan toveis kommunikasjon mellom Flash og en server kan utføres.

Hva måtte læres

Vi har få kunnskaper på forhånd om hvordan denne oppgaven skal løses. Det eneste vi kan som er relevant for selve utførelsen av flashsiden er databaser, som er et av de fagene vi har hatt i dette studie. Denne databasen skal brukes for å lagre artikler og annen relevant data i avisen.

Etter en periode med undersøkelser ble vi enige om å bruke ASP og XML i kommunikasjonen mellom Flash og serveren. For å kunne utføre oppgaven måtte da vi lære oss Flash, ASP og XML, før vi kunne planlegge selve gjennomføringen av flashsiden.

(*se vedlegg K: Statusrapport 1*). De to siste delene XML og ASP kom inn under fag som vi har i dette semesteret.

Vi har god erfaring om denne arbeids måten, fra flere andre prosjekter i studiet. Alle gruppe medlemmene har også jobber sammen i et annet prosjekt i "Elektronisk publisering" som er et av fagene vi har denne våren, dette ga oss erfaringer om hvordan gruppe medlemmene fungerte sammen. Noen av medlemmene har også vært i samme gruppe i en rekke andre prosjekter.

Arbeidsformer

Prosjektgruppen har valgt å hente mye inspirasjon og kunnskaper fra internett. Vi brukte denne kilden fordi det er her den nyeste kunnskapen ligger og den måtte vi ha for å løse oppgaven. I den første fasen av prosjektet fant vi på Internett noen eksempler for hvordan kommunikasjonen mellom Flash og serveren kunne løses.

Vi lette også på nettet etter dynamiske flashsider, men fant ingen som lignet på den vi skulle lage. Fagboken i Flash (*litt. nr. 1 s. 71.*) som vi har brukt mest, har vi bestilt på nettet fra USA. I de norsk bokhandlere som vi besøkte var det ingen bøker i Flash som tok for seg avansert dynamisk flash. Under gjennomføringen av flashsiden brukte vi ofte Internett for å finne svar på problemer som dukket opp.

For eksempel så brukte vi noen dager for å finne ut hvordan vi skulle få en dynamisk meny til å virke i flash og fant til slutt svaret på en webside. Før vi startet på gjennomføringa fasen for nettavisen så brukte vi Internett for å finne ut hvordan en vanlig struktur for en slik side er bygd opp. Linker til en del av de websidene vi besøkte ligger på prosjektets webside "http://prosjekter.hig.no/flash", men en oversikt over alle websidene vi har brukt angående prosjektet er å finne i litteraturlisten.

Organisering av prosjektet

Prosjektet har vi delt opp i flere faser. Vi startet med å komme frem til en problemstilling og vårt hovedmål ble å finne svar på dette problemet. I den neste fasen planla vi hvordan dette prosjektet skulle utføres og hvilke kunnskaper som var nødvendig å lære.

Vi kom frem til måter for hvordan vi kunne finne svar på problemstillingen og lære den kunnskapen som var nødvendig. (*se vedlegg K: Statusrapport 1*). Ansvarsforholdene for administrasjon og faglig del ble fordelt innad i gruppen. (*se Ansvarsforhold, s. 14*). Det ble lagd en tidsplan i form av et gantt-diagram. I dette diagrammet har vi fordelt timer i forhold til det som skal planlegges og utføres i prosjektet.

Møtetidspunkt med oppdragsgiver er også satt inn i denne tidsplan. I den neste fasen som vi har kalt strategi fasen, tilegnet vi oss den kunnskapen som var nødvendig for å kunne gjennomføre oppgaven. Det ble benyttet metoder som vi hadde kommet frem til i planleggings fasen for å lære det vi skulle.

De to fasene planlegging og strategi faller litt inn i hverandre, dette er fordi vi fikk noen nye ideer etter hvert som vi lærte og det krevde litt mer planlegging. I fasen for gjennomføring fordelte vi ansvarsforholdene på nytt, slik at vi fikk ansvar for å gjennomføre hver vår del av flashsiden. I forkant av denne fasen la vi planer for gjennomføringen og siden vi hadde et prosjekt i "klient og serverside programmering" der vi skulle programmere mot en server. I denne oppgaven passet det godt å lage en nettavis i ASP og HTML, som kunne fungere som et forprosjekt for hovedprosjektet. (*se vedlegg L: Statusrapport 2*)

Ansvarsforhold

Ansvars forholdene er delt i administrasjons oppgaver, faglige områder og ansvar for gjennomføring av flashsiden. En av de administrative oppgavene er gruppeleder som må påse at møtereferater og at de aktuelle rapportene blir skrevet. Vedkommende har også ansvar for at gruppen fungere og at de nødvendige oppgavene blir utført. Gruppeleder kan også ta beslutninger der gruppe medlemmene ikke kommer til enighet.

Den andre administrative oppgaven er økonomisk ansvarlig, som holder orden på utgifter og inntekter i form av det oppdragsgiver var villig til å dekke av utgiftene. De faglige områdene vi fordelte var en for databasen, der den ansvarlige fikk i oppdrag å lagde databasen ut fra skisser vi kom frem til på gruppemøter. En annen faglig oppgave gikk ut på sette seg inn i sammenhengen mellom ASP og Flash i forhold til prosjektet.

Den tredje faglige oppgaven var å sette seg inn i XML i forhold til Flash. Den siste oppgaven i denne kategorien var ansvaret for flashsiden og se på mulighetene for å utføre en nettavis i Flash. Under gjennomføringen fordelte vi flashavisen i to deler, en del for admin og en for bruker. Admin-siden er beregnet for ansatte i flashavisen. Det er i denne delen det er mulig å lagre og redigere artikler og andre data som er nødvendig i en slik avis. I siden for brukere av avisen blir data hentet opp fra databasen og presentert på skjermen.

(*se Vedlegg F: Flytskjema for brukersiden, Utførelse Admin-siden s. 42 og Vedlegg L: Statusrapport 2*)

Fordeling av oppgaver

- Gruppeleder: Ragnhild Hammerstad
- Økonomisk ansvarlig: Geir-Olav Goksøyr

- Database / server: Stian Torp
- ASP (database / server): John Veflen
- XML (Flash- server): Geir-Olav Goksøyr
- Flash: Ragnhild Hammerstad

- Admin-siden: John veflen og Ragnhild Hammerstad
- Bruker-siden: Stian Torp og Geir-Olav Goksøyr

Utførelse

Design for utseende

Bakgrunnsbildet som er brukt i flashavisen er lagd i Photoshop og er satt sammen av flere små bilder som hører til de forskjellige artiklene i avisen. Det er meningen at bilde skal illustrere forskjellige nyheter. Bilde er gjennomsiktig med en lys gul bakgrunn slik at det gir følelsen av gammel brevpost, men det er den nyeste teknikken som ligger bak. I Flash ble bilde gjort om til en av fargevalgene, slik at Paint Bucket kunne benyttes for å fylle fargen i en form, som på forhånd var laget i Flash.

Knappene er som linjer på papir og er med å understreker dette estetiske bilde. Når vi skriver på papir så benytter vi ofte gul markerings tusj på viktige ord eller de strekes under med rød penn. Dette gjenskaper vi i flashavisen ved at knappene skifter farge til gult eller får en rød markering rundt teksten, når markøren er over dem eller knappen trykkes ned. Designet kan på denne måten virke beroligende på brukeren, slik at vedkommende får følelsen av at dette er noe kjent.

Fargene i designet er duse og det er få sterke kontraster, slik at det lettere for brukeren av avisen å konsentrere seg om teksten i artiklene. For å gi en god leselighet har vi brukt svart skrift mot hvit bakgrunn. Designet er oversiktlig og det er enkelt å finne frem. Vi har for eksempel med hensikt unngått undermeny i menyen, slik at den er lett å bruke.

Krav til prosjektet

- Det endelige produkt skal være laget i Flash og være dynamisk oppdaterbart.
- Det skal ligge en database i bunnen av applikasjonen.
- Sidene skal kunne oppdateres direkte fra internett. Dette gjelder både endring av eksisterende innhold, samt innleggelse av nytt.
- Det skal være brukt XML til overføring av data.
- Vi skal drøfte fordeler og ulemper i forhold til et tilsvarende produkt laget i HTML.
- Hvis tidsrammene tillater det skal muligheten for visning av lyd og bilder vurderes og eventuelt implementeres.
- Det skal ikke legges vekt på design og utforming av dette. Kommunikasjonen mellom flash og databasen er det som er mest interessant.

Tekniske spesifikasjoner

- Sidene skal kunne lastes ned over et 33.6 moden med akseptabel nedlastingstid.
- Sidene skal automatisk tilpasse seg skjermstørrelser mellom 800x600 og oppover.
- Anbefalt fargedybde er ”tusenvis av farger”.
- Programmeringsspråk som skal benyttes er ASP, XML, SQL og Actionscript.

Funksjonelle krav

- Dynamisk meny som leses inn fra XML. Her er det avsatt plass for maksimalt 12 forskjellige kategorier inkludert hovedsiden. Hvor mange av disse som benyttes bestemmes av administrator av applikasjonen.
- Når brukeren velger en kategori i menyen så skal vedkommende får opp en gruppe av ingresser for de artiklene som kommer inn under dette valget. I hver ingress skal det være en link til den aktuelle artikkelen.
- Overskriftene til de fem siste innlagte artiklene vises i eget vindu i grensesnittet.
- En søkefunksjon skal gjøre alle artikler i databasen tilgjengelig for bruker. Denne avgrenses ved at bruker kan velge ønsket kategori og tidsintervall på søket.

Det som skal foreligge når prosjektet er ferdig er en dynamisk nettavis laget i flash. Den skal inneholde to deler; en administrasjonsdel der avisens journalister kan redigere innholdet i avisen, og en brukerdel der avisen kan leses i sin helhet.

Sammenligning av endelig kravspesifikasjon mot kravspesifikasjonen pr.24.03.01.(vedlegg)

Den endelige kravspesifikasjonen for prosjektet har enkelte endringer i forhold til den første kravspesifikasjonen som ble levert 24.03.

Endringer i index.swf, brukersiden

- Det er oppført at det skal være mulighet for å legge til 3 ekstra valg på menyen. På grunn av at vi har lagt inn en menyknapp statisk (tilbake til førstesiden), er det nå bare mulighet for to ekstra knapper.
- Det står at de ekstra menyvalgene (kategoriene) kan legges inn og fjernes etter behag, for eksempel ved store begivenheter. Dette er ikke mulig. Valgene kan legges inn, men ikke slettes etter at det er lagt inn artikler i databasen under gjeldende kategori. Dette på grunn av relasjoner som blir opprettet i databasen koblet til de nye kategoriene.
- Klokkeslett er ikke tatt med på grunn av oppdateringsmuligheter. (*Nærmere forklart i "Script i index.swf" under Label: dato, s. 26*).
- Et tillegg til kravene i kravspesifikasjon 1 er relaterte artikler. Under hver artikkel vil eventuelle relaterte artikler bli listet ut med tittel og opprettet-tid. Tittelen fungerer her som link på samme måte som praksisen er ellers i avisen.
- Opprinnelig oppsett for søkefunksjonen er endret litt. Det var i utgangspunktet meningen at søkefunksjonens form skulle være tilgjengelig i hele avisen. Dette er endret. Det er opprettet en knapp i grensesnittet som tar brukeren med til søkefunksjonen. Denne ligger nå på kun en frame i filmen, men er tilgjengelig overalt gjennom nevnte knapp. (*Nærmere forklart i "Diskusjon av implementering", s. 20*)
- Et tillegg er 5siste. Dette gir bruker tilgang på direkte link til de fem siste nyhetene uansett hvor han befinner seg i filmen.(Nærmere forklart i "Script i index.swf" label: 5siste)
- Flashfilmen var beregnet å tilpasse seg skjermstørrelser mellom 600x480 og 1280x1024. Dette punktet er forandret til å gjelde fra 600x800 og oppover. Dette på grunn av at teksten i avisen er nesten uleselig i oppløsning 600x480. Oppad har vi ikke funnet noen måte å sette begrensning på tilpasningen til flashfilmen, selv om dette i utgangspunktet var intensjonen.

Endringer i Admin.swf, administrasjonssiden

- I kravspesifikasjonen står det at det skal brukes XML i overføringen av data. Dette har vi imidlertid kuttet ut på adminsiden, på grunn av at det er små mengder informasjon som overføres. Derfor syntes vi det var like greit å bruke bare ASP.
- Vi hadde tenkt at swf filmen skulle tilpasse seg skjermoppløsninger fra 640x480 til 1280x960, men vi fant ut at teksten ble for liten i 640x480. Oppad fant vi ingen måte å begrense størrelsen på.

Utførelse av brukerdelen

Diskusjon av implementering

Implementering av koden i flash ble vanskeligere en det gruppen i utgangspunktet hadde forestilt seg. Tidlig i prosjektet gikk vi til anskaffelse av en del lærebøker i flash og Actionscript. Disse har hjulpet mye, men likevel har vi støtt på svært mange vanskelige problem som det kostet adskillige dager å finne en løsning på. Vi savnet en veileder som kunne flash da dette hadde lettet implementeringen betydelig. I mangel på en slik rådgiver har vi oppsøkt en del newsgrupper og websteder for å få hjelp. Dette har i de fleste tilfeller vært nytteløst, men enkelte tips og triks har vi kunnet dra nytte av.

Mye av vanskelighetene vi har hatt skyldes for lite informasjon og læremateriell. Vi har flittig brukt help-funksjonen i flash, men vi har likevel hatt mange ubesvarte spørsmål som vi ikke har klart å finne noen løsning på før vi tilfeldigvis dumpet borti svaret selv. Dette har kostet oss adskillige ekstra arbeidstimer, og er hovedgrunnen til at vi fikk litt lite tid på slutten av prosjektet.

Feilmeldingsrutinene i flash kunne vært bedret. Det finnes en begrenset mengde feilmeldinger, for eksempel hvis det er en asp-fil flash ikke finner, eller om vi har glemt en {, men er feilen en annen får vi ingenting. Dette kan være litt frustrerende siden vi ikke får vite hva som er galt.

Måten vi løste dette på er ikke fullverdig, men det virket. Inne i scriptet vi jobbet med la vi inn “bokmerker” for å sjekke om det skjedde noe, og hvor det eventuelt skjedde. Vi opprettet et tekstfelt/variabel som vi skrev “bokmerkene” ut i. I tillegg skrev vi ut diverse variable som vi koblet til informasjon som hadde innvirkning på det aktuelle scriptet. På denne måten fant vi etter hvert ut hva som skjedde inne i flash, og fikk rettet feilen. Dette er en svært langsom måte å feilrette på, og når vi i tillegg ikke hadde oversikt over de innebygde funksjoner og objekt i flash og deres virkemåte, ble implementasjonen en tung prosess.

På den annen side har vi lært mye av denne måten å jobbe på. Vi har fått en innsikt i flash som vi ganske sikkert ikke hadde fått hvis vi hadde en rådgiver tilgjengelig som vi kunne spørre så snart vi fikk problemer.

Både slik systemet er nå, og under hele utviklingsfasen, har i tillegg ASP vært en akilleshæl. ASP-filene våre har til tider vært svært sårbare, noe både serveren og input til filene har vært årsak til. Et eksempel er at enkelte ASP-filer må ha “request.querystring” når de andre har virket utmerket med bare “request”. Det finnes flere lignende tilfeller som verken vi eller veileder har kunnet finne grunnen til. Tidvis har det virket som om Higstud5 buffrer filene våre. Skulle vi gjøre en liten forandring måtte vi døpe om filen, ellers fikk vi bare ut den samme filen uten de nye endringene. Dette ble slitsomt i lengden, men vi ble flinke til å finne på nye filnavn.

Utførelse

I brukerdelen hentes all informasjon inn via ASP-generert XML. Dette setter store krav til Actionscriptet som skal lese inn og parse koden inne i flash. I første instans jobbet vi mot statiske XML-dokument som vi prøvde å lese inn. Vi jobbet utifra et eksempel i en lærebok, men vi fant etter hvert ut at dette måtte omarbeides mye for å få det til å virke. Løsningen som vi endte opp med er ASP-filer som genererer XML "on the fly" og som leses rett inn i XML-objekt i flash. Deretter blir dataene formatert og lagt ut i et system av XML-objekt og arrayer. (*Nærmere forklart i "Script i index.swf", s. 26*). De forskjellige elementene i arrayene blir i tur og orden skrevet ut i dynamiske tekstfelt og presentert for bruker.

Dynamisk meny

Den dynamiske menyen var den mest krevende elementet i index.swf. Her leses alle kategorier som ligger i databasen inn via ASP/XML og legges i arrayer. Disse brukes så til å navngi og identitetsmerke movieclippet som dupliseres og legges ut på scenen når filmen lastes. (*Nærmere beskrivelse av koden, "Script i index.swf" s. 26*)

Vanskelighetene vi kom over i utviklingsfasen baserte seg i hovedsak på at vi ikke fikk noe feilmeldinger. Vi lot arrayen som kategoriene lå i bestå av objekter for letter å kunne legge inn identifisering og navn i form av "lapper". Eksempelvis "KategoriArray[ii].Name". Problemet var å koble MovieClipet som skulle utgjøre knappen til objektet. Når vi fikk skrevet ut menyen fikk vi ikke respons når vi prøvde å aktivisere knappene. MovieClipet ville ikke gi fra seg informasjonen som vi hadde lagt inn. Etter hvert fant vi den innebygde funksjonen Arguments[0]. Denne var med på å løse problemet. Den endelige løsningen var å sende med navnet på MovieClipet, ikke clipet sin id eller Name slik som først antatt, og la Arguments[0] snappe den opp. Nå fungerer menyen godt, men på grunn av den langsomme utviklinge kunne vi ønsket oss en bedre oversikt over de forskjellige inneygde funksjoner og objekter i flash en det vi har hatt tilgang til hittil.

Kategorier

Når scriptet som leser inn forsiden var ferdig, var det en grei jobb å skrive om dette til å skrive ut kategoriene. Vi har relativt mye overlappende kode, siden koden som brukes alle steder der XML leses inn har sterke likhetstrekk. Dette kunne muligens vært komprimert ved bruk av MovieClip, men siden kontroll av MovieClip er en ganske komplisert operasjon valgte vi å ha litt redundans i stedet.

Scriptet i kategori mottar en kategoriid fra menyknappen som er blitt aktivert og sender en forespørsel til databasen. Når data mottas blir denne formatert slik at hver tittel har funksjon som en link.

Artikkel og relatert

Begge disse har tilnærmet samme script som de foregående, men her er enkelte ulikheter. Artikkel er det eneste scriptet som skriver ut en artikkel, og dette kan trigges fra nesten alle de andre scriptene. Artikkel henter, i tillegg til selve artikelen, inn forfatternavn og forfatters mailadresse for så å skrive denne ut under artikkelen med "mailto"-funksjonalitet. Her kan altså bruker kontakte den aktuelle forfatter.

Artikkel og relatert henger sammen i den grad at trigges en kjøres begge, men de står i hver sin frame. Grunnen er at når vi prøvde å kjøre to forespørsler til to ulike ASP-filer i smme frame, fikk vi følgende feilmelding: "et script i flash kjører sakte. Vil du avslutte scriptet?" Velger brukeren "ja" på dette punktet kjører filmen videre uten styring og resultatet er ubrukelig. Velger bruker "nei" henger flash seg og må avsluttes med makt. For å unngå dette ble det brukt en frame til hvert script, og det fungerte bra.

Både Artikkel og relatert får Artid tilsendt og sender denne videre i en forespørsel til databasen. Artikkel får artikkelen og relatert får tittel og opprettet-tid på eventuelle artikler med samme tema som den aktuelle. Finnes det ikke relaterte artikler returneres tom XML.

Søk, søkartikkel og relatertsøk

Søk er nesten identisk med index-filen som skriver ut førstesiden. Den henter inn ingresser utifra bestemte søkekriterier som den får tilsendt fra bruker via formen i soekevalg. Disse ingressene skriver den selv ut med link til tittel på vanlig måte. Når en av disse linkene aktiveres sendes Artid til soekartikkel. Det er opprettet et eget script som kjører ut artikkelene og relaterte i søk. Dette er løst på denne måten for å unngå problemer med variable som må deles av flere script samtidig, samt for å unngå å miste informasjon på grunn av dette. Vi opplevde også her en del feilmeldinger med beskjed om at et script kjørte sakte, før vi fysisk splitta ut søkefunksjonen. Selv med en del ekstra redundans føler vi at vi her har fått til den beste løsningen i dette tilfellet.

Testing

Eksternt er prosjektet testet, da i beta, av vår oppdragsgiver. Veileder har også testet løsningen i tillegg av en del venner og bekjente av medlemmer i prosjektgruppa. Vi har valgt å ikke kjøre en større offentlig testsekvens. Grunnen til dette er at det føles unødvendig da Flashavisen, slik som den er nå, ikke er beregnet på et flerbrukersystem. Denne begrensningen ligger i databasen som er en Microsoft Access database. På grunn av dennes manglende kapasitet i flerbrukersystemer vil ikke avisen fungere tilfredstillende ute i markedet. MSAccess-databasen ble valgt utifra at det ikke var nødvendig med en større database siden avisen kun er et testprosjekt for å finne ut om teknologien er noe å satse videre på av vår oppdragsgiver. Ved implementering av, for eksempel, en SQL-database ser ikke prosjektgruppa noe i veien for at Flashavisen vil kunne brukes ute i markedet.

Internt er avisen testet fortløpende gjennom hele utviklingsperioden. Det ble så tidlig som mulig tidlig laget en løsning der vi fikk prøve ut kommunikasjonen gjennom alle ledd, helt fra databasen og ut i flash. Dette for å avdekke eventuelle kommunikasjonsproblemer mellom de forskjellige teknologiene.

Design kode

Her følger en forklaring til bruk av lag i filmen. Index.swf består av to scener: Innlasting og Hoved. Her forklares disse separat:

Hoved:

- Labels: Dette er navnelapper som er brukt til adressering innad i filmen. I tillegg letter de oversikten i scriptlaget.
- Script: Her har vi samlet alle framescript.
- Tilbakeknapp: Tekstknapp som brukes i søkefunksjonen f or å bringe bruker fra valgt artikkel og tilbake til resultat av søk.
- Logoanim: Dette er animasjonen som kjører i øverste venstre hjørne. Denne ligger som eget MovieClip.
- GåTilSøk: Knapp som henter frem søkefunksjonen.
- Sistefelt: Tekstfelt som viser de fem siste innlagte artikler.
- Menyknapper: Statisk menyknapp som tar bruker tilbake til forsiden.
- Kategorifelt: Dynamisk tekstfelt som inneholder informasjon om hvilken del av avisen bruker befinner seg i, for eksempel søk eller 5 siste.
- Datofelt: Dynamisk tekstfelt som viser dagens dato.
- Soek: Input tekstfelt der bruker skriver ønsket søkeord for søkefunksjonen, samt knapp som setter i gang søket.
- DropTid: DropUp-meny der bruker velger tidsperspektiv for søket.
- DropKategori: DropUp-meny der bruker velger kategori for å avgrense søket.
- Artikkelfelt: Dynamisk tekstfelt der alle artikler og ingresser vises. Hovedtekstfelt som brukes gjennom hele filmen.
- Skjulefelt: Lapp som brukes for å dekke over overskrift i tekstfeltet 5 siste.
- Scrollknapper: Scrollknappene som styrer Artikkelfelt. Disse ligger i eget MovieClip.
- Grensesnitt: Grensesnittet.
- Bakgrunn: Legger hvit bakgrunnsfarge i grensesnittet.

Innlasting:

Denne scenen brukes for å gi flash tid til å laste inn menyen. Alle layers som ligger i denne scenen har samme funksjon som i hovedscenen.

- Labels: Dette er navnelapper som er brukt til adressering innad i filmen. I tillegg letter de oversikten i scriptlaget.
- Script: Her har vi samlet alle framescript.
- Sistefelt: Tekstfelt som viser de fem siste innlagte artikler.
- Menyknapper: Statisk menyknapp som tar bruker tilbake til forsiden.
- Kategorifelt: Dynamisk tekstfelt som inneholder informasjon om hvilken del av avisen bruker befinner seg i, for eksempel søk eller 5 siste.
- Datofelt: Dynamisk tekstfelt som viser dagens dato.
- Artikkelfelt: Dynamisk tekstfelt der alle artikler og ingresser vises. Hovedtekstfelt som brukes gjennom hele filmen.
- Skjulefelt: Lapp som brukes for å dekke over overskrift i tekstfeltet 5 siste.
- Scrolleknapper: Scrollknappene som styrer Artikkelfelt. Disse ligger i eget MovieClip.
- Grensesnitt: Grensesnittet.
- Bakgrunn: Legger hvit bakgrunnsfarge i grensesnittet.

Script i Index.swf

Her beskrives de ulike scriptenes funksjon i den rekkefølge de blir trigget når en bruker kommer inn i avisen. (Vedlegg F: Flytskjema for brukerdelen)

Scene: Innlasting.

Label: Meny

Nødvendig input: Kategorier innlest fra XML.

Output: Menyknapper.

Path: >> Meny

Forklaring: Script som leser inn menyen.

Den delen av koden som er beskrevet nedenfor som leser inn og parser XML er tilnærmet identisk gjennom hele index.swf. Derfor blir beskrivelsen her også gjeldende for resten av filmen. Dette til etterretning.

Funksjonen SkrivUtMeny sørger for å parse data hentet fra XML og legge denne i arrayer. Funksjonen jobber seg nedover i XML og legger de to nederste nodene i hierarkiet inn i arrayer.

```
function SkrivUtMeny () {
    // Arrayer og XML-objekt som brukes til å holde informasjon
    KategoriXML = new XML();
    TmpXML = new XML();
    MenyList = new Array();
    KategoriList = new Array();
    // Legger inn data i MenyListe-arrayen.
    MenyList = MenyXML.firstChild.childNodes;
    // Looper gjennom og formaterer data.
```

Når alle data ligger i arrayer kjøres to while-løkker. Den første sjekker på de nest nederste nodene, i vårt tilfelle om vi har noder som heter ”kategori”. Har vi det kjøres neste while-løkke som sjekker hvilke barn som ligger under ”kategori”. Finner den ”kategoriid” kopieres verdien fra denne inn i en array av objekter. Denne blir merket med ”.id”. Finner den ”kategorinavn” legges verdien av denne inn i samme array av objekter som id, men denne blir merket med ”.Name”. Høyden på objektet blir statisk satt til = 20.8. Vi har altså ett objekt med tre deler: Name, id og Height. Dette inneholder informasjonen om en XML-node.

```
while (i<=MenyList.length) {
    if (i<MenyList.length) {
        MenyArray[i] = new Object();
    }
    if (MenyList[i].nodeName.toLowerCase() == "kategori") {
        KategoriXML = MenyList[i];
        KategoriList = KategoriXML.childNodes;
        ii = 0;
        while (ii<=KategoriList.length) {
            TmpXML = KategoriList[ii];
// Hvilke av de dypeste nodene er vi i nå?
            if (KategoriList[ii].nodeName.toLowerCase() == "kategoriid") {
                MenyArray[i].id = unescape(KategoriList[ii].firstChild.nodeValue);
            } else if (KategoriList[ii].nodeName.toLowerCase() == "kategorinavn") {
                MenyArray[i].Name = unescape(KategoriList[ii].firstChild.nodeValue);
                MenyArray[i].Height = 20.8;
```

Funksjonen knapper skriver ønsket antall menyknapper ut i grensesnittet. Menyknappene heter ”Menyfelt” og er hver et lite MovieClip.

En for-løkke sjekker mot lengden av arrayen som alle objektene ligger i. Vi skal ha ut en knapp for hver kategori. Inne i for-løkka tilkalles _root.AttachMovie. Dette er en innebygd funksjon i flash som legger MovieClip inn i en annen film. Eval-funksjonen legger objektenes data inn i hvert MovieClip den legger ut i hovedfilmen.

```
function knapper () {
    for (Counter=0; Counter<MenyArray.Length; Counter++) {
        _root.AttachMovie("Menyfelt", "Menyfelt" add Counter, 200+Counter);
        eval("Menyfelt" add Counter).kat_navn = MenyArray[Counter].Name;
        eval("Menyfelt" add Counter)._x = 56.3;
        eval("Menyfelt" add Counter)._y = (Counter*20.8)+319.4;
        eval("Menyfelt" add Counter).id = MenyArray[counter].id;
```

Funksjonen VisKategori henter inn kategoriid når en av knappene i menyen blir aktivisert. Denne blir hentet inn ved hjelp av Arguments[0], og lagt ut i en link som tilkaller scriptet som skriver ut kategorien som er valgt.

”path” er en variabel som er initialisert helt i begynnelsen av filmen. Denne inneholder den komplette pathen som vi måtte bruke mot server for å få tilkalt filene som vi trenger. For at det skal være lettest mulig å overføre prosjektet og kjøre det mot annen server har vi opprettet en variabel som er det eneste som må endres ved en eventuell forflytning.

```
function VisKategori (menyfelt) {
//utskrift holder rede på kategoriid
utskrift = arguments[0];
ASPfunksjon = path + " kategori4.asp?kategoriid="+arguments[0];
gotoAndPlay ("Hoved","kategori");
```

Scene: Innlasting.
Label: dato
Nødvendig input: Ingen
Output: Dagens dato utlistet i tekstfeltet datofelt
Path: >> Meny >> dato

Forklaring: Script som leser inn dato.
Det er opprettet et datofelt som viser dagens dato i grensesnittet. Dette feltet fylles ved hjelp av innebygde dato-objekter i flash. Her er det opprettet en array som inneholder alle dager og måneder. Dette for å konvertere datoen fra amerikansk skrivemåte.

```
// Arrayer som brukes av dato-funksjonen.

dag = new Array();          maaned = new Array();
dag[0] = "Søndag";          maaned[0] = "Januar";
dag[1] = "Mandag";          maaned[1] = "Februar";
dag[2] = "Tirsdag";         maaned[2] = "Mars";
dag[3] = "Onsdag";        maaned[3] = "April";
dag[4] = "Torsdag";         maaned[4] = "Mai";
dag[5] = "Fredag";          maaned[5] = "Juni";
dag[6] = "Lørdag";          maaned[6] = "Juli";
                            maaned[7] = "August";
                            maaned[8] = "September";
                            maaned[9] = "Oktober";
                            maaned[10] = "November";
                            maaned[11] = "Desember";
```

Her foretas en enkel konvertering av datoen. Vi har også mulighet for å ta med klokkeslett, men dette ble utelatt på grunn av vanskelig oppdatering. Datoobjektene kjøres bare når filmen lastes, og skulle vi få klokken til å gå måtte vi kjøre en kontinuerlig oppdatering av denne. Vi vurderte det slik at dette ville trekke for mye ressurser i forhold til nytteverdien.

```
// Funksjon som henter dato og klokkeslett og legger disse ut i datofeltet på hovedsiden.
function TheDate () {
    idag = new Date();
    date = date+dag[idag.getDay()]+" ";
    date = date+idag.getDate()+". ";
    date = date+maaned[idag.getMonth()]+" ";
    date = date+idag.getFullYear()+" ";

}
TheDate();
klar = "lastet";
gotoAndPlay ("Hoved", "index");
```

MovieClip: Hoved
frame: 5
Label: Index
Nødvendig input: Ingresser og titler fra XML.
Output: Ingresser lagt inn i dag og i går listes ut i tekstfeltet Artikkelfelt.
Path: >> Meny >> dato >> index

Forklaring:
Her leses førstesiden inn i avisen. Dette er den hovedsiden som automatisk blir trigget når en bruker kommer inn i avisen. Her skrives dagens ingresser ut. Når index kjøres tar flashfilmen alltid med 5siste også, og her stopper filmen til brukeren foretar et valg.

MovieClip: Hoved
frame: 7
Label: 5siste
Nødvendig input: Titler og opprettet-tid fra XML.
Output: Ingresser lagt inn i dag og i går listes ut i tekstfeltet Artikkelfelt.
Path: >> Meny >> dato >> index >> 5siste

Forklaring:
5siste lister ut de fem nyeste artiklene i databasen og legger dem i tekstfeltet sistefelt. Dette scriptet blir bare kjørt i forbindelse med index på grunn av begrenset behov for oppdatering. Ny informasjon her kommer kun når en ny artikkel blir lagt inn.

STATUS: Det som vises på skjermen når filmen er på dette stadium er følgende: Ingressene på førstesiden vises, dagens dato vises i datofeltet, de fem siste artiklene vises med tittel og tidspunkt for opprettet og menyknappene er listet ut i forhold til antall forskjellige kategorier i databasen. Nederst på skjermen vises en knapp som bringer bruker til søkefunksjonen. Animasjonen øverst i venstre hjørne er ferdig animert og står nå i ro.

Herifra har bruker fire valg:

1. Han kan velge å lese en av artiklene på førstesiden.
2. Han kan velge å lese en av artiklene i feltet 5siste.
3. Han kan søke etter artikkel.
4. Han kan velge en kategori fra menyen.

Valg 1: Brukeren vil lese en av artiklene på førstesiden:

Filmens videre gang:

MovieClip: Hoved
Frame:13
Label: Artikkel
Nødvendig input: Artid
Output: Artikkelen med forfatter skrives ut i Artikkelfelt.
Path: >> Meny >> dato >> index >> 5siste >> Artikkel

Forklaring:
Artikkel får tilsendt Artid fra index. Artid blir sendt til ASP/XML som sender riktig artikkel tilbake. Denne blir vist i tekstfeltet Artikkelfelt. Dette er den eneste endringen i forhold til forrige status. Navn på forfatter skrives ut nederst i teksten med ”mailto-funksjon”.

MovieClip: Hoved
frame: 15
Label: relatert
Nødvendig input: Artid
Output: Tittel og opprettet-tid til evt relatert artikkel.
Path: >> Meny >> dato >> index >> 5siste >> Artikkel >> relatert

Forklaring:
Relatert får Artid medsendt fra index og sender denne til ASP/XML som sjekker om det finnes artikler som har samme tema som artikkelen som vises for øyeblikket. Finnes det noen sendes tittel og opprettet-tid tilbake . Dette vises som linker under artikkelen i Artikkelfelt.
Blir en av disse linkene aktivert sendes et kall medsendt Artid tilbake til Label: artikkel (Forrige omtalt) og den blir vist der.

Valg 2: Bruker velger å lese en av artiklene i 5siste:

Ved aktivering av en av linkene i 5siste blir Artid medsendt i et kall til Label: artikkel.
Samme som ved aktivering av link fra index/forside.

Valg 3: Bruker velger å søke etter artikkel:

MovieClip: Hoved
frame: 25
Label: soekevalg
Nødvendig input: Ingen, eller en eller flere av følgende: tid, kategori, søkeord.
Output: Fullstendig path som sendes til Soek.
Path: >> Meny >> dato >> index >> 5siste >> Soekevalg

Forklaring:
Ved å trykke på knappen ”Søk etter artikler” nederst til venstre går flashfilmen til Label: soekevalg. Dette er eneste framen der formen for søk er synlig. Bruker fyller ut feltet for søkeord, velger kategori og tidsbegrensning i dropUp-menyene og trykker på knappen ”Søk”.
Dette sender filmen til Label: Soek og sender med en path med de forskjellige søkekriterium vedlagt.

MovieClip: Hoved
frame: 19
Label: Soek
Nødvendig input: path fra Label: soekevalg
Output: Ingresser til artikler som stemmer overens med kriterier for søk.
Path: >> Meny >> dato >> index >> 5siste >> Soekevalg >> Soek

Forklaring:
Mottar en path fra Label: soekevalg som brukes for å trigge ASP/XML som sender tilbake resultatet av søket. Dette vises med ingresser i Artikkelfelt. Scriptet har stort sett samme funksjonalitet som index.

MovieClip: Hoved
frame: 20
Label: soekartikkel
Nødvendig input: Artid
Output: Artikkel med forfatter
Path: >> Meny >> dato >> index >> 5siste >> Soekevalg >> Soek >> soekartikkel

Forklaring:
Samme funksjonalitet som artikkel. Søk har fått sitt eget script for utskriving av artikler på grunn av et ønske om å holde søkefunksjonen adskilt. Dette fordi det oppsto problemer med delte variable og innlesing av flere ASP/XML-filer i samme frame.

MovieClip: Hoved
frame: 21
Label: relatertsoek
Nødvendig input: Artid
Output: Tittel og opprettet-tid til evt relatert artikkel.
Path: >> Meny >> dato >> index >> 5siste >> Soekevalg >> Soek >> soekartikkel >> relatertsoek
Forklaring:
Samme funksjonalitet som Label: relatert. Søk har fått sitt eget script for utskriving av relaterte artikler på grunn av et ønske om å holde søkefunksjonen adskilt. Dette fordi det oppsto problemer med delte variable og innlesing av flere ASP/XML-filer i samme frame.
Unntak: Aktivert link fra relatersoek trigger soekartikkel, ikke artikkel som relatert gjør.

Meny >> dato >> index >> 5siste >> kategori >> artikkel >> relaterte >> soek >> soekartikkel >> relatertsoek >> soekevalg

Valg 4: Bruker velger en kategori fra menyen:

MovieClip: Hoved
frame: 9
Label: kategori
Nødvendig input: Kategoriid og utskrift.
Output: Tittel, ingress og opprettet-tid til artikler.
Path: >> Meny >> dato >> index >> 5siste >> Kategori
Forklaring:
Samme funksjonalitet som Label: Index. Får path tilsendt fra knappen det er trykt på, og skriver ut alle artiklene i valgt kategori fra de siste 7 dagene. Henter nummeret på knapp det ble trykket på i variabelen "*utskrift*", og finner navnet på knappen i *MenyArray[utskrift-1].Name*. Dette navnet skrives ut i tekstfeltet "*Kategorifelt*".

Oversikt over variable i index.swf

Path = Variabel som inneholder fullstendig path til ASP-filer. Opprettet for å forenkle jobben når systemet skal flyttes til annen server. Pathen treng kun å forandres en sted.

ResultatVar = Holder pathen ASPfunksjon (med søkekriterier) for bruk i Label: Soek. Trigges av knappen som bringer bruker fra Soekartikkel og tilbake til søkeresultatet.

Kategori = Holder kategorinavnet som blir valgt i DropUp-menyen i søkefunksjonen.

tiden = Holder tiden som blir valgt i DropUp-menyen i søkefunksjonen.

TempArtid = Får Artid fra Arguments[0] når en link til artikkel aktiveres fra Artikkelfelt.

Artid = Variabel som fylles av hver artikkel sin artikkelid under innlesing av XML. Brukes for å tilkalle artikler fra databasen. Medsendes til ASP.

Sisteartid = Samme som artid, men brukt i 5siste. Måtte ha forskjellige navn for ikke å komme i konflikt.

Sistesjekk = Variabel som blir fylt med et signaturtall for hver frame den er inne i. På denne måten vet Label: Artikkel hver gang hvor kallet kommer fra og setter overskriften i kategorifelt deretter.

ASPfunksjon = Variabel som i hver funksjon som kaller ASP blir fylt med korrekt adresse med eventuelle parameter. Denne blir kjørt i den aktuelle framen som skal ha informasjon fra ASP.

RelatertFunksjon = Som over. Måtte ha forskjellige navn for å ikke komme i konflikt.

Adresse = En variabel som tar imot mailadresse til forfatter som et argument (Arguments[0]). Medsendes som parameter til ASP.

Tekstfelt /variable

Et dynamisk tekstfelt kan også en variabel, og motsatt. Følgende er brukt i index.swf:

Artikkelfelt = Brukes for å vise alle artikler, ingresser og kategorier på skjermen.

Sistefelt = Viser de 5 siste artiklene.

Utskrift = Dette er et usynlig tekstfelt / variabel som mottar kategoriid fra menyen. Også utstrakt brukt til testing.

Date = Inneholder dagens dato.

Kategorifelt = Viser brukeren hvor han er og hvor artikkelen / ingressene han leser kommer fra.

Soekeord = Bruker skriver et ønsket søkeord inn i dette tekstfeltet. Brukes i søkefunksjonen.

Variable i DropUp-menyer

DropUp-menyene er innebygde movieclip i Flash. Variable og implementering av disse er utførlig forklart under punkt ”dropdownbokser” under ”Implementering av administrasjonsdelen” s 51.

MovieClip

LogoXR = Animert logo

DropKategori = DropUp-meny. Dette er et innebygget smartClip i flash. Her brukt som dropUp-meny i kategorifeltet i søkefunksjonen.

DropTid = = DropUp-meny. Dette er et innebygget smartClip i flash. Her brukt som dropUp-meny i tidsfeltet i søkefunksjonen.

MenyFelt = Dette er MovieClipet som blir skrevet ut dynamisk og som på scenen fungerer som knapper i hovedmeny. Clipet består av en knapp og et tekstfelt.

Scrolleknapper = Dette er de to knappene som kontrollerer scrollingen i Artikkelfelt. Disse er satt sammen i et MovieClip for lettere å kunne kontrollere disse med scripting.

Består av to frames, der 1. ramme består av knapper med script:

Scroll _opp
```
on (rollOver) {
    _root.Artikkelfelt.scroll = _root.Artikkelfelt.scroll-1;   }
```

Scroll_ned
```
on (rollOver) {
    _root.Artikkelfelt.scroll = _root.Artikkelfelt.scroll+1;   }
```

og 2. ramme består av et framescript:

```
play ();
_currentframe-1;
```

Det det siste scriptet gjør, er å hoppe til forrige ramme, og spille det som står der. Dermed vil scriptet på knappene bli kjørt en gang til, og tekstfeltet vil da scrolle et hakk. Dette vil gå i en evig loop, til tekstfeltet er ferdig scrollet.

Oversikt over arrayer, XML-objekt og funksjoner

Scene: Hoved

Label:	index		
XML-objekt:	NyhetXML	DispXML	TempXML
Arrayer:	NyhetsListe	DispList	
Funksjoner:	LoadArtikkel	SkrivUt	
Label:	5siste		
XML-objekt:	SisteXML	DispoXML	TempXML
Arrayer:	SisteList	DispoList	
Funksjoner:	sisteUt	LoadSisteArtikkel	
Label:	kategori		
XML-objekt:	NyhetXML	DispXML	TempXML
Arrayer:	NyhetsList	DispListe	
Funksjoner:	SkrivUt	LoadArtikkel	
Label:	Artikkel		
XML-objekt:	NyhetXML	DisList	TempXML
Arrayer:	NyListe	DisListe	
Funksjoner:	SkrivUt	SendMail	
Label:	Relatert		
XML-objekt:	RelatertXML	DispRelXML	TempRelXML
Arrayer:	RelatertsListe	DispRelListe	
Funksjoner:	SkrivRelatertut	LoadRelatertArtikkel	
Label:	Soek		
XML-objekt:	NyhetXML	DispXML	TempXML
Arrayer:	NyhetsListe	DispList	
Funksjoner:	SkrivUt	LoadSoekArtikkel	
Label:	Soekartikkel		
XML-objekt:	NyhetXML	DispoSoekXML	TempXML
Arrayer:	NyListe	DispSoekList	
Funksjoner:	SkrivUtSoek	SendSoekMail	
Label:	RelatertSoek		
XML-objekt:	RelatertXML	DispRelXML	TempRelXML
Arrayer:	RelatertsListe	DispRelListe	
Funksjoner:	SkrivRelatertSoekUt	LoadRelArtikkel	

Forklaring til ASP-filene (bruker)

Index

Funksjon: Skriver ut hovedsiden (forsiden), nyheter fra i dag og igår.

Skriver ut:

<artikkel>artid, tittel, ingress, opprettet</artikkel>

Parametre: Ingen parametre

SQL:

dato = date

dato = dateAdd ("d", -1, dato)

*”SELECT * FROM Artikkel WHERE Opprettet>#" & dato & "# AND Aktiv = 1 ORDER BY Artid DESC"*

Kategori4

Funksjon: Skriver ut artikler i ønsket kategori, de 7 siste dagene.

Skriver ut:

<artikkel>artid, tittel, ingress, opprettet</artikkel>

Parametre: *Kategoriid*

SQL:

dato = date

dato = dateAdd ("d", -7, dato)

*"SELECT * FROM Artikkel WHERE Opprettet>#" & dato & "# AND Aktiv = 1 AND kategoriid="&kategoriid&" ORDER BY Artid DESC"*

Category

Funksjon: Skriver ut alle kategoriene (menyen)

Skriver ut:

<kategori>kategoriid, kategorinavn</kategori>

Parametre: Ingen parametre

SQL:

*"SELECT * FROM Kategori"*

Artikkel

Funksjon: Skriver ut ønsket artikkel

Skriver ut:

<artikkel>artid, tittel, ingress, tekst</artikkel>

<forfatter> forfatteremail, forfatternavn</forfatter>

Parametre: *Artid*

SQL:

Skriver ut ønsket artikkel:

*"SELECT * FROM Artikkel WHERE Artid = " & Request("Artid")*

Skriver ut forfatteren av ønsket artikkel:

"SELECT Forfatter.Navn, Forfatter.Email FROM Forfatter,Artikkel WHERE Forfatter.Forfatterid = Artikkel.Forfatterid AND Artikkel.Artid = " & Request("Artid")

Relaterte5

Funksjon: Skriver ut relaterte artikler til en artikkel

Skriver ut:

<artikkel>artid, tittel, opprettet</artikkel>

Parametre: *Artid*

SQL:

Denne finner ut hvilket tema som artikkelen ligger i:

"SELECT Tema.Temaid FROM Artikkel,Tema,Temakobling WHERE Tema.Temaid = Temakobling.Temaid AND Artikkel.Artid = Temakobling.Artid AND Artikkel.Artid = " & Request.querystring("Artid")

Denne velger ut alle artiklene i dette temaet, ekslusive gjeldende artikkel:

"SELECT Artikkel.Artid, Artikkel.Tittel, Artikkel.Opprettet, Tema.Temaid, Artikkel.Artid FROM Tema INNER JOIN (Artikkel INNER JOIN Temakobling ON Artikkel.Artid = Temakobling.Artid) ON Tema.Temaid = Temakobling.Temaid WHERE (((Tema.Temaid)="&rs("Temaid")&") AND ((Artikkel.Artid) <> "&Request.querystring("Artid")&") AND (Artikkel.Aktiv = 1))"

katsiste

Funksjon: Skriver ut de 5 siste artiklene i en kategori

Skriver ut:

<artikkel>artid, tittel, opprettet</artikkel>

Parametre: *Kategoriid* (Må være <>0)

SQL:

*"SELECT * FROM Artikkel WHERE Kategoriid = ”& Request("Kategoriid") & " AND Aktiv = 1 ORDER BY Opprettet DESC"*

siste

Funksjon: Skriver ut de 5 siste nyhetene

Skriver ut:

<artikkel>artid, tittel, opprettet</artikkel>

Parametre: Ingen parametre

SQL:

*"SELECT * FROM Artikkel WHERE Aktiv = 1 ORDER BY Opprettet DESC"*

Looper 5 ganger og skriver ut disse artiklene.

soeq

Funksjon: Søker i databasen, og skriver ut resultatet.

Skriver ut:

<artikkel>artid, tittel, ingress, opprettet</artikkel>

Parametre: *Ord, Tiden, Kategori*

SQL:

Finner id til kategorien som er valgt i dropupboksen:

"Select Kategoriid FROM Kategori WHERE Navn LIKE '"&kategori&"'"

Søker i databasen:

*"SELECT * FROM Artikkel WHERE (Tekst LIKE '" & "%" & soekeord & "%' OR Tittel LIKE '%" & soekeord & "%' OR Ingress LIKE '%" & soekeord & "%')"*

Sjekker om bruker har valgt en kategori i dropupboksen:
If kategoriid <> 0 then
”AND Kategoriid = " & Kategoriid
end if

Fortsetter med uttrykket:
" AND Opprettet > #" & dato & "# AND Aktiv = 1 ORDER BY Opprettet DESC"

Utførelse av administrasjonsdel

Oversikt over variabler i admin.swf

Variabler i hovedtidslinjen

Innlogget	true eller false for å sjekke om brukeren er innlogget
Passord	Inneholder passordet til brukeren
Forfatterid	Inneholder id'en til brukeren
Rettigheter	Rettighetene til brukeren
Ferdig	Inneholder 1 eller 0 for om artikkelen er ferdig eller ikke.
Aktiv	Inneholder 1 eller 0 for om artikkelen er ferdig eller ikke.
Antall	Antall poster i søkeresultatet.
Soek_kategori	Kategorien som begrenser søket.
Admin	Rettighetene til forfatteren 0(vanlig) eller 2(sjef).
Kategorinavn	Kategorien for artikkel som skal lagres.
Artid	Id'en til artikkelen
Kategori_navn	Brukes til å lagre valgt kategori for redigert artikkel.
Soek_klar	Brukes for å sjekke når alle variablene er lastet.
Temanavn	Temaet for redigert artikkel.
Soek_tid	Tidsavgrensning for artikler i søket.
Rediger_klar	Brukes for å sjekke om alle data er lastet.

Variabler/tekstfelt

Ingress	Ingressen til artikkelen.
Tekst	Teksten til artikkelen.
Brukertype	Teksten "sjef" eller "bruker".
Tittel	Tittelen på artikkelen.
Navn	Inneholder navnet til brukeren
Soek_tekst	Søkestrengen for å søke i artikler.
Email	Email til ny forfatter/ny produsent
Nyforfatter	Navnet til ny forfatter
Nyprodusent	Navnet til ny produsent
Fpassord	Passord for ny forfatter
Tema	Felt for nytt tema

Variabler i Movie clip (navnet på Movie clipet i parentes)

Lagret_svar	Teksten for tilbakemelding til brukeren (Lagret_svar_MC)
Item_spacing	Avstanden mellom menyelementene i søkeresultatet (menu_item_group)
Art_id	Artikkel id (menu_item_group)
Tittelx	Tittelen på artikkelen, hvor x er et nummer (menu_item_group)
Tema_klar	Sjekk om alt er lastet (tema_dropdown)
Kategori	Kategoriene på kommaseparert form (kategori_dropdown, kategori_ny)
Tema	Temaene hentet fra databasen på kommaseparert form (tema_dropdown)
kategoriNy	Kategoriene til dropdownboksen (kategori_ny)
Artikkel_klar	Sjekk om alt er lastet (kategori_ny, kategori_dropdown)

Movie Clip

Lagret_svar_MC	Animasjon for tilbakemelding til brukeren.
Sjef_meny_MC	Inneholder sjefsbruker menyen.
Bruker_meny_MC	Inneholder brukermenyen.
Uptrigger	Movie clip for å scrolle teksten oppover.
Downtrigger	Movie clip for å scrolle teksten nedover.
Soek_downtrigger	Movie clip for å scrolle søkeresultatet nedover.
Soek_uptrigger	Movie clip for å scrolle søkeresultatet oppover.
LogoXR	Animert logo i redigere scenen.
LogoX	Animert logo i sjefsbruker scenen.
Checkbox2	Sjekkboksene i formene.
Menu	Dropdownboksene
Menu item group	Dynamisk meny for søkresultatet.

Forklaring til ASP filene (administrator)

Under er en forklaring til ASP filene, det er forklart hvilken funksjon de har og hvilke variabler de tar inn og returnerer. Variabelnavnene står i parentes.

Admin_soek

Funksjon:
Utfører søket mot databasen ut i fra valgt kategori, tid og søketekst. Dersom det er en vanlig bruker får han bare opp de artiklene han selv har skrevet, men om det er en sjefsbruker får han opp alle artikler som stemmer med søkekriteriene.

Tar inn:
Rettigheter til brukeren (rettigheter)
Forfatterid til brukeren (forfatterid)
Søketeksten som brukeren har skrevet inn (Soek_tekst)
Det valgte tidsintervallet for søket (Soek_tid)
Valgt kategori (Soek_kategori)

Returnerer:
Tittelen til artikkelen (tittelx) x er et nummer fra 1 til antall artikler
Artikkel id (artidx) x er et nummer fra 1 til antall artikler
Antall artikler som ble funnet (antall)
En variabel for å vite når alt er blitt lastet (Soek_klar)

Hent_kategori

Funksjon:
Henter kategoriene fra databasen for bruk i redigere artikkel.

Tar inn:
Ingenting

Returnerer:
Kategoriene på kommaseparert form (kategori)
En variabel for å vite når alt er blitt lastet (artikkel_klar)

Hent_kategori_soek

Funksjon:
Henter kategoriene fra databasen for bruk i søkesiden, skriver første kategorien som "Alle kategorier" for å kunne søke i alle.

Tar inn:
Ingenting

Returnerer:
Kategoriene på kommaseparert form (kategori)
En variabel for å vite når alt er blitt lastet (artikkel_klar)

Hent_tema_rediger

Funksjon:
Henter temaene fra databasen til dropdownboksen i redigere artikkel, første tema som skrives ut er "Samme tema" slik at man kan lagre artikkelen uten å endre tema.

Tar inn:
Ingenting

Returnerer:
Temaene på kommaseparert form (tema)
En variabel for å vite når alt er blitt lastet (tema_klar)

Lagre_artikkel

Funksjon:
Henter dataene fra formen i ny artikkel, finner kategorien og eventuelt tema, så lagrer den artikkelen i databasen.

Tar inn:
Kategorien for artikkelen (kategorinavn)
Tittelen for artikkelen (tittel)
Ingressen til artikkelen (ingress)
Teksten til artikkelen (tekst)
Id'en til forfatteren (forfatterid)
Om artikkelen er ferdig (ferdig)
Om den er aktiv (aktiv)
Tema for artikkelen (temanavn)

Returnerer: Melding om lagringen gikk greit (lagret_svar). Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Lagre_forfatter

Funksjon:
Tar inn data fra formen ny forfatter og lagrer de i databasen.

Tar inn:
Navnet på forfatteren (nyforfatter)
Email til forfatteren (email)
Passord for forfatteren (fpassord)
Om det er en sjefsbruker (admin)

Returnerer:
Melding om lagringen gikk greit (lagret_svar)
Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Lagre_kategori

Funksjon:
Henter kategorien og sjekker hvor mange poster som er lagret i databasen, dersom det er under 13 så lagres kategorien, hvis ikke gis det melding om at det ikke re plass til flere kategorier.

Tar inn:
Innskrevet kategori (kategori)

Returnerer:
Melding om lagringen gikk greit (lagret_svar)
Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Lagre_produsent

Funksjon:
Henter navn og email til produsenten fra formen og lagrer den i databasen. Så gis det tilbakemelding om hvordan det gikk.

Tar inn:
Navnet på produsenten (nyprodusent)
Email til produsenten (email)

Returnerer:
Melding om lagringen gikk greit (lagret_svar)
Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Lagre_redigert

Funksjon:
Henter alle data om den redigerte artikkelen, finner id for valgt tema og tema dersom temanavn er forskjellig fra "Samme tema". Så blir artikkelen lagret i databasen.

Tar inn:
Navnet på kategorien (kategorinavn)
Id'en til forfatteren (forfatterid)
Artikkel id (artid)
Tittelen på artikkelen (tittel)
Ingressen til artikkelen (ingress)
Teksten til artikkelen (tekst)
Om artikkelen er ferdig (ferdig)
Om artikkelen er aktiv (aktiv)
Navnet på temaet (temanavn)

Returnerer:
Melding om lagringen gikk greit (lagret_svar)
Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Lagre_tema

Funksjon:
Henter temaet og sjekker hvor mange som ligger der fra før dersom det er under 23 så legges det inn i databasen, hvis ikke gis det melding om at det ikke er plass til flere.

Tar inn:
Navnet på det nye temaet (tema)

Returnerer:
Melding om lagringen gikk greit (lagret_svar)
Variabel for å sjekke om alle dataene er lastet (rediger_klar)

Log_inn

Funksjon:
Henter brukernavn og passord som skrives inn på innloggingsskjermen og sjekker om den finner tilsvarende poster i databasen. Dersom brukernavn og passord stemmer med det i databasen sender den navnet, rettighetene og id'en til gjeldende bruker tilbake. Hvis den ikke finner noe settes "innlogget" til false brukernavn og rettigheter til null.

Tar inn:
Brukernavnet som er email adressen (navn)
Passordet til brukeren (passord)

Returnerer:
Navnet til brukeren (navn)
Id'en til brukeren (forfatterid)
Rettighetene til brukeren (rettigheter)
Variabel som er true eller false for å sjekke om brukeren virkelig er innlogget (innlogget)

Ny_artikkel

Funksjon:
Henter kategori og tema fra databasen til dropdownboksene for å skrive ny artikkel.

Tar inn:
Ingenting

Returnerer:
Kategoriene (kategoriNy)
Temaene (tema)
Variabel for å sjekke om alt er lastet (artikkel_klar)

Oppkobling

Funksjon:
Inkluderes i alle de andre filen og setter opp databasedrivere og hvilken database som skal brukes.

Rediger_artikkel

Funksjon:
Henter artikkel id'en og henter ut dataene om artikkelen fra databasen. Dataene blir så sendt.

Tar inn:
Id'en til artikkelen: (Art_id)

Returnerer:
Tittelen på artikkelen (tittel)
Ingressen til artikkelen (ingress)
Teksten til artikkelen (tekst)
Artikkel id (artid)
Navnet på kategorien som artikkelen tilhører (kategori)
Variabel for å sjekke om alle data er ferdig lastet (rediger_klar)

Implementering av administratordelen

Vi har valgt å dele hele administrasjonssiden inn i tre scener, dette har vi gjort for å gjøre det mer oversiktlig å jobbe med. Man slipper da å ha alt liggende i en lang tidslinje etterhverandre. De tre scenene har vi kalt logg_inn, redigere og sjefsbruker. Fra starten var det meningen å ha fire scener for å skille mellom sjefsbruker og vanlig bruker, men mot slutten fant vi en annen løsning på dette. Derfor har sjefsbruker scenen et litt misvisende navn ettersom den brukes til både sjef og vanlig bruker.

Forskjellen mellom en vanlig bruker og en sjefsbruker er at en vanlig bruker ikke kan:

- Legge inn ny forfatter.
- Redigere andre sine artikler.
- Legge inn ny kategori.

I admin siden er det et menyvalg for å legge inn ny medieprodusent, dette har egentlig ingen funksjon nå. Det ble laget i starten av prosjektet da det var meningen å få med bilder og video også, slik at vi kunne lagre produsenten til de ulike mediene. Vi valgte å ikke ta den bort da den ikke ligger i veien og ved videre utvikling kan det være aktuelt å bruke den.

"Logg_inn" scenen inneholder en form for å logge på, en skjerm for utlogget og en for feil brukernavn/passord.

"redigere" scenen inneholder en form for å søke blant artiklene, man får opp søkeresultatet og kan velge å redigere en av de.

"sjefsbruker" scenen inneholder former for å legge inn nytt tema, ny kategori, ny forfatter, ny medieprodusent og ny artikkel.

Bruk av lagene

Logg_inn scenen:

Bakgrunn	For bakgrunnsbildet
Form	Inneholder resten av elementene.

Redigere scenen:

Meny	Inneholder de to menyene
Mask	Inneholder en maske for å få til scrolling på søkeresultatet
Form	Inneholder alle form elementene
Bakgrunn_soek	Bakgrunn på søkeresultatet og teksten "Søkeresultat"
Bakgrunn	Bakgrunnsbildet og logoen

Sjefsbruker scenen:

Menylink	Inneholder de to menyene
Form	Inneholder alle form elementene
Bakgrunnsbilde	Bakgrunnsbildet og logoen

Overføring av verdier mellom Flash og ASP

For å overføre variabler fra Flash til ASP og omvendt bruker vi en funksjon som heter "loadVariables", den sender variablene i Flash filmen til ASP skriptet og tar inn igjen variabler som måtte bli returnert fra ASP.

Eksempel:
loadVariables ("Lagre_tema.asp", "_root.Lagret_resultat_MC", "POST");

Denne linjen overfører variablene i filmen til "lagre_tema.asp". Det siste argumentet angir hvilken metode man bruker for å ta i mot informasjonen i ASP fila, som her er "POST". Den laster så variabelen som måtte bli returnert fra ASP inn i movie clipet "Lagret_resultat_MC" som ligger i hovedtidslinjen.

For å få variablene fra ASP inn i Flash skrives de ut på VDS format, altså på formen:

"&variabel1=X&variabel2=X&variabel3=X".

For at dette skal virke er det viktig at ingenting annet blir returnert fra ASP fila. Det vil si at om en viser ASP fila i en browser skal VDS strengen være det eneste som kommer fram.

For å vite når alle verdiene er lastet inn i filmen er det lurt å legge på en variabel på enden av VDS strengen slik at man kan sjekke om den inneholder noe, dersom den gjør det vet man at dataene er ferdig lastet.

Menyløsning

I utgangspunktet var det meningen å lage to scener, en for sjefsbruker og en for vanlig bruker, ettersom de skal ha ulike valg i menyen. Men etter hvert fant vi ut at det var bedre å putte menyene inn i et movie clip. Da kan vi sette den ene menyen synlig og den andre usynlig, på denne måten klarte vi oss med bare tre scener og det gjorde det hele mye mer oversiktlig, og vi slapp å ha så mye liggende dobbelt.

Dette gjør det også mye enklere å forandre ting. Vi får da bare en scene å forandre ting i og når menyen ligger i et movie clip vil både den i redigere og sjefsbruker bli forandret om vi gjør noe med en av dem.

Dropdownbokser

Dropdownboksene var et stort problem å få til Vi hadde noen statiske til å begynne med som virket greit, men det var omtrent umulig å få de til å virke dynamisk. Vi strevet i flere dager før vi plutselig oppdaget at det var et innebygd bibliotek i Flash som inneholdt dynamiske dropdownbokser. Så vi bestemte oss for å bruke disse, men minuset med dem var at de ikke hadde scrollbarer.

Vi gjorde et forsøk på å bygge de om slik at de kunne scrolles, det ga vi imidlertid fort opp da vi fant ut en metode vi kunne bruke for å lage dynamiske dropdownbokser selv.

Måten vi gjorde dette på var å legge en knapp i et movie clip, og putte dette i et annet movie clip. Vi kunne da duplisere movie clipet med knappen for å lage en liste med knapper. Da denne listen med knapper lå inne i et movie clip kunne vi flytte alle knappene ved å sette y posisjonen til movie clipet. Ved å legge en maske over dette slik at bare en del var synlig så det ut som det ble scrollet. Dette er samme metode som vi bruker for å scrolle søkeresultatet.

For å få det hele til å se ut som dropdownbokser satte vi movie clipet med knappene synlig når man klikket på boksen og usynlig når man klikket en av knappene. Men vi fikk noen problemer, blant annet ble den satt usynlig men vi klarte ikke å få den til å dukke opp igjen, og vi hadde problemer med å få overført varabelen fra movie clipet til hovedtidslinjen. Så på grunn av tidspress bestemte vi oss for å bruke dropdownboksen som var ferdige i Flash og bare begrense antallet elementer til 22.

Et annet problem med denne dropdownboksen var at hvis det lå ting under dropdown listen og man klikker på et valg vil man også samtidig klikke på det som ligger under. Vi hadde forskjellige løsningsmetoder på dette, vi kunne:

- Gå til en annen frame der knappen som var i veien var byttet ut med et bilde av knappen.
- Sette knappen usynlig når dropdownboksen var nede(usynlige knapper er ikke klikkbare).
- Prøve å unngå at det ligger knapper akkurat der dropdownboksen kommer.
- Legge en dummy knapp oppå det som ligger i veien som settes synlig,når dropdownboksen er nede (det er bare den øverste knappen som virker).

Vi valgte å unngå den første metoden, ettersom vi da måtte ha alle form elementene over to frames. Så vi forsøkte å sette knappen usynlig når vi klikket på dropdown boksen. Men problemet var at knappen dukket opp igjen samtidig med at man trykket i menyen og da var man like langt. Mens vi forsøkte å finne en løsning på dette fant vi ut at vi kunne lage menyen slik at man må holde museknappen nede for å velge noe i menyen og den spretter opp igjen når man slipper museknappen. Dette fant vi ut at var en god løsning på problemet.

En annen ting vi har merket oss er at de ikke blir oppdatert, om man legger inn et nytt tema, eller en ny kategori. Vi har prøvd å løse problemet ved å nullstille variablene i dropdown boksene, men den ble fortsatt ikke oppdatert, så dersom man legger nytt tema/kategori må man reloade hele filmen for å få oppdatert dropdownboksene.

Tilbakemelding til brukeren

Til dette bruker vi et movie clip(Lagret_svarMC) som inneholder en animasjon, et tekstfelt, og en knapp. Tekstfeltet har som standard teksten "Loading…" slik at hvis det tar litt tid før resultatet kommer ser brukeren at den holder på å laste. Dette movie clipet er satt usynlig som default, og når brukeren klikker på en lagre knapp blir det satt synlig og animasjonen kjøres. Teksten som vises kommer fra ASP filen eller i enkelte tilfeller direkte fra Flash. OK knappen setter det usynlig igjen og setter teksten tilbake til "Loading…".

Checkboksene

Checkboksene består av et movieclip med to frames, i første framen er boksen ikke avkrysset men i den andre er den det. Dette er for at brukeren skal se om den er avkrysset.
Oppå dette movie clipet ligger det en knapp som setter en variabel til en eller null avhengig av hva den var. Var den en settes den til null og omvendt. Samtidig med dette sies det fra til det underliggende movie clipet om at det skal gå til enten frame en eller to.
Settes variabelen til en(påslått) går man til framen i movie clipet hvor boksen er avkrysset.

TAB rekkefølge

En annen ting vi oppdaget var at TAB rekkefølgen var helt vill, om man satte skrivemerket i første tekstfeltet og trykte på TAB hoppet den bort i menyen en tur først før den kom til neste tekstfelt. Dette var noe vi syntes burde virke slik at brukeren slipper å bruke musa for å flytte skrivemerket mellom feltene. Løsningen på dette var at det måtte programmeres. For å få det til å virke laget vi en usynlig knapp som vi koblet til dette scriptet:

```
on (keyPress "<Tab>") {
    if (Selection.getFocus() == "_level0.tittel") {
        Selection.setFocus("ingress");
    } else if (Selection.getFocus() == "_level0.ingress") {
        Selection.setFocus("tekst");
    }
}
```

Dette gjør at om markøren står i "tittel" feltet og man trykker på TAB hopper den til ingress, og deretter "tekst". getfocus og setFocus virker imidlertid bare på tekstfelt så vi har ikke laget noen TAB rekkefølge på knappene.

Vi har nå funnet ut at det er mulig å laste ned et smart clip fra macromedia exchange for å sette TAB rekkefølgen, men vi har ikke hatt tid til å teste dette.

Scrollfelt tekst

Scrolling på tekstfelt er heller ikke automatisk i Flash og må programmeres. Vi bestemte oss for å lage de slik at når man holder musepekeren over pilene i scrollfeltet, skulle teksten scrolles. Vi oppdaget at det ikke var like enkelt som vi håpet, vi forsøkte først å legge scriptet som utfører scrollingen på selve knappen i en "on (rollover)" setning men da scrollet den et hakk i det vi førte musepekeren inn på knappen og ikke noe mer.

For å løse dette problemet brukte vi et movieclip(uptrigger og downtrigger) som går i kontinuerlig loop over to frames når man holder musepekeren over scrollknappen. I den ene framen movie clipet looper over ligger koden for å scrolle teksten, dermed kjøres denne koden flere ganger så lenge man holder musepekeren over scrollknappen. Når man så tar bort musepekeren fra scrollknappen sier man fra til movie clipet at det skal gå til en frame med stop.

Det er veldig greit å scrolle tekstfelt i flash, vi brukte denne koden:

```
/tekst:scroll = /tekst:scroll-1;
```

Scrollfelt søkeresultat

Da vi skulle lage scroll på søkeresultatet fant vi ut at det var litt verre ettersom vi da måtte scrolle et movie clip og da kunne vi ikke bruke "scroll" funksjonen i Flash.

Vi laget en maske som gjør at bare en del av listen med artikler er synlig, masken står stille og vi flytter listen bak denne, dermed ser det ut som den scrolles.

Vi brukte samme metode som for scrollfelt på tekst, med et movie clip som looper(se scrollfelt tekst ovenfor). Vi laget to nye movie clip(soek_uptrigger og soek_downtrigger), de er identiske med uptrigger og downtrigger som vi bruker for å scrolle tekst, med unntak av koden på frame 2. Den byttet vi ut med dette som flytter movie clipet oppover :

```
scroll = (_root:antall * (-40));
scroll = (scroll + 540)
if (Number(_root.Soek_MC._y)>= scroll) {
   setProperty (_root.Soek_MC, _y, getProperty(_root.Soek_MC, _y)-15);
}
```

Den første linjen tar antall elementer i menyen og ganger med avstanden mellom disse, så plusses høyden av den synlige delen av feltet på dette. Dette må gjøres for at movie clipet skal stoppe på riktig plass, ellers ville det scrollet ut av skjermen og blitt borte. Så settes y posisjonen til movieclipet så lenge plasseringen er større eller lik det tallet vi har i "scroll".

Dynamisk meny med søkeresultatet

For å få til mulighet for å velge å redigere en artikkel i søkeresultatet laget vi det som en dynamisk meny. Vi lagde et movie clip(menu_template_clip) med en knapp og et dynamisk tekstfelt som skulle inneholde tittelen til artikkelen, dette puttet vi inn i et annet movie clip (menu item group). På denne måten kunne vi scrolle alle menyvalgene ved å flytte dette movie clipet(se scrollfelt søkeresultat over). Variablene i søkeresultatet kommer på formen:

&artid1=X&tittel1=X&artid2=Y&tittel2=Y……&antall=Z

Dette blir lastet inn i hovedtidslinjen og etterpå tatt inn i movie clipet. Antall forteller hvor mange artikler som ble funnet, slik at vi vet hvor mange ganger vi skal duplisere movie clipet. Så legges artid1 og tittel1 inn i den første kopien av movie clipet, artid2 og tittel2 i den neste osv. På knappen ligger et skript som kaller "rediger_artikkel" som henter artikkelen med gjeldende artikkelid og går til formen for å redigere.

Testing, kvalitetssikring

Vi har testet adminsiden i Internet Explorer og den ser ut til å virke fint. Det er også Explorer vi har brukt til testing underveis så vi er ganske sikre på at den skal virke med denne.

Da vi skulle teste den i Netscape communicator 4.7 fikk vi problemer med å logge inn det som skjedde var at vi fikk denne feilmeldingen:

HTTP Error 405
405 Method not allowed

Etter å ha sjekket litt på internett ble vi anbefalt å installere siste versjon av Flash Pluginen, da vi gjorde det så alt ut til å virke bra. Vi har prøvd dette på flere maskiner og vi er ganske sikre på at det var pluginen som var problemet.

Så testet vi i Netscape 6.01, men her kan det være problemer med å lagre ting i databasen. Vi har også her siste versjon av Flash pluginen innstallert, men det er et eller annet som ikke virker som det skal. Noen ganger virker det bra en liten stund, men så får vi problemer med å lagre ting i databasen og hele Netscape krasjer. Vi har ikke testet siden i Opera, så vi vet ikke om den vil virke der. Dette på grunn av tidspress og mangel på direkte tilgang til programvare.

Avslutning

Konklusjon

Vi planla å finne svar på problemstillingen ved å lage en dynamisk nettside som kunne presentere nyheter som ligger i en database. Det vi gjør som er annerledes og nytt, er å lage den dynamiske nettsiden i Flash. Problemstillingen vår var å finne ut hvordan informasjon som ligger i en database kan redigeres, lagres og presenteres i en flashfilm.

Vi får svart på problemstillingen på flere måter i de løsningene vi har valgt. Artikler blir hentet ut fra databasen ved hjelp av ASP og XML, og deretter blir resultatet presentert i Flash ved hjelp av ActionScript. For å legge inn nye forfattere, kategorier og artikler i databasen blir ASP og ActionScript brukt. I administratorsiden er det også mulighet for redigering av artikler, slik at data blir hentet fra databasen og vist i denne flashfilmen. På denne måten har vi fått til toveis-kommunikasjon mellom databasen og Flash. De løsningene vi har valgt er etter vår mening et godt svar på problemstillingen.

Vi har et sluttprodukt som i store trekk samsvarer med det som ble satt som kriterier i forprosjektrapporten og i kravspesifikasjonen. Vi har til dels gått bort fra, til dels lagt til elementer i det endelige produkt. Det vi har endt opp med, er en nettavis i Flash med gode muligheter for å oppdatere og vise informasjonen basert på data fra en Microsoft Access database via ASP og XML.

- Nettavisen er helhetlig. Vi har lagt vekt på at informasjonen skal vises i samme grensesnitt, og at alle knapper har samme utforming. Dette er med på å løfte helhetsinntrykket av nettavisen.
- Flashfilmene er lagt opp for å være enkle og intuitive i bruk.

Drøftinger / diskusjoner

Resultater

Et av målene for dette prosjektet var å finne ut om Flash egnet seg som en arvtaker for HTML for å vise dynamisk innhold, og hvilke bakdeler og fordeler en slik eventuell løsning har. Den største bakdelen med Flash er at det krever mye programmering for lite resultat. Det vil si at Flash er tungvint å jobbe med i forhold til HTML, der man har et ”tag”-basert kodesystem. Flash har et programmeringsspråk kalt ActionScript, som ligner meget på JavaScript. Dette språket er mye mer komplisert å kode enn om man skulle kode HTML. De mange tidslinjene man kan ha i Flash gjør denne prosessen vanskeligere, da man trenger å ha oversikten over alle elementene til enhver tid, og sammenhengen mellom dem. Å referere til et annet objekt bør ikke nødvendigvis være lett.

Et sannsynlig skille på brukerne vil mest sannsynlig være store tunge selskaper kontra vanlige ”hjemmebrukere”. Det er en meget bratt læringskurve for Macromedia Flash. Dette fører til at de fleste vanlige brukere ofte går lei før man faktisk har utrettet noe. Men for store firmaer vil Flash være en løsning å vurdere:

- Støtte av XML. Det er forholdsvis enkelt å importere data som ligger i XML-format. Dessverre har av og til rekkefølgen av innhenting av poster mye å si om man skal inkludere disse i funksjoner i Flash. For eksempel linking i Flash: Koden hvor man sjekker om det er en artikkelid eller en tittel er sekvensbestemt, selv om man bruker en If-setning, og ”Artid” kommer før ”Tittel” i XML. En Artikkelid som skal sendes med som parameter i et funksjonskall må hentes ut før man henter posten med teksten som skal trykkes på.
- Støtte av ASP. Ved hjelp av ActionScript kan man trigge ASP-filer. Et minus kan være at ASP-filene må trigges ved full adresse, da Flash ikke takler logiske adresser i dette tilfellet. Dette har vi løst ved å sette inn en variabel som inneholder pathen, som blir satt foran alle filnavn.
- Støtte av HTML ver. 1. Denne versjonen er tatt med på grunn av tekstformatering, i stedet for at Macromedia måtte lage sin eget system. Det går rykter om at dette skal forbedres til neste versjon av Flash.
- Gode animasjonsmuligheter. Flash har muligheter for å animere tekst, grafikk og bilder på en meget enkel og profesjonell måte. Hvert objekt har hver sin tidslinje, slik at man kan få en del mangedimensjonale arrayer av tidslinjer, da et objekt (movieclip) kan inneholde mange objekter.
- Støtte for vektorgrafikk. Grafikken i Flash er vektorbasert.. Dette gjør Flash raskt og kompakt å jobbe med, på grunn av at mye av grafikken regnes ut ved hjelp av avanserte matematiske logaritmer og funksjoner.

Vi har tilpasset prosjektet mest i forhold til Internet Explorer og til skjermstørrelse 1024*768, men den fungerer også bra i Netscape. Brukerdelen er testet i Opera og virket greit der. Skjermstørrelsen spenner fra 800*600 og oppover. Ved 640*480 blir muligens teksten noe uleselig.

I brukerdelen laget vi dropupmenyer for to søkekriterier; tidsintervall og kategori. Dette gjør det enklere for brukeren å få opp de mest aktuelle treffene. I tillegg kommer det frem en tilbake-link, slik at man kan gå tilbake til det søkeresultatet man fikk uten å måtte søke på nytt.

Vi har også laget en del som er forbeholdt administratorer. Her må man logge seg på med brukernavn og passord før man kommer videre i systemet. Også her har vi lagt vekt på brukervennlighet ved å bruke dropdownbokser og form-elementer. Når administrator for eksempel velger ”ny artikkel” på menyen vil han få opp en form hvor vi har lagt opp til at han/hun kan velge hvilket tema og hvilken kategori artikkelen skal legges under.

Ønsker man å endre/slette en artikkel får man opp et søkefelt hvor man kan søke på et ord. Deretter får man opp en side med linker til treffene og når man velger en av artiklene, kommer formen opp med den aktuelle artikkelens innlagte verdier. På den måten kan man enkelt endre et av feltene eller slette hele artikkelen.

Forskjeller på Flash kontra HTML

Som nevnt i tidligere rapporter har vi laget en nettavis i HTML ved hjelp av ASP. Denne nettavisen ble laget i faget Klient- og Serversideprogrammering mot WWW. Med utgangspunkt i dette, skulle det være mulig å se på forskjeller mellom de to systemene:

- HTML krever ingen plugin. Flash krever en shockwaveplayer- eller en Flashplayerplugin fra Macromedia for å kunne vise en flash-film. Det tar ca. 8 minutter på et 56K modem å laste ned shockwave, mens det tar 1 minutt for Flash. Men de fleste brukerne går nok for Shockwave, da denne inneholder Flash.
- HTML bruker standardbaserte formselementer. I Flash derimot, kan man for eksempel lage sine egne knapper, sette på funksjon på disse, flytte de slik man vil rundt på scenen, og animere.
- I HTML må man fysisk skrive inn lengde og høyde på elementer. Grafikken i Flash er vektorbasert, slik at Flash regner ut lengde og høyde ved hjelp av avanserte matematiske logaritmer og operasjoner. Dette muliggjør skalering av objekter på en enkel og flytende måte.
- HTML er enklere å implementere. HTML-kode er såpass lett, at man kan forestille seg hva man får ut på skjermen mens man programmerer. Flash derimot, har en meget høy læringskurve, fordi man bruker en del tid på å forstå kode, og programmere denne. I tillegg gjør de mange forskjellige tidslinjene i Flash at det hele kan bli uoversiktlig. I Flash jobbes det mer visuelt enn i HTML.
- Flash har støtte for ASP og XML. Dette gjør det enklere å importere data fra disse formatene. HTML kan bli skrevet ut av ASP, men ASP kode kan ikke ligge i HTML. HTML kan trigge ASP i bruk av forms, men for at kompilereren skal forstå at det kommer ASPkode i et dokument, MÅ det være et ASP-dokument. Det samme gjelder for XML. Et XML-dokument kan bli skrevet ut som HTML ved hjelp av XSLT, som er et stylesheet for XML, og det kan bli skrevet direkte ut i browser. Men det kan ikke inkluderes i et HTML dokument uten hjelp.

ASP-avisen kontra Flashavisen

- Hvor raskt kommer teksten frem til brukeren? I ASP-avisen kjøres ASPscriptet direkte, men det tar tid å presentere dette på grunn av at mye HTML kode skal prosesseres og vises for brukeren. Flashavisen kjører ASP-scriptet, dette skal returnere XML av resultatet, og dette skal igjen deles opp i Flash. Men dette går omtrent like fort, da Flash kun trenger å konsentrere seg om teksten, og ikke bygge opp hele scenen på nytt.
- Prosjekter basert på HTML er enklere å dele opp enn prosjekter basert på Flash. Dette på grunn av at i HTML kan man fordele arbeidsoppgavene ut til personer med forskjellig arbeidsområde, mens i Flash jobber man ofte mot kun en film. Denne kan inneholde flere scener, men da må partene samarbeide meget tett. Som vanlig blir man både designer og programmerer i Flash.
- Man har større muligheter designmessig i Flash enn i HTML. I ASP-avisen fikk man det litt tunge tabell-preget over nettavisen, der man tydelig kunne se ”skillelinjer” mellom de forskjellige delene. I Flashavisen trengte man ikke holde seg innefor visse rammer. Dessuten er knappene i ASP-avisen typiske systemknapper, slik at de ikke henger sammen med resten av nettavisen. I Flashavisen derimot, lagde vi våre egne knapper, slik at de ble en del av det helhetlige bildet.

En løsning for å bruke begge teknologiene på best mulig måte er å bruke Flash mot bruker, som ofte krever en del brukervennlighet og design, mens man bruker ASP/HTML på administratorsiden. Flashavisen har bedre muligheter designmessig, med det kreves mer programmering enn ASP-avisen. Dette førte til at Flashavisen var tung å lage, for eksempel på grunn av at det er mye mer arbeid med å lage formelementer i Flash enn det er i HTML. Men en fordel med Flash er om at om man har laget for eksempel et movieclip, kan dette brukes mange ganger uten å forandre noe.

Når bilder er tatt med, så vil vi på grunn av de store fordelene Flash har for design, anbefale en kombinasjon der HTML/ASP brukes i administrator-siden, og Flash i brukersiden.

XML var tungvint å jobbe med i Flash, selv om det også var ganske lett å få skrevet ut ting når man først forsto tankegangen. Det krevde en del mer programmering for å få inn XML, men det blir enklere å overføre hovedprosjektet fra et system til et annet. Dessuten er nå de forskjellige dataene som kommer inn til Flash ferdig lagt i bås”.

Alternative løsninger og valg underveis

En av de store fordelene vi fant med Flash var at man kunne lage animasjoner av tekst og bilder meget lett. Dette er vanskelig, om ikke umulig, i HTML/CSS. Animering gir en hyggeligere ramme rundt det hele, men i vårt tilfelle, der vi skulle lage en nettavis, kunne animasjoner være irriterende for leseren. Vi valgte derfor å ikke ha en utpreget bruk av slike.

Alle tekstfeltene er i HTML-format. Dette gjør formatering av tekst mye enklere, da teksten kan legges formatert i databasen. Videre formatering av teksten kan utføres ved hjelp av ActionScript eller Flash.

Vi valgte å gå vekk fra å skrive ut de 5 siste nyhetene for den gjeldende kategori grunnet tidspress. Det hadde ikke vært noe problem å implementere koden, da denne er ganske lik mye av den koden vi har fra før.

Vi hadde planlagt en loading-funksjon som lastet inn hele flashfilmen før brukeren kom inn i hovedgrensesnittet, men også denne ble kuttet ut på grunn av tidspress.

En stort avvik fra forprosjektrapporten er muligheten for å vise bilder, lyd og video i Flash. Vi fant ut, ved hjelp av nyhetsgrupper og litt forskning på internett, at dette ikke lot seg gjøre hvis man ikke vurderte dyre løsninger i henhold til programvare eller avansert koding i PHP. Veileder på skolen kunne ikke hjelpe oss med slik PHP-programmering, og det ville ha tatt altfor lang tid om vi skulle ha funnet ut av det selv. Det fulgte med en prøveversjon av Macromedia Generator ved den versjonen av Flash vi brukte, men vi fikk aldri tid til å sette oss inn i denne eller prøve den.

Grunnen til at Flash ikke kan importere bilder, lyd og video ”on-the-fly” er fordi Flash skal være raskt og godt komprimert. Derfor er det ingen importeringsfiltre inkludert. En eventuell løsning på problemet kan være å ha en link i Flash som åpner et javascript-vindu med bilde. Her tok vi ikke stilling til lyd og video.

Kritikk av oppgaven

- Arbeidsmengden i prosjektet ble større enn vi trodde i utgangspunktet. På grunn av at prosjektgruppen var i en opplæringsfase, tok enkelte operasjoner og programmering lenger tid enn det som hadde vært normalt, i motsetning til om man kunne Flash godt fra før.
- Da vi begynte med prosjektet, hadde ¾ av gruppen aldri jobbet med Flash. Actionscript var ukjent for alle. I begynnelsen gikk mye tid med til å sette seg skikkelig inn i programmet Macromedia Flash 5, og virkelig få oversikt over mulighetene med de forskjellige verktøyene. I tillegg måtte vi lære oss ASP og XML, noe som også førte til tidspress på slutten.
- Flash har ingen gode feilmeldingsrutiner, det vil si tilbakemeldinger til bruker om hva som er galt, eller hvorfor koden ikke fungerer. Dette gjorde feilretting og koding av Actionscript nesten umulig. Flash sa ifra om den ikke fant inkluderte filer, eller om det var syntaks-feil i koden. For å kunne gå rundt dette problemet, lagde vi noen tekstfelt som vi ga variabelnavn, og skrev stadig vekk ut resultater i disse, slik at vi til enhver tid visste om noe fungerte eller ikke, og hvorfor.
- Problemer med serveren ga også lenger tid på utarbeidelse av ASP og Flash. Av og til ville serveren kjøre den ”gamle” filen, selv om endringer var gjort. Dermed måtte man endre navn på fila man prøvde å kjøre hver gang man skulle prøve noe nytt. (Dette gjaldt i programmeringsprosessen, ikke når scriptet funket.)

Videre arbeid, nye (hoved)prosjekter

Forslag til endringer:

- Det kunne vært fordelaktig med en loading-funksjon som laster inn filmen før den vises. Dette er til dels implementert i administratordelen av prosjektet, men mindre i brukerdelen.
- Scrollefunksjonen har potensiale for forbedringer. Det hadde vært mulig å lage en markør som viser hvor langt ned i artikkelfeltet man har scrollet, og en funksjon som muliggjorde dra-og-scroll.
- Det er mulig å utvide med bilder, lyd og video, men dette krever enten avansert programmering i PHP for å generere .SWF-filer ut av eksisterende jpg og gif-bilder, eller dyr programvare som Swift Generator eller Macromedia Generator.
- Utskriving av de siste nyheter i en bestemt kategori mangler. ASP-filen er ferdig laget, men det som gjenstår er å finne en løsning på hvordan man skal få både de siste nyhetene og de siste nyhetene i en kategori inn i det samme feltet, eventuelt to felt, med mulighet for å scrolle opp og ned disse feltene. Hvis man går for muligheten med kun 1 felt, må man kunne oppdatere feltet om man går ut av en kategori, og f.eks inn i en annen.
- Søkeresultatet i adminsiden kunne vært forbedret ved å legge inn et felt som viser hvor mange artikler som ble funnet. Når søket ikke resulterer i noen artikler står fortsatt knappen for å redigere der, denne kunne i dette tilfellet settes usynlig.
- Dropdownboksene i adminsiden burde ha scrolling
- Tilstanden til sjekkboksene (ferdig og aktiv) kunne hentes ut av databasen og settes riktig i formen for redigere artikkel.

EVALUERING

Organisering

Vi valgte å dele prosjektet inn i forskjellige faser. De forskjellige fasene kommer inn under mål, planlegging, strategier og gjennomføring, samt evaluering. Det første vi gjorde var å komme til enighet om en problemstilling, der målet ble å finne svaret på dette problemet. Neste fase var å legge planer for hvordan vi kunne nå dette målet.

I strategi fasen tilegnet vi oss den kunnskapen som vi i fasen for planlegging hadde kommet frem til, var nødvendig for mål oppnåelse. Denne inndelingen har fungert bra, men det hadde vært en fordel med litt mer tid på gjennomføringen, slik at vi kunne ha startet tidligere med sluttrapporten. På grunn av en eksamen og et annet prosjekt så fikk vi gjort lite med Flashavisen i starten av gjennomføringa.

I slutten av denne fasen ble det jobbet meget hektisk, fordi programmeringen som ligger bak flashsiden krevde mer energi enn det vi hadde planlagt. En annen grunn til at det gikk med mer tid enn forventet var at vi ikke hadde noen vi kunne spørre om hjelp i Flash, slik at det gikk med mye tid til å finne ut ting som egentlig var ganske enkelt. På den andre siden så kunne vi ha lært mer Flash i strategi fasen enn det vi gjorde. I de andre fasene har alt fungert bra, planene vi la har stort sett blitt realisert og vi kom i mål til riktig tid.

Under planleggingen ga vi hverandre ansvar for forskjellige områder. Dette ansvaret tok alle medlemmene meget alvorlig og gjorde en god jobb med den delen de hadde. Denne inndelingen hindret en del dobbelt arbeide. En i gruppen fikk ansvaret for databasen og han lagde denne, etter flere diskusjoner i gruppen der alle kom med synspunkter for hva som burde være med. På denne måten fikk de andre medlemmene tid til å jobbe med det de hadde ansvar for. Det er lite hensiktsmessig at fire stikker jobber med databasen når det bare er nødvendig at en person utfører dette oppdraget. (se *"Fordeling av oppgaver" s. 15*)

Denne inndelingen kan kritiseres på den måten at enkelte områder kan bli glemt. For eksempel så burde vi ha satt oss inn i actionscript litt før, men da vi delte inn ansvarsområder så visste vi ikke hvor krevende dette språket var. Vi har lært litt forskjellig etter hva vi hadde ansvar for. For eksempel så lærte han som hadde ansvaret for XML mye om dette språket, mens de andre ikke fikk fordypet seg mye på dette området.

Organisering av gjennomføring

Under fasen for planlegging hadde vi funnet ut at vi kunne bruke prosjektet i et annet fag vi hadde, som test for Flashavisen. I dette prosjektet lagde vi en dynamisk nettavis i HTML og ASP, der testet vi ut deler av strukturen og databasen som vi hadde lagd for Flashavisen. Dette prosjektet kalte vi for ASP-avisen. I fasen for gjennomføring delte vi inn ansvaret på nytt. Flashavisen hadde vi delt i to, en del for brukere av avisen og en for administrasjon.
Denne inndelingen brukte vi også på ASP-avisen. To av oss lagde den delen for administrasjonen, mens de andre to lagde den for brukere. Dette var effektivt tidsmessig og vi fikk gjort det vi skulle, men det ble litt for lite kommunikasjon mellom medlemmene.
Enkelte avgjørelser ble tatt for de forskjellige delen i siden, uten at alle gruppe medlemmene ble konfrontert med dette. For eksempel så ble det lagt inn funksjoner i nettavisen som ikke var planlagt eller diskutert i gruppen på forhånd. Dette førte til at noen av medlemmene følte seg overkjørt av andre og det var vanskelig å kritisere det som hadde blitt gjort. For eksempel så ble det laget en oversikt over de 5 siste kategorier før alle medlemmene ble spurt om de syntes at dette skulle være med på nettsiden.

Under utførelsen av Flashavisen førte også denne manglende dialogen til at det ble gjort noe dobbelt arbeide. Det er lett å utfører de forandringene som er nødvendig etter hvert som ideene dukker opp og da blir det glemte å spørre de andre i gruppen når de ikke er i nærheten. Hvis vi hadde delt inn gjennomføringen litt mer detaljert og jobbet tettere i lag så hadde vi løst dette problemet, men en slik inndeling kunne også ha ført til at vi hadde forstyrret hverandre i arbeide, slik at det har gått med mer tid enn nødvendig.

Evaluering av prosessen

I perioden før vi startet med gjennomføringen av Flashavisen lærte vi mye om Flash ved å vurdere og kritisere andre flashsider. Vi fikk også mange gode tips da vi lette på Internett etter løsninger for hvordan vi kunne svare på problemstillingen. Det var også en del gode tips å hente fra oppdragsgiveren for hvordan vi skulle gjøre forarbeide. De mailet oss en del URL til websider som vi fikk bruk for både før og da vi laget flashsiden.

En annen metode vi benyttet for å lære Flash, var å gi hverandre i oppgave at alle i gruppen skulle laget hver sin flashside og at den ble lagt på dere hjemmeside. Dette fungerte bra og nesten alle lagde en slik side, men det var bare to av medlemmen som la flashsiden de hadde laget ut på sin hjemmeside. Denne oppgaven burde vært fulgt opp med en til i litt vanskeligere grad, men vi hadde begrenset tid og tyngre Flash fikk vi lært senere i prosjektet.

Under gjennomføringen fikk vi et stort utbytte av at vi hadde laget en dynamisk nettavis i ASP og HTML. En del av ASP-koden vi skrev til denne avisen, kunne også etter forandringer brukes i Flashavisen. Det var stort sett bare SQL-setningene vi kunne ta med oss over i Flashavisen. De nye delene som vi kom frem til i ASP-avisen, ble også brukt i hovedprosjektet, slik som relaterte artikler og en oversikt over de 5 siste artiklene som er lagt inn i databasen.

Utførelsen av flashsiden foregikk på to plasser. Index.html ble laget på sør-byn studenthjem og Admin.html ble laget på skolen. I utgangspunktet hadde vi planlagt at hele gjennomførelsen av flashavisen skulle foregå et grupperom som ver hybel vi hadde leid på
Sørbyen studenthjem.

Disse ekstra utgiftene ble ikke så store siden oppdragsgiveren sa seg villig til å betale halvparten av husleien.Denne hybelen ble leid på grunn av for få grupperom på skolen og det var begrenset hvor mange PCer vi fikk legge inne Flash. Vi så det også som en fordel at gjennomføringen foregikk på en plass, slik at samarbeidet kunne bli bedre.

Forandringen av denne planen ble gjort fordi det blant annet manglet en PC på hybelen og fordi grupperommet var for lite. Flashavisen har krevd mye koding i ASP, XML og Actionscript, som ligner litt på JavaScript. Vi har høstet mange nye kunnskaper på dette prosjektet og brattest var lærings kurven under gjennomføringen.

Prosjekt gruppen har nesten klart å overholde tidsplan som er illustrert i et gantt-diagram.(*se Vedlegg G: Gantt-diagram…*). Det eneste som bryter med denne planen er at betaversjon ble en uke forsinket og derfor måtte vi også utsette det siste møtet med oppdragsgiver. Denne forsinkelsen forplantet seg utover i resten av gjennomføringen, slik at vi fikk litt mindre tid på sluttrapporten enn det som var planlagt. Det er ingen hensikt i å lage en revidert utgave av gantt-diagram, når det er så få forandringer å sette inn. Dette er å betegne som små avvik i planen som ikke har hatt noen større innvirkning på gjennomføringen.
I utgangspunktet hadde vi forventet at det skulle bli flere forskyvninger i forhold til planen derfor må vi si oss godt fornøyd med at vi har klart å holde tidsplan i den grad som vi gjorde.

Årsaken til at betaversjon ble litt forsinket var i første rekke at vi hadde satt oss litt for lite inn i actionscript og at serveren som vi benyttet krasjet en del ganger. Det har også gått med flere timer til gjennomføringen enn det som var planlagt, men det er bestandig vanskelig å beregne hvor mange timer som er nødvendig, når vi på forhånd ikke har nok kunnskaper om det som skal utføres. (*se Vedlegg I: Revidert ressursplan)*

Prosjekt som arbeidsform

Det er mange ulemper, men mest fordeler med prosjekt som arbeidsform. Blant fordelene kan vi trekke frem at det er en bedre utnyttelse av ressursene til hver enkelt person. Hvis gruppen er riktig organisert, så får den enkelte ansvar for det området som vedkommende er flinkest i. For eksempel så kan ansvaret fordeles slik at den som har mest erfaring med design får denne delen, mens den personen som liker programmering best får ansvaret for dette. På denne måten er det en større mulighet for at sluttproduktet bli bedre på flere områder, enn hvis bare en person hadde gjort oppgaven alene.

Hvis vi ikke fokusere på sluttproduktet, men det vi har lært under prosessen, så kan en person lære mer ved å gjøre oppgaven alene. Problemet med å gjøre oppgaven alene er at det hadde tatt mye lengre tid, derfor måtte oppgaven blitt mindre utfordrende. Den personen hadde også mistet anledningen til lære av de andre medlemmene i gruppen. På den andre siden så hadde vedkommende blitt tvunget til å gjøre alle oppgavene alene og kunne da heve sitt nivå på flere områder.

For eksempel så kunne en som er meget flink i programmering men uten evnen til å takle design, blitt bedre på det området som vedkommende ikke takler. En annen fordel med å jobbe alene er at det er lettere å holde oversikten over prosjektet, mens i en gruppe er det større sjanser for at den enkelte medlem mister oversikten.

Noen av bakdelene med å jobbe i gruppe for å utføre er prosjekt er at det går med mer tid til administrering jo større gruppen er. Det er også et problem hvis enkelte medlemmer i gruppen ikke fungerer i lag og hvis noen ikke har evnen til å fungere i en gruppe. I slike tilfeller kan det som skal produseres komme i skyggen av intriger innad i gruppen.
Hvis enkelte ikke gjør det de skal og henger på de andre i gruppen, oppstår det ofte gnisninger og det blir vanskelig å styre gruppen.

Det er også vanskelig å få gruppen til å fungere godt hvis medlemmene kjenner hverandre for godt og bruker for mye energi på det sosiale. I slike sammenhenger oppstår det ofte situasjoner der den enkelte holdet til bake det de mener og gir en kritikk av andres arbeide der de unnlater å fortelle sannheten i redsel for å bli uvenner med resten av gruppe medlemmene. Det kan trekkes frem mange flere argumenter for og mot denne måten å
arbeide på. En av konklusjonen er at sluttproduktet har størst mulighet til å bli best med prosjekt som arbeidsform, hvis prosjekt gruppen er i riktig størrelse og fungerer slik den skal.

Subjektiv opplevelse av hovedprosjektet

Denne evalueringen har vi valgt å dele inn slik at alle i gruppen skriver litt om hvordan de har opplevd prosjektet. Vi har alle ulik bakgrunn og kompetanse, derfor har vi også opplevd prosjektet forskjellig. Det er derfor mest riktig at alle medlemmene i gruppe, gir hver sin subjektive evaluering.

Subjektiv evaluering Ragnhild Hammerstad, gruppeleder

Det har vært moro å være gruppeleder i denne gruppen, fordi alle gjorde det beste de kunne på de områdene de hadde ansvar for. Alle i gruppen har være flink til å møte opp på gruppe møter og engasjert seg i prosjektet. Ansvars forholdene ble tatt på alvor og de oppgavene hver enkelt skulle gjøre ble utført både før og under gjennomføringen. Det har også vært et særdeles godt humør på de fleste, derfor var det bare en fornøyelse å gå på gruppe møte og arbeide i lag med denne gjengen.

Det eneste som har vært litt vanskelig var å vurdere hvilke kunnskap de forskjellige satt inne med. Enkelte ganger var det lett å gå i fellen og tro at enkelte viste mer enn de gjorde. Dette skapte et feil bilde og førte til at andre i gruppen ikke kom frem med det de kunne. Under fasen for gjennomføringen ble dette problemet eliminert.

Personlig så har jeg lært mye under prosjektet og er spesielt glad for at jeg fikk denne anledningen til å lære meg Flash. Jeg har lyst til å fortsette å jobbe med Flash og fordype meg på dette området. Dette prosjektet har også gitt meg lyst til å fortsette med denne måten å jobbe på, i andre grupper med nye prosjekter. Jeg tror ikke det blir slik at denne gruppen tar et nytt oppdrag slik at den fortsetter å eksistere.

Subjektiv evaluering Stian Torp

Opplevde gruppearbeidet meget positivt, både med henhold til samarbeid med oppdragsgiver og resten av gruppa. Labarbeidet har hovedsakelig blitt delt inn i enkeltoppgaver som hver person har jobbet med individuelt, men vi har allikevel jobbet tett sammen for å sikre et mest mulig helhetlig sluttprodukt.

Hvis vi hadde hatt mer erfaring, og ikke lærte programmet og programmeringskode mens vi holdt på med prosjektet, tror jeg vi lettere kunne ha satt opp hele prosjektet. Situasjonen førte til at ideene for implementering kom underveis, og vi måtte gjøre en del utvelgelser og endringer gjennom hele prosessen.

Subjektiv evaluering John Veflen

Egentlig er jeg ikke så glad i gruppearbeid som arbeidsform, det har lett for å bli slik at en på gruppa bestemmer alt og noen gjør ingenting. Men dette synes jeg har fungert veldig greit på dette prosjektet, kanskje fordi vi har hatt hvert vårt område som vi har hatt ansvaret for. Jeg synes også det er veldig greit å kunne fordele ting som skal gjøres slik vi har gjort så ikke alle sitter foran en datamaskin og krangler.

Mot slutten av prosjektet har vi kanskje vært litt for mye splittet i to grupper, og jeg tror nok det hadde vært en fordel for oss om vi hadde hatt litt bedre kommunikasjon mellom brukerside delen og admin delen. Men alt i alt synes jeg samarbeidet har gått veldig bra.

I starten tenkte jeg vel ikke helt over hvor mye ekstra arbeid det kom til å bli når vi skulle bruke et program som vi ikke kunne fra før, men som utbytte for det ekstra arbeidet har vi jo lært oss mye om Flash. Det hadde vært greit om vi hadde hatt en person på skolen som kunne Flash. Det har vært en del unødvendig arbeid som kunne vært unngått om vi kunne fått hjelp, men vi har brukt internett flittig og funnet mye informasjon der.

Subjektiv evaluering for Geir-Olav Goksøyr

Prosjekt som arbeidsform synes jeg er en grei måte å jobbe på. Man får utnyttet hvert gruppemedlems sterkeste sider. Prosjektarbeidet setter store krav til gruppeleder, og medlemmene for øvrig. Spesielt gjelder dette samarbeidsevner, humør og toleranse. På dette prosjektet synes jeg at gruppa har fungert bra. Vi har hatt våre diskusjoner og debatter, men generelt sett har det vært en positiv gruppe. Måten vi har delt arbeid og ansvarsområder på synes jeg har vært den absolutt mest

effektive med en slik oppgave som en todelt avis tross alt er. På den annen side har kanskje delingen spesialisert oss litt mer enn ønskelig.

Produktet vi har laget synes jeg har blitt svært bra. Jeg har problemer med å peke på ting som burde være annerledes hvis vi skulle lage den en gang til. Jeg er således svært fornøyd med forarbeidet som vi har gjort, og kunnskapen vi har tilegnet oss. Med tanke på at jeg ikke visste hva flash var når vi begynte, har jeg lært utrolig mye, og jeg er innstillt på å utvikle mine kunnskaper i flash videre etter at jeg er ferdig på HIG. Oppdragsgiveren vår har på alle måter vært en god støtte i prosjektet. Jeg tror at resultatene vi har kommet frem til i dette prosjektet kan være nyttige å ta med seg videre.

Litteraturliste

1. J. Scott Hamlin & David J. Emberton : Flash 5 Magic with ActionScript, New Riders Publishing 2001
2. Russell Chun : Macromedia Flash Advanced for Windows & Macintosh, Peachpit Press 2001
3. Cheryl Brumbaugh-Duncan : Macromedia Flash 5 from Scratch, Que Publishing 2001
4. Leete & Flinkelstein : Flash 5 for dummies, IDG Books 2001
5. Milburn & Croteau : Flash 4 Web Animation f/x & Design, The Coriolis Group 2000
6. Øivind Kolloen : Aktive Server Sider, kompendium
7. Simon North & Paul Hermans : Lær XML på 21 dager, Tano Aschehoug 2000
8. Ed Tittel & Frank Boumphrey : XML for dummies, IDG Books 2000
9. http://www.macromedia.com
10. http://www.flashkit.com
11. http://www.w3schools.com
12. http://www.hig.no
13. http://www.asp101.com
14. http://www.visualintensity.com
15. http://www.asptoday.com
16. http://polar-lights.com
17. http://www.e-mergence.co.uk/
18. http://www.hig.no/at/data/client_server/index.phtml
19. http.//www.deja.com
20. http://www.were-here.com/
21. Nyhetsgruppene alt.macromedia.flash, no.it.programmering.asp og comp.text.xml på serveren news.online.no.

Innholdsfortegnelse Vedlegg