Arbeidsnotater fra Høgskolen i Buskerud nr. 66

ARBEIDSNOTAT

ARBEIDSNOTAT

Utvikling av emnet "Applikasjonsutvikling for internett"

En analyse av endringer i faglig innhold i emnet

Gunnar Syrrist

HØGSKOLEN i Buskerud

Arbeidsnotater fra Høgskolen i Buskerud

Nr. 66

Utvikling av emnet

Applikasjonsutvikling for internett
-
En analyse av endringer i faglig innhold i emnet.

av

Gunnar Syrrist

Hønefoss, november 2008

ISSN 0807-447X

Innhold

Forord

Høgskolen i Buskerud (HiBu) har undervist i applikasjonsutvikling for internett i flere år (web applikasjonsutvikling). Emnet dreier seg om å gjøre studentene kjent med bruk av web servere og de markerings- og programmeringsspråk som er aktuelle i denne sammenheng.

Vi har i stor utstrekning basert oss på bruk av Microsoft Internet Information Services[1] (IIS) som plattform for web server.

Tidligere kollega Nils Garli underviste i emnet da jeg begynte på HiBu, og i flere år samarbeidet vi om undervisningen i emnet. Etter at han ble instituttleder, og senere har gått av med pensjon, har jeg vært alene om å undervise i emnet.

[1] Studentene blir alternativt anbefalt å benytte Apache, se http://www.apache.org/.

Innledning

Til programmering av web applikasjoner benyttes programmeringsspråk eller skriptspråk[2] som enten kjører på web server eller på web klient. Web applikasjoner benytter seg av http protokollen på web server (eventuelt https for sikkerhetsformål). Web server tar imot henvendelser om å overføre innholdet på en angitt webside gjennom denne protokollen, og formidler denne siden til klienten.

En webside som inneholder skriptkode (for eksempel Javascript eller VBscript) som skal utføres på klienten, blir overført som den er, med denne koden. En webside som inneholder kode (for eksempel VBscript/ASP eller PHP[3]) som skal utføres på web server, vil først få denne koden utført på serveren, og den resulterende webside blir sendt til klienten.

De websider som utvikles skal kunne vises i den nettleser[4] den aktuelle klient velger å benytte. Siden de fleste nettlesere kjenner Javascript (ECMAscript, som er en ECMA[5] standard), bør man velge dette skriptspråk ved programmering av skriptkode som skal utføres på klienten, siden VBscript[6] bare er kjent for Microsofts Internet Explorer. Det språket man benytter til å programmere kode som skal utføres på webserver er bare avhengig av hvilken webserver man har applikasjonen liggende på. Siden vi hadde Microsoft servere, var det mest naturlig å bruke VBscript, dvs. ASP kode.

Av aktuelle markeringsspråk som studentene skulle gjøre seg kjent med, var det til å begynne med HTML (hypertext markup language), senere kom CSS (cascading stylesheets), XML (extensible markup language) og XHTML (Extensible hypertext markup language)[7].

På andre webservere enn IIS var det ikke VBscript som ville være det aktuelle programmeringsspråk for applikasjoner, men tidligere CGI (utviklet i perl) og noe senere PHP. Både perl og php er språk som har likheter med programmeringsspråket C og Java (selv om Java vel typisk anses som et språk på høyere nivå enn C).

Tidlig etter år 2000 ble det annonsert at Microsoft ville komme med en ny web plattform til erstatning for ASP, nemlig ASP.NET. .NET ble markedsført som web plattform for framtida.

På HiBu installerte vi den nyeste versjon av IIS, som ga mulighet for å kjøre applikasjoner med ASP.NET. Vi begynte å gjøre oss kjent med den, og tok med elementer fra .NET inn i undervisningen i emnet.

Tilgjengelig litteratur påpekte en del prinsipielle forskjeller mellom ASP og ASP.NET, men det ble etter hvert tydelig at vi måtte lage en vurdering av hva vi burde basere oss på som plattform for vår undervisning i web applikasjonsutvikling. Internasjonalt tydet mye på, gjennom noen år, at Apache tok mer av markedet for web servere, og Apache kjenner ikke VB eller VBscript, men for eksempel PHP.

Til nå hadde vi vurdert at likheten ved bruk av ASP og PHP var såpass stor, at vi ikke ville gi studentene noe ekstra fordeler ved å lære begge språk. De ville kunne tilpasse seg den ene omgivelsen om de behersket den andre. Om, og i hvilken grad, det samme var riktig i relasjonen mellom ASP.NET og PHP var mer uklart, og måtte undersøkes.

---

[2] Den typiske forskjell på et programmeringsspråk og et skriptspråk er at et programmeringsspråk resulterer i et program som blir kompilert/oversatt én gang.
[3] Se http://www.php.net/.
[4] For eksempel Firefox, Safari, Opera, Google Chrome eller Internet Explorer.
[5] Se http://www.ecma-international.org/.
[6] Se http://msdn.microsoft.com/en-us/library/t0aew7h6(VS.85).aspx.
[7] Alle forklaringer finnes på http://www.w3.org/.

Små eksempler brukt i samband med undervisningen, ville ikke kunne gi en tilstrekkelig bakgrunn til å vurdere dette, vi måte skaffe oss kunnskaper om utvikling av en reell applikasjon basert på den nye plattformen.

Sommeren 2004 var undertegnede i kontakt med Ringerike Næringsforum (RNF), fordi de ønsket å få utviklet et nytt websted. De hadde et eksisterende websted, men det var vanskelig å holde innholdet ved like.

Å gjennomføre et utviklingsprosjekt for RNF, ville gi nettopp muligheten for å vurdere om ASP.NET var bedre enn ASP for utvikling av web applikasjoner, og i hvilken grad dette ville være en plattform å foretrekke også i framtida (i stedet for Apache/PHP). Jeg valgte derfor å bruke RNF som formål for mitt FoU arbeid i 2004 og 2005, for å skaffe den ønskede erfaring.

Formålet med prosjektet var å forbedre det websted RNF hadde, undersøke hvilke tjenester de kan benytte seg av ut over de som ble brukt, samt undersøke utnyttelse av nyere funksjonalitet i konstruksjon av moderne web applikasjoner (ASP.NET og XML/XSL/XSLT).

Det har også vært en strategisk målsetning med prosjektet ved at Ringerike Næringsforum representerer en kanal for HiBu til å bli bedre kjent i nærområdet, og derigjennom skape bedre grunnlag for framtidige prosjekter for næringslivet i Ringeriksregionen.

Prosjektet ble startet på vår lokale server som også benyttes som kjerne i undervisningen i emnet. Høsten 2004 skulle første versjon av applikasjonen gjøres tilgjengelig for RNF hos deres ISP (internet service provider), Telenor. Det viste seg da at Telenor ikke hadde noen server som kjørte ASP.NET, og ikke hadde noen nær forestående planer om å sette opp og tilby en slik server. Flere av de sider som applikasjonen for RNF inneholdt, måtte derfor omprogrammeres til ASP, for at applikasjonen skulle komme på lufta, og det ble en forsinkelse av flere av funksjonene i applikasjonen. De funksjoner som var ment å utnytte funksjonalitet i ASP.NET ble spesielt forsinket.

Det var selvfølgelig dumt ikke å undersøke om deres ISP faktisk kjørte servere med ASP.NET, men jeg tenkte ikke på muligheten en gang, da ASP.NET allerede hadde vært i markedet et par år. Telenor tok opp sine første ASP.NET servere i februar 2005 (meg bekjent), og det var heller ikke problemfritt å benytte deres server den første perioden.

På denne bakgrunn vurderte jeg det slik at applikasjonen for RNF måtte videreføres mest mulig på ASP som fundament, og at en evaluering av ASP.NET måtte gjøres på et annet grunnlag.

Jeg ble omtrent samtidig, høsten 2004, spurt om å ta et verv som hallkoordinator for Schjongshallen (ishallen i Hønefoss), siden to av mine sønner spilte ishockey. Dette ville omfatte ansvaret for fordeling av treningstider i hallen, og utleie av istid i perioder av året hvor ikke Ringerike Ishockeyklubb (RIK) brukte ishallen fullt ut.

Jeg vurderte det slik at de Excel filer som var brukt til publisering av treningstider i hallen til nå, kunne forbedres og forenkles ved å benytte web applikasjonsutviklingsverktøy, treningstider og kamptider kunne gjøres tilgjengelig på internett, og potensielle leietakere kunne se hvilke tider som var tilgjengelig.

Denne utviklingen startet jeg med på kveldstid. Men da det viste seg at prosjektet for RNF ikke kunne gi den erfaring jeg trengte for vurdering av ASP.NET, bestemte jeg meg for å benytte prosjektet for RIK som plattform for å foreta denne vurdering.

Sammendrag

For å få et grep på hvilken funksjonalitet den aktuelle applikasjon omfatter, kan det være fordelaktig å ta en titt på vedlegget som beskriver applikasjonen før rapporten leses i sin helhet.

Som jeg nevnte innledningsvis innholdt tidlig litteratur en punktvis oppstilling av hva som måtte gjøres med en ASP webside for at denne skulle kunne kjøre under ASP.NET. Det ble fort tydelig at en slik punktvis oppstilling ikke på noen måter klargjorde hva som var forskjellen mellom ASP og ASP.NET.

Dette var ikke en oppdatering eller en ny versjon av et programmeringsspråk, men et nytt språk. På flere områder ser ASP.NET til forveksling ut som ASP, på andre områder er forskjellene store. Framfor alt må man tenke løsning av problemet sitt på en annen måte.

Som jeg nevnte innledningsvis hadde vi tidligere besluttet å lære studentene ASP, i stedet for PHP. Dette var et valg som ikke i stor utstrekning påvirket hvilke av språkene de kunne arbeide med i framtida. Når det gjelder forholdet mellom ASP.NET og PHP, er forskjellene i tankegang fundamentale, som de også i stor utstrekning er mellom ASP og ASP.NET.

Tatt i betraktning at ASP.NET var et proprietært produkt (som jo også ASP var), og at overføringsverdien til andre åpne produkter var minimal, så ville det måtte foretas en prinsipiell vurdering av om vi for framtida skulle undervise i Microsoft baserte metoder, eller om vi skulle bytte til metoder basert på åpne standarder og produkter.

Den prinsipielle vurdering var enkel: Vi bør basere oss på åpne standarder.

Den praktiske vurdering var nesten like enkel: Vi bør basere oss på programmering i PHP, som har overføringsverdi til ASP (og flere andre språk), og relasjonene mellom PHP og Java / Javascript er mer iøynefallende enn hva tilfellet er mot ASP og ASP.NET.

Det kan legges til, at etter at XML også i større grad er vurdert som et aktuelt verktøy for applikasjonsutvikling for internett, så har XSL (Extensible Stylesheet Language, som er knyttet til XML) visse likhetstrekk med ASP.NET, hva gjelder hvordan anvendelser programmeres. Men denne likheten er litt for perifer til at jeg legger noe særlig vekt på den i forhold til det prinsipielle og praktiske nøkkelspørsmålet som er presentert ovenfor.

Vi har i stor utstrekning basert oss på bruk av FrontPage til utvikling av web applikasjoner. At vi inntil videre fortsetter å benytte det, har små konsekvenser for valg av web utviklings-plattform. En utvikling mot ASP.NET ville i større grad invitert til å benytte Microsoft sitt neste utviklingsmiljø, VisualStudio. Det er i stor utstrekning utbredelse og lisens politikk som har gjort at vi ikke har valgt å bruke andre utviklingsverktøy, for eksempel Dreamweaver.

Valg av utviklingsverktøy ser jeg som et valg som kan gjøres uavhengig av valg av webserver plattform, og er derfor ikke med i denne vurdering.

På den andre side har vi i andre sammenhenger også gjort vurdering av hvilke utviklings-verktøy vi skal bruke i forbindelse med flere av våre emner.

De enkleste endringer fra ASP til ASP.NET

Flere lærebøker jeg har sett på presenterer de enkle endringer man må gjøre med en fil av type .asp for at den skal kunne kjøres under ASP.NET.

For det første må det påpekes at en webserver som kjører ASP.NET, håndterer .asp filer akkurat som før. Man trenger ikke endre dem, men man trekker da heller ingen fordeler av overgangen til ASP.NET.

En fil som skal utnytte ASP.NET er ikke av type (file extension) .asp, men .aspx.

Det språket man benytter til å kode skriptet på serveren er ikke lenger VBscript, men VB (Visual Basic). Skriptkoden kan skilles fra HTML kode ved en <script language="vb" runat="server"> . . . </script> seksjon, eller som før med kortformen <% . . . %>. **Men**: Deklarasjon av funksjoner og subrutiner kan bare gjøres i seksjoner omsluttet av <script> tag. Mer om subrutiner og funksjoner følger senere.

I ASP.NET må parametre til funksjoner og subrutiner alltid omsluttes av parenteser. Dette var ikke nødvendig i flere sammenhenger tidligere, og var derfor ofte ikke gjort.

Variable som skal brukes i koden må deklareres først (default). Dette var ikke nødvendig tidligere, men var mulig, og er en god vane. Globale variable må defineres i en seksjon med <script . . .> . . . </script>.

I ASP ble variable implisitt deklarert av typen variant, dvs. at variabelen hadde til enhver tid den type som det den refererte. I ASP.NET skal variable eksplisitt deklareres, og i de fleste sammenhenger er det nødvendig å angi typen.

Nøkkelordene set og let finnes ikke i VB. I stedet bruker man en vanlig tilordning (=). Set ble for eksempel brukt i forbindelse med variable som skulle peke på objekter.

I ASP ble parametre til funksjoner og subrutiner overført ”by reference”, i ASP.NET blir de overført ”by value”. I de aller fleste tilfeller spiller dette ingen rolle for programmereren. De fleste har ikke programmert på en måte som medfører noen forskjell.

Dette er kun en liste over små ting som må gjøres med en ordinær ASP side for at den skal kunne kjøre som en ASP.NET side. Dette ser ikke spesielt plagsomt ut, noen ergrelser blir det selvfølgelig underveis under omlegging. Men dette gir heller ikke rekkevidden på endringen fra ASP til ASP.NET, og antyder ingenting om motivasjonen for endringen.

VB er et sterkt typet språk. I enkelte sammenhenger vil du kunne basere deg på implisitt konvertering mellom forskjellige typer, men oftere enn før må du foreta eksplisitt casting (konvertering) av verdier.

En ting som er endret vesentlig har med datoer og klokkeslett å gjøre. Har du en applikasjon som inneholder dette, må dette oftest programmeres på nytt. Men på dette området er det også forskjeller mellom programmeringsspråk generelt.

Under ASP har man enkelte server side objekter, som for eksempel request objektet, response objektet og server objektet, som brukes under programmering av avanserte server side applikasjoner. Her har det også skjedd noen endringer.

Under avansert server side programmering inngår også bruk av transaksjoner. Her er det også gjort endringer. Jeg går ikke i dybden på dette her i notatet, heller ikke i særlig grad går jeg inn på server side objektene.

Å bruke request objektet til å legge til innhold på en side, ved response.write, eller å sende kontrollen til en annen side, ved response.redirect, fungerer omtrent som før.

Skjemaer, overføring av data fra klient til server

Skjemaer, HTML <form> konstruksjoner, brukes til å gi mulighet for overføring av informasjon fra en klient til webserveren. Et skjema kan inneholde felter av mange forskjellige typer, og kan inneholde få eller mange datafelter.

Det er en gylden regel å kontrollere at informasjon fylt inn i et skjema er på riktig form. En god applikasjon vil foreta de fleste kontroller hos klienten, før informasjon blir sendt til serveren. Men dette er ikke alltid mulig, en del ting må kontrolleres på serveren. Dersom webserveren fant at det var oppgitt informasjon på en ikke akseptabel måte i en ASP side, så ville kontrollen bli sendt tilbake til brukeren. En dårlig applikasjon ville gi brukeren en ”fornuftig” melding om feilen (muligens), for deretter å be brukeren om å fylle inn alle dataene på nytt. En god applikasjon ville sørge for at de data som var akseptert, eller ikke hadde blitt kontrollert, ble satt inn i skjemaet når brukeren fikk melding om hva som måtte endres. Jobben å sørge for dette for brukeren var ofte krevende programmering.

I ASP.NET er filosofien endret. Et skjema beholder de data som er fylt inn, dersom kontrollen kommer tilbake til det samme skjemaet. <form> taggen har fått et nytt attributt, runat="server", som ber serveren om å fylle inn dataene i det som kalles sidens ”viewstate”. Dette vil i formen introdusere et ”hidden” felt som kalles ”viewstate”, som webserveren kan bruke til å fylle inn de aktuelle data for klienten (se på HTML kilde koden for en .aspx side med en form). Det er ikke lett å se hvordan de aktuelle data er representert i dette hidden elementet.

For at en <form> skal virke under ASP.NET, må <form> taggen, og alle skjemafeltene, inneholde attributtet runat="server". Hvis du inkluderer et skjema på en side uten runat="server", så vil det virke som før.

Det skjedde en endring i HTML for en del år siden. Mange tagger kunne ha et attributt name. Det ble på et tidspunkt enighet om at det skulle kunne være flere elementer som skulle refereres ved navn, men nå skulle navnet (identifikasjonen) oppgis ved attributtet id.

Sider i ASP som mottar data fra et skjema, refererer til feltene i skjemaet ved det navnet som er oppgitt ved attributtet name, PHP også. Under ASP.NET refereres feltene ved navnet som er oppgitt ved attributtet id, som også benyttes i Javascript.

En side som inneholder runat="server" inneholder vanligvis ikke et attributt action=" . . . ", det er meningen at kontrollen skal tilbake til den samme siden. Derfor vil normalt sider som inneholder skjemaer være av type .aspx, og ikke .html som de ofte var tidligere (det var heller ikke uvanlig å ha en .asp side som sendte kontrollen tilbake til seg selv gjennom det riktige action attributtet).

I denne sammenheng tar jeg ikke opp problemet med forhåndsutfylling av felter i et skjema første gang det vises (hvis for eksempel formålet med skjemaet er å oppdatere opplysninger i et register). Dette må programmeres som før, men mer komplekst i noen tilfeller.

Den store endringen i forbindelse med skjemaer er at man tar utgangspunkt i at dataene i skjemaet skal bli der, inntil kontrollen går videre til en annen side. En tidligere løsning med et skjema som en .html fil som sendte kontrollen til en .asp fil, kan med de endringer som er vist innledningsvis stort sett fungere, uten å bry seg med poenget med viewstate.

På enkelte websider valgte man tidligere å ha flere former. Hvis du benytter deg av server-side <form> tagger, kan det bare forekomme én per webside. Dermed er det ikke lenger så nødvendig å teste hvilken submit knapp som er klikket. Det kan være aktuelt å teste om siden er lastet tidligere eller ikke, dvs. inneholder en viewstate. Dette kommer senere.

Sessions, webserveren kan huske brukere

Den protokollen som spesifiserer hvordan kommunikasjon mellom en klient og en webserver skal foregå heter http (hypertext transfer protocol). Dette er i utgangspunktet det som kalles en request – response protokoll, dvs. at klienten sender en forespørsel til en webserver, og får et svar tilake; ferdig med det. Hvis samme bruker sender en ny forespørsel, har serveren i utgangspunktet ikke informasjon om at det er samme bruker.

Det er mange typer anvendelser hvor man, for eksempel, ønsker å kontrollere at en bruker har rett til å benytte webstedet. Dette forutsetter en mekanisme for autentisering av brukere, og en måte å huske at en bruker er blitt autentisert (er den vedkommende gir seg ut for). Flere andre anvendelser trenger tilsvarende funksjonalitet.

For å kjenne igjen den samme bruker over flere ”besøk” er begrepet session etablert. Dette er en mekanisme som ikke er opprettet med .NET, men som har vært der på webserveren både under ASP og PHP. For de som kjenner til hva en cookie er, kan det være på sin plass å nevne at denne teknologien også baserer seg på at webserveren har sessions.

Er det så noe problem med dette i forhold til overgang fra ASP til ASP.NET? Ja! Dersom du ønsker å foreta en gradvis overgang fra ASP til ASP.NET, og er avhengig av sessions, har du et problem. I en situasjon hvor du har noen .asp filer og noen .aspx filer, må du være oppmerksom på at disse filene ikke kjøres i samme session! Dersom du har foretatt autentisering av en bruker under ASP, så må du ta vare på opplysning om dette både i din ASP og din ASP.NET session. Hvis du har blanding mellom en av disse teknologiene og PHP, vil du også ha dine .php filer i en annen/tredje session, men dette er ikke særlig vanlig.

Rettigheter på webserveren

Et annet aspekt som til en viss utstrekning ligner på problemet med sessions, er problemet som oppstår i forhold til rettigheter på webserveren, rettigheter til utførelse av skript og rettigheter til å lagre filer på serveren, for eksempel opplasting av bilder.

En bruker som ber om en fil gjennom http på en webserver, har ingen rettigheter på webserveren. Det er webserveren (som er et program eller en service, og ikke en maskin) som utføres på serveren, og det er dette programmet som er logget inn på serveren; som derigjennom har rett til å bli utført på serveren. På en webserver som kjører ASP, heter denne brukeren ”IUSR_”, etterfulgt av navnet på serveren (dette er i hvert fall det vanlige). Derfor er det denne brukeren som må ha rett til å utføre skript på de aktuelle kataloger, og eventuelt lagre filer på kataloger som er avsatt til dette.

På en server som skal kjøre ASP.NET heter denne brukeren ASPNET. Rettigheter til filer og kataloger som skal kjøre ASP.NET må derfor bli satt opp for denne brukeren også. Dette blir lett et irritasjonsmoment. I PHP brukes samme bruker som under ASP.

global.asa, web.config og machine.config

For avansert programmering av et websted, kunne man benytte muligheten til å legge inn opplysninger i en fil ved navn global.asa på roten i webstedet under ASP. For eksempel ble informasjon om forbindelse til en database lagt inn i global.asa. Programmering av denne filen er ikke helt identisk med tidligere, men endringene er av mindre betydning. Noe informasjon er nå lagt inn i filene machine.config og web.config i stedet. Den siste av disse gir mulighet til å legge inn opplysninger som er spesifikke for en gitt applikasjon, ikke bare for webstedet. Det finnes ingen slike filer under PHP (applikasjonsspesifikke).

Håndtering av events "på server"

Event er et begrep som vanligvis brukes om det som skjer på en webside i nettleseren, ikke på serveren, men i denne sammenheng er det bare server side events som er interessante.

Vi kan skille mellom forventede events og uventede events. For eksempel vil det at en bruker klikker på en submit knapp i en form være en forventet event, at innholdet i et felt forandres likedan. Dersom en post i et register blir forsøkt lest når denne posten ikke eksisterer, vil dette være en uventet event.

Forventede events vil vi programmere eventhåndteringsrutiner for, som knyttes opp til et element i HTML, mens uforutsette events må fanges i programkode.

Den konstruksjon i VB som benyttes til å fange uforutsette events ser slik ut:

```
try
        <!-- en del kode, muligens kan en uforutsett event inntreffe -->
catch
        <!-- kode som skal utføres dersom en event inntreffer et sted i koden ovenfor -->
end try
```

Dersom du for eksempel skal programmere en anvendelse som lagrer eller henter data i en database, vil denne programmeringsteknikken gi større muligheter til å gi en aktuell bruker forståelige meldinger om hva som har skjedd.

Denne måten å programmere sikrere applikasjoner er annerledes enn tilsvarende i ASP (og PHP), men er ikke ukjent i en del andre programmeringsspråk.

Forskjellen er også klar når du skal programmere eventhåndteringsrutiner (server side events finnes ikke under ASP eller PHP). Når du ønsker å fange en server side event, angir du dette ved at du legger til i koden for en submit knapp:

```
<input type="submit" id="reg" value="Registrer" onserverclick="rutine" runat="server" />
```

Det er en selvmotsigelse å si at det forekommer et klikk på serveren. Poenget er at brukeren klikker på en knapp, og kontrollen skal til serveren, og den angitte rutine skal kalles. Attributtet onclick kan ikke brukes, det blir fanget av nettleseren.

Eventhåndteringsrutinen må være definert i en <script> seksjon som påpekt tidligere. I tillegg får et kall på denne subrutinen med seg to implisitte parametre. Derfor må en deklarasjon av subrutinen ta med spesifikasjon av disse parametrene. Deklarasjonen av subrutinen rutine vil derfor se ut som følger:

```
Sub rutine (sender as Object, e as EventArgs)
        <!-- kode som skal utføres -->
End Sub
```

Dersom du skal deklarere en subrutine som du skal kalle fra ASP.NET kode, er det ingen implisitte parametre å ta hensyn til. Om du ønsker noen eksplisitte parametre med i en event rutine, må disse spesifiseres i tillegg til de to implisitte parametrene. Forskjellige type rutiner kan ha forskjellig spesifikasjon på typen parametre og skal være med.

Det finnes en forhåndsdefinert rutine Page_Load, som kalles når en side blir lastet. Der kan du teste om det er første gang siden lastes eller ikke ved bruk av Page.IsPostBack.

Dette er ikke i nærheten av hvordan tilsvarende utføres i ASP eller andre server side språk. Det er en klart proprietær måte å løse problemet. Og det er svært forskjellig fra hvordan det gjøres i andre sammenhenger.

HTML kontroller og Web kontroller

I ASP.NET er det introdusert to grupper av hjelpemidler som kalles HTML kontroller og Web kontroller. En HTML kontroll er en HTML tag som har med attributtet runat="server" og hvor server side funksjonalitet er knyttet til denne. Web kontroller blir beskrevet nedenfor.

I ASP.NET er det innført noe som kalles code-behind filer. Meningen sies å være å skille presentasjon eller innhold på en webside fra funksjonaliteten på websiden, hva gjelder server side funksjonalitet. Dette er forskjellig fra alle andre server side måter å programmere på.

En HTML kontroll og en Web kontroll inngår som en del av HTML koden på en side, ved for eksempel:

```
<asp:Literal id="meld" runat="server"></asp:Literal>      <!-- eller kortformen -->
<asp:Literal id="meld" runat="server" />                  <!-- er en Web kontroll -->
<input id="username" type="text" size="20" />             <!-- er en HTML tag, mens -->
<input id="username" type="text" size="20" runat="server" /> <!-- er en HTML kontroll -->
```

Web kontroller er ofte konstruksjoner som ikke er en direkte omskriving av en HTML tag, og har andre attributter. For eksempel vil HTML taggen ovenfor som en Web kontroll være:

```
<input id="username" type="text" size="20" />             <!-- er HTML taggen, mens -->
<asp:Textbox id="username" size="20" runat="server" /> <!-- er en Web kontroll. -->
```

Det er ikke uten videre lett å skille disse, og det er noen overlappinger. Når det er fordelaktig å bruke den ene eller den andre kontrollen er ikke alltid klart, men dersom du har data i en database som skal bindes til kontrollen, vil du være avhengig av å bruke en Web kontroll.

Å sette et prefiks på en ”tag” er en mekanisme som også er innført i samband med XML og XSLT. Det er også andre likheter.

Enkle Web kontroller: Label og Literal

En HTML kontroll og to enkle web kontroller vil forklare litt om hensikten.

```
<span id="melding" runat="server">En melding</span>
<asp:label id="melding2" runat="server">En andre melding</asp:label>
<asp:literal id="melding3" runat="server"En tredje melding</asp:literal>
```

Disse forekommer altså som en del av HTML koden på en webside. Det trenger ikke være noe innhold i taggen i utgangspunktet. I skript seksjonen er det mulig å fylle inn (annen) tekst.

```
melding.innerText = "En ny melding"
melding2.Text = "En ny andre melding"
melding3.Text = "En ny tredje melding"
```

Under ASP var den måten man la ut server genererte meldinger som del av HTML koden, at man med response.write, evt. <%= . . %>, produserte den koden som skulle vises. Denne måten å programmere på kan stort sett brukes ennå, med noen forbehold. Slik det ble gjort under ASP er stort sett likt med hvordan det er i andre språk, for eksempel PHP.

Det finnes en mengde andre HTML og Web kontroller, for eksempel Web kontrollene Linkbutton, TextBox, CheckBox. Bruk av disse i serverskript vil medføre at du programmerer websiden på en måte som ikke ligner på noe du vil gjøre i andre server side programmerings-språk. Alle kontrollene har en del egenskaper (attributter) og metoder assosiert med seg.

Andre aspekter ved Web kontroller

I HTML er det to forskjellige måter å bruke <select> (dropdown liste med alternativer), én linjes dropdown lister, og multi linje lister. Attributtet size angir om det er det ene eller andre. Det er forskjellige Web kontroller for de to formålene under ASP.NET.

```
<asp:DropDownList id="minliste" runat="server" />
<asp:ListBox id="minliste2" runat="server" />
<asp:DataList id="minliste3" runat="server" />
```

Men heller enn å angi en tom liste, vil du ofte angi alternativene, slik som

```
<asp:DropDownList id="musikk_kategori" runat="server">
        <asp:ListItem Text="Country" value="country" />
        <asp:ListItem Text="Pop" value="pop" />
        <asp:ListItem Text="Klassisk musikk" value="klassisk" />
</asp:DropDownList>
```

Men hva er poenget med dette? Det virker jo bare tungvint. Ja, etter min smak også. Hvis jeg på nytt påpeker at dette har en likhet til XML (XSL), så la oss se på en annen likhet også.

Som del av en konstruksjon kan du lage forskjellige (deler av) visninger for forskjellige tilfeller, og du kan angi flere forhåndsdefinerte eventhåndteringsrutiner.

```
<asp:DataList id="liste3" runat="server"
        RepeatColumns="3" GridLines="Both"
        OnUpdateCommand="DataList_UpdateCommand"
        OnCancelCommand="DataList_CancelCommand"
        OnDeleteCommand="DataList_DeleteCommand"
        OnEditCommand="DataList_EditCommand">
<ItemTemplate>
   <table><tr>
        <td><asp:LinkButton Text="Edit" CommandName="edit" runat="server" /></td>
        <td>Navn:</td>
        <td><%# Container.DataItem("Navn")%></td>
        </tr>
   </table>
</ItemTemplate>
<EditItemTemplate>
   <table style="background-color:#a0a0ff;"><tr>
        <td>Navn:</td>
        <td><asp:TextBox id="Box1" runat="server"
                Text='<%# Container.DataItem("Navn_1")%>' /></td>
        </tr><tr>
        <td><asp:LinkButton Text="Oppdater" CommandName="update" Runat="server" />
        <asp:LinkButton Text="Slett" CommandName="delete" Runat="server" />
        <asp:LinkButton Text="Avbryt" CommandName="cancel" Runat="server" /></td>
        </tr>
   </table>
</EditItemTemplate>
</asp:DataList>
```

Her er det flere forhold som er verdt å kommentere. For det første vil konstruksjonen med Container.DataItem bli kommentert senere, dette refererer til innhold fra en database.

For det andre angis det to måter å vise fram innholdet på, en enkel visning i listeform, og en visning av et element som skal redigeres. Hvis brukeren klikker på knappen med teksten ”Edit” (fra Linkbutton), vil dette elementet bli vist fram etter malen (templaten) EditItemTemplate, mens resten av elementene fremdeles blir vist fram etter malen ItemTemplate. Ved første visning blir alle elementer vist etter malen ItemTemplate.

Programmering ved hjelp av templates er ikke brukt før i server side programmering, men metoden ligner på den som benyttes i XSLT under XML.

Som det går fram av listingen er det knyttet eventhåndteringsrutiner til fire mulige events: Edit, Cancel, Update og Delete. Du trenger ikke benytte disse, men kan velge den eller de som er relevant for din applikasjon. De oppgitte rutiner må være deklarert i en <script> seksjon, og eventen inntreffer når for eksempel en LinkButton med den aktuelle kommando blir klikket.

En titt på en av rutinene knyttet til event Edit ser enkel ut.

```
Sub DataList_EditCommand(sender as Object, e as DataListCommandEventArgs)
        List2.EditItemIndex = e.Item.ItemIndex
        BindDataList
        MeId("")
End Sub
```

Det som gjøres er å ta vare på indeksen til det elementet som er ønsket endret. BindDataList er et rutinenavn (ganske standard), og man benytter denne rutinen til å fylle inn data i den aktuelle listen (på nytt).

Eventhåndteringsrutinen for DataList_CancelCommand er tilsvarende enkel. Men det ligger langt utenfor formålet for dette notatet å beskrive det som må gjøres i DataList_DeleteCommand og DataList_UpdateCommand. Du må gjøre mer enn du var vant til under ASP, og det ligner ikke på hvordan du programmerer tilsvarende i noen annen server side applikasjon.

I denne sammenheng vil jeg likevel legge til at det finnes noen svært omfattende Web kontroller, for eksempel er det en kalender kontroll som har stor verdi.

```
<asp:Calendar id="velgdato" runat="server" style="font-size:0.8em" width="100%"
        NextPrevFormat="ShortMonth" SelectionMode="DayWeek"
        SelectWeekText="Vis"
        onselectionchanged="Date_Selected">
 <DayStyle CssClass="dag" />
 <SelectedDayStyle ForeColor="red" />
 <TitleStyle Font-Bold="true" Font-size="1.5em" />
 <NextPrevStyle Font-Size="0.8em" Font-Bold="true" VerticalAlign="Bottom" />
 <OtherMonthDayStyle ForeColor="Gray" />
</asp:Calendar>
```

Dette er ikke på langt nær en uttømmende liste over egenskapene til denne kontrollen. To kommentarer likevel: Det er ikke lett å se hvorfor noen egenskaper angis som attributt til kontrollen og mens andre egenskaper angis ved lokale tagger. Og selv om kontrollen kan benytte CSS til å angi visning av elementer har de valgt også å kunne angi stil til hvert enkelt element på egen måte.

Å lage sin egen funksjon for dette formålet krever selvfølgelig en del programmering, men under ASP eller PHP ville man måtte gjøre det. Det er likevel mulig å tenke seg andre egenskaper ved en kalender som denne kontrollen ikke håndterer, for eksempel har den ikke mulighet for visning av ukenummer.

Bruk av databaser

De fleste websteder er ikke statiske, de kan ta imot data fra en bruker, de kan vise fram oppdatert innhold, de skal kunne foreta kontroll av en brukers identitet, eller lignende. I de fleste slike sammenhenger brukes en database.

Det finnes mange forskjellige databaser, Access (filorientert database), MySQL, Oracle og SQLServer for å nevne noen. Forskjellige programmeringsspråk har mer eller mindre egne måter å bruke den ene eller andre type database, for eksempel kjenner PHP spesielt godt til (har egne funksjoner for kommunikasjon med) MySQL. De fleste skriptspråk har muligheten for å kommunisere mot en database gjennom en ODBC kobling, gjennom at det blir angitt et ODBC navn for referanse til databasen på webserveren. Databasen kan enten ligge på webserveren, eller befinne seg på en dedisert databaseserver.

Det første som må gjøres er å oppgi en såkalt connectionstring, som inneholder tilstrekkelig informasjon til å knytte applikasjonen til databasen, og kanskje vise at du har rett til å gjøre det. En connectionstring ser forskjellig ut avhengig av skriptspråk, men er framfor alt avhengig av hvilken database du skal kommunisere med.

I ASP oppgis ofte en connectionstring i global.asa, om den ikke ligger i den enkelte webside, direkte eller indirekte. Under ASP.NET ligger den tilsvarende i web.config eller i den enkelte webside. I PHP ligger den typisk i den enkelte webside.

Når jeg sier at den ligger i den enkelte webside, kan dette gjøres effektivt ved å inkludere innhold fra en egen fil i hver side som skal benytte databasen, dermed vil de faktiske opplysninger om en connectionstring forekomme bare ett sted (programmeringsteknikk).

Under ASP trengte du ett objekt for å betjene forbindelsen til databasen:

```
Set RS = Server.CreateObject("ADODB.RecordSet")
```

Dette objektet var kjent på forhånd. Under ASP.NET trenger du to objekter:

```
Dim objConnection as SQLConnection=New SQLConnection(connString)
Dim objCommand = New SQLCommand(sql, objConnection)
```

Disse objektene er ikke direkte tilgjengelig, men man må angi gjennom en import spesifikasjon at man trenger deres innhold, slik er det med mange andre forhold også.

De objektene du trenger i PHP er forhåndsdefinert, det ligger en gang for alle i en fil php.ini hvilke ting som skal være tilgjengelig. Den enkelte webside trenger ikke angi hva som skal benyttes, heller ikke i ASP.

Det er flere forskjeller mellom hvordan en applikasjon programmeres mot en database i ASP, i ASP.NET og i PHP. PHP er nok den enkleste, men ASP har flere likheter mot PHP enn det har mot ASP.NET. Der hvor det er en fundamental forskjell oppdager du når du skal lage HTML kode ut fra innholdet i en database. Forskjellene mellom ASP og PHP er små.

For det første trenger du en webkontroll, se ovenfor. For det andre må du knytte data i en eller flere tabeller i en database (eller fra et view) til den aktuelle web kontroll. Du må angi hva nøkkelen i den aktuelle tabellen heter. Det er denne som identifiserer det enkelte Item. Du må i Page_Load rutinen, eller BindDataList, programmere opphentingen av data fra den aktuelle tabell.

Konklusjon

Jeg har belyst de elementer i ASP, ASP.NET og delvis PHP som jeg ønsker å basere meg på for å avgjøre hva som skal være plattform for videre undervisning i emnet ”Applikasjons-utvikling for internett”, som del av vårt IT bachelorstudium på HiBu.

Bygging av et websted består av

- utforming av det HTML innhold de forskjellige websider skal ha, og den struktur webstedet skal ha (de sider det skal inneholde),
- utforming av den design (fortrinnsvis gjennomgående) hver enkelt webside skal ha, og utforming av de CSS filer som trengs til dette formål, samt den grafikk som inngår,
- programmering av den kontroll som skal gjennomføres på data bruker skal registrere på webstedet, gjennomført på klienten og utformet i Javascript,
- design av database for å ta vare på aktuelle data, og
- programmering av den funksjonalitet som trengs på webserver for å ta vare på data, autentisere brukere til deler av applikasjonen, og kunne vise fram à jour innhold.

Det kan se ut som mye av det som inngår ikke blir berørt av valget av plattform for server side programmering. Dette er bare tilsynelatende. Mye arbeid med bygging av en applikasjon vil gå med til det siste punktet, og særlig for den type applikasjon som er avanserte på dette punktet vil valg av plattform være vesentlig.

> **På bakgrunn av de ting jeg har trukket fram i denne rapporten, mener jeg at den videre undervisning i emnet må basere seg på bruk av PHP til server side programmering. Da faller det også naturlig å basere seg på bruk av MySQL som tilhørende database. Emnet bør inneholde presentasjon av ASP.NET, samt bruk av andre databaser enn MySQL, men det vil være vesentlig for ikke å gjøre arbeidsbelastningen for stor for studentene at tilstrekkelig tydelig fokus holdes.**

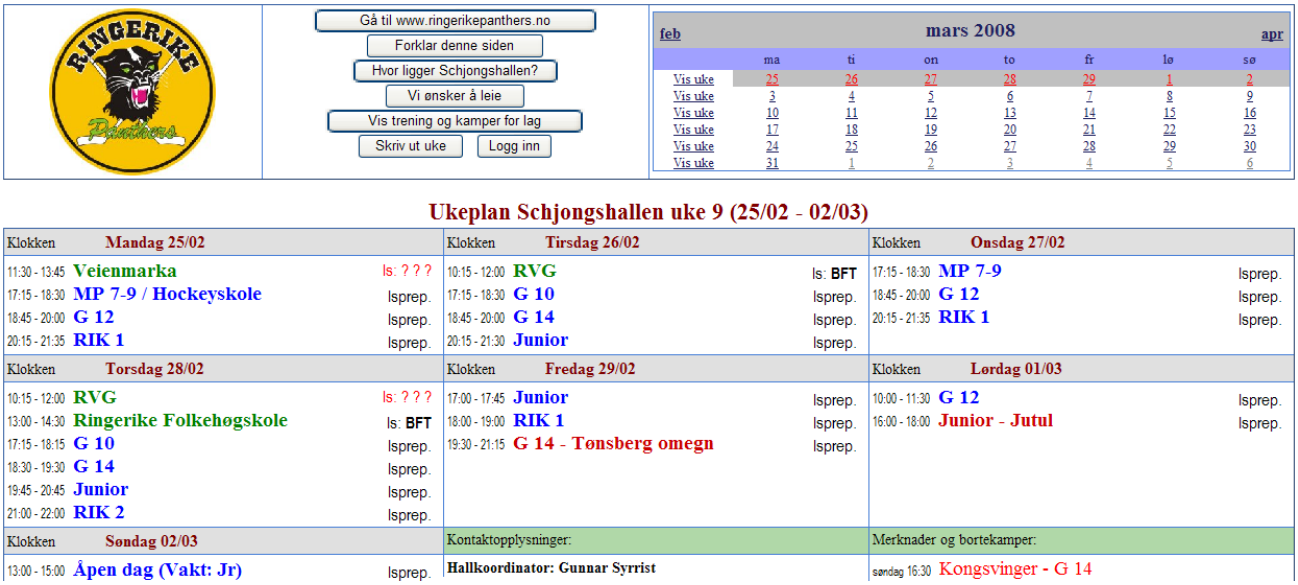
ASP.NET er en proprietær teknologi med liten overføringsverdi til andre server skriptspråk, slik som ASP, PHP, CGI eller JSP. PHP er en åpen og veldokumentert teknologi.

ASP.NET blander sammen HTML skripting og server side skripting, på en mer uoversiktlig måte enn andre server side skriptspråk, og i mye større grad. PHP har et klarere skille mellom HTML innhold og server side funksjonalitet.

ASP.NET er bare tilgjengelig på Microsoft webserver (Internet Information Services). Apache er i ferd med å bli den dominerende webserver plattform, uavhengig av operativsystem. PHP og MySQL er tilgjengelig på alle plattformer.

Vedlegg: Websted ”Ukeplan Schjongshallen”

Bildet nedenfor er tatt ved en skjermdump etter opplasting av webstedet i nettleser.



Dette er den åpne del av applikasjonen, hvor klubbens medlemmer kan slå opp treningstider og kamptider for eget lag, og hvor alle kan se når ishallen er belagt.

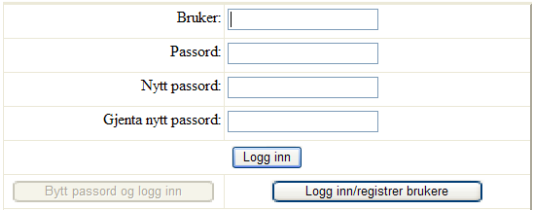
Øverst i høyre hjørne finner man en kalender, som er en av de svært nyttige elementene ASP.NET tilbyr. Øverst angis aktuell måned, og øverst til venstre eller høyre kan man bla til foregående, hhv. neste måned. Linken med teksten ”Vis uke” benyttes til å fylle den nedre del av skjermen med ukeplan for den uka. Andre hjelpemidler av denne type som er benyttet, beskrives i rapporten.

For øvrig nevner jeg spesielt knappen ”Vi ønsker å leie”, som brukes til å bygge opp informasjon vi forutsetter å motta fra en klubb (for eksempel) som ønsker å leie istid hos oss, og sender en e-post til undertegnede med dette ønsket. Da får vi lettere de opplysninger vi trenger i forbindelse med avtale om utleie.

Kalenderen benyttes av potensiell leietaker til å angi dato for ønsket leie, pluss at vi får med nødvendige opplysninger før e-post sendes.

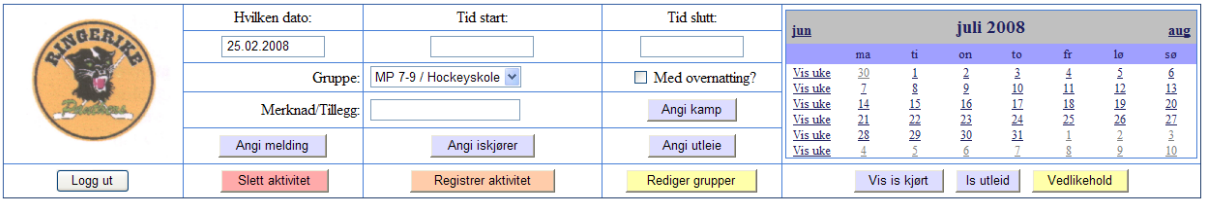
På både den første siden vist, og denne for eventuell leie, ligger det noen knapper som i det vesentlige er til forklaring av sidens funksjonalitet for brukeren.

På den første siden vil knappen ”Logg inn” gi brukeren (les administrator eller autorisert person ellers) mulighet til å komme inn i den lukkede del av applikasjonen. Avhengig av om bruker er en administrator, eller en annen autorisert bruker, vil han/hun komme inn i forskjellige deler av applikasjonen.



Administrasjonsdel av applikasjonen

Det er spesielt ved oppdatering av utnyttelse av ishallen, til trening, til konkurranse, til utleie eller til andre formål, at webapplikasjonen gir store fordeler i forhold til den tidligere bruken av Excel regneark til å produsere oppslag med ukeplan, i tillegg til at tilgjengeligheten av en oppdatert ukeplan for den som ønsker det på et gitt tidspunkt, avlaster hallkoordinator for mange telefoner.



Administrasjonsmodulen inneholder funksjoner for å legge inn eller endre treningstider, meldinger, kamper og utleie. I en ishall er det et spesielt behov for iskjørere, personer som er autorisert til å kjøre ispreppareringsmaskinen. Disse personer har en binding mot et eller flere av lagene i klubben i tillegg til at ansvaret for preparering av isen i andre sammenhenger avtales i hvert enkelt tilfelle.

Applikasjonen brukes også i kommunikasjon med iskjørerne. Alle iskjørere har autorisasjon til å kunne logge seg inn i applikasjonen, se hvilke oppgaver de har forestående, samt sette seg på oppgaver andre ennå ikke har satt seg på.

Når leietakere skal faktureres etter leie, gir applikasjonen den informasjon regnskapsfører i klubben trenger i denne sammenheng.

Til forståelse av de ting som er omtalt i denne rapporten skulle dette være tilstrekkelig beskrivelse av applikasjonen til å se rekkevidden av de vurderinger som blir gjort.

Referanser

Dette er en liste av litteratur som har vært eller blir benyttet i undervisningen i emnet.

Alex Homer, Dave Sussman, Brian Francis, ”Professional Active Server Pages”, Wrox Press Ltd 1999, ISBN 1-861002-6-10.

Stephen Walter, ”ASP.NET Unleashed”, Sams Publishing 2004, ISBN 0-672-32542-x.

Svend Andreas Horgen, ”Webprogrammering i PHP”, Stiftelsen TISIP og Gyldendal Norsk Forlag 2005, ISBN-13 978-82-05-34578-2.

Harvey M. Deitel, Paul J. Deitel, Tem R. Nieto, ”Internet & World Wide Web: How to Program”, Prentice Hall 2002, ISBN 0-13-030897-8.

Chris Bates, Web Programming: Building Internet Applications, John Wiley & Sons Ltd, 2006, ISBN 978-0-470-01775-3.

Høgskolen i Buskerud
Postboks 235
3603 Kongsberg
Telefon: 32 86 95 00
Telefaks: 32 86 98 83

www.hibu.no

HØGSKOLEN
i Buskerud