Datavetenskap

Björn Andersson

Martin Meijer

ASP och PHP

En jämförelse mellan de båda teknikerna

Examensarbete, C-nivå 10 poäng

2003:25

ASP och PHP

En jämförelse mellan de båda teknikerna

Björn Andersson

Martin Meijer

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

Björn Andersson

Martin Meijer

Godkänd, 2003-06-04

Handledare: Mari Göransson

Examinator: Stefan Lindskog

Sammanfattning

Denna uppsats behandlar de båda skriptspråken ASP och PHP. Dessa tekniker är två av de mest populära idag när det gäller att skapa dynamiska hemsidor på Internet. Denna typ av webbsidor gör det möjligt för den enskilde Internetanvändaren att styra hur sidorna ska se ut. En väldigt vanlig applikation som gör en hemsida dynamisk, är de populära gästböckerna. Den andra typen av webbsidor är de statiska hemsidorna, vars innehåll inte kan förändras av Internetsurfaren. Dessa sidor fungerar därför mest till för att informera användarna.

Uppsatsen är av helt utredande karaktär vilket i sin tur innebär att vi inte genomfört någon programmeringsuppgift. Därefter har vi sammanställt en jämförelse mellan skriptspråken och det är denna utvärdering som avrundar uppsatsen tillsammans med vår slutsats. För att göra jämförelsen mellan språken har vi varit tvungna att gå in på andra delar än just ASP och PHP. Detta dels för att själva få klarhet inom ämnet och dels för att förklara för läsaren på ett sådant sätt att ingen information skulle kunnas misstolkas. Exempel på områden som vi tar upp vid sidan om ASP och PHP är hur skriptspråk generellt fungerar, statiska hemsidor, HTML samt hur dynamiska hemsidor fungerar.

ASP and PHP

A comparison between the two techniques

Abstract

This essay considers the two scripting languages ASP and PHP. These techniques are two of the most popular when creating dynamic webpages on the Internet today. This type of webpage enables the possibility for the individual Internet user to control the appearance of the pages. A very common application, which makes the pages dynamic, is the popular guestbook's. The other type of webpage is called a static webpage, and its content can not be changed by Internet users visiting the page. These webpages are therefore mostly used to present information to the user.

This essay is strictly of evaluating character, which means that we have not carried out any programming assignment. Afterwards we made a comparison between the scripting languages and this evaluation is the part that rounds up this essay along with our conclusion. To be able to make this comparison between these to scripting languages we had to describe some other sections besides ASP and PHP. We did this partly because we had to understand them ourselves and partly because we should explain them to the reader in such a way that nothing could be misinterpreted. Some examples of areas around ASP and PHP are how scripting languages generally works, static homepages, HTML and how dynamic homepages work.

Innehållsförteckning

Figurförteckning

Tabellförteckning

1 Disposition

Detta första inledande kapitel i C-uppsatsen ”ASP och PHP – En jämförelse mellan de båda skriptspråken” kommer att fungera som en intakt till uppsatsen och sammanfattar kort uppsatsens resterande kapitel. Läsaren kommer alltså i detta kapitel få en övergripande översikt om hur vi valt att lägga upp arbetet.

I kapitel 2 kommer vi att gå igenom introduktionen till denna C-uppsats. Vi tar här upp vad syftet med uppsatsen är, vilka avgränsningar som vi bestämde oss för, vilka metoder vi använde oss av, alltså varifrån vi fick informationen, samt vad vi har haft för grund i vår framställning av uppsatsen.

Kapitel 3 handlar om statiska hemsidor och i det kapitlet får läsaren reda på vad en statisk hemsida är och hur de kan skapas. I kapitel 4 fortsätter vi prata om statiska hemsidor, indirekt genom att behandla HTML. Vi förklarar vad HTML är, dess historia samt en beskrivning av syntaxen och dess struktur. När det gäller syntaxen kommer en högst generell bild att ges, i princip kommer vi bara nämna HTML’s viktigaste beståndsdelar. Vi valde att göra på detta viset, då uppsatsens främsta avsikt var att utreda ASP och PHP. I och med detta kapitel anser vi oss ha betat av de statiska hemsidorna.

Kapitlen 5, 6 och 7 handlar om dynamiska hemsidor. I kapitel 5 tar vi upp dynamiska hemsidor genom att förklara vad dynamiska hemsidor är, beskriver dess historia, hur de fungerar samt hur användarna skapar dynamiska hemsidor. Vi introducerar även ett avsnitt som handlar om skriptspråk, både på serversidan respektive klientsidan. I kapitel 6 går vi således vidare med ASP, som är ett av de dynamiska skriptspråken vi kommer att redogöra för i denna uppsats. Vi beskriver ASP i detalj, där vi bland annat tar upp historien bakom ASP, beskriver vad ASP är, berättar hur ASP fungerar samt ger en översikt över dess objektmodell. Vi beskriver samtidigt ASP.NET och ställer ASP och ASP. NET mot varandra för att visa några skillnader. Anledningen till vi avsatte ett avsnitt som berörde ASP.NET, var att detta troligen inom kort kommer att ersätta det klassiska ASP. I kapitel 7 avslutar vi vår behandling av dynamiska skriptspråk genom att gå igenom skriptspråket PHP. Vi tar upp hur PHP kom till, vad det är, hur det fungerar samt beskriver PHP’s arkitektur. Vi ger även en introduktion till syntaxen i PHP som vi mest riktar till Hög-Data AB. Eftersom vår kontaktperson på Hög-Data AB bad oss att jämföra ASP och PHP, och att Hög-Datas anställda nästan uteslutande bara hade kunskaper inom ASP, tyckte vi att det skulle vara

lämpligt med att illustrera hur PHP’s syntax är upplagd. I kapitel 8 ställer vi de två dynamiska språken ASP och PHP mot varandra och redogör skillnader mellan dem men även deras likheter. Vi har valt att presentera ett antal aspekter som vi tyckte var intressanta att jämföra och detta beskrivs kortfattat i kapitel 8.

Kapitel 9 behandlar uppsatsens olika slutsatser, ur vårt perspektiv. En rekommendation till Hög-Data AB samt en sammanfattning av vårt arbete som helhet finns där.

I bilagorna, som återfinns längst bak i uppsatsen, har vi samlat en helhetsbild av ASP’s objektmodell, en lista med förklaringar samt förkortningar på ord som kan vara svåra att förstå, ett kodexempel i PHP samt ett kodexempel i ASP.

2 Introduktion

2.1 Syfte och bakgrund

Hög-Data AB i Karlstad, vars officiella hemsida är *www.hogdata.se*, gav oss i uppgift att göra en undersökning om två tekniker för att skapa dynamiska hemsidor, nämligen Active Server Pages (ASP) och PHP: Hypertext Preprocessor. ASP som är den äldsta tekniken av de båda är dessvärre på väg ut och ASP.NET tar nu över istället. Därför har vi valt att även gå in och klarlägga skillnader mellan ASP och ASP.NET.

Hög-Data AB har under åren de varit verksamma, använt sig av ASP tekniken när de har utvecklat sina dynamiska hemsidor. PHP är ett relativt nytt sätt att skapa dynamiska hemsidor på. För något år sedan så fanns det inte så många som visste vad PHP var. Men intresset för PHP har ökat närmast explosionsartat det senaste året och därav kommer även Hög-Datas intresse för just PHP. Personalen på Hög-Data kan i framtiden tänka sig att gå över till PHP om det nu visar sig att det är ett snabbt, enkelt och säkert programmeringsspråk. Men steget till PHP är långt pga. att de flesta anställda på Hög-Data AB är väl insatta i hur ASP fungerar och är nöjda med det systemet. Samtidigt är de som är ansvariga för webbplatsen nyfikna på hur PHP fungerar i stort och vad som har fått det att väcka så stort intresse på bara några år.

Vi kommer först och främst att rikta oss till Hög-Data AB när vi skriver denna rapport, men alla intresserade är naturligtvis välkomna att ta del av denna uppsats. Resultaten som vi kommer fram till i uppsatsen kommer att ligga till grund för vår rekommendation av vilket skriptspråk Hög-Data AB bör använda. När vi gör vår bedömning av vilket som vi anser vara det bästa kommer vi även ha i åtanke att Hög-Data AB just nu har alla applikationer i ASP. Arbetet går därför ut på att ta reda på skillnader och likheter i alla möjliga aspekter. Exempel på aspekter är t.ex. användarvänlighet, uppdateringsmöjligheter och utveckling, prestandan hos de båda skriptspråken m.m.

2.2 Avgränsning

Vårt arbete går ut på att ta reda på skillnader mellan ASP och PHP, och sen även göra ett större programmeringsexempel i PHP. Det senare fick vi i uppgift att göra i mån av tid och därför har vi valt att begränsa oss så att vi utför själva huvuduppgiften först.

Huvuduppgiften är att jämföra skriptspråken och att ta reda på skillnader och likheter. Extrauppgiften, om vi nu kan kalla den för det, är att vi sedan ska konstruera ett programmeringsexempel i PHP. Eftersom Hög-Data nu använder ASP skulle självklart uppgiften komma att bli en ”översättning” av ASP- koden till ett identiskt fungerande PHP-skript. Det troliga är dock att vi endast hinner med att ta med exempelkod till PHP, så att Hög-Data får en inblick i hur det kan vara att utveckla webbsidor i PHP.

2.3 Metod

Vi kommer att läsa böcker och ta hjälp av personal på Hög-Data AB för att utreda de frågor som vi vill ha svar på. Vi kommer även att söka efter information på Internet och vi kommer även att ta kontakt med Viktor Jonsson, en före detta student vid Karlstads Universitet som numera jobbar som IT-tekniker för Studentlitteratur. Han har skrivit ett antal böcker inom PHP området. En bok är ”Webbprogrammering med PHP” [12] som kommer att ge oss en stor del av den kunskapen som vi behöver om PHP- programmering och allt vad det innebär. Vi kommer också förhoppningsvis kunna ordna en intervju med Viktor Jonsson och genom den vägen få lite mer kunskaper om PHP. Denna intervju kommer troligtvis att ske via e-mail eller via telefon eftersom Viktor Jonsson arbetar i Lund.

Vi kommer även att ta kontakt med två anställda på Hög-Data AB, vilka vi kommer att använda oss av i vår jämförelse. Deras information kommer vi att ha med som en del i jämförelsen.

Metoden som vi kommer att använda oss av när vi tar fram skillnader och likheter med de båda teknikerna är att läsa information och sedan sammanställa den. Vi kommer att redovisa denna information i ett separat kapitel. Det här kapitlet tillsammans med det kapitel som fungerar som vår rekommendation till Hög-Data AB, kommer att avsluta vår uppsats och är därför de två sista kapitlen. Men redan i kapitlen som behandlar ASP och PHP kommer vi att komma in på vissa viktiga detaljer som skiljer skriptspråken åt.

3 Statiska webbsidor

I detta kapitel kommer vi att beröra statiska webbsidor, vad det är och hur det fungerar. Vi kommer även att redogöra vilket programspråk som är vanligast förekommande, samt ge ett enkelt exempel som illustrerar hur en statisk webbsida fungerar. Nästa kapitel är helt avsatt för detta språk, som också kallas för markeringsspråk.

3.1 Vad är statiska webbsidor?

En statisk hemsida är en sida där en besökare inte kan ändra eller påverka sidans innehåll. Statiska hemsidor lämpar sig bäst för mindre projekt, så som exempelvis sidor där användare av Internet skall kunna hämta någon slags information såsom text eller ladda ned olika filer. De statiska sidorna kan inte förändras av sidans besökare. Med det menar vi att de inte kan lägga till data av något slag. Termen "statisk" betonar att innehållet i webbsidan, d.v.s. HTML-koden, förblir oförändrad ända tills någon webbsideredigerare går in och modifierar text och kod för hand.

För statiska hemsidor är följande filtyper vanliga som innehåll:

- HTML-text
- PDF-dokument
- GIF- och JPG- bilder
- Programfiler i komprimerad form, t ex ZIP mm

Informationen ovan är hämtad från hemsidan med titeln *”Underhåll av webbplats med PC”* [24]. Den vanligaste typen av statiska hemsidor är de textbaserade. I de textbaserade hemsidorna kan användaren bara ta del av innehållet och därigenom inte förändra det. Punktlistan ovan illustrerar bara några exempel på filer som kan användas i statiska hemsidor. Det finns fler, men dessa är de vanligaste filtyperna. I regel finns möjligheten att lägga upp vilka filer som helst på en statisk sida om avsikten är att besökaren enbart ska kunna sparas till sin egen dator.

I kapitel 4 kommer vi att redogöra för hur en annan typ av hemsidor kan konstrueras, nämligen de sidor som kan bete sig på ett dynamiskt vis. Vad dessa hemsidor kan erbjuda

återkommer vi till, och efter det kommer läsaren också förstå varför de kallas för dynamiska sidor. De statiska hemsidorna används sålunda inte numera i lika stor utsträckning som tidigare. Den egentliga målgruppen för statiska webbsidor idag, som för övrigt också kanske är den största, är för privatpersoner. I nästa avsnitt följer en närmare beskrivning av hur de statiska sidorna fungerar.

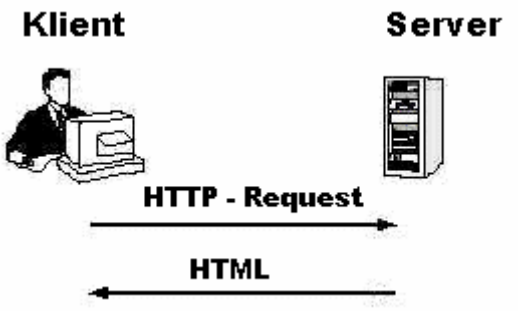
3.2 Hur fungerar statiska webbsidor?

En statisk hemsida är en sida som består av ett enkelt HTML- dokument. HTML, som står för **H**yper**T**ext **M**arkup **L**anguage, är ett programmeringssätt för att skapa just statiska hemsidor. Givetvis kan användaren sedan ta sig till andra sidor via s.k. länkar. En länk kan ta användaren till en annan sida, men den kan också utgöra en ”väg” till ett annat objekt, som exempelvis ett program eller en bild. Dessa filer (bilder, ljud, program etc.) finns i samma webbserver redo för sidans alla besökare.

Som vi nämnde tidigare är de statiska sidorna numera mest till för att visa enkla textdokument. De används också mycket till att hänvisa att en viss sida har upphört eller att den har flyttat till en annan adress. I det senare alternativet finns ofta en länk som tar dig till den nya adressen, eller automatiskt tar dig till den nya sidan efter ett visst tidsintervall.

De statiska webbsidorna är helt skrivna i HTML, och detta språk kommer vi att redogöra för i nästa kapitel. Det finns många likheter mellan en HTML- sida och en PDF- fil. (PDF är en typ av fil som enbart illustrerar text och bilder, vanligen lokalt på en dator). Användarna kan inte förändra innehållet i vare sig en HTML- sida (om inte källkoden finns tillgänglig) eller i ett PDF- dokument (om inte källtexten finns tillgänglig).

Statiska webbsidor finns normalt på en server. På samma server finns dessutom sidans eventuella bilder och ljudfiler beroende på hur sidan är skapad. En enkel bild på hur det fungerar när Internets användare begär en webbsida från en server illustreras i Figur 3.1.



Figur 3.1 Statisk hämtning av webbsida

Figur 3.1 visar hur interaktionen mellan klienten och servern fungerar. Klienten, d.v.s. du som användare, begär en hemsida (eng. http – request) i samband med att webbläsarens adressfält fylls i med en giltig adress. Servern, dvs. den dator som tillhandahåller hemsidan, skickar tillbaka den önskade filen, vanligen en HTML- fil. Detta är en högst förenklad bild över hur förloppet går till och lägg dessutom märke till att detta endast gäller för statiska sidor. I kapitel 5, som handlar om dynamiska webbsidor, ser processen lite annorlunda ut.

I nästa kapitel, kapitel 4, kommer vi att beskriva hur det vanligaste redskapet, d.v.s. HTML, kan användas för att skapa statiska hemsidor. Utan detta språk kan helt enkelt inte hemsidor konstrueras, vare sig de är statiska eller dynamiska. Men vi kommer ännu så länge bara beskriva hur HTML traditionellt har använts, dvs. ur ett statiskt perspektiv.

3.3 Sammanfattning statiska webbsidor

I detta inledande kapitel om statiska webbsidor, har vi kortfattat berättat vad en statisk hemsida är och hur den fungerar. Eftersom statiska sidor fortfarande används, framförallt i stor utsträckning av privatpersoner, skulle det vara fel och säga att det är en omodern metod att skapa hemsidor i. Men som vi kommer att se i uppsatsens senare kapitel finns det betydligt bättre och roligare sätt att skapa hemsidor i nu för tiden. Men eftersom det är relativt enkelt att skapa en hemsida som är statisk, kommer säkert denna metod att finnas kvar under en lång tid framöver.

4 HTML

Följande kapitel kommer att ta upp de statiska sidornas varumärke, nämligen HTML och HTML- kodning. Som en följd av att vi berättade om statiska webbsidor i det tidigare kapitlet, där vi nämnde att HTML är den överlägset vanligaste metoden för detta ändamål, kommer detta kapitel bl.a. ta upp hur tillvägagångssätten är för att skapa enklare hemsidor. Dessutom, som ett avslutande avsnitt analyseras framtiden för HTML och de statiska hemsidorna.

4.1 Historia

Till detta avsnitt är följande 3 källor använda, [6], [16] och [19]. HTML står, som tidigare nämnts, för **H**yper**T**ext **M**arkup **L**anguage. Termen ***Hyper*** innebär att ett HTML-dokument kan bli kodat för att ge läsaren möjlighet att komma åt andra html-dokument genom användningen av hyperlänkar. Termen ***Markup*** refererar till textformateringen i ett html-dokument. Båda dessa termer underlättar förståelsen om hur HTML fungerar.

Allt eftersom Internet tog form var det viktigt att det skulle finnas ett vanligt format för alla hemsidor att följa så att samtliga operativsystem och webbläsare skulle kunna hantera alla hemsidor. Detta var den främsta avsikten med att skapa HTML. HTML används för att skapa enklare hemsidor. De var från början menade att enbart hantera textbaserade hemsidor, dvs. statiska hemsidor. På senare tid så finns även möjligheten att hantera bilder m.m. i HTML.

HTML är framtaget ur språket SGML som betyder **S**tandard **G**eneralized **M**arkup **L**anguage. SGML är ett starkare och mer kraftfullt språk än HTML, men det är samtidigt ett mer komplext, tidsödande och svårbegripligt språk. Det var speciellt därför som det skapades ett enklare språk (HTML) som i grunden var utvecklat från SGML, men med inriktning på enkelhet och användarvänlighet. Dessutom har det visat sig att problem kan uppstå i samband med utvecklandet av webbsidor.

Det var under sommaren 1991 som den första specifikationen av HTML släpptes ut på marknaden. Som vi tidigare nämnde hade SGML använts som underlag vid framställning av HTML. Den första upplagan av HTML hette HTML 1.0 och designades av Tim Berners-Lee. Upplagan stödjer de flesta operativsystem, exempelvis Unix/Linux, Windows och Macintosh.

Allt eftersom tiden har gått så har nyare versioner av HTML tagits fram och för närvarande så är det HTML 4.0 som används. På senare tid har det tagits fram en ny version som har

egenskaper från både HTML och SGML, som senare kom att kallas XML (e**X**tensible **M**arkup **L**anguage). Det var mellan åren 96- 97 som det första förslaget kom fram om hur XML kunde användas. Det var då W3C (The World Wide Web Consortium) [23] som inledde ett försök att utarbeta en förenklad version av SGML, för att få bukt med de olika problem och begränsningar som har upplevts med HTML.

Syftet med XML var att förena enkelheten hos HTML med styrkan hos SGML. Detta eftersom begränsningar och i vissa fall stora brister upplevdes av HTML's användare, mycket pga. dess statiska karaktär. I dagsläget är säkerheten, pålitligheten och egenskapen att det ska vara lätt att uppdatera informationen i de olika systemen de kanske viktigaste aspekterna när det gäller Internetrelaterade lösningar. I februari 1998 togs den första fastslagna versionen av XML i bruk.

4.2 Struktur och syntax

Informationen till detta avsnitt är hämtad ur [2] och [23]. Vi kommer att berätta hur HTML är uppbyggt och behandla de viktigaste bitarna angående språket HTML. Olikt SGML och XML är HTML ett fixerat programspråk, med det menas att HTML har en klart definierad struktur, i form av HTML taggar, som alla HTML dokument måste följa.

Till och börja med kan vi nämna att det inte behövs några märkvärdiga utvecklings-verktyg vid konstruktion av webbsidor. Windows anteckningar (eng. notepad) fungerar alldeles utmärkt. Annars finns det en hel uppsjö av editorer, som tillhandahåller olika funktioner och olika egenskaper. Exempelvis kan en önskad egenskap hos en editor vara att den färgsätter olika nyckelord i ett programspråk. Detta används för att förtydliga HTML-koden, speciellt om mycket kod och text förekommer om vartannat.

Skulle inte kunskaper finnas hos användaren om hur HTML fungerar rent syntaxmässigt, finns grafiska program som gör det möjligt att skapa snygga webbsidor utan att kunna någonting rent kodmässigt. Exempel på sådana program är Microsofts Frontpage och Macromedias Dreamweaver. För mer information se sidorna www.microsoft.com/frontpage respektive www.macromedia.com.

Nu ska vi börja med att förklara HTMLs beståndsdelar. Som vi tidigare nämnde är ett HTML- dokument uppbyggt av taggar. Det finns väldigt många olika taggar och därför kommer vi endast att beskriva de obligatoriska (d.v.s. de som bygger upp ett vanligt HTML-dokument) samt de vanligast förekommande.

Utseendet på dessa är säkert redan bekant, men för att ändå visa det ser de ut som följer:

Starttagg: < … >

Sluttagg: </ … >

Punkterna ska sedan ersättas med något fördefinierat ord. Tecknen inom taggarna kan skrivas med stora eller små bokstäver. Som kan ses arbetar de allra flesta HTML- taggar i par (starttag - sluttagg). Det finns dock undantag och de kommer vi till vi till lite senare.

Ett HTML- dokument startar alltid med <HTML>. På liknande sätt ser då slutet på samma dokument ut så här: </HTML>. Inom dessa taggar finns hela webbsidans kod och information. Därefter kan HTML- programmeraren med fördel använda <HEAD> taggen, men det är inget måste. Head- taggen, eller webbsidans huvud som det också heter, innehåller information om sidan till Internets användare och sökmotorer. Dessutom används taggen till att definiera vissa funktioner, om skriptspråk används i HTML- dokumenten.

Inom huvudet läggs också sidans titel och det görs med <TITLE>. Ett bra riktmärke för hur sidans titel ska namnges är att den ska vara kort och koncis men ändå att den ska vara tillräckligt tydlig för att sidans besökare ska få ett hum om vad som den handlar om. Därefter avslutas både titel- och headtaggen. Den tagg som följer är <BODY> och det är detta som representerar webbsidans kropp, alltså inom <BODY> och </BODY> kommer allt visas på själva sidan. Det vanliga är att body- taggen är den största och avslutas i regel just innan HTML- dokumentet fullbordas. De taggar som vi här har redogjort för, är de som bildar stommen till en statisk hemsida. För mer information om HTML's olika versioner och vilka taggar som finns, hänvisar vi till W3C's hemsida, www.w3.org.

4.3 Vad händer sedan?

Vad händer efter HTML 4.0? Kommer det att lanseras någon ny version, en HTML 5.0?
Det var i slutet av 1997 som HTML 4.0 såg sitt ljus och sedan dess har ingen ny version av HTML tagits fram. Den senaste revideringen skedde i slutet av december 1999. Eftersom vi under avsnittet angående historian, i det inledande avsnittet i detta kapitel, berättade att XML var en bättre lösning än HTML när mer avancerade webbsidor ska byggas, är det troliga svaret nej. Och enligt W3C, är svaret följande:

> "Vid konferensen **HTML och dess framtid** som hölls i San Jose i Kalifornien i april 1998, diskuterade man hur nästa generation av HTML skall se ut. När konferensen var slut hade man kommit fram till, att en ny, utökad version blir svår att göra. Det är även krångligt att anpassa HTML 4.0 till XML. Det förslag som till slut antogs innebär att man inte gör någon ny version av HTML, utan i stället startar på ny kula, med nya kommandon som är helt och fullt anpassade till XML." [1]

Det är Jonas Ahlberg som har översatt denna notis och originalet finns publicerad på W3C's hemsida [23]. Dock har vi hittat ovanstående artikel på Jonas Ahlbergs hemsida [1]. Han fortsätter sin artikel med att berätta om XML och andra delar av SGML- familjen och hur dessa kommer vidareutvecklas. XML är likt HTML uppbyggt av en grunduppsättning av fördefinierade element men den viktiga skillnaden är att hemsideutvecklarna här kan bygga på språket med egendefinierade taggar. Det går också att utöka HTML, vilket gjordes under en period, men detta ledde till en viss inkompabilitet mellan olika HTML- verktyg [16]. Det är bland annat därför som användare av Internet ibland stöter på meddelandet ”Denna sida visas bäst i Internet Explorer” på vissa webbplatser vilket i värsta fall kanske innebär att Netscape inte kan användas. Detta pga. att sidan skapades med taggar som bara stöds av en leverantör. Som vi klargjorde tidigare var ju tanken med HTML att det skulle vara ett språk som kunde användas oavsett vilken webbläsare klienterna hade.

Så, denna anledning tillsammans med att ett mer kraftfullt verktyg behövs för att skapa hemsidor med, är de största anledningarna till att utvecklare av webbplatser troligtvis övergår mer och mer till XML. Men HTML kommer med största sannolikhet att användas som tidigare, då främst av privatpersoner och mindre företag. XML kommer att bli standarden för de riktigt stora och komplexa projekten och ytterligare en viktig punkt med XML- dokument är att de är hårdvaru- och plattformsoberoende samt att de har fullständigt stöd för Unicode. Det innebär att dokument kan utväxlas mellan användare världen över oberoende av operativsystem och teckenuppsättning. [6]

4.4 Sammanfattning HTML

I detta kapitel har vi kunnat läsa om den i särklass vanligaste metoden som finns vid skapandet av helt statiska hemsidor. Bland annat har vi stiftat bekantskap med de grundläggande HTML- taggarna, alltså de taggar som är obligatoriska i ett HTML- dokument,

och vad som troligen sker i framtiden. Detta kapitel har fungerat som en fortsättning på det förra, som behandlade statiska hemsidor, och i nästa kapitel kommer vi att lära oss vad dynamiska hemsidor är och hur de fungerar. Tanken är att kapitel 3 och 5 ska, rent uppläggsmässigt följa varandra, så att läsaren lätt ska kunna bilda sig en uppfattning om vad skillnaderna är mellan statiska och dynamiska hemsidor.

5 Dynamiska webbsidor

I detta kapitel kommer vi att beröra de viktigaste aspekterna beträffande dynamiska hemsidor. Resterande del av uppsatsen kommer på ett eller annat sätt handla om dynamiska webbsidor och vi kommer i kapitel 6 och 7 gå in på ett av uppsatsens egentliga område, nämligen att redogöra för de två absolut vanligaste teknikerna för att skapa dynamiska hemsidor, ASP och PHP. I kapitel 8 återfinns sedermera själva jämförelsen mellan skriptspråken. Så, detta kapitel kommer att fungera som en upptakt till uppsatsens återstående kapitel.

5.1 Historia

Som vi utrett i tidigare kapitel, så var Internet i ”begynnelsen” statiskt, d v s ett Internet som inte utnyttjade att vissa delar av en viss hemsida kunde genereras automatiskt. Då fungerade det helt enkelt på det viset att en klient någonstans runt om i världen begärde att få se en specifik hemsida. Den server som då hade tillgång till den aktuella sidan skickade över informationen på enklast möjliga vis. Informationen till detta avsnitt är hämtade ur källorna [17] och [22]

För att göra Internet något mer dynamiskt började webbutvecklare utnyttja s.k. CGI (Common Gateway Interface) - applikationer. CGI är en mekanism som låter webbläsaren kommunicera med en webbserver genom att exekvera olika applikationer på serversidan. Resultatet av exekveringen blir en konvertering till HTML formatet, så att en webbläsare kan behandla dokumentet. Detta HTML dokument skickas sedan till klientens dator.

Men CGI’s brister insågs ganska tidigt, eftersom metoden ganska kraftigt slöade ned webbservrarna i takt med att allt fler människor runt om i världen började ansluta sig till Internet, och därför lät Netscape bygga NSAPI (Netscape Server Application Programming Interface). Det nya programmet var inte bara ett svar på CGI’s ineffektivitet, där nu flera program kunde köras i samma process, utan det erbjöd dessutom fler avancerade funktioner och möjligheter. Nackdelen var att programmet bara kunde användas på någon av Netscapes servrar och att webbservrarna var instabila. Trots att utvecklingsspråket var C/C++ blev NSAPI inte lika populärt som CGI, mycket pga. av dessa nackdelar. ISAPI (Internet Server Application Programming Interface) är Microsofts variant till NSAPI och är ett interface till IIS (Internet Information Server). Mer om detta i nästa kapitel, det som handlar om ASP.

Enligt boken *ASP in a nutshell* [22] tog CGI alltså Internet och World Wide Web från att vara en plats där personer bara kunde dela information till en modern och framförallt mer intressant media. Tyvärr kan vi inte ge ett konkret exempel på en sådan webbplats, men denna teknik ledde till att Internet växte i popularitet och dessutom började nu skriptning på allvar på klientsidan genom klientens webbläsare. Framförallt kunde Netscapes JavaScript och Microsofts VBScript användas.

Men efter att nackdelarna med CGI så småningom insågs, har webbutvecklarna idag betydligt bättre tekniker att tillämpa. Två av dessa, ASP och PHP, kommer att kunna studeras lite närmare i de följande kapitlen.

5.2 Vad är dynamiska webbsidor?

I kapitel 3 redogjorde vi för hur statiska webbsidor var och hur de fungerade. Läsaren fick också en kort introduktion till hur enkla sådana kunde skapas i kapitel 4.

Som namnet avslöjar är de dynamiska hemsidorna mer flexibla och enklare att underhålla. Sidor som är dynamiska kan vara helt unika för varje förfrågan från en klient. Med dessa dynamiska sidor kan nu Internet- användare vara med och påverka sidans innehåll. Detta innebär i praktiken att de dynamiska webbsidorna kanske uppdateras flera gånger per dag. Det innebär i sin tur att en sida av dynamisk natur är mer intressant och eftertraktad än hemsidor som uppdateras mera sällan. Ofta söker klienter efter ny information på Internet och ser de att en hemsida är uppdaterad för väldigt länge sedan, kan det lätt bli så att sidan väljs bort till förmån för en annan, senare upplagd hemsida. Känslan kan då vara att denna uppdaterade sida har mer aktuell information att delge.

Exempel på dynamiska webbsidor är de nu för tiden vanliga och ytterst populära gästböckerna. En gästbok på Internet är en möjlighet för en webbsidas besökare att lämna en hälsning eller ett kort meddelande till webbsidans upphovsmakare. Gästböcker är relativt vanliga på hemsidor som privatpersoner har skapat, för att på så vis kanske få en liten koll på vad sidans användare tycker om utseendet, innehållet osv.

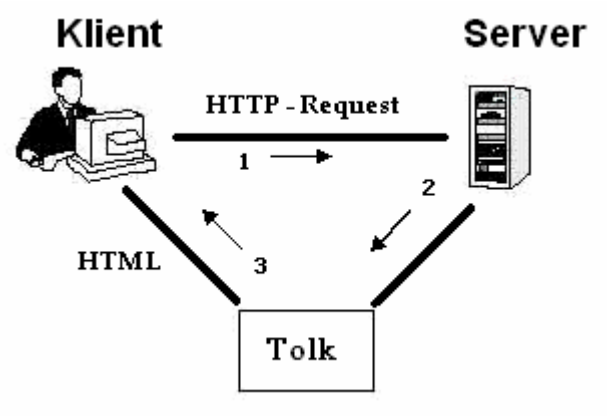
De alltfler Internetbutikerna måste också använda sig av dynamisk stomme. Databaser är mer eller mindre ett måste för de flesta affärer på nätet i dag. Databaser kan användas till mycket, bl.a. för att lagra information om vilka produkter de olika butikerna erbjuder, närliggande information om produkten, hur många produkter affären har i lager mm. I samband med att butikerna vill få sina produkter sålda, måste kunderna på något sätt registrera sig online. Detta sker också vanligtvis via en webbserver, som sparar uppgifterna i

databasen. Kunderna får skapa sig ett konto, avsett endast för deras bruk. Uppgifterna kan sedan oftast redigeras endast om kunden har tillgång till lösenordet som är förknippat med kontot.

Detta var bara två exempel på vad dynamiska hemsidor kan erbjuda och det finns många fler.

5.3 Hur fungerar dynamiska webbsidor?

De dynamiska webbsidorna fungerar inte riktigt på samma sätt som de statiska sidorna gör, dvs. de sidor som enbart är kodade i HTML. Om exempelvis en person som använder Internet vill hämta en sida som är dynamisk, ska vissa delar av koden tolkas på ett speciellt sätt. Mer om detta kommer vi att ta upp i de två följande kapitlen som handlar om ASP och PHP. Men tills vidare kan följande bild betraktas:



Figur 5.1 Dynamisk hämtning av webbsida

Bilden visar väldigt förenklat hur förloppet går till. När klienten begär en webbsida, kopplas han/hon till servern som hemsidan finns lagrad på. Som vi i senare kapitel kommer in på, känner servern av vad för slags webbapplikation som den ska behandla, och det gör den med hjälp av filändelsen. En statisk hemsida har vanligen filändelserna *.html* eller *.htm*, medan en dynamisk sida exempelvis kan ha ändelserna *.asp* eller *.php*. När det väl har kommit till kännedom att sidan inte är statisk, måste delar av hemsidans kod tolkas. De delar som då kollas, kallas skript och är skrivet i något skriptspråk. (Vi kommer att få bekanta oss med skriptspråk i nästa avsnitt.) När tolkningen är klar skapas en ny HTML- sida, beroende på vilken typ av skriptmetod som används, som sedan sänds till klienten.

Eftersom vi vid det här laget har nämnt ordet skriptspråk så många gånger utan att egentligen förklara vad det är, ska vi i nästa delkapitel klargöra detta genom att beskriva de två olika alternativ som används, nämligen server-side scripting samt client-side scripting.

5.4 Skriptspråk

Det är skriptspråken som gör det möjligt att skapa dynamiska webbsidor. Skript inbäddat i den vanliga HTML- koden tog Internet till en helt ny dimension, i och med att mer avancerade och interaktiva webbplatser nu kunde skapas. Informationen för hela delkapitel 5.4 är hämtade ur [3], [21] och [22].

Det finns många olika typer av skriptspråk. Exempel på några är PerlScript, Python, Awk, VBScript och JavaScript, där de två sistnämnda är de mest populära. En anledning till det är troligtvis att de används flitigt inom ASP, som vi kommer att berätta mer i detalj i nästa kapitel. Det finns dock en sak som alla skriptspråk har gemensamt och det är hur en webbläsare känner igen ett skript. För att instruera ett HTML- dokument om detta, skrivs starttaggen <SCRIPT>. All skriptkod ska sedan finnas mellan starttaggen och sluttaggen (</SCRIPT>), eftersom webbläsaren förväntar sig att ett skript ska finnas där.

Eftersom det kan finnas Internet- användare som fortfarande använder sig av äldre webbläsare, som inte känner igen skripttaggen, finns det ett sätt att kringgå detta problem. Det löses med hjälp av att använda sig av HTML's kommentartecken, <!-- … -->. Allt inom denna tagg kommer därmed att tolkas som om det vore en kommentar, och ignoreras av webbläsaren. Nu är det så att om klienterna har en gammal webbläsare, kan dessa inte utföra skripten ändå. Detta gäller även om kommentartaggen i HTML- dokumentet finns med. Taggen användas egentligen endast i det syfte att det inte ska inträffa något fel i samband med att en äldre webbläsare stöter på skripttaggen.

Att både ASP och PHP är skriptspråk innebär att de är designade till att utföra något endast efter att någon slags händelse har inträffat. Dessutom tillämpar både ASP och PHP en metod som kallas sever-side scripting, alltså skripten körs och exekveras på servern. Den andra varianten är att all skriptkod exekveras på klientens dator och kallas därför för client-side scripting på engelska. I följande avsnitt kommer läsaren att kunna stifta bekantskap med dessa två metoder. Här kommer vi att ta upp för- och nackdelar med dem båda samt ge exempel vi vilka skriptspråk som hör till vilken kategori.

5.4.1 Skriptning på klientsidan (client-side scripting)

Som tidigare nämndes, exekveras alla skript som är client-side på klienten dator. Detta innebär att ett skript exekverar först när hela webbsidan är nedladdad på användarens dator eller när en länk är aktiverad. Client-scripting kan alltså användas för att avlasta webbens olika servrar. Det var för övrigt därför som scriptning på klientsidan skapades. Fördelarna med att exekvera skript på klientmaskinen, är naturligtvis för att reducera serverns arbete vilket i sin innebär att ”trafiken” på nätet kan minska. De begränsningar som finns i samband med klientskript är att de inte kan användas till alltför komplexa uppgifter, såsom interagerande med databaser, och det faktum att koden för skripten inte får vara någon hemlighet, då de är synliga för alla användare. En nackdel kan vara att användarna alltid måste ha en uppdaterad webbläsare, för att klara av de senaste skripten.

Vilka skriptspråk tillhör då denna kategori? Det kanske mest kända är JavaScript och det utvecklades av Netscape i samarbete med Sun Microsystems. För ytterligare läsning se de officiella hemsidorna, www.javascript.com, home.netscape.com och www.sunmicrosystems.com. JavaScript hette från början Livescript men fick senare namnet JavaScript eftersom det påminner om programspråket Java. JavaScript är för övrigt också utvecklat med hjälp av Java.

Ett annat väldigt populärt skriptspråk på klientsidan är VBScript och det är en förkortning av Visual Basic Scripting Edition. Detta skapades av Microsoft och detta skriptspråk är baserat på programspråket Visual Basic. Ett tredje skriptspråk är JScript. Även denna är skapad av Microsoft och är väldigt lik JavaScript.

5.4.2 Skriptning på serversidan (server-side scripting)

Server-side scripting är skript som körs och exekveras på en webbserver och är alltså motsatsen till skriptning på klientsidan. Skripten som exekveras på servern ska ta hand om de uppgifter som inte kan lösas med hjälp av ett klientskript, dvs. de lite mer komplexa problemen.

Det som händer vid server-side scripting är att skriptkoden i ett HTML- dokument ersätts med resultatet av skriptkoden och sedan skickas dessa returnerande värden (från funktioner och anrop i skriptet) och den övriga HTML- koden tillbaka till användaren vid klientdatorn. Därmed har klienten ingen aning om hur det har gått till, eftersom det enda som skickades tillbaka var värden. Det innebär därmed också att koden är gömd för alla användare.

En klar fördel med skriptning på serversidan är att användarna inte behöver ha den senaste modellen av webbläsare för att vara säker på att klienterna ska kunna ta del av Internets alla

sidor. Detta eftersom att allt arbete att tolka skripten sker på servern och att användarna endast får tillbaka ett vanligt enkelt HTML- dokument, vilket alla webbläsare klarar av. Nackdelarna är självklart att webbsidan kan slöas ned markant om många användare utnyttjar applikationen samtidigt. Detta leder till att webbsidans ansvariga eventuellt är tvungna att byta till mer en kraftfull server för att klara av alla klienter.

Vilka språk tillämpar då denna typ av skriptning? Jo, bland annat ASP och PHP, som vi resterande del av uppsatsen kommer berätta om. Andra exempel är Perl (Practical Extraction and Report Language), som utvecklades för att skapa CGI- skript, och JSP (Java Server Pages).

5.5 Sammanfattning dynamiska webbsidor

Detta kapitel har tagit upp vad en dynamisk hemsida är och hur den fungerar. Efter kapitel 3, som berörde statiska hemsidor, kunde läsarna lära sig om HTML i kapitel 4 och på ett liknande sätt följer nu den, eller de vanligaste metoderna för att skapa motsvarande dynamiska webbsidor. Nästa kapitel, kapitel 6, behandlar den tidigare av de två teknikerna, ASP och i kapitel 7 berättas det om PHP, som är ett yngre språk, ursprungligen skapad för Unix maskiner.

6 ASP

Informationen i detta kapitel är hämtat ur boken *ASP in a nutshell* [22] om inget annat anges. Under detta kapitel kommer läsaren att kunna stifta bekantskap med Microsofts, vars officiella hemsida är www.microsoft.com, sätt att skapa dynamiska hemsidor. Ett ASP-exempel finns i bilaga D och exemplet beräknar arean av en triangel. Exemplet är varken stort eller märkvärdigt men ska ge läsaren en inblick i hur ASP-kod ser ut.

Det klassiska ASP kommer troligtvis inom kort helt ersättas av ASP.NET, men eftersom HögData i Karlstad främst var intresserad av en jämförelse mellan ASP och PHP är det i första hand ASP som tas upp i detta dokument. Dock kommer ett litet avsnitt beröra vissa skillnader mellan nuvarande version av ASP och ASP.NET. Dessutom kommer nästa kapitel, det om PHP upplevas som lite större än detta om ASP. Det finns också en tanke bakom detta och den är att eftersom Hög- Data nu använder ASP, och därmed kan ASP- tekniken, så behöver tyngdpunkten inte ligga på ASP- kapitlet. ASP kan dessutom användas mycket olika beroende på vilket skriptspråk som används. Skriptspråk, som behandlades i förra kapitlet, är en väsentlig punkt i ASP, eftersom användare av ASP i praktiken kan välja vilket skriptspråk som han/hon känner sig bekväm med. Nedanför följer några punkter, som ska ge en kort introduktion till vad ASP egentligen är. Den sista punkten kommer vi att behandla utförligt i avsnitt 6.4.

- Kraftfull utvecklingsapplikation
- Enkelt att använda
- Väldigt kraftfullt om vid samtidig användning av ActiveX och/eller Java komponenter
- Använder sig av något skriptspråk, vanligen VBScript eller JavaScript
- Egentligen inget programspråk eller utvecklingsplattform
- ASP är en teknik skapad av Microsoft
- Löser vissa komplexa problem som kan uppstå vid användandet av CGI
- Har tidigare varit plattformsberoende, men det finns numera mjukvaror som gör det möjligt att använda andra servrar än just Microsofts operativsystem.
- ASP bygger på en objektmodell kännetecknande av sex väsentliga objekt:

  Application, ObjectContext, Request, Response, Server och Session

6.1 Historia

För att råda bot på CGI's ineffektivitet, introducerade Microsoft ett alternativ till CGI, nämligen ISAPI (Internet Server Application Programming Interface). Anledningen till detta var att ISAPI skulle komma att ha viktig fördel gentemot CGI. Ett betydande problem med CGI är att när en klient frågar efter en applikation som finns på en server, så öppnas en separat instans av en CGI- applikation för varje begäran. I och med detta så kan denna server lätt överbelastas om den ska betjäna många klienter. ISAPI har löst detta problem med att använda DLL (Dynamic Link Library)- filer. Genom att varje ISAPI- tillämpning använder sig av en DLL- fil som automatiskt laddas in i serverns minne så fort en begäran gjorts av en första klient, undviks på detta sätt att samma applikation ”går den långa vägen” för varje klient. DLL- filen finns sedan i minnet så länge som den inte explicit raderas. Detta gör att ISAPI ökar effektiviteten av framförallt servens minne vilket i sin tur leder till att ISAPI-tillämpningar normalt är snabbare och mer pålitliga ur responstid sätt, vid en jämförelse med motsvarande CGI- applikationer. När en ISAPI- applikation väl laddats in i minnet stannar programmet där och därmed snabbar på processen för nästkommande användare.

Ytterligare en fördel med ISAPI applikationer är att det tillhandahåller olika filter. Dessa filter blir anropade av servern så fort någon klient gjort en http- förfrågan (eng. http- request). Filtret instruerar webbservern hur den ska behandla en sådan begäran och är vanligen en DLL. Exempel på filter är enligt boken *ASP in a Nutshell* [22]:

- De som utgör ett säkerhetslager mellan klient och webbserver och fungerar som ett alternativ till autenticitet där webbens användare ska ange användarnamn och lösenord.
- De filter som kan tolka strömmar av info från servern och presenterar informationen i ett annat format än originalet. Exempel är ASP.DLL, som är ett filter som förklarar serverkod i ett skript, begärt av en klient, och skickar en tolkning till klienten.

6.2 Vad är ASP?

I samband med att IIS 2.0 (Internet Information Server) allt mer användes, introducerade Microsoft en teknologi som kallades Denali och det är detta som idag kallas ASP. ASP är en teknologi där dynamiska hemsidor lätt kan skapas. Det var Microsoft som ursprungligen var företaget bakom idén och är en betydligt bättre variant på att generera dynamiska webbsidor än föregångaren CGI (Common Gateway Interface).

ASP är inte ett språk som exempelvis HTML och C/C++ är. En lämplig beskrivning kan istället vara att ASP kan ses som en teknik som gör det möjligt att lägga in programmeringsspråk (egentligen skriptspråk, exempelvis VBScript eller JavaScript) i ett HTML- dokument. Det är detta som leder till att hemsidan blir dynamisk.

ASP arbetar alltså tillsammans med något skriptspråk. Skriptspråken, som för övrigt presenterades i avsnittet 5.4 i föregående kapitel, kan väljas godtyckligt bland de som marknaden erbjuder eller så kan webbutvecklarna välja att producera ett eget skriptspråk och använda detta istället. Gemensamt för skriptkoden är att den placeras inom speciella taggar i ett vanligt HTML- dokument. Det är sedan denna kod som tolkas av webbservern när en klient begär en sida som är uppbyggd enligt ASP's princip. ASP tillämpar server-side scripting, dvs. alla skript tolkas och körs på servern vilket leder till att klienten inte kan se hur skripten ser ut i sin webbläsare. All ASP- skript finns inom vissa taggar:

```
<%
        ...
%>
```

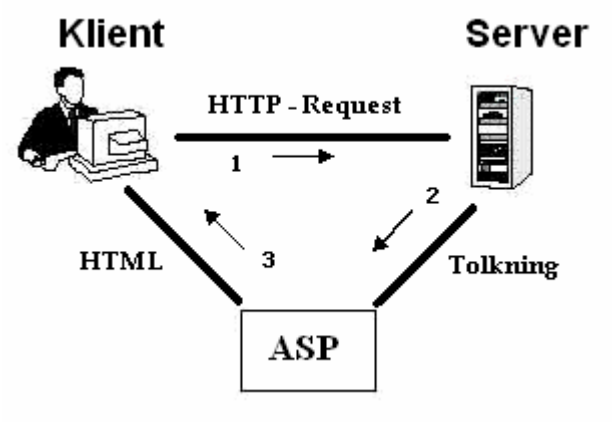
Punkterna ersätts sedan med ASP- koden. Förenklat går det att säga att ASP är inkapslat i en enda DLL- fil. Den är i storleksordningen ca 300 kB och filen är döpt till ASP.DLL. Filen är ett ISAPI- filter och finns i IIS's minne.

6.3 Hur fungerar ASP?

Så, när en användare begär en fil eller hemsida, i exempelvis adressfältet eller via en länk med ändelsen *.asp* på en server, kommer användarens webbläsare att anropa servern. Eftersom det är en ASP- fil vet servern att det begärda dokumentet först måste genomgå en viss procedur, innan sidan kan skickas till klienten. De filer som innehar ASP- ändelsen innehåller ju inte bara HTML- kod, utan det finns även ett skript inbäddat. Detta gör att innan klienten kan se

hemsidan, måste en programtolk exekvera ASP- koden och det är det här som kallas för ett skript. Den fil som tidigare nämnts, ASP.DLL, är det som utgör programtolken.

Det som händer är att ASP.DLL laddas in i serverns minne och exekverar den aktuella skriptkoden på serversidan. När exekveringen på servern är klar genereras en ny HTML- sida helt utan ASP- kod. Det är först därefter som hemsidan kan skickas till klientens webbläsare. Det användaren sedan kan se är endast vanlig HTML- kod, eftersom ASP- koden exekverades på servern. Oftast är det just HTML som genereras, men det kan också vara någon annan standard t ex XML.



Figur 6.1 Hämtning av webbsida som är av asp- typ

Figuren ovan visar enkelt förfaringssättet för hur det går till när en klient begär en hemsida som är skriven i ASP. Bilden ovan kan misstolkas en aning, speciellt när det gäller pilarna som är numrerade 2 och 3. Det är på det viset att innan klienten får upp webbsidan på sin dator, kommer tolken, här illustrerad som boxen som är märkt ASP, returnera ett resultat av skriptet tillbaks till servern. Servern i sin tur genererar en ny sida och skickar den till klienten. Som Figur 6.1 här illustreras kan personerna som studerar den, lätt få den uppfattningen att det skulle vara tolkens uppgift att vidarebefordra hemsidan till klienten, men det är det alltså inte.

Den servergenererade koden kan, beroende på hur avancerad hemsidan är upplagd, innehålla olika typer av kommandon som servern kan exekvera. Det är först efter exekveringen av dessa kommandon som HTML- dokumentet kan överföras till klientens dator.

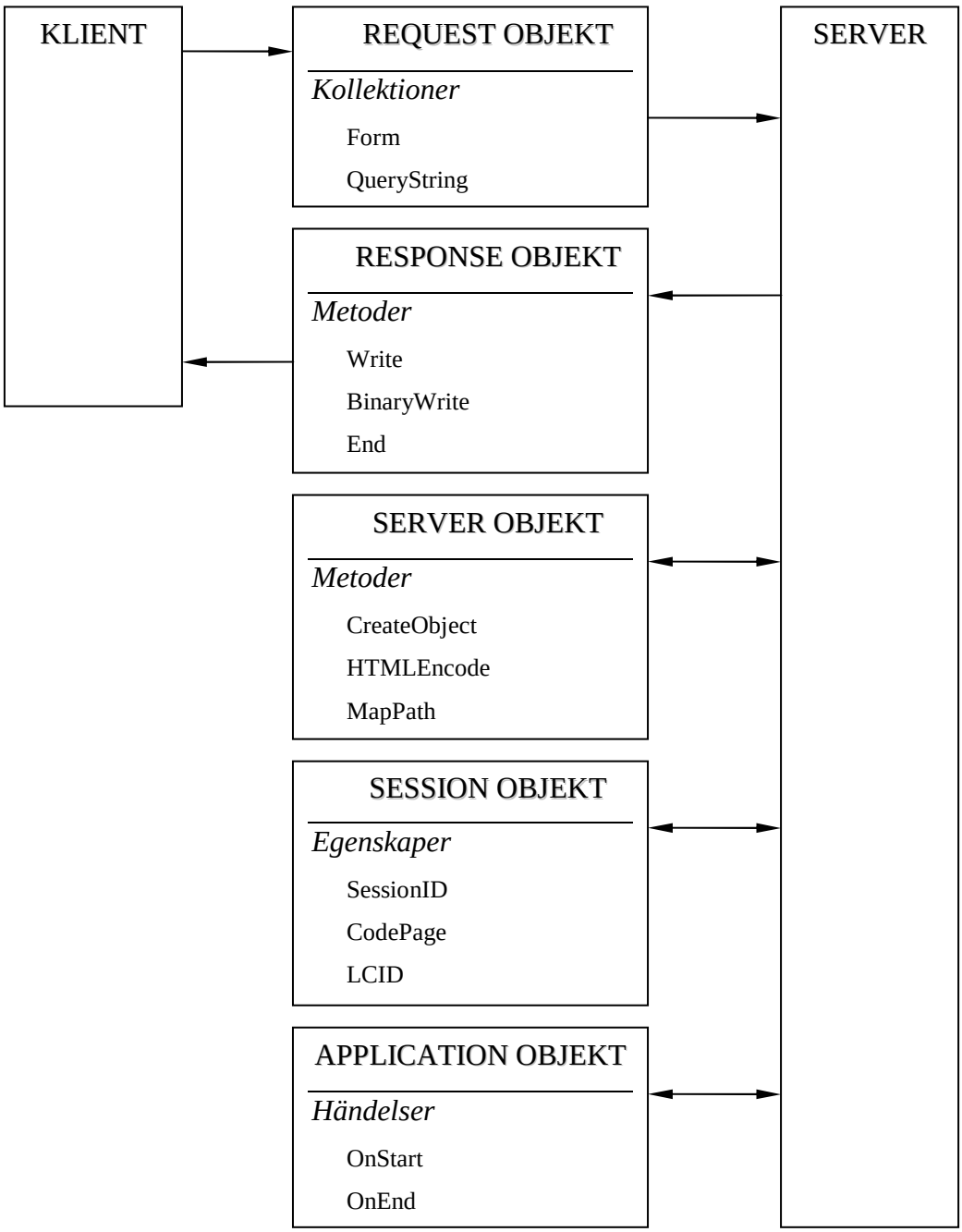
Ett exempel på ett ASP- kommando är:

```
<%=  Time() %>
```

Detta kommando används i samband med att visa aktuell tid.

6.4 ASP Objekt modell

ASP är uppbyggd enligt en viss modell som kallas ASP Objekt Modell. Denna modell består av 6 objekt och alla objekt är delar av ASP.DLL. Objekten finns för övrigt alltid tillgängliga för ASP- applikationerna. Dessa objekt blir instruerade på olika sätt beroende på vilket skriptspråk som används. Gemensamt för alla objekt är att de innehåller ett stort antal metoder och egenskaper. Det är dessa delar som skiljer de olika objekten åt och det är samtidigt dessa egenskaper och metoder programmeraren praktiskt arbetar med.



Figur 6.2 En sammanfattad ASP objekt modell

Bilden ovan illustrerar ASP objekt modell. Observera att bilden inte på något sätt är komplett utan att den mest är till för att få en övergripande överblick över hur ASP är uppbyggt. De attribut som objekten är utrustade med är de vi kommer att förklara i kommande underrubriker. För ett fullständigt schema över hur ASP- modellen ser ut, med alla objektens attribut, hänvisar vi till bilaga B. Avsnitten som följer nu berättar lite mer om vad varje objekt är och vad en programmerare kan utföra med dem.

6.4.1 Request Objekt

Detta är kanske det viktigaste av de sex objekten i ASP- modellen. Detta objekt representerar sättet ASP agerar med klientens http request (begäran). Informationen går från användarna till webbservern. Objektet innehåller all information som finns mellan användaren och servern där hemsidan finns lagrad. Exempel på information är bland annat vilken webbläsare klienten använder, vilken programvara servern utnyttjar, vilken slags uppkoppling som förbinder datorerna, vilken IP adress klienten sitter på osv. Det är alltså i http headern som den mesta informationen finns. En viktig parameter i http headern är den första raden som bland annat beskriver vilken metod som kommer användas, *POST* eller *GET*. GET används för att få tillbaka någon form av dokument vilket inte POST kan användas till. POST kan däremot utnyttjas till att skicka innehållet i ett HTML formulär till servern men det kan för övrigt kommandot GET också användas till. En god regel är dock att använda POST i HTML-formulär, som kan komma att förändras. POST är en säkrare metod och dessutom mer effektiv och därför bör inte GET användas när något ska förändras.
Den generella kodstrukturen för Request- objektet är:

```
<%

  strName = Request(”name”)

%>
```

`strName` bildar en variabel som tilldelas det värde som returneras från `Request`. Detta sätt att ange objektet är det enklaste, eftersom det inte behövs specificera någon kollektion.

Som den kompletta ASP- bilden över alla objekt i ASP i bilaga B illustrerar, innehåller Request- objektet 5 stycken s.k. kollektioner (eng. collections). Vi ska bara nämna två av dem, nämligen *Form* och *QueryString*. Form används i samband med att programmeraren i ASP tänker utnyttja POST som metod, inom FORM- taggen i ett HTML- dokument. QueryString nyttjas därmed förstås om GET istället anges.

Både Form och QueryString har de tre egenskaperna *item, key och count*. De fungerar på liknande sätt för bägge de kollektionerna. Item representerar värdet på ett element, key representerar namnet på ett element och count returnerar hur många element som existerar i kollektionen. Ett exempel skulle kunna vara:

```
<%

   strKeyName = Request.Form.Key(3)
   strKeyValue = Request.Form.Item(strKeyName)

%>
```

Exemplets första rad tar reda på namnet det tredje elementet i Form- kollektionen. Den andra raden tilldelar variabeln `strKeyValue` elementets värde. QueryString fungerar på samma sätt med den skillnaden att programmeraren byter ut Form mot QueryString.

Vi avslutar request- objektet med en kort tillbakablick som sammanfattning. Request-objektet har till uppgift att behandla de förfrågningar som användaren begärt på en sida eller applikation.

6.4.2 Response Objekt

Request- objektet kan ses som ett inmatningsobjekt. Detta innebär då att Response- objektet är den raka motsatsen, ett utmatningsobjekt. Ett kanske lättare sätt att se på detta objekt är att det kan liknas som ett svarsmeddelande till det som klienten tidigare skickade begäran (eng request) om. Alltså, informationen förflyttas från servern till klienten.

Det är detta objekt som gör att ASP arbetar dynamiskt. Respone- objektet kan därför sägas utgöra en representation av vad för slags information som överförs till klientdatorn, eftersom informationen kan komma att förändras från gång till gång. Det kan exempelvis gälla att själva sidan förändras eller att hemsidans klienter blir omdirigerad till alternativa sidor.

Som ASP- bilden i bilaga B visar, ses att objektet innehåller många olika egenskaper och metoder. Vi kommer inte att gå in på dem här men vi vill bara upplysa om att response objekt är ett kraftfullt verktyg när det gäller kontrollen om hur och vad som skickas tillbaka till klienten.

Viktiga metoder i detta objekt är *Write, BinaryWrite och End*. Det är också dessa metoder som finns med i ASP's översiktsbild, Figur 6.2. Kortfattat vill vi poängtera att de två första metoderna gör det möjligt att skriva information direkt till response kroppen (eng. response body) som sedan klienten kan ta emot. BinaryWrite utför ingen teckenkonvertering, och programmeraren måste själv se till att använda denna metod om applikationerna som används skriver binära värden, så att rätt information når klienten. Det End gör är att fullständigt avsluta processen av ett server-skript som exekverar och skicka innehållet av response

objektets buffert till klienten. Eventuell kod efter detta anrop, ignoreras.

```
<%

  Response.Write (”Hello World”)
  Response.Write(Request.Form(”name”))

%>
```

Det översta exemplet, alltså den översta kodraden, skriver helt enkelt ut `Hello World` direkt på skärmen medan det undre skriver ut ett värde som är beroende av ett formulär.

6.4.3 Server Objekt

Detta är objektet som helt och hållet koncentrerar sig på vad servern utför. Med andra ord har detta objekt inte något med klienten att göra. Allt som objektet ansvarar för utförs på servern [11]. Objekten som arbetar på servern erbjuder funktioner eller metoder som exempelvis att kunna skapa nya objekt, *CreateObject()*. Det som görs då är att det skapas nya objekt-instanser på server- sidan. När ett program anropar *CreateObject()*, har programmet sedan möjlighet att lagra det nyss skapade objektet i en variabel med hjälp av kommandot Set. Den generella syntaxen för *CreateObject()* är:

```
<%

  Set objMyObject = Server.CreateObject(”strProgId”)

%>
```

`objMyObject` är namnet på den variabel som innehåller referensen till det instansierade objektet. `strProgId` representerar ett program- ID för den klass som önskas instanseras ifrån.

Exempel på andra metoder som kan användas på serversidan är *HTMLEncode*, vilket gör det möjligt att uppvisa källkoden på en HTML- sida, och *MapPath* med vilken klienterna kan ta reda på sökvägen till en speciell fil.

6.4.4 Session Objekt

Detta objekt bibehåller temporär information för en specifik användare, dvs lagrar den information som är användarspecifik för enskilda användare under hela tiden som personen använder hemsidans tjänster. Det leder till att användaren slipper påvisa sin identitet varje gång han/hon begär nya åtgärder från servern.

För att allt detta ska fungera, skrivs en s.k. cookie från session- objektet och denna sänds sedan till användarna varje gång de har använt sig av någon ASP- applikation. Läsaren kan tänka sig cookies som information inbakade i små paket, vars information lagras på användarens dator och är av bestående form. [9]

En cookie från en ASP- applikation innehåller en ASP Session ID, vilket är ett unikt värde eller sifferserie. Denna används för att servern ska kunna para ihop en viss användare med en viss speciell instans av Session. Det går till på det viset att så fort en klient utför en begäran på sidan så kommer servern att kunna ta emot den cookie som den först skickade till användaren. [18]

SessionID:et fungerar som en variabel och tilldelas varje enskild användare. SessionsID:et sparas dels i serverns minne och användaren får även informationen genom cookien. För att identifiera en klient, jämförs värdet i cookien med värdet i serverns minne. *SessionID* är en egenskap i Session objektet. Nedanför illustreras ett exempel på hur ett sådant användar- ID kan bestämmas:

```
<%

  Dim UserSessionID = Session.SessionID

%>
```

Dim är helt enkelt ett sätt i VBscript att deklarera variabler och fungerar som i Visual Basic. Dessutom finns möjligheten att med hjälp av Session objektet bestämma en specifik användares behörighets- nivå och för mer avancerade applikationer har användaren också möjlighet att skräddarsy en webbsidas utseende efter dennes val. *CodePage* och *LCID* är exempel på egenskaper som kan tillgodose olika användares önskemål.

Sessionsobjektet är helt beroende av cookies, som först skickas ut till användarna för att vid ett senare tillfälle skickas tillbaka till servern.

6.4.5 Application Objekt

Detta objekt representerar en ASP- applikation och det finns bara ett applikationsobjekt för alla användare. Det innebär att informationen inom detta objekt delas av hemsidan eller applikationens samtliga klienter. Det är egentligen det som skiljer Application objektet från Session objektet. I Session gav ju cookies varje klient värden, specifika just för honom/henne. Annars påminner Application objektet väldigt mycket om Session objektet eftersom båda objekten sparar information [18].

Ett applikationsobjekt blir initierat när den förste klienten av ASP- applikationen begär någon fil från sidan. Detta kan göras med hjälp av filen GLOBAL.ASA. Vi kommer inte att redogöra för denna fil här utan hänvisar till A.K Weissinger bok, *ASP in a nutshell* [22].

En detalj som ASP- programmeraren bör vara medveten om är att all sparad data går förlorad om han/hon startar om datorn eller om webbservern kraschar [11].

Förutom att Session- objektet mer eller mindre är ett användarspecifikt objekt och att Application- objektet är tillgängligt för samtliga användare, fungerar objekten på ett likartat sätt. Och en titt på bilden i bilaga B igen, visar att objekten har flera liknande drag, som exempelvis att båda objekten har händelserna *OnStart* respektive *OnEnd*. Detsamma gäller för kollektionerna. Det som skiljer sig här är objektnamnet.

Vi tar bara upp att detta objekt består av två händelser: *Application_OnStart* och *Application_OnEnd*. OnStart är händelsen som aktiveras när den allra första användaren surfar in på en hemsida som använder ASP. OnStart initierar information som behövs för en specifik applikation. OnEnd är händelsen som bara kan utföras av en administratör eller liknande, där denne explicit tar bort en applikation från serverns minne. Med andra ord, OnEnd "städar" upp serverns minne, genom att händelsen tar bort applikationer som inte längre ska användas. Båda händelserna anropas endast en gång per applikation.

6.4.6 Object Context

Detta objekt används för att kunna interagera med Microsofts Transaction Server (MTS). Genom att använda objektets båda metoder och dess båda händelser finns möjligheten att skapa ett transaktionsskript. Skripten var en viktig del av ASP 2.0 och koden i skriptet skulle antingen lyckas fullständigt som en enda enhet eller också misslyckas som en enhet. För att beskriva det tydligare, ska vi återge ett exempel som Weissinger tog upp i sin bok *ASP in a nutshell* [22].

Tänk er ett skript som består av två sektioner där ena sektionen tar bort en post i en inventarietabell (lagertabell) och där den andra lägger till poster i någon försäljningstabell. Bara om båda sektionerna lyckas, lyckas själva skriptet. Skulle det vara på det viset att bara en av sektionerna lyckades, misslyckas själva skriptet. Ett ganska enkelt exempel som visar att så fort en butik säljer en vara av en speciell produkt, ska lagret automatiskt minska. För vad hade hänt om butiken hade sålt varor utan att lagret uppdaterats? Jo, lagret hade troligtvis tagit slut innan butikens anställda hade hunnit beställa hem nya varor.

Alla transaktioner i ASP-applikationer kontrolleras av MTS. Utan denna support, hade applikationerna själva manuellt fått uppdatera sina databaser. För att visa att skripten ska vara av transaktionstyp måste ASP- programmeraren lägga till denna kodrad:

```
<%@ TRANSACTION = Required %>
```

Dessutom ska denna rad finnas först i skriptet. Objektet har, som vi nämnde, två metoder *SetComplete* och *SetAbort*. SetComplete signalerar om transaktionen slutfördes på ett riktigt sätt och SetAbort används för att avbryta en transaktion. När denna metod anropas, är det ett tecken på att transaktionen slutfördes på ett felaktigt sätt. Personer som programmerar i ASP ska dock vara medvetna om att när SetComplete anropas innebär det inte heller nödvändigtvis att hela transaktionen är genomförd. Transaktionen är fullständig först efter att all kod är genomgången utan anrop till SetAbort och detta gäller om programmeraren inte explicit angett att hans/hennes program har anropat SetComplete.

6.5 ASP kontra ASP.NET

ASP har under en väldigt lång tid varit en både användbar och populär metod att skapa avancerade webbsidor och interaktivitet på Internet, men flera viktiga begränsningar i den klassiska utgåvan har motiverat utvecklingen mot en ny ASP- version, ASP.NET.

W3Schools Online Web Tutorials [15] tar upp några skillnader mellan ASP och ASP.NET, framförallt vilka nya egenskaper ASP.NET är begåvat med. Webbsidan tar upp ett antal punkter bland annat vilka språk som stöds och att fler komponenter och kontroller finns tillgängliga.

I klassiska ASP, kunde webbutvecklaren använda bl.a. skriptspråken VBScript och JavaScript (JScript är Mircosofts svar på JavaScript). I ASP.NET kan programmeraren fortsätta att använda dessa, men en väsentlig skillnad i ASP.NET är att det stödjer hela Visual

Basic (och alltså inte VBScript). Dessutom kan nu de som väljer att skapa sina hemsidor i ASP.NET använda språket som är på frammarsch, nämligen C # (C sharp).

Eftersom ASP.NET är kompatibelt med Visual Basic och eftersom Visual Studio.NET tillhandhåller ett rikt utbud på utvecklingskomponenter mm, kan utvecklarna enkelt skapa snygga och avancerade hemsidor. Detta är en klar fördel för ASP.NET, eftersom Visual Studio.NET är ett komplett programpaket innehållande många program för utvecklande av mjukvaror av olika slag.

Nya kontroller, såsom objektorienterade indatakontroller, exempelvis programmerbara listboxer, och olika operationer på datamängder, exempelvis sortering, finns nu att tillgå i ASP.NET. I och med att ASP.NET är objektorienterat leder det till att koden kan användas mer effektivt samt att det är lättare att underhålla och uppdatera.

Genom att Visual Basic stöds fullt ut i ASP.NET, kan programmeraren nu på ett enkelt sätt ta till vara på de händelser som kan uppkomma i samband med att han/hon använder webbapplikationens olika komponenter. Och apropå ASP.NET's komponenter, är de nästan helt baserade på XML, som vi tidigare berättat lite grann om.

Beträffande användarautentisering, så stöder ASP.NET formulärbaserad sådan, med cookiehantering och automatisk omdirigering av misslyckande logins, om sådana skulle inträffa.

En annan viktig detalj med ASP.NET är att det erbjuder en högre grad av skalbarhet (eng scalability) vilket innebär att programmeraren har möjlighet att dela eller ”skala” en applikation på flera olika servrar. Detta innebär att kommunikationen mellan olika servrar har ökat och blivit stabilare. Dessutom leder detta till att större projekt, som kanske utvecklas i stora grupper, har lättare att samarbeta med varandra eftersom de kan dela upp arbetet på ett bättre sätt nu, när personerna inom arbetsgrupperna kan använda flera filer för en och samma webbplats. Exempelvis är det numera inga problem att separera HTML från ASP- kod i olika filer.

För att öka på prestandan kompileras ASP.NET- koden och detta görs efter första begäran av en ASP.NET- sida. Därefter cachas en kopia in i serverns minne. Tidigare utgåvor av ASP tolkade (eng. interprets) koden rad för rad varje gång en sida anropades, vilket gjorde att en viss ineffektivitet smög sig in. En annan sak som också hindrar prestandan är att variabler i klassiska ASP var löst typade, vilket leder till att variablerna blir bundna till en typ först när koden körs. Dessutom innebär detta att det blir svårare att hitta fel när företagen eller privatpersonerna utvecklar nya program.

Något väldigt användbart i ASP.NET är att det tillhandahålls en debuggningsfunktion, vilket inte fanns i någon av de tidigare versionerna av ASP. Det många användare då gjorde var att använda sig av Responseobjektet och metoden Write, som vi berättade om i förra avsnittet, för att spåra hur exekveringen gick. Men i ASP.NET kan semantiska fel enkelt spåras upp med hjälp av debuggningsfunktionen.

Ytterligare en fördel med ASP.NET är att servern inte behöver startas om när en omkonfiguering av en applikation har gjorts. Detta kan dessutom göras medan applikationen i fråga är i gång och exekverar. Allt om konfigurationen lagras i XML- filer, och påvisar ännu en gång att ASP.NET är ytterst XML- baserat.

En liten nackdel dock skulle väl vara att den klassiska ASP- koden inte är helt kompatibel med ASP.NET, och det innebär att applikationer skrivna i en tidigare version av ASP måste justeras för att fungera under den nya plattformen.

De applikationer som skapas i ASP.NET har filnamnet *.aspx*, och inte *.asp* (som program skapade i traditionella ASP har) vilket gör det möjligt att använda program från både ASP och ASP.NET på en och samma server.

6.6 Sammanfattning ASP

ASP, som detta kapitel har handlat om, är och har varit ett väldigt vanligt sätt att få sina webbsidor att bli dynamiska. I och med att Microsoft lanserade ASP.NET kan risken vara stor att det klassiska ASP mer och mer kommer att ersättas av just ASP.NET. I ASP.NET har utvecklarna på Microsoft rättat till och framförallt tillhandahållit nya funktioner som gör det enkelt för webbutvecklarna att göra attraktiva webbplatser. Som vi nämnde i inledningen, finns ett kodexempel i bilaga D som beräknade en triangels area. I nästa kapitel kommer vi att rikta blickarna mot ett annat skriptspråk, som också gör hemsidor dynamiska, nämligen PHP.

7 PHP

Det föregående kapitlet handlade om ASP. Som då kunde ses använde sig ASP av ett speciellt skriptspråk för att skapa dynamiska hemsidor med. Exempel på skriptspråk är JavaScript och VBScript, och dessa nämndes kort i kapitel 5 i stycket 5.4, skriptspråk. Detta kapitel kommer att handla om PHP och hur hemsideutvecklarna kan använda detta språk för att generera hemsidor av dynamisk karaktär. PHP fungerar ungefär på samma sätt som ASP eftersom även detta skriptspråk tillämpar server-side skriptning, men skillnaden är att PHP är sitt eget skriptspråk, dvs. det är inbyggt i språket. Till detta kapitel finns också ett kodexempel och detta återfinns i bilaga C. Exemplet är ett PHP-skript som krypterar ett meddelande.

I detta kapitel kommer vi att ta upp historia, vad PHP är och hur PHP kan användas. Kapitlet kommer i mångt och mycket att följa upplägget från föregående kapitel, det om ASP. Varför vi har valt att lägga upp kapitlen om ASP och PHP på ett så sammanfallande vis, är för att läsaren själv ska kunna dra vissa paralleller och slutsatser. Utifrån dessa ska läsaren sedan kunna bilda sig en egen uppfattning om de båda skriptspråken. Detta är en viktig del i uppsatsen, då kapitel 8 sedan kommer att bestå av skillnader men även de likheter som vi stött på under resans gång, mellan de båda skriptspråken. Ett avsnitt i detta kapitel kommer att ta upp PHP’s syntax lite mer ingående. Ett sådant avsnitt fanns inte i ASP. Tanken är att ge Hög-Data AB en mycket kort introduktion till hur program utvecklade i PHP kan se ut, rent syntaxmässigt. Vi inleder detta kapitel som vi gjorde i ASP, nämligen med lite punkter om PHP.

- Öppen källkod dvs. gratis att hämta hem.
- Med PHP finns möjligheten att skapa avancerade webbsidor dynamiskt.
- Har ett bra stöd för många databaser, ex MySQL.
- Ett stort förråd av färdiga funktioner.
- Förhållandevis lätt att lära sig, speciellt om kunskaper i C/C++ finns
- Plattformsoberoende, dvs. fungerar i alla operativsystem.
- Språket är svagt "typat", dvs. typomvandling är möjlig.

7.1 Historia

Språk som hade använts inom webbutveckling innan den första versionen av PHP kom ut var t.ex. Perl och ASP. Dessa språk används även idag men antalet hemsidor som är programmerade i PHP har ökat närmast explosionsartat. Informationen till detta stycke om PHP's historia har vi hämtat ur källorna [3], [7], [10], [12] och [13] .

Den första versionen av PHP kom ut 1994 och skrevs av Rasmus Lerdorf. Syftet var att han ville se vilka som var inne och tittade på hans CV på Internet. Det började med att han skrev ihop några Perl- skript som sedan växte till en samling som senare kallades för ”Personal Home Page”. Detta var den första benämningen till förkortningen PHP. ”Personal Home Page” döptes efter en tid om till PHP: Hypertext Preprocessor. Det var bl.a. i den första version som webbutvecklarna kunde använda sig av gästböcker och räknare, som vid den här tiden blev väldigt populära då dessa visar hur många besökare en viss hemsida hade haft.

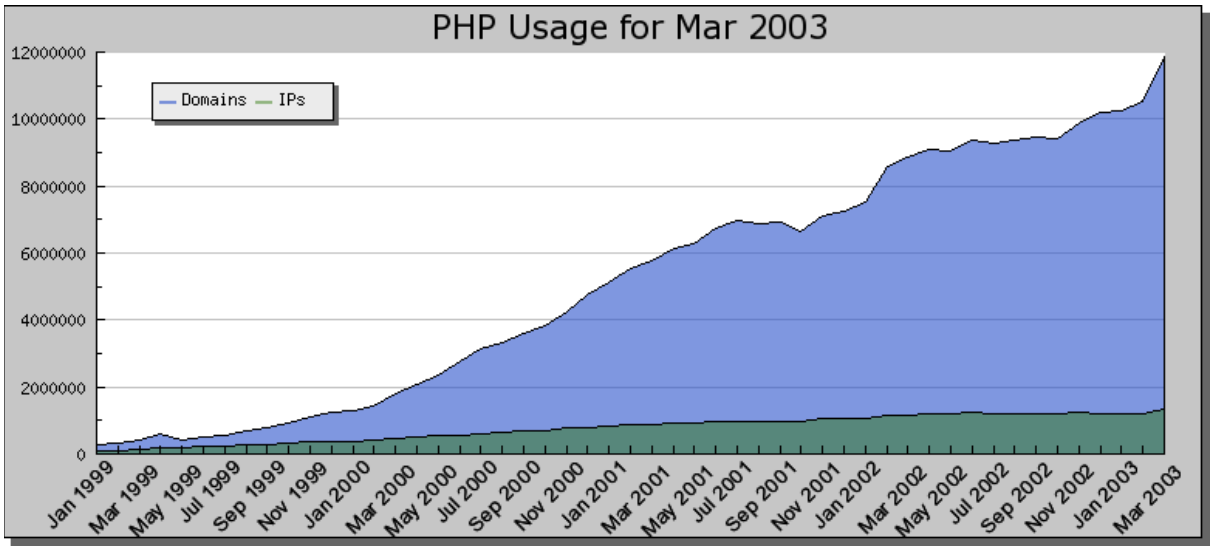
I takt med att intresset ökade för PHP så skrevs 1995 en så kallad skriptmotor, som fick namnet Zend. Zend kommer att beskrivas utförligare vid ett senare skede i detta kapitel. Två av de stora orsakerna till detta intresse för PHP var delvis enkelheten med att använda PHP som språk samt nyfikenheten att prova hur det fungerade. Efter det att skriptmotorn Zend konstruerades dröjde det inte länge innan utvecklarna bakom PHP byggde in stöd för att hantera indata från HTML-formulär, en så kallad Form Interpreter (FI). En översättning till det svenska språket skulle bli ”formulärtolk”. Det var efter att en sammanförning av PHP och FI som resultatet blev PHP/FI. Denna andra version, som också kom att kallas PHP2, var mer kraftfull än föregångaren och förutom att den kunde tolka HTML-formulär så var en viss databasinteraktivitet inbyggd.

Intresset ökade kraftigt och kort därefter utvecklades PHP3. Zeev Suraski och Andi Gutmans var två av utvecklarna bakom PHP3 som tillsammans skapade en ny parser till PHP. En förenklad förklaring av vad en parser är så är det ett program som går igenom källkod och konverterar detta till objektkod. När PHP3 kom hade språket utvecklats rejält till ett mer professionellt verktyg för att skapa dynamiska webbplaster. Mycket nyheter i PHP3 satte en stor prägel till att PHP överhuvudtaget blev en stor hit för många webbutvecklare.

När PHP4 kom ut på marknaden år 2000 så fanns det bl.a. stöd för sessionshantering.
PHP4 är för närvarande den versionen som används även om PHP5 kommer att släppas till sommaren. PHP version 5 släpps då i sommar som vi tidigare nämnt och då kommer Zend att

uppdateras till version 2. Den största skillnaden mellan PHP4 och PHP5 är att PHP5 kommer att stödja objektorientering (OO) fullt ut till skillnad från den nurvarande versionen av PHP som är objektbaserat. Vidare kommer PHP5 även att ha en automatisk minneshantering tillgänglig. Den är betydligt snabbare än sin föregångare. (Här är källan Viktor Jonsson, personligen)

Bilden nedan visar hur utvecklingen för PHP fortlöpt. Bilden visar bland annat hur många olika domäner som utnyttjar PHP. I mars 2003 var antalet ca 12 miljoner [20].



Figur 7.1 PHP's utveckling genom åren

7.2 Vad är PHP?

Ursprungsversionen av PHP är som tidigare sagt skapat av Rasmus Lerdorf 1994. PHP är ett så kallat skriptspråk som tolkas på serversidan. Detta kallas, som vi kunde läsa i kapitel 5, för ett server-side scripting språk. Vanliga programmeringsspråk kompileras (konverteras) till maskinkod innan körning men skriptspråk tolkas direkt vid körning. Informationen till detta avsnitt är främst funnet ur källorna [3], [7], [10] och [12].

All källkod till PHP är öppen, vilket betyder att koden distribueras fritt på Internet och är därmed fri för alla privatpersoner och företag att ladda ner och använda. I och med att all källkod är gratis och att webbutvecklarna i sin tur slipper oroa sig om diverse licenser har detta inneburit att Apache- servern, som är den vanligast förekommande webbservern inom PHP tekniken, har blivit väldigt populär. Fördelen med att PHP distribueras som öppen

källkod är att uppdateringar kan göras av vem som helst som kan PHP och att alla har tillgång till dessa uppdateringar. På så sätt så kommer det ut nya uppdateringar väldigt ofta, och dessa är oftast väldigt nyttiga.

PHP och HTML skrivs ofta i samma fil vilket betyder att PHP- koden kan skrivas tillsammans med HTML koden. Detta förfarande liknar alltså ASP, där ASP- kod också blandades med HTML- kod i ett och samma dokument. Det enda som programmeraren måste ta hänsyn till när han/hon skriver PHP koden är att all PHP- kod måste finnas inuti specifika PHP taggar:

```
<php?
        …
?>
```

I de nyare versionerna av PHP, räcker det med att ange taggarna :

```
<?
        …
?>
```

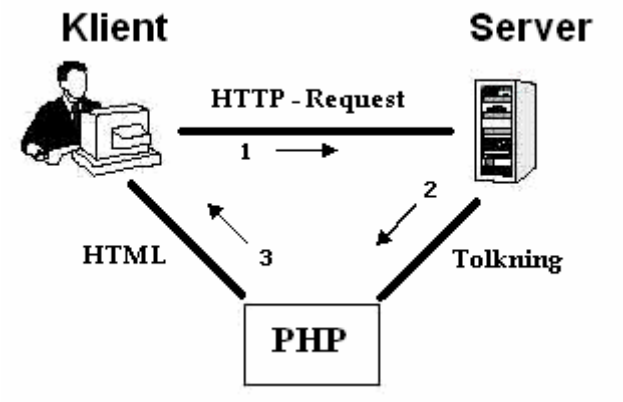
Detta förfaringssätt, att ange att skripten kommer att finnas inom dessa taggar, liknar också ASP. Den enda skillnaden är att ASP's taggar ser lite annorlunda ut, se kapitel 6.2.

PHP är relativt lätt att lära sig p.g.a. att det påminner mycket om programspråken Perl och C. PHP liknar även ASP, men det finns vissa skillnader som vi kommer att ta upp längre fram i nästkommande kapitel, nämligen kapitel 8. PHP är också svagt "typat", vilket innebär att variabler av en typ, utan någon typomvandling, enkelt kan anta en annan typ. Vi kommer att upp detta mer i detalj i ett senare avsnitt i detta kapitel.

7.3 Hur fungerar PHP?

Eftersom PHP liknar ASP och tillika tillämpar server-side scripting, alltså att alla skript körs och exekveras på servern, påminner PHP om ASP och dess arbetssätt. Som vi tidigare sagt i kapitel 5, innebär server-side scripting att alla skript är "dolda" för Internetanvändarna. Det enda som klienterna har tillgång till vid sina datorer är de värden som genererats av skripten. Informationen till detta avsnitt om hur PHP fungerar, har vi funnit i [3] och [10].

Om omfattande studier genomförs i PHP's arbetssätt i jämförelse ASP's arbetssätt, skulle säkerligen stora detaljer upptäckas. Huvudidén är i alla fall lika: skripten exekveras på servern som sedan genererar en ny HTML-sida innehållande de värden som erhölls vid exekveringen. Läsaren känner därför igen bilden nedanför från ASP- kapitlet, med den skillnaden att webbsidor skapade med PHP, ska ligga på en server som hanterar PHP- kod.

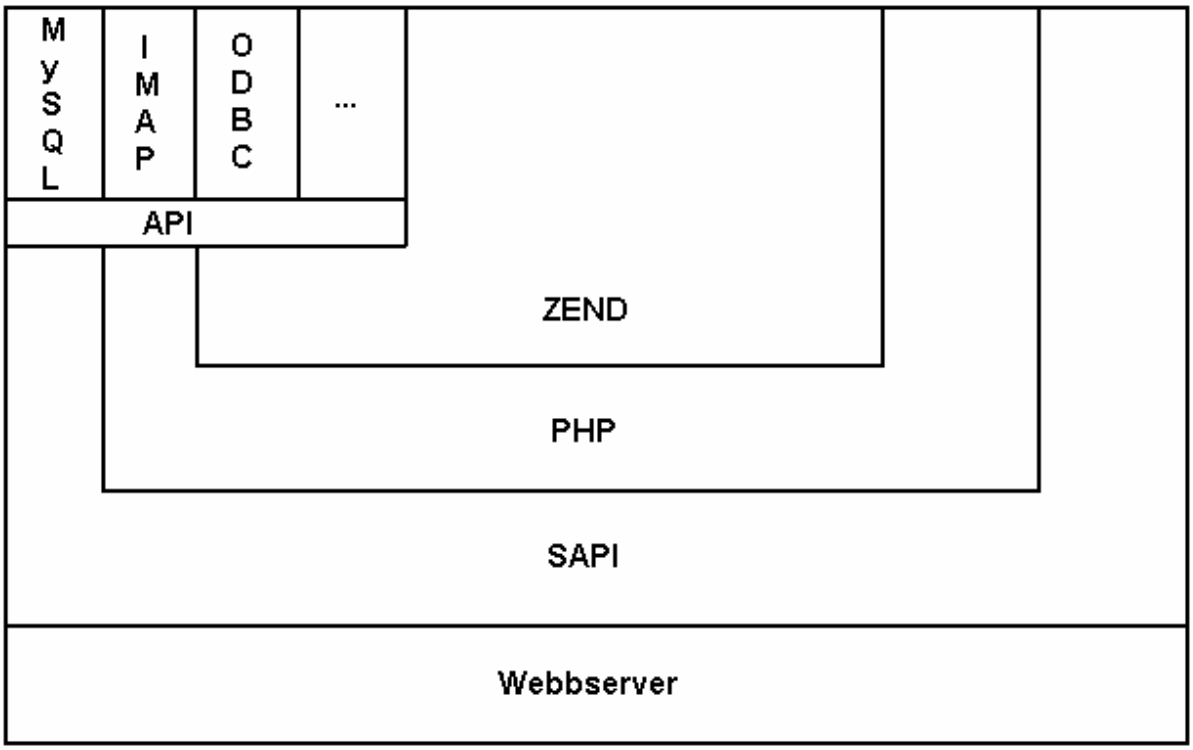


Figur 7.2 Hämtning av webbsida som är av php- typ

Så, sammanfattningsvis säger vi att PHP- tolken, eller parsern som många litteraturer föredrar att säga, tolkar eller parsar all PHP- kod i en fil med filändelsen *.php*. Om det skulle finnas någon HTML- kod i denna fil, ignorerar parsern/tolken detta och koncentrerar sig helt på det som finns inom taggarna `<php? ... ?>`. Därefter exekveras koden inom taggarna för att sedan ersättas med resultatet av koden. Det är sedan dessa värden samt HTML- koden som slutligen sänds tillbaka till användaren.

7.4 PHP's arkitektur

PHP består av en rad olika komponenter som tillsammans bildar en helhet. Vi har för avsikt är att först ge en generell bild av hela arkitekturen för att därefter behandla de olika beståndsdelarna och visa hur de hör ihop. I detta avsnitt kommer vi endast att behandla en källa, nämligen boken *Webbprogrammering med PHP* av Viktor Jonsson [12]. Figuren nedan visar PHP's arkitektur.



Figur 7.3 PHPs Arkitektur

Figuren är hämtad ur Viktor Jonssons bok [12]. Tänk dig att användaren finns ovanför bilden. Kommunikationen sker från toppen nedåt i bilden, det vill säga ned mot det understa lagret, det lagret som är märkt *Webbserver*. Detta innebär att användaren kan kommunicera med alla lager förutom Webbservern.

PHP har stöd för många *externa komponenter*, och några av dem finner läsaren längst upp till vänster. De som i bilden är angivna är MySQL, IMAP och ODBC (Open Database Connectivity). Punkterna anger att det finns fler. Dessa externa komponenter kommunicerar med PHP genom ett så kallat *tillägg*, ett API (Application Programming Interface). Ett API är ett programmeringsgränssnitt mot ett program eller bibliotek. Bilden visar två stycken tillägg. Dels API, som vi just nämnde, men också SAPI som är PHPs gränssnitt mot webbservern. API bildar då gränssnittet mellan de externa komponenterna och de övriga komponenterna. I bilden kan läsaren även se, att ovanför SAPI- lagret finns PHP- lagret. Detta är själva kärnan och den binder samman de olika komponenterna, förutom Webbservern. I tidigare versioner var själva skriptmotorn, numera kallad Zend, en del av PHP- kärnan, men från och med de senare versionerna av PHP är den alltså en egen komponent.

7.4.1 Skriptmotorn Zend

Skriptmotorn har funnits så länge som PHP har existerat men det var först i PHP version 3, mer känd som PHP3, som Zend utvecklades till självständig skriptmotor och bildade en egen komponent. Den skriptmotor som fanns ***i PHP3*** innan Zend var väldigt bra på att exekvera små webbapplikationer, alltså skript som endast bestod av ett fåtal rader kod. Men eftersom exekveringen utfördes allteftersom koden lästes, medförde detta att vid mer komplexa webb-applikationer och kanske med återkommande loopar så blev exekveringen för långsam.
Av denna anledning designades den dåvarande skriptmotorn om ännu en gång av utvecklarna Zeev Suraski och Andi Gutmans, som vi tidigare nämnt var två av upphovsmakarna bakom PHP3. Resultatet blev Zend som för närvarande är den skriptmotor som PHP utnyttjar. Här följer några fördelar med Zend jämfört med tidigare skriptmotorer.

- Den är 5 till 200 gånger snabbare
- Den är modulariserad, det vill säga uppdelad i klart distinkta delar
- Den är nästan helt kompatibel med PHP3
- Den är utbyggbar med olika plug-ins
- Den är oberoende av PHP

Zend går enkelt att bygga ut och den kan också användas i andra applikationer pga. dess oberoende design. Det var också därför som utvecklarna bakom Zend ville döpa den till något speciellt. Själva namnet Zend tillkom genom en ordlek med namnen på de båda utvecklarna Zeev och Andi. Senare grundade även upphovsmännen bakom Zend tillsammans med några andra personer ett företag, *Zend Technologies Ltd*, som för övrigt är världsledande inom PHP-utveckling än idag.

Tidigare pratade vi om varför skriptmotorn i PHP3 gjordes om till Zend. Vi nämnde då att det var på grund av att exekveringen gick för långsamt. Det som är den stora skillnaden mellan ***den tidigare*** skriptmotorn i PHP3 och Zend är att exekveringssättet att *läsa raderna samtidigt* som exekvering av dem, sker först efter att den har *läst alla rader* i den nya skriptmotorn Zend. Nu sker alltså en kompilering innan själva exekveringen görs och på så sätt går exekveringen snabbare. Detta är inte den enda fördelen som finns utan även andra fördelar, som till exempel att funktionsanrop inte behöver komma efter själva funktionen finns. Detta möjliggörs genom att Zend redan har läst koden en gång så den vet vad som kommer längre ner vilket inte är en allt för vanlig egenskap hos många andra skriptmotorer.

I punktlistan ovan, som behandlade fördelar med Zend gentemot andra skriptmotorer står det att Zend är utbyggbart med olika plug-ins. Ett exempel på en sådan plug-in är *Zend Optimizer*, som optimerar koden och gör exekveringen 40 till 100 procent snabbare. Det är bland annat detta som gör PHP till det absolut snabbaste HTML-inbäddade skriptspråket.

7.4.2 PHP-tillägg

PHP är ett mycket populärt skriptspråk och det beror mycket på att PHP har stöd för ett flertal vanliga program och programmeringsbibliotek. Det är till exempel väldigt enkelt att koppla databaser till PHP, för att sedan kommunicera med dessa genom olika funktioner som finns inbyggda. Det krävs inte någon speciell kunskap för att använda dessa funktioner eftersom de är så skickligt utformade. Men självklart ingår det även en hel del funktioner i PHP utan att några tilläggsprogram behövs, och exempel på en sådan funktion är hanteringen av XML-dokument. Det finns olika tillägg, även kallade PHP- tillägg, som bestämmer vad PHP skall ha stöd för. De PHP-tillägg som skall vara med från början bestäms oftast vid kompileringen av PHP. Dessa PHP-tillägg länkas då till PHP så att en omkonfiguration av systemet inte är nödvändigt.

Det vanligaste sättet att implementera PHP på, är att kompilera och installera en egen version av PHP. Men det finns även ett annat sätt och det är att skapa så kallade dynamiskt inladdningsbara PHP-tillägg, vilka kan laddas in i PHP vid behov.

7.4.3 Webbserverabstraktionslagret SAPI

SAPI (Server Application Programming Interface) är, som vi tidigare nämnt, ett gränssnitt mellan PHP och webbservern. För att använda PHP över Internet måste en webbserver finnas tillgänglig. Denna webbserver kan sedan enkelt kopplas till PHP. SAPI har stöd för många webbservrar bl.a. Apache, Roxen, IIS, AOL server och vissa Netscape- servrar. Vad som menas med detta är då att PHP kan bli en modul, en enhet tillsammans med webbservern istället för en ensam enhet och behöva köras som ett enskilt program. Det är detta som gör exekveringen mer effektiv.

PHP är så kallat plattformsoberoende och detta innebär att det kan användas på de flesta operativsystem med de vanligaste webbservrarna. Detta möjliggjordes då av SAPI som infördes i PHP4.

Den vanligaste installationen av PHP är att köra PHP på en dator med operativsystemet Linux, med webbservern Apache och databasen MySQL. Denna lösning brukar förkortas

LAMP, vilket är en förkortning för Linux, Apache, MySQL, PHP. *LAMP* går också, i vissa fall, under det mer kända namnet Red Hat Linux.

7.5 Introduktion till PHP's syntax

Eftersom Hög-Data huvudsakligen var intresserade av en jämförelse mellan ASP och PHP, och där kunskaperna inom området PHP hos de anställa var ganska liten, beslutade vi oss för att i detta delkapitel ge en kort presentation av främst PHP's syntax. Vi kommer att behandla vilka PHP's olika datatyper är, hur språkets funktioner kan användas, hur variabler deklareras mm. Som läsaren snart kommer att upptäcka finns stora likheter med programmeringsspråket C/C++. Detta betyder att de som har kunskaper inom detta välkända programspråk egentligen utan problem kan sätta sig ned och skapa dynamiska hemsidor i PHP. All information för detta avsnitt har hämtats ur [12], om inget annat angetts.

7.5.1 Variabler

Förutom Viktor Jonssons bok "Webbprogrammering med PHP" [12] , kommer detta avsnitt som behandlar variabler i PHP även använda sig av källan [7].

Variabler i PHP skrivs med ett dollartecken ($) före själva variabelnamnet, d.v.s. det inleds alltid med $, t ex `$variabel`. Då erhålls alltså en variabel som heter "`$variabel`". De tecken som är tillåtna i variabelnamn är, **a-ö, A-Ö, _** (underscore) och **0-9**. Siffrorna däremot, får inte inleda ett variabelnamn. Observera att använda de svenska bokstäverna å, ä och ö i variabler inte är att föredra pga. att en del versioner inte stöder svensk stavning i variabelnamn. I PHP är variablerna *case sensitive*, vilket betyder att PHP gör skillnad på stora och små bokstäver, exempelvis så skulle `$Variabel`, `$VARIABEL` och `$variabel` vara tre olika variabler.

När en programmerare utvecklar program är det viktigt för att andra personer, som eventuellt skall läsa den kod som är skriven, att förstå vad funktioner och variabler avspeglar. Så med andra ord kan det sägas att variablerna skall spegla dess innehållet.

Variabler behöver inte deklareras med någon särskild datatyp, vilka vi beskriver i nästkommande avsnitt, eftersom variabeln ges en datatyp när den tilldelas ett värde. Så, om exempelvis variabeln `$tal` får värdet 6 blir variabeln av datatypen heltal (eng. integer).

7.5.2 Datatyper

Som läsaren kunde se i föregående avsnitt, nämnde vi att variabler inte behöver deklareras till en speciell typ eftersom de tilldelas en datatyp när den ges ett värde. I andra språk,

exempelvis C och C++ så måste variabeln tilldelas en specifik datatyp innan denna variabel kan anta ett värde, men det behövs alltså inte i PHP. Därmed är variabelns datatyp beroende av vilket värde som ges variabeln.

Det finns åtta stycken datatyper i PHP. Av dessa är tre stycken grunddatatyper och de andra fem är speciella datatyper. Nedanför finns en lista på de datatyper som finns. De 3 första är grunddatatyperna *Integer*, *Double* och *String*. De övriga fem utgör de speciella datatyperna.

- **Integer** - Heltal
- **Double** - Flyttal
- **String** - Sträng av tecken
- **Objekt** - Lagra instanser av klasser
- **Array** – Vektor av värden (tal)
- **Bool** - Sant eller Falskt
- **Null** - Strängslut, variabeln är tom
- **Resource** - innehåller en identifierare till en extern resurs

Datatypen *String* används för att representera icke numeriska värden, så som bokstäver och andra tecken. String, som betyder *sträng* på svenska, används oftast i samband med teckensträng. Strängar är ett vanligt alternativ för att lagra namn eller hela meningar.
Studera följande exempel:

```
$string = ”4003”;          // Lagras som tecken, det vill säga en sträng
$integer = 4003;           // Lagras som ett tal, här som ett heltal
$double =  4003.1;         // Lagras som ett tal, här som ett flyttal (decimaltal)
```

För att skriva kommentarer i PHP, används dubbla slashtecken ”`//`”, som exemplet visar. Detta kan bara användas radvis och alternativet är ”`/*  …  */`”, där kommentaren skrivs istället för punkterna. Exemplets egentliga mening var att visa skillnaden mellan hur en variabel blir deklarerad som en sträng och hur den blir deklarerad som ett tal. Som kommentaren till variabeln `$string` säger, används citationstecken för att skapa en sträng. Variabeln `$integer` bildar ett heltal (eng. integer). Den sista variabeln `$double` är av

typen *double*, eftersom talet på vänster sida om likhetstecknet är specificerat till ett decimaltal. Här nedanför följer ett exempel på hur strängar kan användas för att lagra namn.

```
$namn = martin;            // fungerar
$Namn = martin meijer;     // fungerar ej
$NAMN = ”martin meijer”;   // fungerar
```

Den första och den sista av de tre raderna fungerar. Men det gör inte den andra, eftersom PHP måste ha citationstecken före och efter strängen, om den innehåller mellanslag. Datatypen *Array* används för att lagra flera värden. En array kan därför sägas vara en variabel som består av flera variabler. Ett vanligt svenskt uttryck för array är vektor.

```
$array[0] = ”Martin”;
$array[1] = 15;
$array[2] = ”Meijer”;
```

I detta exempel skapas en array, som lagrar tre element. På första platsen finns namnet ”Martin”, andra platsen innehas av heltalet 15 för att sedan avslutas med ”Meijer” på tredje platsen. Indexsiffrorna inom hakparenteserna hade för övrigt kunnat utelämnas, eftersom uppräkningen sker automatiskt. Det kan tyckas konstigt att det skulle gå att initiera en array som faktiskt kan lagra värden av olika datatyper men det är faktiskt möjligt i PHP, till skillnad från exempelvis C/C++.

```
$array = array(0 => ”Martin”, 15, ”Meijer”);
```

Ovanstående rad gör exakt samma sak som det tidigare exemplet. Operatorn ”=>” samt nollan innebär att arrayens startvärde ligger på plats nummer noll. Därefter räknas platsnumret upp med ett för varje värde som har angetts. Datatypen *Objekt* är den mest komplicerade datatypen i PHP. Detta på grund av att den kan innehålla någon av de andra datatyperna. Objekt kan även innehålla funktioner, och i association med objekt kallas funktionerna för *metoder*.

För de läsare som har kunskaper inom något objektorienterat språk, exempelvis Java, brukar det heta att ett objekt skapas ur en klass. Klassen fungerar då som en mall för det skapade objektet. Objektet kommer då att vara utrustat med alla de metoder som klassen eller

mallen har. I denna uppsatsen kommer vi inte att redogöra för hur det går till för att skapa en klass, utan nöjer oss med att det görs med nyckelordet `class`. Viktor Jonsson har i sin bok ”Webbprogrammering i PHP” [12] ägnat ett helt kapitel om objekt och klasser, och därför hänvisas intresserade läsare dit för ytterligare information. Det är för övrigt klasserna i PHP som gör språket till ett objekbaserat språk. En av de största fördelarna med att ett språk är objektorienterat är att det ges tillfälle att återanvända kod. Detta leder till att utvecklingstiden för webbapplikationer kan minska.

En variabel av datatypen `bool` kan bara anta värdena 0 och 1. För att lagra ett booleskt värde, används nyckelorden `TRUE` och `FALSE`. `TRUE` representerar 1 och `FALSE` 0. Nedanstående exempel sätter variabeln `$test` till att vara sann.

```
$test = TRUE;
```

`Null` är datatypen som används för att påvisa att en variabel inte blivit tilldelad något värde, se exemplet under:

```
$Test = Null;
```

Variabeln `$Test` är således en variabel som är tom. Både `bool` och `null` används främst i jämförelseoperationer. *Resource* är datatypen som används i samband med de s.k. externa komponenterna, som kunde läsas i avsnittet som behandlade PHP’s arkitektur. Även här refererar vi till Viktor Jonssons bok för ytterligare fakta.

7.5.3 Funktioner

PHP har över 2000 inbyggda, fördefinierade funktioner, som hanterar allt från databaser till bildmanipulering. Dessa inbyggda funktioner är inte programmerade i PHP utan ursprungligen i C. Funktionerna är konstruerade för att lösa deras uppgift på ett effektivt sätt. Därför ska utvecklare i PHP alltid först kontrollera om en funktion finns fördefinierad i språket, innan han/hon sätter sig ner och konstruerar en egen. Förutom att det kan vara komplicerat att skapa egna funktioner, så kan detta också leda till att funktionerna inte är optimerade som de redan inbyggda tillämpningarna är. Programmering i PHP handlar till stor del om att kunna lära sig hantera dessa inbyggda funktioner, veta deras kapacitet samt att känna till hur de på bästa möjliga sätt kan kombineras med funktionerna och algoritmerna som är egenutvecklade. Det optimala är att med hjälp av de inbyggda funktionerna förena

dem så att PHP- programmeraren till slut får den algoritm som eftersträvas. Skulle detta innebära problem, finns möjligheten att skapa egna funktioner med PHP, precis som i C/C++. Senare i detta avsnitt kommer vi att kunna se hur en funktions syntax ser ut.

Som tidigare nämndes i kapitlet var API en förkortning av Application Programming Interface och fungerade som ett gränssnitt mot ett program eller bibliotek. Flera av de inbyggda funktionerna i PHP ingår i de olika API:erna. Det innebär att det är enkelt att hantera den funktionalitet som programmet eller biblioteket tillhandahåller när PHP-utvecklare använder de funktioner som API:et tillhandahåller. Den generella syntaxstrukturen för en egendefinierad funktion ser ut så här:

```
function funktionsnamn(parameter_1, … , parameter_n)
{
        funktionskropp
        return värde;
}
```

De fetstilta orden, **`function`** och **`return`**, är så kallade nyckelord. En användardefinierad funktion inleds alltid med det reserverade ordet **`function`**. Därefter följer funktionens namn, och likt variabelnamnen, som vi talade om i ett tidigare skede i detta kapitel, bör dessa också namnges efter vad de är menade att göra. Inom parenteserna tar funktionen emot noll eller flera parametrar. En parameter är oftast en variabel. För att sedan deklarera funktionens *scope* används tecknet ”`{`” för att inleda funktionen och ”`}`” avslutar den samme. Ett scope anger alltså funktionens livslängd eller räckvidd. Inuti ”`{}`” finns själva funktionen och här utförs allt det som funktionen är ämnad till att genomföra. Det sista som händer i en funktion innan den avslutas är att den vanligvis skall returnera ett värde, ett värde som blivit skapat i funktionen. Observera att funktioner inte nödvändigtvis behöver returnera något värde.

Den stora vitsen med funktioner är att programmen blir uppdelade i olika programdelar, även kallade moduler. De läsare som har kunskaper från bland annat C/C++ ser tydliga paralleller med PHP- syntaxen ovan. Det enda som skiljer är att i C/C++ måste funktionerna deklarera vilken returtyp som ska användas för varje specifik funktion och att parametrarnas datatyp inom parenteserna måste anges. I praktiken innebär detta att nyckelordet **`function`**, som för övrigt skall användas vid alla funktioner i PHP, ersätts med ett annat nyckelord i

C/C++. Detta nyckelord ska sedan stämma överens med det värde som returneras från funktionen. I PHP behöver alltså inte funktionerna i förväg veta vilken typ som ska returneras.

Variabler deklarerade inuti en funktion sägs vara *lokala*. Med detta menas att dessa variabler inte finns tillgängliga utanför funktionens scope. Livslängden eller räckvidden för de lokala variablerna är till dess att den funktion de skapades i inte längre körs. Den andra typen av variabler är de *globala*, vilka skall deklareras utanför en funktion. Dessa globala variabler kan accessas i hela programmet och de har därmed en livslängd som löper genom hela webbapplikationens körning.

De inbyggda funktionerna fungerar på samma sätt som de användardefinierade när det gäller att anropa dem. Ett enkelt men samtidigt tydligt exempel följer:

```
mail(”nisse@manpower.se”, ”ämne”, ”Här följer själva meddelandet...”);
```

Som kan ses ur ovanstående exempel behandlas funktionen `mail()`, och denna används i PHP för att skicka ett e- mail. Som säkert kan urskiljas ur exemplet ovan är funktionens första argument e- mailadressen dit meddelandet ska. Andra argumentet är ämnet, det vill säga det mottagaren ser när han/hon loggar in på sin mail. Det sista argumentet är själva meddelandet.

Alla fördefinierade funktioner i PHP är redan inkluderade i språket, vilket innebär att det bara är att anropa dem vid behov. Det finns bra dokumentation om alla inbyggda funktioner, bland annat på hemsidan www.php.net [20]. Dokumentationen är koncentrerad på att beskriva funktionernas anropssyntax samt vilket returvärde som eventuellt skickas tillbaka. Som vi tidigare nämnde har PHP över 2000 inbyggda funktioner och `mail()` var bara en av dem. Vi kommer inte att behandla några ytterligare här, men alla funktioner finns samlade på www.php.net [20].

7.5.4 Operatorer

Operatorer används vid bearbetning av data och är något av det mest elementära i ett program. Operatorer i PHP fungerar på ungefär samma sätt som i andra högnivåspråk. Det innebär att personer med kunskaper från språken C/C++ och Java inte kommer att stöta på några nyheter. En operator används i en operation för att manipulera data och på så vis erhålla nya värden. Ett enkelt exempel på en operation skulle kunna vara `1 + 2`. +- tecknet är operatorn i operationen och talen `1` och `2` är operander. Operander motsvaras alltid av ett värde även då de inte anges i klartext. Med detta menar vi att variabler också skulle kunna fungera som operander.

Operatorerna är indelade i olika kategorier, och till de vanligaste operatorerna hör de aritmetiska. Till denna grupp av operatorer tillhör + (addition), - (subtraktion), * (multiplikation), / (division) och % (restdivision). De aritmetiska operatorerna kallas också för de matematiska eftersom de till 80 % utgör de fyra räknesätten. Den sista femtedelen är restdivisionen och även denna är vanlig inom matematiken. En annan grupp av operatorer är jämförelseoperatorerna och alla dessa är samlade i tabellen under.

Operator	Betydelse	Exempel
==	Lika med	$a == $b
<	Mindre än	$a < $b
>	Större än	$a > $b
<=	Mindre än eller lika med	$a <= $b
>=	Större än eller lika med	$a >= $b
!=	Inte lika med	$a != $b
<>	Inte lika med	$a <> $b

Tabell 7.1 PHP's jämförelseoperatorer

Jämförelseoperationerna används till att jämföra två operander. Resultatet värderas till en boolesk betydelse, antingen sant eller falskt. Ett krav är dock att operanderna är av samma datatyp. Återigen, läsare som är bekanta med C/C++ känner igen alla operatorer i Tabell 7.1. En tredje grupp av operatorer är de logiska operatorena och dessa används för att kombinera flera villkor i ett uttryck. Till de logiska operanderna hör && (logiskt och), || (logiskt eller), ! (logiskt icke) och xor som betyder exklusiv eller. Istället för &&, || och ! tillåts programmeraren att skriva and, or respektive not. Studera följande exempel:

```
$a = 5;
$b = 10;
/* ex 1 */ print (($a == 5) && ($b < 8)); // falskt
/* ex 2 */ print (($a == 5) || ($b == 0)); // sant
/* ex 3 */ print (($a == 5) xor ($b > 8)); // falskt
/* ex 4 */ print !($a > 15) // sant
```

Enligt exempel 1, ska hela uttrycket vara sant om $a är lika med 5 och att $b är mindre än 8. Men $b är ju initierad till 10 och är alltså större än 8. Printsatsen är felaktig ty båda operationerna måste vara sanna för att uttrycket ska bli sant och följden blir att inget skrivs ut. Exempel 2 är sant, eftersom $a är lika med 5. $b är visserligen inte lika med 0, men eftersom

printsatsen i exempel 2 innehåller ett logiskt eller istället för ett logiskt och, blir hela uttrycket sant. Detta innebär att en 1:a skrivs ut i webbläsaren. I exempel tre, är båda delsatserna sanna, ty `$a` är lika med 5 och `$b` är större än 8. Men eftersom ett exklusivt eller finns mellan satserna, blir hela uttrycket falskt. Ett uttryck som är exklusivt eller blir endast sant om någon av satserna är sanna. Det sista exemplet är sant, eftersom `$a` är mindre än 15.

Den absolut vanligaste operatorn är dock tilldelningsoperatorn `=`. Den används exempelvis alltid vid initieringar av variabler Vi ska inte behandla fler operatorer i detta avsnitt, men i nästa avsnitt, det som handlar om villkor, kommer vi att återknyta till vissa av dessa operatorer igen.

7.5.5 Villkorssatser

Vanliga villkorssatser i PHP, likt C/C++, är `if-` satserna. Med sådana satser ger programmeraren programmet möjligheten att bli vägledd rätt. Vad vi menar med detta är att programmet ska utföra något speciellt beroende på vad en variabel har för värde. Det är bland annat här jämförelseoperatorerna kommer in i bilden. De används nämligen i samband med villkorssatserna för att ett program ska fatta olika beslut. Studera följande exempel:

```
if ($a == 2)
        print ” Variabeln är lika med 2”;
else
        print ” Variabeln är inte lika med 2”;
```

Exemplet ovan skriver ut meddelandet ”`Variabeln är lika med 2”` om variabeln `$a` är lika med 2. I alla andra fall skrivs meddelandet `”Variabeln är inte lika med 2”` ut i webbläsaren. Vidare finns möjligheten att kombinera `if-` satser med logiska operatorer. Detta skulle kunna se ut så här:

```
if (($a == 2) && ($b == 3))
{
```

```
        //gör endast om $a är 2 och $b är 3…
}
else
{
        //… annars utför detta
}
```

I det ovanstående exemplet utförs `if`- satsen om och endast om `$a` är lika med 2 och `$b` är lika med 3. Skulle dessa variabler innehålla andra värden, exekveras `else`- satsen.
I vissa fall kan `if`- satserna ersättas med en så kallad `switch`- sats och det är när ett och samma värde testas mot flera andra värden. Men en `switch`- sats kan inte jämföra annat än lika med. Det innebär att denna sats endast kan användas i begränsade situationer. Exempel på när en `switch`- sats är lämplig att använda är när en variabel ska jämföras med många olika värden. Beroende av variabels värde, ska en viss procedur utföras. Vi ska i ett förenklat visa hur syntaxen ser ur vid användandet av en `switch`- sats.

```
switch ($variabel)
{
        case 1: // något utförs
        break;
        case 2: // något annat utförs
        break;
        .
        .
        .
        case n: // ytterligare något annat utförs
        break;
        default: // utförs när inget annat matchades
        break;
}
```

En `switch`- sats inleds alltid med nyckelordet `switch`. Därefter följer ett antal fall, `case`, och i varje av dessa fall jämförs `$variabel`. Där kommentarstecknen finns, ska programmeraren placera den kod som associeras till det värde `$variabel` har. När programmet hittar ett värde i en `case`- sats som stämmer överens med variabelns värde, exekveras koden som följer efter. Två andra nyckelord som också förekommer är `break` och `default`. `Break` är till för att avbryta switch- satsen efter att ett av alla fallen har

exekverats. `Default` samlar upp alla andra fall, fall som då inte explicit angetts i något av de andra case- satserna. `Default` ska alltid placeras sist i en switch- sats.

7.5.6 Loopar

En loop används när ett program ska utföra något flera gånger efter varandra. Ett väldigt vanligt användningsområde är när något ska skrivas ut, exempelvis innehållet i en array. En loop består av ett block av PHP- kod som exekveras ett givet antal gånger eller till dess att ett villkor uppfylls. De vanligaste typerna av loopar är `for`- och `while`- loopen. Det finns två andra varianter på dessa; `foreach` och `dowhile`- loopar. Vi nöjer oss dock med att visa exempel på de två förstnämnda.

```
for ( $i = 1; $i <= 5; $i++)
{
       print ”$i <br>”;
}
```

Exemplet ovan visar en typisk användning på en `for`- loop. En `for`- loop är vanligtvis uppbyggd enligt en viss syntax; initiering av en variabel, ett villkor, en räknare och inom `{}` sker det som loopen ska utföra. I detta exempel, skrivs variabeln `$i` ut följt av ett enterslag (
). Räknaren, som motsvaras av `$i++`, ökar värdet på `$i` varje gång som loopen körs och enligt exemplet skulle detta bli 5 gånger. När `$i` får värdet 6, bryts loopen, eftersom villkoret för att loopen ska köras är att variabeln ska vara mindre än eller lika med 5. Så, utskriften från denna `for`- loop blir:

```
1
2
3
4
5
```

En titt på hur motsvarande exempel skulle kunna skrivas med en `while`- loop, se exemplet nedan.

```
$i = 1;
```

```
while ($i <= 5)
{
        print ” $i <br>”;
        $i++;
}
```

Det enda som skiljer är sättet att skriva koden. I fallet med en `while`- loop måste initieringen av variabeln ske innan själva loopen börjar. Därefter kontrolleras variabelns värde och loopen exekveras så länge som `$i` är mindre än eller lika med 5. I `while`- loopens kodblock skrivs variabeln ut och för att öka variabelns värde, måste `$i++` placeras sist i loopen.

7.5.7 Typomvandling

Typomvandling innebär helt enkelt en omvandling av en datatyp till en annan. På så sätt tvingas en variabel att ändra sin datatyp. Det kan gå till på två olika sätt; det ena sättet som ofta kan skådas i C/C++ - kod, vilket innebär att programmeraren specifikt deklarerar vilken typ variabeln ska ha, och det andra som utförs med hjälp av PHPs egna fördefinierade funktion, `settype()[7]`. För att tydligare visa hur detta kan gå till, studera följande exempel:

Variant i C/C++ - stil:

```
$pris = 4.56;
$pris = (int)$pris;
```

Variant med funktionen `settype()`:

```
$pris = 4.56;
settype($pris, integer);
```

I det översta exemplet, varianten i C/C++, tilldelas variabeln `$pris` en datatyp av typen flyttal (eng. double), eftersom högerledet är ett decimaltal. Raden under anger att `$pris` sedan antar ett värde av datatypen heltal (eng. integer). Det undre exemplet, varianten med `settype()`, gör exakt samma sak. Det enda som skiljer är sättet att skriva det på.

7.5.8 Escapesekvenser

Meningen med escapesekvenser är att ”fly” från olika begränsningar som språket har genom att använda olika teckensekvenser. Studera följande exempel nedan:

```
$string = 'What\'s up doc?';
print "Variabeln \$string innehåller
       Snurre Sprätts vanliga fråga \"$string\"<br>\n";
```

I exemplet ovan initieras `$string` till en sträng med innehållet `What's up doc?`. Dock görs detta med apostrofer, istället för citationstecken som är det vanliga. Problemet i exemplet ovan är att strängen innehåller en apostrof i *What's*. Om slashtecknet framför apostrofen, vid initieringen av strängen, hade utelämnats, hade PHP tolkat denna apostrof som strängslutet. Men i och med att programmeraren anger ”\”, vet PHP att nästkommande teckens egentliga betydelse ska ignoreras. Det finns motsvarande slashtecken i printsatsen också, då för att ta bort betydelsen för bland annat `$`. I printsatsen innebär det att dessa tecken kan skrivas ut, vilket hade varit omöjligt utan slashtecknet (`\`). Det är detta som kallas för escapesekvenser och de används för att inte PHP ska bli förvirrat av vissa tecken. Här följer en tabell på de vanligaste escapesekvenserna, som är hämtad ur [12]:

Escapesekvenser	Betydelse
`\ '`	Apostrof
`\ "`	Citationstecken
`\ \`	Backslash
`\ $`	Dollartecken
`\ n`	Ny rad
`\ r`	Radmatning (Carriage return)
`\ t`	Tabulering

Tabell 7.2 Escapesekvenser i PHP

Exemplet ovan hade kunnat lösas enklare utan användning av escapesekvenser. Det som då skulle förändras var initieringen av strängen. Istället för att använda sig av apostrofer, kunde vanliga citationstecken utnyttjas. Det hade då sett ut så här:

```
$string = ”What’s up doc?”;
```

7.6 Sammanfattning PHP

I detta kapitel har vi sagt att PHP är ett skriptspråk som exekverar skripten på servern och att det genom åren framtagits flera olika versioner av PHP. Den nuvarande versionen är PHP4, men kommer inom kort att ersättas av den helt objektorienterade versionen PHP5. Vi har beskrivit den generella arkitekturen av PHP och skrivit om den senaste skriptmotorn Zend. Vi har tagit upp hur PHP fungerar och beskrivit vad PHP är. Dessutom har vi i avsnitt 7.5 tagit upp hur PHP’s syntax ser ut och behandlat enkla exempel. Vi tog där bland annat upp vilka PHP’s olika datatyper var, hur variabler deklareras, hur funktioner skrivs mm. Vi har sagt att PHP är ett svagt typat språk med över 2000 funktioner samt stöd för en hel del databaser. Som kunde läsas i inledningen, finns ett kodexempel i PHP som krypterar ett meddelande i bilaga C.

I nästa kapitel kommer nu själva jämförelsen mellan ASP och PHP. Uppsatsen avslutas sedan med ett resultat eller mer en rekommendation till Hög-Data, om vilket språk som vi anser att de i framtiden ska utnyttja.

8 ASP kontra PHP

I detta kapitel kommer vi att ta upp viktiga skillnader mellan ASP och PHP. Vi kommer också att visa på vilka plan som språken inte skiljer sig, det vill säga vilka likheter som finns. Eftersom vi i tidigare kapitel relativt utförligt berättat vad ASP och PHP är och hur de fungerar, kommer vi i detta kapitel mer visa skillnaderna och likheterna på mer ett kortfattat vis. Vi kommer alltså inte här ge några djupare diskussioner, utan mer kortfattat berätta vad som skiljer/inte skiljer de båda skriptspråken. I detta kapitel har vi förutom böcker och Internet använt oss av kunskaper men även synpunkter hos två av Hög-Datas anställda, varav David Hall som vid denna tidpunkt var praktiserande och Victor Norgren, som har byggt Hög-Datas webbplats. David Hall hade sedan tidigare kunskaper i PHP. Han höll på med att lära sig ASP på Hög-Data, när denna uppsats skrevs. Därför kunde hans synpunkter vara viktiga, eftersom han enklare kunde jämföra ASP och PHP på ett bättre sätt jämfört med Victor Norgren, då han enbart hade kunskaper inom ASP- tekniken. Däremot hade han mer kunskaper i ASP än David Hall, eftersom han hade större erfarenhet av ASP- programmering. En tredje person, Viktor Jonsson, har också bidragit med en del information, då främst inom PHP- området. Informationen har vi fått från honom via e- mail. Notera att han även har skrivit boken ”*Webbprogrammering med PHP*” som vi har använt oss av i kapitel 7.

8.1 Plattformsoberoende eller plattformsberoende

Detta är en av de viktigaste punkterna. När vi sökte information på Internet om just ASP och PHP, kunde vi inte undgå att lägga märkte till alla forum som handlade om vilket av skriptspråken som var bäst. Det som var ett hett ämne var just på vilka maskiner och därigenom vilket operativsystem som ASP och PHP kunde köras på. Eftersom inlägg i olika forum inte alltid är ur en helt objektiv synvinkel, var vi tvungna att kontrollera uppgifterna därifrån i böcker och med Hög-Datas båda webbanställda. Det som fastslogs var att ASP nästan helt är bunden till Microsoftprodukter medan PHP har stöd i betydligt fler plattformar. Det innebär att detta är ASP’s stora nackdel, just att så få plattformar stöds. Dock ska det tilläggas att PHP’s extra funktioner inte med säkerhet fungerar i andra operativsystem än Unix/Linux. I vissa av dessa miljöer (operativsystem) tappas vissa bibliotek bort och även vissa rättigheter kan försvinna och därigenom faller viss funktionalitet bort.

Det finns visserligen tredjepartslösningar som exempelvis Chili!ASP, som gör det möjligt att använda ASP utanför Microsofts produkter. Chili!ASP's officiella hemsida är www.chilisoft.com. Nackdelen med mjukvara som Chili!ASP och liknande är att de inte är officiella Microsoft produkter, och att användare av dessa program inte kan vända sig till Microsoft för support. Men även om ASP mer eller mindre är bundet till enbart Microsoft produkter innebär detta oftast inget problem. Detta eftersom att Microsoft är ett så pass stort och utbrett företag i världen idag och att dess produkter används av många stora och medelstora företag världen över.

8.2 Kostnad

Kostnaden är också en aspekt som skiljer ASP från PHP en del. Kostnaden för att utveckla webbapplikationer i ASP respektive PHP är i sig gratis. Allt programmeraren behöver är någon slags editor, där han/hon kan skriva koden i och sådana finns i en hel uppsjö på Internet för gratis nedladdning. Det som kostar pengar är mjukvaran till servern (samt naturligtvis själva servern), men det är för enbart ASP's mjukvara som användarna måste öppna sin plånbok. PHP's programvara finns tillgänglig på PHP's officiella hemsida www.php.net [20]. När det gäller ASP måste företagen eller privatpersonerna som använder sig av detta skriptspråk betala för en licens, som då innebär att företaget eller personerna lagligt kan använda Microsofts tjänster. PHP's programvara får användas gratis så länge som webbsidans ansvarig inte tjänar pengar på att utnyttja den. Då ska någon slags avgift erläggas, enligt David Hall. Detta har sin grund i att källkoden i PHP är ”open source”, vilket i praktiken innebär att vem som helst får vara med och påverka hur PHP ska se ut. Men det innebär samtidigt att människor världen över inte får tjäna pengar på vad andra personer en gång i tiden har utvecklat. I Hög-Datas fall råder vi företagets ansvariga att noggrant kontrollera vilka regler som gäller, eftersom Hög-Data är ett säljande företag.

8.3 Säkerhet

En viktig detalj numera är hur säkra webbsidorna är. Detta eftersom det finns programmerare som tycker om att sabotera andra personers hemsidor. Eftersom ett företag är speciellt utsatt för intrång av olika slag, är det viktigt att ett skriptspråk som ASP och PHP tillhandahåller en god säkerhet. När vi frågade David Hall, svarade han med att PHP är lite bättre utrustat när det gäller hur säker en hemsida kan bli om han jämförde med ASP. Däremot var han och Victor Norgren eniga om att säkerheten till stor del berodde på programmeraren som skapade

hemsidorna. Dessutom tillade Victor Norgren att skript på serversidan, så kallade server-side scripting, var en ganska säker metod. Som tidigare kunde läsas är både ASP och PHP skriptspråk som exekveras på servern. Skript på servern behandlade vi i avsnitt 5.4.2 i denna uppsats.

En annan viktig detalj som Victor Norgren tog upp, var att säkerhetsaspekterna i ASP till stor del vilade på hur webbservern var installerad och konfigurerad. Han menar att om användarna gör korrekta inställningar redan från början, är ASP en relativt säker metod. David Hall berättade vidare att eftersom det så ofta kommer ut nya versioner av PHP, täcks de säkerhetshål, som uppkommit i tidigare versioner, över inom en ganska snar framtid. Kanske till och med samma dag, eftersom vem som helst är välkommen att delge just sin lösning. Men båda är överens om att webbserverns olika inställningar är ytterst viktiga när det gäller säkerheten i både ASP och PHP. I de olika böckerna vi läst om ASP och PHP har vi inte funnit så mycket information som vi först trodde, just angående säkerheten. De böcker vi har använt oss av har i bästa fall avsatt ett avsnitt till hur programmeraren ska skapa så säkra webbplatser som möjligt. I och för sig är dessa böckers främsta syfte att läsaren ska få lära sig utveckla dynamiska hemsidor, inte att bygga dem så säkra som det bara går. Tyvärr har vi inte hittat någon bra litteratur som koncentrerar sig på säkerheten, vare sig det gäller ASP eller PHP. Den bok som vi tycker har varit bäst på säkerhetsaspekten är Larry Ullmans bok *PHP For the world wide web* [21]. Där har författaren angett, i en bilaga, några webadresser, som den intresserade kan besöka för mer information. Och eftersom säkerheten på Internet blir viktigare och viktigare, bör framförallt skriptningen på servern vara en betydligt säkrare metod vid jämförelse med klientskripningen. Som vi tidigare sagt är både ASP och PHP exempel på tekniker som tillämpar skripning på serversidan.

8.4 Installationen av servermjukvara

För att en webbserver ska kunna fungera på ett bra sätt krävs rätt programvara. Vi kommer i detta avsnitt inte att berätta hur användarna ska gå till väga för att installera den programvara som respektive skriptspråk behöver, utan vi nöjer oss med en redogörelse för vad som skiljer dem åt.

Vi nämnde tidigare att en av ASP's stora nackdel var att tekniken var plattformsberoende. Dock finns det andra möjligheter att installera och köra andra mjukvaror, exempelvis Chili!ASP, som då skulle göra det möjligt att köra ASP på andra plattformer än just Microsofts. Dock kan dessa tredjepartslösningar inte alltid motsvara alla förväntningar, då de

inte utvecklas av Microsoft vilket kan leda till att Chili!ASP och liknande produkter inte stödjer ASP fullt ut. Väljs ASP's ”officiella” programvara IIS, som enligt Victor Norgren är enkelt att installera, erhålls en webbserver som klarar allt som ASP är ämnat för. Victor tillägger att användaren bör göra egna, personliga inställningar när en ny sida skapas. Detta för att bland annat få optimal säkerhet. Om vi jämför med installationen i PHP, skulle detta vara ett värre moment, enligt David Hall. Dessa uppgifter angående installationen som Victor Norgren och David Hall har givit oss återfinns även i Viktor Jonssons bok [24]. Dessutom är programmeraren tvungen att utföra vissa moment innan servern fungerar som den ska. Han tillägger att mjukvaran för ASP är av ”Plug n Play”-modell, vilket gör det enkelt för användaren att installera. De flesta av Microsofts mjukvaror nu för tiden är just av denna modell, vars översättning skulle bli ”att det bara är att plugga i och köra”. En liten fördel med ASP i det här fallet är att programvaran redan finns i de olika versionerna av operativsystemet Windows. Detta heter PWS och det står för Mircrosoft Personal Webserver. Jämfört med IIS innebär dock PWS begränsade möjligheter, men fungerar alldeles utmärkt om webbapplikationen endast ska köras på en lokal dator.

8.5 Utveckling och framtid

Utvecklingen inom ett tekniskt område kan vara väldigt viktigt för såväl privatpersoner som företagare när de ska välja vilken teknik som de vill nyttja. Med utveckling menar vi om, och i så fall hur, språket kommer utvecklas. Med det menar vi om regelbundna uppdateringar planeras. En annan sak som är viktig inför framtiden inom en viss teknik, är om det kommer att finnas några fördelar gentemot andra tekniker. Denna rubrik, utveckling, går hand i hand med nästa rubrik, som är uppdateringsmöjligheter.

När det gäller utvecklingen på ASP- sidan, har ju en ny metod tagits fram, ASP.NET. Avsnitt 6.5 handlade om viktiga skillnader mellan ASP och ASP.NET. Där kunde det läsas att den nya metoden bland annat hade stöd för hela Visual Studio, där bland andra Visual Basic ingår. Det klassiska ASP har endast stöd för skriptvarianter av VB och Java, d.v.s. VBScript eller Javascript. Det innebär att det nu öppnas större möjligheter och i synnerhet som ASP.NET också stödjer det nya programspråket C#. ASP.NET kommer därför att tillhöra framtiden inom ASP- tekniken, men eftersom den klassiska modellen av ASP inte är helt kompatibel med ASP.NET, kommer det säkerligen dröja ett tag innan den ASP.NET slår igenom på allvar. Det tror i alla fall Victor Norgern. Han fortsätter med att ASP och skriptspråket VBScript nu är uppe i version 3.0.

När det gäller PHP är nästa stora version att vänta PHP5. Till skillnad från ASP, släpps det betydligt fler versioner av PHP. En anledning till detta att många människor världen över hjälper till att forma PHP med sina egna lösningar. Detta leder till att vid varje uppdatering hos PHP sker det inte så stora förändringar som det gör hos ASP. Inom ASP sker uppdateringarna mera sällan och här kan skillnaderna, mellan varje version, vara så pass stora att gammal ASP- kod kanske inte är kompatibel med de nya funktionerna. Det leder till att programmeraren i värsta fall kanske blir tvungen att skriva om vissa skript. I PHP släpps versionerna tätare, vilket innebär att väldigt små justeringar görs mellan varje version. Detta i sin tur leder till att programmeraren egentligen hela tiden har uppdaterade funktioner tillgängliga. Det säger David Hall. Enligt vad Viktor Jonsson erfar kommer PHP5 att vara helt objektorienterat, som vi tidigare nämnt i uppsatsen. Dessutom kommer den nya versionen vara begåvat med en automatisk minneshanterare, som automatiskt, under tiden programmet exekverar, kan ”städa upp” olika minnesutrymmen och därmed exempelvis kontrollera vilka variabler som används.

8.6 Uppdateringsmöjligheter

I förra avsnittet talade vi om att PHP’s olika versioner släpptes oftare än ASP’s, men att det samtidigt innebar att skillnaderna var färre jämfört med ASP’s olika utgåvor. De uppdateringar som Microsoft har utvecklat för ASP kan inte användas gratis till skillnad från motsvarande i PHP. PHP’s uppdateringar kan laddas ned gratis från Internet, medan en avgift får erläggas vid varje uppdatering beträffande ASP. Men ska dock ha i åtanke att företag och personer som använder sig av PHP- skript inte får tjäna pengar på deras webbplats, eftersom koden i många fall är utvecklad av privatpersoner som inte har erhållit ett öre för de funktioner som de har skrivit. ASP- koden är utvecklad hos anställda på Microsoft, som får betalt för sitt arbete. Därigenom måste kunder hos ASP och Microsoft generellt skaffa licenser till sina program. Licenserna är ett kvitto på att mjukvaran används på ett lagligt vis. I och med företag som Microsoft arbetar med licenser på sina utvecklande produkter, finns möjligheten för kunderna att få support, om problem skulle uppstå. Detta fungerar inte på samma vis när det gäller PHP, eftersom det egentligen inte finns någon som är ytterst ansvarig för programvaran. Det finns gott om information både på Internet och i böcker när det gäller både ASP och PHP, men visst kan det ibland underlätta att få en direkt och relevant lösning på de problem som uppstår. Detta är lite av PHP’s svaghet.

8.7 Användarvänlighet

Användarvänlighet hos olika tekniker är en faktor som är väldigt personlig i många fall. För att åter nämna Internets alla forum, finns det gott inlägg som både är positiva och negativa beträffande ASP's och PHP's användarvänlighet. Det som generellt tas upp är att ASP har en rejäl fördel gentemot PHP, eftersom ASP stödjer olika språk. De vanligaste, vilka vi redogjorde för i avsnitt 5.4, är VBScript och JavaScript. Det innebär att hemsideutvecklarna har en stor valmöjlighet, just att det ges utrymme att relativt fritt välja ett språk som utvecklaren behärskar. I PHP är det ju ett språk som gäller och där finns inte denna valmöjlighet. Enligt Victor Norgren fungerar ASP dessutom likadant oavsett vilket skriptspråk som väljs, det enda som skiljer är syntaxen språken emellan. Men Victor Norgren, liksom många inlägg på olika forum på Internet, rekommenderar programmerarna att inte blanda olika skriptspråk, eftersom de i alla fall oftast kan utföra samma saker. Blandas olika skriptspråk i ASP- koden, skulle det kunna innebära att koden blir inkonsekvent och att kodens läslighet minskar.

Ytterligare en nackdel med PHP är att PHP, enligt David Hall, har många olika funktioner som i princip utför samma sak. Detta kan leda till att den orutinerade PHP- användaren kan känna en viss komplexitet och att förvirring lätt kan uppstå i samband med att programmeraren inte vet vilken algoritm som är bäst vid varje given tillfälle. Å andra sidan skulle detta kanske innebära en viss fördel för mer rutinerade PHP- programmerare, då programmerarna enklare kan utnyttja de funktioner, som helt och hållet är optimal för ett visst användningsområde. Men enligt David Hall är ASP's syntax lättare i den meningen att programmeraren inte har så stor valfihet att fixa till ett problem.

Något som båda skriptspråken är begåvat med är att det finns ett stort antal fördefinierade funktioner. Eftersom många olika människor världen över har varit med och byggt upp PHP, har detta språk fler funktioner, och som vi tidigare varit inne, på gör flera av PHP's funktioner i stort sett samma sak. Funktionerna är mer specialiserade i PHP, medan funktionerna i ASP kan ha fler användningsområden. Men David Hall medger att han känner att ungefär samma resultat kan uppnås oavsett vilken teknik som tänker utnyttjas. Men han upplever att skripten i PHP generellt sett blir större än motsvarande i ASP när VBScript användes. Detta skulle i praktiken innebära att antalet kodrader i PHP är fler än för motsvarande applikation i ASP.

8.8 Kompatibilitet och portabilitet

Vi har varit lite inne på denna punkt tidigare, då vi pratade om plattformsberoendet respektive plattformsoberoendet. Vi kom där fram till att ASP är betydligt mer beroende av Microsofts operativsystem än PHP skulle vara av sitt Unix- system. Men eftersom Microsoft har den största marknaden, innebär detta oftast inget problem. En av de vanligaste åsikterna om programvaror som är tillverkade av Microsoft är att de är dyra och dessutom det ofta påträffas fel. Men vad som ofta glöms bort är att många av Microsofts produkter och komponenter ofta är kompatibla med de övriga av Microsofts ”verk”. Ett bra exempel på detta är ASP.NET, som exempelvis stödjer hela Visual Studio.

Själva koden i ASP är omöjlig att flytta eftersom programmeraren enbart kan flytta kod mellan två datorer som kör samma mjukvaruplattform, ex Windows med IIS. PHP:s kod går att flytta i princip mellan alla möjliga plattformar. Har programmeraren dock använt sig av systemspecifika funktioner eller egenskaper, kan en viss konvertering behövas. Det kan exempelvis handla om filhantering. Informationen till detta stycke har vi fått från Viktor Jonsson personligen i form av ett mail.

8.9 Prestanda

Prestandan hos de båda skriptspråken skulle, enligt Viktor Jonsson, vara väldigt svårt att mäta, åtminstone att få så rättvisa siffror som möjligt skulle erhållas. Förutom att det krävs mycket tid och två identiska applikationer i ASP och PHP skulle det krävas att utrustningen runt omkring programmen, var identisk. Detta p.g.a. att så många externa faktorer påverkar, som exempelvis valet av webbserver. För att dessutom överhuvudet taget se någon skillnad i exekveringstid skulle applikationerna också vara relativt stora. Det vanliga enligt Viktor Jonsson är att servrarna, Apache och IIS jämförs istället. Men han tillägger att dessa tester ofta är missvisande. Tyvärr har vi inte funnit några siffror på ASP och PHP ur prestanda synpunkt, men enligt [21] ska PHP vara ett av de bästa alternativen beträffande exekveringen och även att programmera och utveckla program i, för den delen. Nu är det väl så, att information och vissa åsikter från personer som helt fördrar ett språk framför ett annat, får läsarna ta med en nypa salt. Dock upplevde David Hall också att PHP var en något snabbare metod jämfört med ASP när det gällde hur snabbt skripten exekverade. Men han tillade att det gäller främst på större tillämpningar, där skripten innehåller många tidskrävande operationer. Eftersom den klassiska versionen av ASP är en äldre teknik än PHP, talar det mesta för att PHP skulle vara ett, ur exekveringssynpunkt, bättre alternativ än ASP. Men eftersom vi inte

har några tydliga bevis på det, utan mest har personliga åsikter och erfarenheter som bakgrund, lämnar vi denna fråga öppen.

9 Slutsats

Vår slutsats grundar sig på de kunskaper och den information som vi har tagit fram genom att undersöka ASP och PHP. Vi har även rådfrågat två anställda på Hög-Data AB, nämligen David Hall och Victor Norgren, för att få deras åsikt i frågan. De olika aspekterna som vi utgick ifrån i vår jämförelse utvidgade sig något då det skulle vara svårt att mäta laddningstider och svarstider. Utifrån den information som vi lyckades få fram om ASP och PHP så har vi försökt att presentera uppgifterna i denna uppsats på ett så objektivt synsätt som möjligt. Med denna information grundar vi vår rekommendation till Hög-Data AB, när det gäller valet av skriptspråk för deras del. Vår rekommendation till Hög-Data AB blir därför att fortsätta utveckla webbsidor i ASP. Alternativt kanske ASP.NET, eftersom detta förfaringssätt att skapa dynamiska hemsidor tillhör framtiden. Nackdelen är att ASP.NET inte är kompatibel med den klassiska ASP- varianten. Det skulle innebära att Hög-Datas webbprogrammerare skulle behöva lära sig ett helt nytt språk.

Rekommendation bygger på olika aspekter beträffande de olika språken så som exempelvis mängden kod mellan de båda skriptspråken men även användarvänlighet samt säkerhet. En av de viktigaste anledningarna till att vi tycker att Hög-Data AB skall använda sig av ASP i fortsättningen är för att ASP utvecklas av ett av världens största mjukvaruföretag; Microsoft. Även ASP.NET är framtagit av Microsoft. PHP däremot, har inget respekterat företagsnamn att luta sig tillbaka på. Detta är en nackdel vid supportfrågor och liknande. David Hall, som har kunskaper inom PHP- området, tycker personligen att ASP är en bättre modell för Hög-Data AB än PHP. En anledning till att han kände så är att ett så stort företag som Microsoft ligger bakom ASP. Microsoft är ju trots allt marknadens ledande företag inom mjukvaror. Därigenom finns alltid möjligheten att vända sig till dem om några problem uppstår. Denna utväg finns egentligen inte när det PHP, eftersom det är användarna själva som utvecklar mycket av produkten. Dessutom uppstår problem för säljande företag i samband med att koden i PHP är gratis, som vi tidigare nämnde. Det är mycket detta som har gjort att företag, speciellt säljande företag, väljer ASP framför PHP. Detta innebär att företag som skriver program i PHP för att sedan tjäna pengar på dessa, måste ha någon form av licens för den kod som de inte utvecklar själva.

Däremot används PHP i stor utsträckning av privatpersoner, som endast vill göra sin hemsida mer attraktivare. Det är därför PHP har ökat närmast explosionsartat de senaste åren. Nu för tiden räcker det oftast inte med att en sidas innehåll är bra, utseendet och möjligheterna till att enkelt förändra sidans innehåll är nog så viktiga avseenden. PHP är dessutom gratis för alla privatpersoner, som inte tänkt tjäna pengar på den PHP- kod som används. Ytterligare en orsak till att PHP blivit så populär bland privatpersoner, är att många människor världen över är bekanta med programspråket C/C++. Eftersom PHP's syntax är snarlik, finns stora möjligheter till att utveckla tilltalande webbsidor i en snabbare takt, än om programmerarna först var tvungna att lära sig ett nytt skriptspråk.

När vi frågade David Hall och Victor Norgren på Hög-Data AB, var de övertygade om att det bästa för Hög-Data AB var att fortsätta webbutvecklingen i ASP. Eventuellt skulle en server förberedas för PHP- skript om behovet skulle uppkomma, men att PHP skulle ersätta ASP-tekniken helt och hållet trodde de var uteslutet. Istället trodde de att PHP skulle kunna fungera som en kompletterande länk till ASP. När vi frågade dem om ASP.NET, var de tveksamma till denna teknik också. Anledningen var att de tyckte att språket såg för komplicerat ut. Det skulle därmed innebära att ASP och ASP.NET rent språkmässigt skiljer sig åt ganska mycket.

Referenser

Referenserna är ordnade i bokstavsordning. Ordningen är efter efternamn.

[1] J. Ahlberg, *Jonas Webresurs – Sveriges bästa webbskola sedan 1998!*, www.jonasweb.nu , 030316

[2] K. Andersson, *CCS HTML-skola*, www.ccsweden.se/webskola/home.html, 030316

[3] A. Appu, *The Making Use of PHP*, Wiley Publishing Inc, ISBN: 0-471-21973-8, 2002

[4] K. Francis, J. Kauffman, J. Llibre, D. Sussman, D. & Ullman, C. *Beginning Active Server Pages 2.0*, Wrox Press Ltd, Birmingham, 1999

[5] F S data, *F S data, driftsäkert webbhotell, www hotel, webshop, sql, webhotell, egen domän…*, www.fsdata.se/manual/manual10.shtml#PHP_vad, 030325

[6] I. S. Graham, L. Quin, *XML Specification Guide*, Wiley computer Publishing USA, 1999

[7] J. Hall, *Internetprogrammering DAA 751 - Serverside*, www.masda.vxu.se/users/jha/kurser/daa751/jhweb/serverside/serverside.htm, 030324

[8] R. Henriksson, *http://www.cs.lth.se/Education/Courses/EDA070/vecka6/ohRoger.pdf*, http://www.cs.lth.se/Education/Courses/EDA070/vecka6/ohRoger.pdf, 030327

[9] S. Hettihewa, *Lär dig Active Server Pages 2.0 på tre veckor*, Pagina Förlags AB, Sundbyberg 1999

[10] T. Höjemo, *PHP ett skriptspråk för webbservrar*, www.snt.nu/kurser/phpintro/index.php?, 030321

[11] T. Höögh, C. Mattson, *Active Server Pages och dynamiska hemsidor*, c- uppsats vid Karlstads Universitet (institution Informatik), 1501518887, december 2000

[12] V. Jonsson, *Webbprogrammering med PHP*, Studentlitteratur, Karlstad 2001

[13] Svenska Linuxföreningen, *selinux – Svenska Linuxföreningen*, www.se.linux.org/artiklar/programmering/php/php-del-1.phtml, 030325

[14] M. Luiga, K. Melin och A-K Wåhlin, *PHP kontra ASP*, www.prego.se/exjobb, 030325

[15] A. Mhapsekar, *W3Schools Online Web Tutorials*, http://www.w3schools.com, 030408

[16] Williamssons Offset, *Vad är XML?*, Statskontoret 1998:6, Solna, 1999

[17] M. Omander, *Din Guide till Internet programmering*, Bonnier Icon, Angered, 1998

[18] E. Ronne, ASP *Active Server Pages*, Docendo läromedel AB, Stockholm, 1999

[19] T. Sjöstedt, *Torgny Sjöstedts sidor och länkar för studerande*, www.hisvux.se/torgny, 030315

[20] The PHP Group, *PHP: Hypertext Preprocessor*, www.php.net, 030415

[21] L. Ullman, *PHP For The World Wide Web*, Peachpit Press (Addison Wesley Longman), ISBN: 0-201-72787-0, 2001 (det engelska originalet samt en svensk översättning)

[22] A. K Weissinger, *ASP in a nutshell: A Desktop Quick Reference*, O'Reilly & Associates ISBN: 1-56592-490-8, 1999

[23] W3C, *World Wide Web Consortium*, www.w3.org, 030402

[24] P. Åkesson, *Underhåll av webbplats med PC*, www.abc.se , 030313

A Förklaringar och förkortningar

I den första bilagan kommer vi att ge läsaren en kort översikt över vanligt förekommande förkortningar och uttryck i denna uppsats.

APACHE – Är en gratis webbserver som fungerar på många olika plattformar. Ursprungligen utformad för UNIX.

API – *Application Programming Interface*. Är ett programmeringsgränssnitt mot ett program eller bibliotek.

ASP – *Active Server Pages*. En metod som är utvecklad av Microsoft för att skapa dynamiska hemsidor.

ASP.NET – En ny teknik för att skapa dynamiska hemsidor, också den utvecklad av Microsoft.

CGI – *Common Gateway Interface*. En äldre teknik för skapande av dynamiska hemsidor.

CLIENT-SIDE – Skript som körs på klientsidan, den sida som användaren befinner sig på.

C# - *C sharp*. Är ett relativt nytt programmeringsspråk som är utvecklat av Microsoft. Är till för att kombinera enkelheten hos VB med kraftfullheten hos C++. Utvecklat speciellt för att skapa webbapplikationer.

C++/C – Båda är populära programmeringsspråk. C är det äldsta av de två. C++ stödjer Objektorientering vilket C inte gör.

DLL – *Dynamic Link Library*. En speciell fil som innehåller resurser som kan bli länkade till program vid körning.

DYNAMISKA HEMSIDOR – Så kallade levande sidor. Användarna kan skicka och ta emot information till sådana sidor.

HTML – *HyperText Markup Language*. Är ett språk som det går att skapa statiska hemsidor med.

HTTP – *HypeTtext Transfer Protocol*. Är ett protokoll som ligger till grund för överföring av dokument i World Wide Web.

IIS – *Internet Information Server*. Microsofts vanligaste webbservrar.

IMAP – *Internet Message Access Protocol*. Är en metod att komma åt e-post på en delad mail-server.

ISAPI – *Internet Server Application Programming Interface*. Microsofts modell för kommunikation mellan http-servern och databasen eller andra applikationer.

JAVA – Ett relativt nytt programmeringsspråk. Skapades av Sun Microsystem i USA.

JAVASCRIPT – Ett av de två vanligaste skriptspråken som används i ASP.

JSP – *Java Server Pages*. Är en HTML mall som liknar ASP fastän konceptet är annorlunda. Java Server Pages är ett kompilerat språk istället för ett tolkat språk, som PHP och ASP är.

LAMP - *Linux, Apache, MySQL, PHP* , vilket är den vanligaste installationsformen när det gäller PHP.

MTS – *Microsofts Transaction Server*. Är en server operation som lyckas eller misslyckas som helhet, även om den görs i olika steg.

MYSQL – En av de mest använda databaserna, används i samband med PHP.

NSAPI – *Netscape Server Application Programming Interface*. En specifikation för kommunikation mellan Netscapes http –server och andra databaser och applikationer.

ODBC - *Open Database Connectivity*. Är ett API som tillåter programmeraren att arbeta med databaser utan att först skriva kod för att kunna kommunicera med dem. ODBC är en drivrutin som fungerar som en medlare mellan dig som programmerare och databasen.

OPEN SOURCE – En programvara där källkoden är synlig och fri att ändra i.

PERL - *Practical Extraction and Report Language*. Ett språk med kraftfull stränghantering främst för att ta ut information från text.

PHP – *Hyperrext Preprocessor* (tidigare *Personal Home Page*). En ”open scource” teknik för att skapa dynamiska hemsidor.

SAPI - *Server Application Programming Interface*. Microsofts programmeringsgränssnitt mellan applikationer och deras Internetserver.

SERVER-SIDE – Skript på serversidan är den sida som servern befinner sig på.

SGML – *Standard Generalized Markup Language*. Internationell ISO standard som gett upphov till HTML och XML.

STATISKA HEMSIDOR – Statiska sidor kan ej bli ”levande”, kan bara visa information utan att själv ta emot någon.

SKRIPT – Ett förenkelt programmeringsspråk designat för att utföra speciella eller begränsade uppgifter.

VB – *Visual Basic*. Visual Basic är ett Windows baserat programspråk som gör det lättare att programmera i.

VBSCRIPT – Ett av de två vanligaste skriptspråken som används i ASP och är en skriptvariant på Visual Basic.

W3C – *World Wide Web Consortium*. Arbetar med att ta fram gemensamma protokoll för webben.

WWW – *World Wide Web*.

XML – *eXtensible Markup Language*. En blandning av SGML och HTML för publicering och lagring av information.

ZEND – Är en skriptmotor som konstruerades till PHP3. Utvecklad av Zeev Suraski och Andi Gutmans.

B ASP’s fullständiga objekt

I denna bilaga presenterar vi ASP’s fullständiga objekt. Med fullständiga menar vi att alla objekts ”funktioner” här finns angivna. Objekten är uppbyggda av fyra olika kategorier; *egenskaper, kollektioner, metoder och händelser*. Dock är det bara session- objektet som innehåller funktioner från alla dessa kategorier. Bilagan ska en ge en helhetsbild över ASP, och ska endast fungera som en upplysning för läsaren om vad ASP tillhandahåller. För en detaljerad beskrivning av alla funktionerna, hänvisas till boken *ASP in a nutshell* [1].

Eftersom alla sex objekt inte fick plats på samma sida, har vi valt att representera 4 av dem på nästa sida. De är Application objekt, Response objekt, ObjectContext objekt och Server objekt. På denna sida återfinns Request objektet och Session objektet.

Session Objekt

Egenskaper

- CodePage
- LCID
- SessionID
- Timeout

Kollektioner

- Contents
- StaticObjects

Metoder

- Abandon

Händelser

- Session_OnStart
- Session_OnEnd

Request Objekt

Egenskaper

- TotalBytes

Kollektioner

- ClientCertificate
- Cookies
- Form
- QueryString
- ServerVariables

Metoder

- BinaryRead

Händelser

- -

Response Objekt

Egenskaper

- Buffer
- CacheControl
- CharSet
- ContentType
- Expires
- ExpiresAbsolute
- IsClientConnected
- PICS
- Status

Kollektioner

- Cookies

Metoder

- AddHeader
- AppendToLog
- BinaryWrite
- Clear
- End
- Flush
- Redirect
- Write

Händelser

-

ObjectContext Objekt

Egenskaper

-

Kollektioner

-

Metoder

- SetComplete
- SetAbort

Händelser

- OnTransactionCommit
- OnTransactionAbort

Application Objekt

Egenskaper

-

Kollektioner

- Contents
- StaticObjects

Metoder

- Lock
- Unlock

Händelser

- OnStart
- OnEnd

Server Objekt

Egenskaper

- ScriptTimeout

Kollektioner

-

Metoder

- CreateObject
- HTMLEncode
- MapPath
- URLEncode

Händelser

-

C Exempelkod i PHP

Denna bilaga visar ett exempel på ett PHP-skript, som kryptrerar ett meddelande. Källan är hemsidan foppa.cse.kau.se/peter/forelasningar/f6/ och utskriften från skriptet ses längst ned på denna sida.

```
<?php
$nyckel = "ADBB02";
$text = "Webbutveckling";
//TripelDES är algoritmen
//CBC är läget
//@ Gör så att vi slipper varningen vid kryptering och dekryptering
//mcrypt_encrypt krypterar
//mcrypt_decrypt dekrypterar
//bin2hex omvandlar från binär till hexadecimalform
//mcrypt_encrypt krypterar nämligen till binärtformat

@$krypteradtext = mcrypt_encrypt(MCRYPT_TripleDES, $nyckel, $text, MCRYPT_M
ODE_CBC);
$hexkrypteradtext = bin2hex($krypteradtext);
echo("<B>$text</B><BR>\n");
echo("<B>$hexkrypteradtext</B><BR>\n");
@$dekrypteradtext = mcrypt_decrypt(MCRYPT_TripleDES, $nyckel, $krypteradtex
t, MCRYPT_MODE_CBC);
echo("<B>$text</B><BR>\n");
echo("<B>$dekrypteradtext</B><BR>\n");
?>
```

Utskriften, i webbläsaren, blir:

Webbutveckling

2bdeb54bc3cf4abc43d08d7b9d8d6cee

Webbutveckling

Webbutveckling

D Exempelkod i ASP

Detta är, till skillnad från PHP-skriptet i bilaga C, ett helt komplett exempel, med HTML-taggar och ASP-kod. Exemplet räknar ut en triangels area, där användaren själv får ange triangelns sidlängder. Resultatet av koden nedan, återfinns efter kodexemplet. Källan är hemsidan www.ivarssons.nu/petter/asp/asp_area_calculating.asp

```
<html><head><title>Ytberäkning, Tentauppgift 1</title></head><body>
<%
    Select Case Request.Form("typ")
            Case "triangel"
                    Call triangel()
            Case "rektangel"
                    Call rektangel()
            Case "cirkel"
                    Call cirkel()
            Case Else
                    If Request.Form("doldis") ="felkoll" Then
                            Call frmIngenVald()
                    Else
                            Call frmStart
                    End IF
    End Select

'--detta körs som default
    Sub frmStart()
%>
            <h1>Ytberäkning</h1><h2>Ange typ av grafisk form</h2>
            <form action = "asp_area_calculating.asp" method = "post" name = "frmBase">
            <input type = "hidden" value = "felkoll" name = "doldis">
            <input type = "radio" name = "typ" value = "triangel">Triangel<br>
            <input type = "radio" name = "typ" value = "rektangel">Rektangel<br>
            <input type = "radio" name = "typ" value = "cirkel">Cirkel<br><br>
            <input type = "submit" value = "Jag har valt">
```

```
            </form>
<%
   End Sub

   ’--detta körs om ingen är vald
   Sub frmIngenVald()
%>
            <h1>Ytberäkning</h1>
            <h2><font color = "red">Du måste ange typ av geometrisk figur</font></h2>
            <form action = "asp_area_calculating.asp" method = "post" name = "frmBase">
            <input type = "hidden" value = "felkoll" name = "doldis">
            <input type = "radio" name = "typ" value = "triangel">Triangel<br>
            <input type = "radio" name = "typ" value = "rektangel">Rektangel<br>
            <input type = "radio" name = "typ" value = "cirkel">Cirkel<br> <br>
            <input type = "submit" value = "Jag har valt">
            </form>
<%
   End Sub

   '--TRIANGEL
   Sub triangel()
   If Request.Form("berakna") = "" Then
%>
            <h2>Beräkna triangels area</h2>
            <form action="asp_area_calculating.asp"method="post"name="frmTriangel">
            <input type = "hidden" name = "typ" value = "triangel">
            <input type = "hidden" name = "berakna" value = "berakna">Triangelns bas
            <input type = "text" name = "bas"><br> Triangelns höjd
            <input type = "text" name = "hojd" ><br><br>
            <input type = "submit" value = "Sänd för beräkning">
            </form>
            <a href = "asp_area_calculating.asp">Kör programmet från början<a>
<%
   Else
```

```
            Response.Write "<h2> Triangelns area är = " & Request.Form("bas") *
            Request.Form("hojd") / 2 & "</h2>"
            Response.Write "<a href=""asp_area_calculating.asp"">Kör programmet
            från början<a>"
            End IF
    End Sub

%>
</body></html>
```

Så här kommer det att se ut om användaren nu har valt att beräkna en triangel.
När sedan värden angivits, trycker användaren på knappen ”sänd för beräkning”.

Beräkna triangels area

Triangelns bas 4

Triangelns höjd 5

Sänd för beräkning

Kör programmet från början

Svaret blir då genererat på en ny sida, som ser ut så här:

Triangelns area är = 10

Kör programmet från början