TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

En jämförelse mellan PHP och C# i .Net

Gisela Kind

Louise Svennberg

EXAMENSARBETE 2009

Grafisk design och webbutveckling

TEKNISKA HÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

A comparison between PHP and C# in .Net

Gisela Kind

Louise Svennberg

Detta examensarbete är utfört vid Tekniska Högskolan i Jönköping inom ämnesområdet Grafisk design och webbutveckling. Arbetet är ett led i den treåriga kandidatutbildningen. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Ämneshandledare: Markus Unger
Metodhandledare och examinator: Anette Karltun

Omfattning: 15 hp (C-nivå)

Datum: 2009-05-10

Arkiveringsnummer:

Postadress:	Besöksadress:	Telefon:
Box 1026	Gjuterigatan 5	036-10 10 00 (vx)
551 11 Jönköping		

Abstract

In this report we compare the two programming languages PHP and C# in .Net. They are both used to create dynamic websites and on the web there's a debate going on about which of these languages you're recommended to use. With this report we want to make ourselves a more scientific base of which of these languages that is the most used language and which languages that are the most suitable for creating dynamic websites. We have two purposes with this report:

1. To compare the usage of the programming languages C# in .Net and PHP at web agencies, advertising agencies and combinded agencies in Sweden today.
2. To get a better insight in the languages partly theoretical and partly through making a practical comparison of how you program in those languages.

We have chosen to use four methods to reach our purposes; interviews, survey, literature study and practical work. To among else find out which of PHP and C# in .Net that are the most common language, at web agencies, advertising agencies and combined agencies in Sweden today, we created a survey which we sent out to 100 companies in five counties. The interviews were done before the survey was designed since they were the foundation for the questions in the survey. All theoretical background is built on our literature study but we also got a little help from our literature study during our practical work. During our practical work we created two similar websites, one in each language to practically to be able to compare how the both codes are written and to make ourselves an opinion about how it feels to program in the both languages.

From the survey we found out that PHP is the most common language at the different types of agencies that we studied. We also found out that the companies who answered the survey believes that PHP still will be the most common language in Sweden in year 2015 of the two studied languages PHP and C# in .Net.

The survey also showed that good qualities in PHP are among others the simplicity, open source and platform independent. Worse qualities in PHP are that it lacks in troubleshooting, has low performance and too many versions.

We think that PHP overall is the easier language to program in, in comparison with C# in .Net since the way you write the code in PHP feels more understandable and logical. Many of the advantages and disadvantages we experienced consists with the agencies opinions regarding PHP, but also with the opinions regarding C# in .Net since it's fast, compiled and a well structured language. Worse qualities in C# in .Net we experienced are that it's own by Microsoft, less spread and has complicated code.

The conclusions of this study is that PHP is the most used programming language at the web agencies, advertising agencies and the combined agencies that this study included, that the both programming languages differ from eachother more than we predicted both theoretical and programmatically and PHP is better suited for less complicated programming while C# in .Net is more suited for advanced programming.

Sammanfattning

I detta examensarbete gör vi en jämförelse mellan de två språken PHP och C# i .Net. Båda används för att skapa dynamiska hemsidor och på Internet pågår en debatt om vilket av dessa språk som man helst ska använda sig av. Vi vill med detta arbete skapa oss en mer vetenskaplig grund till vilket av dessa språk som är det mest använda språket och vilket som är det mest lämpade språket vid skapandet av dynamiska hemsidor. Vi har i detta examensarbete två syften. De är:

1. Att jämföra användningen av de båda programmeringsspråken C# i .Net och PHP på webbyråer, reklambyråer samt kombinerade byråer i Sverige idag.
2. Att få en bättre förståelse för de båda språken dels teoretiskt och dels genom att göra en praktisk jämförelse mellan hur man programmerar i språken.

Vi valde att använda oss av fyra metoder för att uppnå våra syften; intervjuer, enkätundersökning, litteraturstudier samt programmeringslaborationer. För att jämföra användningen av de båda programmeringsspråken PHP och C# i .Net på webbyråer, reklambyråer och kombinerade byråer i Sverige idag, skapade vi en enkät som vi skickade ut till 100 byråer i fem olika kommuner. Intervjuerna genomförde vi innan vi utformade enkäten då de låg som grund för frågorna vi ställde. All teoretisk bakgrund bygger på våra litteraturstudier men vi fick även lite hjälp från litteraturstudierna till våra programmeringslaborationer. Under programmeringslaborationer skapade vi två likadana hemsidor; en i vardera språket för att rent praktiskt kunna jämföra hur koderna skrivs och för att själva skapa oss en uppfattning om hur det känns att programmera i de båda språken.

Av enkäten fick vi reda på att PHP är det mest använda språket på de olika typerna av byråer vi undersökte. Vi fick även reda på att byråerna som svarade på enkäten tror att PHP fortfarande kommer vara det mest använda språket i Sverige år 2015 av PHP och C# i .Net.

Enkäten visade också att bra egenskaper med programmeringsspråket PHP är bland annat att det är enkelt, har öppen källkod och är plattformsoberoende. De sämre egenskaperna i språket är att det har bristfällig felsökning, låg prestanda samt att det finns för många versioner av språket.

Överlag fann vi att PHP är det lättare språket att programmera i jämfört med C# i .Net då sättet man skriver det på känns mer lättförståeligt och mer logiskt i PHP. Många av de fördelar och nackdelar vi upplevde stämmer överens med byråernas åsikter om PHP, men även med åsikterna om C# i .Net då det är snabbt, kompilerat och ett välstrukturerat språk. De sämre egenskaperna med språket upplevde vi är att det är Microsoft-ägt, mindre spritt och har komplicerad kod.

Slutsatserna av studien är att PHP är det mest använda programmeringsspråket på de webbyråer, reklambyråer samt kombinerade byråer som ingår i studien, att de båda programmeringsspråken skiljer sig från varandra i fler avseenden än vi förutspått både teoretiskt och programmeringsmässigt och PHP passar bättre för mindre komplicerad programmering medan C# i .Net lämpar sig bättre för mer avancerad programmering.

Nyckelord

Programmeringsspråk, Enkätundersökning, Programmering, PHP, C# i .Net

Innehållsförteckning

1 Inledning och bakgrund

Som en del av vårt tredje år, kandidatpåbyggnadsåret, inom utbildningen Grafisk design och webbutveckling vid Jönköpings Tekniska Högskola ingår det att skriva ett examensarbete.

Under de två första åren i utbildningen fick vi lära oss programmeringsspråket C# i .Net under våra programmeringskurser. Motiveringen till att vi skulle lära oss språket C# i .Net istället för PHP (som var ett alternativ), var att det skulle bli det mest förekommande språket på marknaden, enligt den programansvarige och programmeringsläraren. Efter att ha sökt arbete i vår bransch som grafisk designer upptäckte vi dessvärre att de flesta reklambyråer, webbyråer och kombinerade byråer istället efterfrågar medarbetare med kunskaper i språket PHP och nästan aldrig i språket C# i .Net som vi har skaffat oss kunskaper i. Vi fick därför uppfattningen att PHP är ett mycket mer aktuellt och mer använt språk än C# i .Net inom vår bransch och vi ville undersöka om så är fallet. Vi beslutade oss för att jämföra användningen av dessa båda språk i examensarbetet.

I dagsläget finns det många programmeringsspråk att använda sig av vid skapandet av dynamiska hemsidor. Många programmeringsspråk både liknar och bygger mycket på varandras standarder. Har man väl lärt sig att använda sig av ett programmeringsspråk blir det betydligt lättare att bygga på sin kunskapsbank med andra språk. Alla programmeringsspråk har naturligtvis sina fördelar och nackdelar när det gäller till exempel uppbyggnad, plattformsberoende och användbarhet. För att orientera sig och hitta rätt i programmeringsspråksdjungeln ansåg vi att Internet borde vara en bra informationskälla.

När vi sökte på Internet fann vi flera olika forum som diskuterar fördelar och nackdelar mellan både språken C# i .Net och PHP samt vilket av dessa språk som anses vara det mest användbara språket vid skapandet av hemsidor. Tyvärr verkar det som att alla argument på dessa forum är byggda utan någon riktig vetenskaplig grund, utan enbart är byggda på insändarnas egna åsikter. Insändarna argumenterar bara för ”sitt eget” språk, som de är vana att programmera i. Från dessa forum kan man inte dra några väl underbyggda slutsatser men man får en vag uppfattning om vilket språk de flesta anser vara det bättre. I detta examensarbete vill vi göra en mer grundlig jämförelse mellan de två programmeringsspråken.

1.1 Syfte och frågeställningar

Det finns två syften med detta examensarbete:

1. Jämföra användningen av de båda programmeringsspråken C# i .Net och PHP på webbyråer, reklambyråer samt kombinerade byråer i Sverige idag.
2. Få en bättre förståelse för de båda språken dels teoretiskt och dels genom att göra en praktisk jämförelse mellan hur man programmerar i de båda språken.

De båda syftena är uppdelade i följande frågeställningar:

Frågeställningar för det första syftet:

1. Vilket är i dagsläget det mest förekommande språket av C# i .Net och PHP på webbyråer, reklambyråer och kombinerade byråer som ingår i studien?
2. Finns det någon skillnad i val av programmeringsspråk beroende på vilken av de utvalda storstadskommunerna byråerna ligger i?
3. Finns det någon skillnad angående vilket språk som används beroende på om det är en webbyrå, reklambyrå eller en kombinerad byrå?
4. Vilka egenskaper i språken anser byråerna vara bättre respektive sämre?
5. Vilket av dessa språk tror byråerna kommer att vara det mest använda språket år 2015 och varför?

Frågeställningar för det andra syftet:

6. Vilka är skillnaderna och likheterna mellan de båda språken; PHP och C# i .Net, teoretiskt och programmeringsmässigt?
7. Vilka fördelar och nackdelar finns med att programmera i C# i .Net respektive PHP enligt våra egna bedömningar?

1.2 Avgränsningar

Det finns totalt tre sätt att använda sig av PHP, vi avgränsar oss bara till ett av dessa; Serverbaserad programmering.

1.3 Disposition

I kapitel två går vi igenom den teoretiska bakgrunden för de båda språken. Där redogör vi för deras grunder så som uppbyggnad, bakgrund och annan bakomliggande fakta. Vi går sedan vidare till att beskriva hur vi gått tillväga under arbetets gång och vilka metoder vi valt för att ge svar åt våra frågeställningar, i kapitel tre. I resultat-delen under kapitel fyra redovisar vi svaren på varje frågeställning i den ordning vi angivit dem. I kapitel fem redovisar vi våra slutsatser och reflekterar över vårt arbete; vad vi fick fram, vad vi kunde ha gjort annorlunda och hur arbetet flutit på.

2 Teoretisk bakgrund

2.1 PHP

PHP är ett ”open source” (öppen källkod) och HTML-inbäddat språk, med andra ord skrivs koden direkt i HTML-koden och är tillgänglig för alla att ta del av. PHP-koden skrivs alltså inte i ett separat dokument, men för att ändå hålla isär de olika språken skrivs all PHP-kod inom PHP-taggar (*<?php...?>*) och varje kodavsats avslutas med semikolon [9]. Med HTML kan man enbart skapa enkla statiska hemsidor, men med språket PHP öppnar man dörrarna till möjligheterna att skapa dynamiska hemsidor med olika funktioner och lättare administrering. Dessa funktioner kan bland annat vara databashantering, filhantering, men man kan även anpassa hemsidor efter årstider, datum och veckodag [6].

PHP är ett tolkat programmeringsspråk och kallas därför för ett skriptspråk. Det som skiljer skriptspråk från renodlade programmeringsspråk, till exempel C# i .Net, är hur exekveringen (körningen) av koden sker. Vid skriptspråk tolkas koden direkt vid exekveringen (*se figur 1*) och eventuella fel i koden upptäcks först efteråt, när man till exempel besöker hemsidan. Vanliga programmeringsspråk måste först omvandlas (kompileras) till maskinkod innan koden kan exekveras (*se figur 2*). Denna process gör det lättare att hitta eventuella fel i koden då all kod kontrolleras under kompileringen istället för under exekveringen [10].

Det finns tre olika huvudsätt att använda sig av PHP; Serverbaserad programmering (*läs mer om detta i avsnitt 2.1.2.1*), radkommando-programmering och/eller klientbaserat grafiskt gränssnitt [10].

2.1.1 Bakgrund

Förkortningen PHP står numera för *Hypertext Preprocessor*. Det sist nämnda ordet i betydelsen (*Preprocessor*) syftar på att PHP omvandlas till HTML innan hemsidan visas i webbläsaren (*se avsnitt 2.1.2.2*). HTML står för *HyperText Markup Language* och är det språk som alla hemsidor är uppbyggda av. Originalbetydelsen av PHP stod för *Personal Home Page* då det först skapades för personlig användning, skräddarsytt för dynamiska hemsidor, av Rasmus Lerdorf. Han skapade språket 1994 för att han ville imponera på dem som besökte hans hemsida [6]. Detta gjorde han genom att på sin hemsida använda sig av bland annat en gästbok, en gästräknare samt andra hemside-element som var ”häftiga” när webben inte var så utvecklad som den är idag [12]. Allteftersom språket växte och började användas mer professionellt ändrades betydelsen för PHP från originalbetydelsen till det numera använda uttrycket [6].

Den PHP-version som används idag ser inte likadan ut som den Rasmus Lerdorf skapade 1994. Språket har utvecklats och skrivits om, det finns idag sex olika versioner av PHP. Det var först när version PHP3, som släpptes i juni 1998, som användandet av PHP började sätta igång på riktigt [10]. För att möta den ständigt växande marknadens behov tog två utvecklare initiativet att helt och hållet tänka om när det gäller hur språket PHP fungerade och var uppbyggt. Resultatet av Zeev Surask och Andi Gutmans arbete blev versionen PHP4. Det var först i denna version av PHP som språket fick ett objektorienterat stöd (*läs mer om objektorientering i avsnitt 2.2.1*).

Detta objektorienteringsstöd ansågs först vara en ganska dålig idé men resulterade i att många vana objektorienteringsanvändare faktiskt lockades över till PHP från deras dåvarande språk med objektorienteringsstöd. Med lanseringen av version PHP4 tog sig språket ett stort steg framåt i sin mognad då versionen erbjöd användarna mer styrka och skalbarhet än vad de tidigare versionerna hade samt erbjöd fler nya egenskaper [13].

Den mest synliga egenskapen med version PHP5 som lanserades den 13 juli 2004 är det extremt förbättrade objektorienterade stödet [15]. Översiktligt sett innehåller den femte versionen annars en rad andra förbättringar och nya egenskaper. När ett språk har kommit så här långt i sin utveckling brukar det oftast börja associeras med strukturen för ett moget programmeringsspråk [13].

Ända sedan den 11 november 2005 har PHP6 varit under utveckling. Det har tagit lång tid men det är en avancerad process då man kan ana vissa attitydförändringar i språket. Stora PHP-utvecklare vill ändra på sättet man skriver koden och på så vis ”tvinga” användarna att skriva snyggare och bättre kod. Ingen vet säkert hur PHP6 kommer att se ut men det man har förstått är att det inte kommer att vara lika bakåtkompatibelt som tidigare versioner [14]. Det betyder att man inte helt kan använda sig av projekt gjorda i tidigare versioner, utan koden måste omjusteras till PHP6. Detta är ganska ovanligt då de flesta program är bakåtkompatibla som till exempel Microsoft Office Word 2007 där man kan använda sig av det gamla formatet *.doc* men det nya formatet *.docx* kan man inte använda sig av i de äldre versionerna [9]. Trots att PHP6 fortfarande är under utveckling så finns det idag en testversion att få tag på om man redan nu vill utforska de nya funktionerna och göra sig bekant med det nya sättet att skriva koden. Vill man få tag på testversionen finns den att hämta på hemsidan ”http://snaps.php.net”.

Från att vara ett nästan helt okänt begrepp har PHP växt på bara några år och har nu lyckats bli en av de populärare serverbaserade skriptspråken. Orsakerna till detta är att språket är snabbt, stabilt och lätt att lära sig. På grund av den snabbt växande framgången upptäcker fler och fler användare styrkan i det språk som är under ständig utveckling världen över [9]. Användare har varierade orsaker till varför de använder sig av just PHP. Alla argument verkar däremot ha en tendens att oftast falla in under fyra olika huvudkategorier; dess praktiska förmåga, dess styrka, alla dess möjligheter samt språkets kostnad [13].

Rent professionellt används PHP av både grafiska designers och programmerare. Båda yrkesrollerna föredrar olika egenskaper i språket. Grafiska designers uppskattar språkets tillgänglighet och bekvämlighet samtidigt som programmerare uppskattar dess flexibilitet och hastighet [10].

2.1.2 Teknisk redovisning

2.1.2.1 *Serverbaserat*

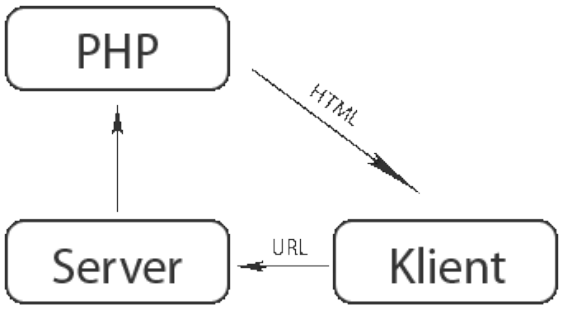
Det första man bör göra innan man börjar programmera i språket PHP, är att installera ett stöd för PHP på webbservern, som man planerar att använda sig av. Har man sedan tidigare ingen webbserver måste man först installera en sådan. Detta stöd på servern krävs för att man som användare ska kunna se och interagera med hemsidan i sitt rätta skick via webbläsaren, exempelvis Internet Explorer. Utan detta PHP-stöd på servern kan man endast se kodraderna man programmerat, utan några grafiska gränssnitt (utseenden) i webbläsaren. Det stödet man installerar på sin webbserver kallas för en applikationsserver och den fungerar som en tolk när PHP-koden på hemsidan körs och omvandlas till HTML-kod (*mer om det i exekveringsavsnittet 2.1.2.2*) [9].

Språkets verkliga styrka är att det fungerar som en applikationsserver. Förutom att den fungerar som en tolk vid exekveringen (körningen) av PHP-koden innehåller applikationsservern flera andra teknologier. Applikationsservern arbetar som ett program som sammanslår flera olika teknologier till ett hopslaget paket. Tre exempel på teknologier, förutom tolken, är ett robust programmeringsspråk, tillgången till databaser för permanent datalagring samt stödet för olika Internetprotokoll som e-post och HTTP (*HyperText Transfer Protocol*) [8]. Teknologin i PHP som handhar tillgången till databaser har stöd för de flesta olika typer av databaser som exempelvis MySQL och Microsoft Access [9].

Med språket PHP kan man utveckla och köra funktioner på flera olika plattformar, alltså operativsystem som Unix, Windows, Macintosh och OS/2. PHP ger även möjligheten för användarna att flytta omkring webbprojekt mellan olika operativsystem utan att koden i princip aldrig behöver ändras [6].

2.1.2.2 *Exekvering*

Med klientbaserade språk menas att allt som utförs på hemsidor, sker enbart på användarens dator och hemsidan hämtas därför inte från en server [6]. Med PHP är det tvärtom, där sker allt via en applikationsserver istället. Det vill säga då klienten vill besöka en hemsida skickas en begäran om hemsidan från klientens dator till en applikationsserver. PHP-delen på servern söker då genom hemside-filerna efter alla existerade PHP-taggar (*<?php...?>*) och kör de kodrader som hittas. Därefter skickas filerna tillbaka till klienten som då har översatts till enbart HTML-kod för att klienten ska kunna besöka hemsidan i sin webbläsare (*se figur 1*) [8].



Figur 1. Hur koden tolkas vid exekveringen [6].

För att visa tydligare hur PHP översätts till HTML under exekveringen visar vi här ett exempel. PHP koden ser ut på följande vis:

```
<html>
  <head>
    <title>PHP-sida</title>
  </head>
  <body>
    <?php $string = 'World!'; ?>
    <h1>Hello, <?php echo $string ?> </h1>
   </body>
</html>
```

Ovanstående exempelkod i PHP använder en funktion (echo) för att skriva ut variabeln $string som innehåller ordet "World!" och vid exekveringen till HTML-kod får man ut nedanstående exempel.

```
<html>
  <head>
    <title>PHP-sida</title>
  </head>
  <body>
       <h1>Hello, World! </h1>
   </body>
</html>
```

Som synes finns ingen PHP-kod i den nedre slutgiltiga HTML-versionen av filen. Inga funktioner finns kvar utan PHP-taggarna har blivit ersatta med HTML-kod och kan därför visas i webbläsaren [8].

2.2 C# i .Net

För att lättare kunna förstå begreppet C# i .Net och hur det är uppbyggt samt vad varje del i produktionen ansvarar för, har vi valt att dela upp detta avsnitt i tre olika underrubriker. De delar som ingår i begreppet C# i .Net är; programmeringsspråket C#, tekniken ASP.Net och systemkomponenten .Net Framework.

.Net Framework är ett utvecklingsverktyg som innehåller programmeringsspråk (C#), serverteknologier (ASP.Net) och utvecklingsmiljöer (programvaror). C# är ett av .Net-språken att programmera hemsidor i. .Net Framework exekverar koden och omvandlar den till HTML och ASP.Net är tekniken som ser till att hemsidan kan visas i en webbläsare då den fungerar som en server.

2.2.1 C# (C sharp)

Anders Hejlsberg, en av världens mest kända utvecklare av programmeringsspråk, värvades till Microsoft år 1996. I sin roll som arkitektchef på Microsoft ledde han en arbetsgrupp som år 2000 presenterade det nya programmeringsspråket C# (språket är alltså Microsoft-ägt). Språket är en stark och modern efterföljare till språket C. Fler programmeringsspråk härstammar från kärnspråket C, som till exempel C++ och Java. Detta medför att dessa fyra språk (C, C#, C++ och Java) är mycket lika i sina koder, element och i sin uppbyggnad. C# innehåller det bästa från flera olika språk då det bygger på Javas förenklingar, C++ flexibilitet och populära mekanismer från Visual Basic [5]. Har man väl lärt sig språket C# får man automatiskt kunskaper i språket C [4]. Programmeringsspråket C# är ett objektorienterat språk som ingår i tekniken ASP.Net och är då även automatiskt en del av systemkomponenten .Net Framework [2]. Med programmeringsspråk utan objektorientering kan det lätt uppstå fel vid ändring av kod. En ändring i koden kan orsaka ett fel på ett annat ställe i koden eftersom alla koder är beroende av varandra. Detta avhjälps med att dela upp koderna i olika klasser samt att skriva funktionerna på var sitt ställe så att de sedan kan anropas från olika håll i projektet. Med objektorienterat språk menas alltså att man gör koderna mindre beroende av varandra och det reducerar även mängden av kod [11]. Till sist kan nämnas att C#-kod skrivs i separata dokument och inte i HTML-koden [5].

Inlärningskurvan för C# är dock inte lika brant som för språken C och C++ då C# har skapats med större enkelhet och lättare användning i åtanke [2]. Exempelvis behöver man inte i språket C# själv hantera tilldelning och borttagning av minne då C# är ett styrt språk där Common Language Runtime (*se avsnitt 2.2.3.1*) sköter dessa uppgifter istället. Däremot har man som C#-användare nästan all den kraft som är tillgänglig för användarna av programmeringsspråket C++ [3]. De klassiska buggarna som lätt uppstår med C++ har man åtgärdat i C# [5].

ECMA (European Computer Manufactures Association) är en organisation som bland annat CLI-standardiserade (*läs mer om CLI i avsnitt 2.2.3.1*) språket C# år 2001. Organisationen uttalar sig på följande vis om C# [5]:

- Tre viktiga faktorer med språket är dess robusthet, varaktighet och programmerarens produktivitet.
- Språket är avsett för utveckling av applikationer, både på stora och små plattformar.

- C#-applikationer är ekonomiska när det gäller processorkapacitet och behov av minne men språket är däremot inte avsett att tävla med kärnspråket C:s prestanda och storlek.
- Det är viktigt att språket har stöd för internationalisering samt att möjligheten att migrera från exempelvis C eller C++ finns.

2.2.2 ASP.Net

ASP.Net är inte ett programmeringsspråk utan en serverbaserad teknik som man använder sig av för att skapa dynamiska hemsidor i. I ASP.Net kan man istället använda flera olika programmeringsspråk för att skapa dynamiska hemsidor och då använder man sig av de redan existerande innovationerna som ingår i .Net Framework [2]. Tekniken ASP.Net är den mest avancerade webbutvecklingsplattform som hittills har skapats och den är nästa generation av ASP (*Active Server Pages*). ASP-.Net har dock omkonstruerats från grunden för att skapa en helt och hållet ny och mer flexibel uppbyggnad för webbutveckling. ASP.Net bygger alltså inte på ASP vilket gör att det inte är bakåtkompatibelt, med andra ord; det som gjorts i ASP.Net kan inte köras i det klassiska ASP [7]. Det som gör ASP.Net så revolutionerande är att tekniken är baserad på Microsofts .Net plattform, det vill säga: .Net Framework [3].

Då ASP.Net är en serverbaserad teknik innebär detta att all den kod som finns på en hemsida körs på en webbserver och inte på användarens (klientens) dator. När webbläsaren, exempelvis Internet Explorer, vill öppna en ASP.Net-fil (.aspx) skickas en förfrågan till servern. Där tar ASP.Net-motorn emot förfrågan och läser av och exekverar koderna i filen (*läs mer om detta i avsnitt 2.2.3.2*). Till sist skickas filen, som då omvandlats till HTML, tillbaka till webbläsaren och hemsidan visas upp [5].

2.2.3 .Net Framework

.Net Framework startades som ett projekt inom Microsoft i slutet av 1990-talet. De tidigare utvecklingsverktygen var helt olika och Microsofts egna mjukvaruprodukter exekverade på olika sätt. Projektet .Net Framework gick ut på att skapa en ny modell för objektorienterad programutveckling och en ny modell för exekvering av mjukvarorna. Det fanns ett genomgående krav i projektet; att exekveringsmodellen skulle anpassas för Internet [5].

.Net Framework är en systemkomponent från företaget Microsoft som innehåller flera stycken underkomponenter; programmeringsspråk, serverteknologier och klientteknologier samt utvecklingsmiljöer [7].

- Exempel på programmeringsspråk som .Net Framework innehåller [2]:
  - C#
  - Visual Basic
  - JavaScript
- Exempel på server- och klientteknologier som .Net Framework innehåller [7]:
  - ASP.Net
  - Windows Forms (Windows skrivbordslösningar)
  - Compact Framework (PDA / Mobiltelefonslösningar)
- Exempel på utvecklingsmiljöer som .Net Framework innehåller [7]:
  - Visual Studio .NET
  - Visual Web Developer

Det finns en inlärningströskel som man måste passera om man väljer att lära sig ett programmeringsspråk som ingår i .Net Framework. Denna inlärningströskel är olika hög beroende på hur väl språket passar .Net Framework. C# är avsiktligt designat just för att passa .Net Framework och är sannolikt det mest ändamålsenliga språket [5].

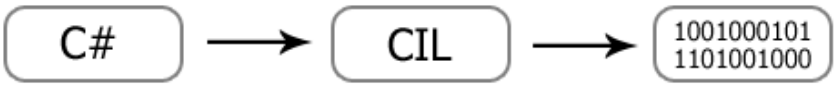
2.2.3.1 *Common Language Infrastructure (CLI)*

CLI ingår i .Net Framework och är ett samlingsnamn för fyra olika tekniker [5]:

- **Common Type System** (CTS) är tekniken som håller reda på de olika typerna, exempelvis klasser, och deras egenskaper. Denna del av CLI är den som utvecklare har störst intresse för.
- **Common Language Specification** (CLS) är en standard för vilka språkmekanismer som är obligatoriska respektive valfria att ingå i ett språk.
- **Common Intermediate Language** (CIL) är mellanformatet som skapas vid kompileringen från ursprungsspråket, innan mellanformatet kompileras till maskinkod. Det får inte finnas kvar några spår av ursprungsspråket i dessa mellanformatsfiler, vilket gör att de kan användas av alla, oberoende av vilket programmeringsspråk som var ursprungsspråket.
- **Virtual Execution System** (VES) är exekveringsmiljön där skräpsamling samt kompilering till maskinkod sker. VES bestämmer även hur klasserna i koden ska laddas. För programutvecklare spelar skräpsamlingsfunktionen i VES en avsevärt stor roll. Funktionen raderar nämligen automatiskt objekt, vid exekveringen, som inte används vilket medför att utvecklare själva inte behöver hålla reda på och radera objekt manuellt.

2.2.3.2 *Exekvering*

I .Net Framework ingår även Common Language Runtime (CLR) som skapar en miljö för körning av kod som är skriven i ett .Net-kompatibelt programmeringsspråk. Det spelar ingen roll vilket språk som används, bara de följer standarden CLS. Koden skriven i valfritt språk kompileras först till mellanspråket CIL. Detta mellanspråk är inte exekverbart utan koden måste kompileras en gång till, denna gång till maskinkod för att kunna exekveras (*se figur 2*). Kompileringen till maskinkod sker vid programstart. Först när koden kompilerats till maskinkod utförs exekveringen där maskinkoden omvandlas till ren HTML-kod [5].



Figur 2. Kompilering från C# till maskinkod via mellanspråket CIL.

En av de största fördelarna med .Net Framework är att medan klassiska tekniken ASP begränsade utvecklare till skriptspråk (med dessas begränsningar), låter ASP.Net utvecklarna att arbeta med valfritt .Net-kompatibelt programmeringsspråk. Detta betyder att den kod man skriver i ASP.Net kompileras för bättre prestanda och man kan utnyttja de avancerade språkegenskaperna till fullo [3].

.NET Frameworks nyckelord [3]:

- Lättare och snabbare programmering
- Reducerad mängd av kod
- Större klassbibliotek
- Bättre stöd för utvecklingsverktyg

3 Metod och genomförande

Från det att vi började med examensarbetet till det att vi lämnade in rapport tog det sammanlagt tio veckor. Vi valde att arbeta med de olika delmomenten parallellt med varandra för att på vis få variation i arbetet och för att ingen tid skulle gå till spillo.

3.1 Metodval

För att kunna svara på våra frågeställningar och för att uppnå vårt syfte har vi valt följande metoder; intervju, enkät, litteraturstudier samt programmeringslaborationer. Intervju och enkät valdes som metoder för att kunna svara på frågeställningarna kopplade till det första syftet medans litteraturstudier och programmeringen har genomförts för att kunna svara på frågeställningarna kopplade till det andra syftet.

3.1.1 Intervju

Vi började med att utforma frågorna till intervjuguiden (*se bilaga 2*). Efter att intervjuguiden diskuterats med vår metodhandledare utförde vi intervjuer på fyra olika webbyråer och reklambyråer i Jönköping och Borås; två byråer i vardera stad. Anledningen till att dessa städer valdes är på grund av att vi båda har personliga anknytningar till dem. Intervjuerna genomfördes för att vi skulle få tips och idéer, om fler frågor som vi kunde ställa samt fler förslag på svarsalternativ att ge till enkäten.

3.1.2 Enkät

Denna metod valdes för att kunna besvara frågeställningarna som är knutna till det första huvudsyftet då dessa frågeställningar baserar sig helt på byråernas användning och åsikter om programmeringsspråken. Enkäten (*se bilaga 1*) utformades efter att intervjuerna genomförts och vi valde att beröra de områden, som diskuterades under intervjuerna, som kändes relevanta för enkäten. Enkäten skickades ut till de systematiskt utvalda webbyråerna, reklambyråerna och de kombinerade byråerna i Sveriges största kommuner; Stockholm, Göteborg, Malmö, Uppsala och Linköping [1]. Vi valde att göra ett systematiskt urval i respektive kommun och den skickades ut till 100 utvalda byråer; 20 olika byråer per kommun.

Vi använde oss av sökmotorn ”http://www.hitta.se” för att göra vårt urval. Söker man på ”reklambyrå” utan att specificera kommun får man 2000 träffar vilket är det totala antalet registrerade reklambyråer i Sverige på ”http://www.hitta.se”. Söker man istället på ”webbyrå” får man 1572 träffar. Av totalt 3572 byråer valde vi som sagt systematiskt ut 100 byråer av dessa till att ingå i vår studie.

Vi var intresserade av 20 byråer i varje kommun; tio reklambyråer och tio webbyråer. Det träffantal vi fick dividerade vi med talet tio och resultatet av divisionen ligger till grund för var n:te byrå vi valde till vår studie. I Stockholm finns 1160 reklambyråer registrerade och vi tog då alltså var 116:e byrå i träfflistan. Antalet registrerade webbyråer i samma kommun är 412 stycken, då resultatet av divisionen blev 41,2 var vi tvungna att avrunda och detta resulterade i att vi valde var 41:e webbyrå. Sökproceduren upprepades för varje kommun.

Observera att vi inte sökte på de kombinerade byråerna då detta sökord inte gav några träffar. Från resultatet av divisionen fick vi däremot kontakt med flera byråer som ansåg sig vara kombinerade byråer genom att de fanns registrerade under exempelvis ”webbyrå”. Vi vill poängtera att eftersom vi valde byråer registrerade som antingen ”reklambyrå” eller ”webbyrå” hade vi ingen uppfattning om vilka av dessa som programmerar på företaget och inte.

3.1.3 Litteraturstudier

Parallellt med hela examensarbetets gång genomförde vi litteraturstudier om de två olika språken; PHP och C# i .Net. Detta skedde främst genom besök på biblioteken i både Borås och Jönköping. Vi hittade även information från flera olika Internetsidor. Informationen vi samlade från dessa hemsidor har vi granskat, sett kritiskt på och jämfört med informationen vi fått från de böcker vi studerade. Vi studerade flera olika böcker om respektive språk för att öka reliabiliteten i vårt examensarbete. Litteraturstudier var ett naturligt metodval för att kunna besvara delar av det andra huvudsyftet, främst den teoretiska bakgrunden men det låg även som grund för programmeringsarbetet vi utförde.

3.1.4 Programmeringsarbete – Skapande av hemsidor

För att kunna tillämpa våra teoretiskt införskaffade kunskaper rent praktiskt samt för att besvara frågeställningarna under andra huvudsyftet skapade vi två hemsidor; en i vardera språk. Hemsidorna gjordes för att vi skulle kunna se skillnader och likheter mellan språken, men även för att vi själva skulle få en uppfattning om eventuella fördelar och nackdelar samt få en uppfattning om vilket språk vi ansåg vara mest lättanvänt. Hemsidorna har en enkel uppbyggnad som bland annat innehåller ett inloggningssystem och en nyhetsfunktion. Programvaran vi använde oss av för att skapa hemsidan i C# var Visual Studio 2008 och för PHP använde vi oss av Adobe Dreamweaver. Hemsidorna som vi skapade gjordes enbart för vårt eget bruk och vi hade på så vis inte någon kravspecifikation. Detta medförde att vi inte hade så mycket att ta hänsyn till, i jämförelse med om vi hade arbetat mot ett företag.

4 Resultat och analys

I redovisningen av resultatet har vi valt att besvara våra frågeställningar i den ordning vi angav dem. För att förtydliga vilken frågeställning vardera avsnitt handlar om har vi återgivit dem i början av varje stycke. Vi har valt att i det här kapitlet inte redovisa samtliga frågor från enkäten då alla frågor inte är av relevans till frågeställningarna för första huvudsyftet.

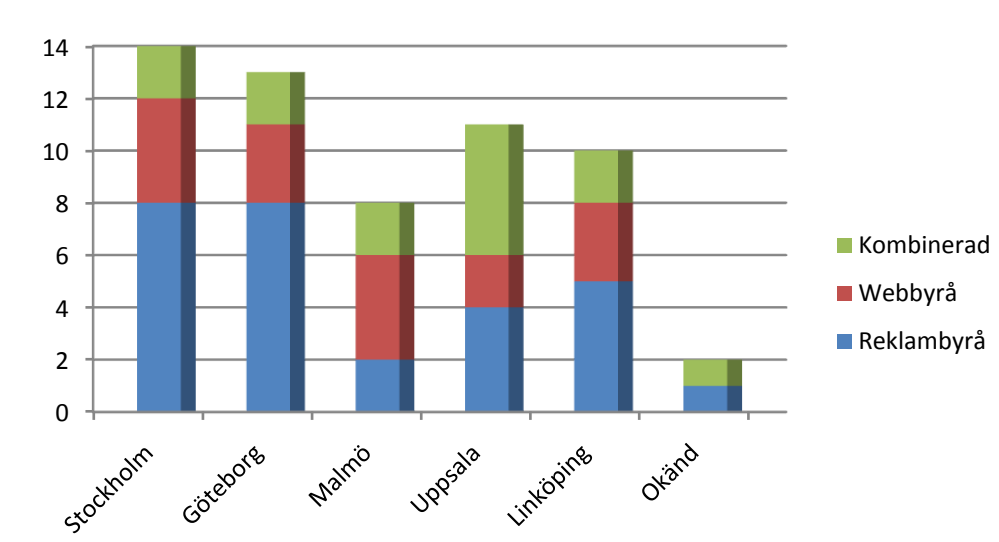
4.1 Huvudsyfte 1

Att jämföra användningen av de båda programmeringsspråken C# i .Net och PHP på webbyråer, reklambyråer samt kombinerade byråer i Sverige idag.

4.1.1 Frågeställning 1

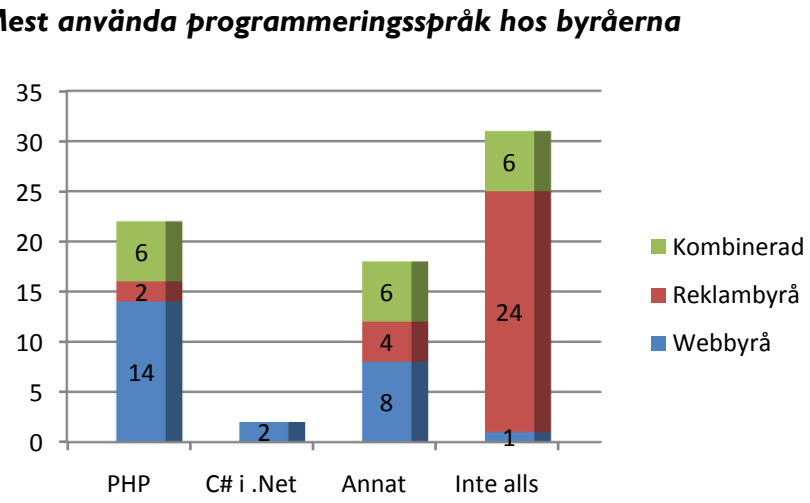
Vilket är i dagsläget det mest förekommande språket av C# i .Net och PHP på webbyråer, reklambyråer och kombinerade byråer som ingår i studien?

Av de totalt 100 systematiskt utvalda byråer i Sveriges fem största kommuner, svarade totalt 58 byråer. Svaren vi fick in i vår databas var av bra spridning då vi fick relativt lika många svar från vardera kommun (*se figur 3*). Det tog sammanlagt tre veckor och två påminnelsemejl att få in dessa 58 svar från byråerna.



Figur 3. Svarsfrekvensen för respektive kommun (n = 58).

Staplarna visar förutom svarsfrekvensen per kommun även antal svar från respektive typ av byrå per kommun. Stapeln Okänd beror på att just de två byråerna inte angav sitt företagsnamn när de fyllde i enkäten. Detta resulterar i att vi inte vet vilken kommun de representerar. Företagsnamnen hjälpte nämligen oss att placera byråerna i de olika kommunerna, då vi inte hade någon fråga i enkäten om geografisk plats. Som synes fick vi flest svar från Stockholm och minst antal svar från Malmö.

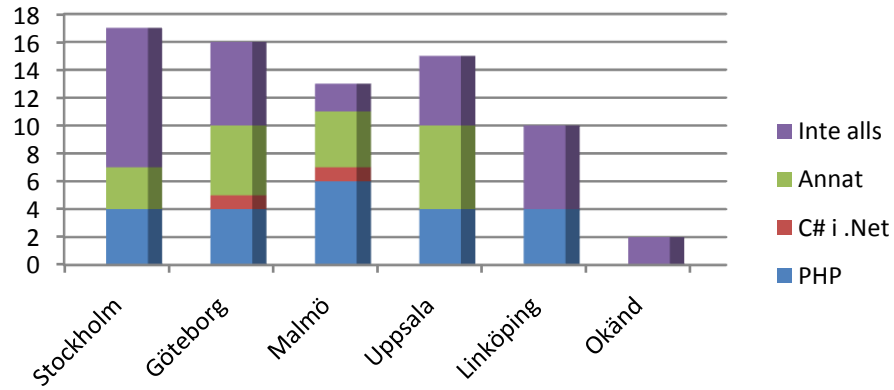


Figur 4. Användningen av PHP och C# i .Net hos de olika byråerna (n = 58).

Av ovanstående diagram ser man klart och tydligt att PHP är det mest använda programmeringsspråket på de olika typerna av byråer som vi undersökt. Endast två byråer har angivit att de använder sig av C# i .Net. Detta är en markant skillnad mot de totalt 22 byråer som använder sig av PHP. Observera att en byrå kan ha angivit i enkäten att de använder sig av både PHP och C# i .Net. Frågan i enkäten var ”Vilket/vilka programmeringsspråk använder ni er av idag?”. Byråerna hade till denna fråga utöver PHP och C# i .Net som svarsalternativ även ”Annat språk” och ”Vi programmerar inte alls i företaget”. Vi ser en tydlig tendens mot att PHP är det mest använda språket på svenska byråer idag.

4.1.2 Frågeställning 2

Finns det någon skillnad i val av programmeringsspråk beroende på vilken av de utvalda storstadskommunerna byråerna ligger i?

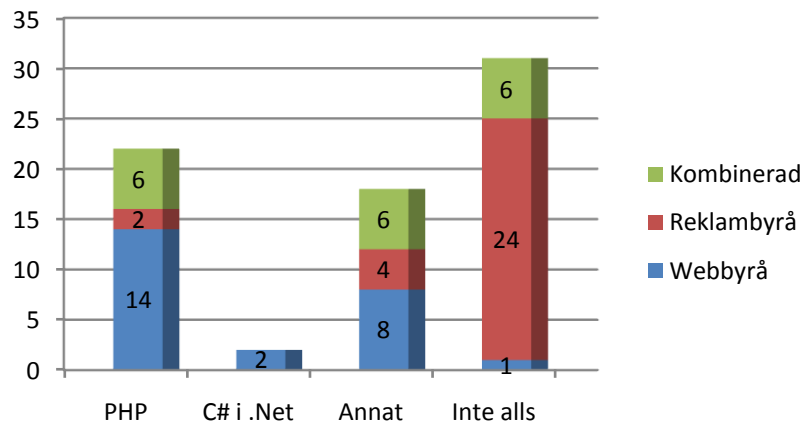


Figur 5. Användningen av de olika i språken i vardera kommun (n = 58).

Stapeln Okänd i diagrammet finns där av samma anledning som beskrevs i första frågeställningen. Som man kan se är C# i .Net det minst använda språket hos de byråer som vi har fått svar från. Diagrammet visar även att majoriteten av byråerna som svarat angett att de inte programmerar alls. Av 58 svar är det hela 31 stycken som angav det. En byrå kan ha registrerat att de använder sig av fler än ett programmeringsspråk på sitt företag. Vi ser att i Linköping använder de svarande byråerna bara sig av PHP eller inget programmeringsspråk alls. Dock har endast tio av totalt 20 utvalda byråer i den kommunen svarat, vi vet alltså inte vad de tio övriga byråerna använder sig av. I Stockholm är fördelningen mellan PHP och andra programmeringsspråk jämn. Diagrammet visar att de byråer som svarat att de använder sig av C# i .Net har sin verksamhet i Göteborg och Malmö. Överlag ser man utifrån diagrammet ingen tendens till någon större lokal skillnad mellan vilket programmeringsspråk som används på byråerna. Då vår svarsfrekvens på denna fråga ändå är 58 procent kan detta peka på att val av programmeringsspråk faktiskt inte beror på kommun.

4.1.3 Frågeställning 3

Finns det någon skillnad angående vilket språk som används beroende på om det är en webbyrå, reklambyrå eller en kombinerad byrå?



Figur 6. Användning av programmeringsspråk hos de olika byråerna (n = 58).

Detta diagram är baserat på samma fråga som första frågeställningen. I det här diagrammet ser vi däremot alla svarsalternativ. Observera att i detta diagram kan en och samma byrå ha kryssat i ett eller flera programmeringsspråk. Exempelvis kan en byrå ha kryssat i att de använder sig av både PHP och annat språk. Stapeln Annat innehåller flera olika programmeringsspråk som byråerna angivit att de använder.

Av diagrammet kan man dra slutsatsen att flertalet reklambyråer svarat att de inte programmerar alls på företaget, de som däremot svarat att de programmerar har angett att de använder sig av PHP eller annat språk. Vi ser att alla webbyråer förutom en programmerar och att PHP är det mest använda på just den byråtypen.

De kombinerade byråerna är väldigt spridda över de olika staplarna. Med andra ord verkar det som om de kombinerade byråerna inte använder sig av något fast programmeringsspråk, utan att valet av programmeringsspråk istället är varierande. Trots att Annat-stapeln innehåller flera olika programmeringsspråk väger PHP-stapeln upp både Annat och C# i .Net vilket kan tyda på att det är väldigt använt språk på byråer i Sverige.

De som svarat att de inte programmerar alls behövde inte medverka i enkäten längre och är därför inte medräknade i svarsfrekvensen i resterande frågeställningar. Då det är 31 byråer som inte programmerar kommer antal möjliga svarande i fortsättningen att vara 27 byråer.

4.1.4 Frågeställning 4

Vilka egenskaper i språken anser byråerna vara bättre respektive sämre?

Nedanstående egenskaper är rangordnade efter hur många byråer som angav just den egenskapen som en anledning. Siffran inom parentesen anger antalet byråer som i sina enkätsvar lyfte fram den egenskapen. Observera att en byrå kan ha angett fler egenskaper på denna öppna fråga. Det här systemet med antal svar inom parenteser gäller för alla listor i resterande delar av rapporten.

4.1.4.1 *PHP*

Bra egenskaper (n = 23 bf = 4):

- *Enkelt (9)* - Språkets syntax är enkel och inlärningskurvan är låg. Det är även lätt att förstå sig på funktionerna.
- *Open source (5)* - Öppen källkod.
- *Webbservrar (5)* - De flesta webbservrarna har stöd för PHP, därför är sidor skapta i PHP lätta att driftsätta, med andra ord: publicera på internet.
- *Välspritt (4)* - Språket är populärt och används av många.
- *Gratis (3)* - PHP är gratis att få tag på, att installera och att använda sig av.
- *Produktivt (3)* - Både tidsbesparande, resurssnålt och ekonomiskt för kunder.
- *Inbyggd funktionalitet (3)* - Innehåller många bra funktioner.
- *Dokumentation (3)* - PHP har bra dokumentation, även online.
- *Kraftfullt (2)* - Kan utföra mycket och kräver inte mycket prestanda.
- *Skalbart (2)* - Passar till både små och stora produktioner.
- *Exempelkod på Internet (2)* - Det finns mycket bra hjälp att få på Internet.
- *Stora möjligheter (2)* - Begränsningarna för vad PHP kan göra är få.
- *Buggfritt (1)* - En buggfri miljö innebär att koden som är skriven fungerar felfritt utan oväntade fel som inom programmering kallas buggar.
- *Dynamisk typning (1)* - Sättet man skriver koden på.
- *Ständig utveckling (1)* - Då det är open source kan alla utveckla det lokalt.
- *Kreativt (1)* - Man kan göra mycket med språket.
- *Plattformsoberoende (1)* - Koden kan skrivas på alla operativsystem.
- *Integration mot MySQL (1)* - PHP arbetar optimalt mot MySQL-databaser.
- *Kontrollerbart (1)* - Vet man vad man gör är det lätt att ha kontroll i språket.
- *Stabilt (1)* - Det håller i alla lägen.

Sämre egenskaper (n = 15 bf = 12):

- *Lite komplext (2)* - Koden kan vara svår.
- *Lösningarna (2)* - Lösningarna blir inte alltid så snygga med PHP.
- *Ej kompilerat (1)* - Koden kompileras inte till skillnad från C#-kod.
- *Utvecklingsmiljö (1)* - Det finns ingen nämnd standardiserad programvara.
- *Teckenkodning (1)* - Kan bli problem med hur man skriver koderna.
- *Dynamisk typning (1)* - Sättet man skriver koden på.
- *Ingen felsökning (1)* - Svårt att veta vart och varför problem uppstår i koden.
- *Växt för mycket (1)* - Det har blivit för stort och populärt.
- *Prestanda (1)* - Hur mycket utrymme och kapacitet PHP kräver.
- *Högre inlärningströskel än ASP (1)* - Svårare att lära sig PHP än ASP.
- *Inte UTF-8-vänligt (1)* - Problem med att visa å, ä och ö på hemsidor.
- *Mycket kod (1)* - Ibland kan det behövas mycket kod för enkla funktioner.
- *För många versioner (1)* - Problem vid uppgradering till nya versioner.
- *Utvecklingsverktyg (1)* - PHP ligger inte riktigt i fas med ASP/.Net-språk angående utvecklingsverktyg.

Som vi ser fick vi in många åsikter om PHP, båda bra och sämre, vilket kan tyda på att PHP är ett välkänt begrepp hos byråerna som ingick i vår studie. Av 27 byråer valde fyra byråer att inte svara på de bra egenskaperna och totalt tolv som inte angav några sämre egenskaper. Det höga bortfallet på sämre egenskaper kan ha flera orsaker, till exempel att man inte anser att språket har några sämre egenskaper eller att man inte svarat på grund av lathet.

4.1.4.2 *C# i .Net*

Bra egenskaper (n = 7 bf = 20):

- *Microsoft-ägt (2)* - Det blir ett beroende mellan C# i .Net och Microsoft, men detta beroende gör att språket integrerar utmärkt med Windows.
- *Snabbt (2)* - Funktioner utförs snabbt och gör inte datorn långsam.
- *Kompilerat (1)* - Språket omvandlas till maskinkod innan exekveringen.
- *Utvecklingsmiljö (1)* - Standardiserad och bra programvara till C# i .Net.
- *Välstrukturerat språk (1)* - Språket har en logisk och genomtänkt struktur.

Sämre egenskaper (n = 8 bf = 19):

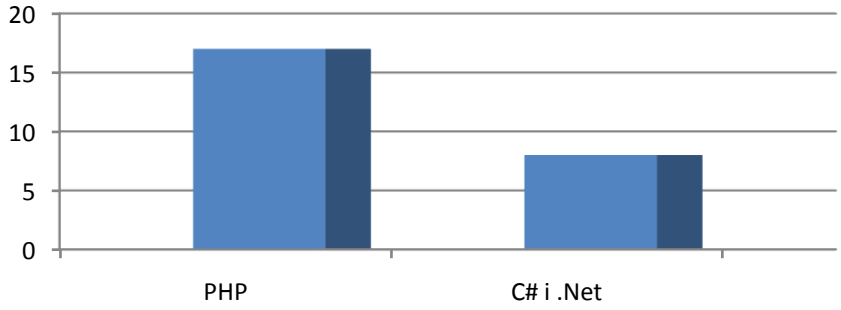
- *Microsoft-ägt (3)* - Att språket är Microsoft-ägt är en avgörande faktor. Detta kan leda till att man tvingas använda deras produkter.
- *Mindre spritt (1)* - I dagsläget används språket inte av så många användare.
- *Långsammare vid webbfunktionalitet än PHP (1)* - Kräver mer kapacitet.
- *Webbhotell (1)* - C# i .Net stöds inte alltid av alla webbhotell.
- *Komplex exekvering (1)* - Vid exekvering omvandlas koden till ett mellanspråk och sedan till maskinkod.
- *Sämre klientbaserad kod (1)* - Kod som utförs i klientens webbläsare är sämre.
- *Omständligt (1)* - Lite väl avancerat ibland.

Åsikterna om C# i .Net var väldigt få i jämförelse med åsikterna om PHP. Av totalt 27 som hade möjlighet att svara var det 20 som inte angav några bra egenskaper i språket och 19 som inte angav några sämre egenskaper. Det är väldigt höga svarsbortfall och då inte allt borde bero på lathet kan man istället anta att svarsbortfallet beror på att C# i .Net inte är ett välkänt och använt språk (*se figur 6*), detta kan resultera i att personer saknar uppfattningar om språket.

Observera att åsikterna är delade om det är en bra eller en sämre egenskap att språket är Microsoft-ägt. Man kan anta att egenskapen är en avgörande faktor till om man vill använda sig av språket eller inte.

4.1.5 Frågeställning 5

Vilket av dessa språk tror byråerna kommer att vara det mest använda språket år 2015 och varför?



Figur 7. Språket som byråerna tror kommer användas mest år 2015 (n = 25 bf = 2).

Frågan i enkäten var ”Vilket av följande språk tror ni kommer vara störst år 2015 i Sverige?” och alternativen till frågan var PHP och C# i .Net. Staplarna ovan visar att majoriteten av de svarande byråerna tror att PHP kommer vara det mest använda programmeringsspråket i Sverige år 2015. PHP fick mer än dubbelt så många röster som språket C# i .Net. Totalt sett trodde 17 av de 25 byråer som svarade, att PHP kommer vara ledande medan endast åtta byråer tror på att C# i .Net kommer vara det mest använda programmeringsspråket år 2015.

Efter att företagen kryssat i vilket språk de trodde skulle vara det mest använda år 2015 i Sverige, fick de svara på följdfrågan ”Varför tror ni det?”. På denna öppna fråga angavs följande anledningar som svar för språket PHP (n = 14 bf = 3):

- Språket är open source, det har alltså öppen källkod (5)
- Det är lätt att lära sig (3)
- PHP är gratis att använda sig av (2)
- Det går snabbt att utföra mycket (2)
- Kräver ingen dyr serverlösning (1)
- Går lätt att utveckla lokalt (1)
- De flesta ungdomar föredrar PHP (1)
- Det är smidigt (1)
- Språket är redan idag populärt (1)
- PHP utvecklas för alla (1)
- Passar dagens fria tänkande (1)
- Plattformsoberoende (1)

Att öppen källkod är den mest frekventa omnämnda anledningen kan bero på att de inte vill känna sig låsta till vissa operativsystem eller programvaror. I listan finns även fler anledningar som styrker detta, exempelvis plattformsoberoende och att det passar dagens fria tänkande. Då byråerna som trodde på PHP var 17 stycken, är det bara det antal som kunnat svara på denna fråga.

Följande anledningar angav byråerna till varför de tror att språket C# i .Net kommer att vara det mest använda språket i Sverige år 2015 (n = 5 bf = 3):

- C# i .Net är ett Microsoft-ägt språk vilket gör det stort då ett starkt företag ligger bakom det (2)
- Fler utbildas idag i språket (1)
- Det finns en stor marknad för C# i .Net (1)
- Det finns stöd för realtidsprogrammering med gemensamma resurser (1)
- Hört av flera programmerare att det kommer att vara störst i framtiden (1)

Av åtta byråer som trodde på C# i .Net var det endast fem som angav anledningar till varför de trodde det. Återigen hamnar Microsoft-ägt högst upp i listan, precis som i avsnitt 4.1.4.2.

4.2 Huvudsyfte 2

Att få en bättre förståelse för de båda språken dels teoretiskt och dels genom att göra en praktisk jämförelse mellan hur man programmerar i de båda språken.

4.2.1 Frågeställning 6

Vilka är skillnaderna och likheterna mellan de båda språken; PHP och C# i .Net, teoretiskt och programmeringsmässigt?

4.2.1.1 *Teoretiska skillnader och likheter mellan PHP och C# i .Net*

De båda språken är programmeringsspråk men då PHP exekveras på ett annat sätt än ett traditionellt programmeringsspråk som C# i .Net, kallar man det istället för ett skriptspråk. Exekveringen i PHP utförs snabbt då det bara består av ett steg (PHP-koden exekveras till HTML), medan C# i .Net har två steg innan koden ens kan exekveras till HTML (från C# till ett mellanformat och sedan till maskinkod).

Medan C# i .Net är ägt av Microsoft och låst till deras produkter är PHP tillgängligt för alla på samtliga operativsystem. PHP är även fritt för användarna att själva utveckla språket lokalt då det har öppen källkod.

PHP är skräddarsytt för att kunna skapa dynamiska hemsidor. Det är inte C# i .Net eftersom det även kan användas till programmering med andra ändamål som till exempel program till mobiltelefoner.

En annan väsentlig skillnad är var koden skrivs i de olika språken. När man programmerar i PHP skrivs koden i PHP-taggar direkt i HTML-koden. Programmerar man istället i C# i .Net skrivs koden i ett separat dokument och beblandas på så vis aldrig med HTML-koden.

Något som stämmer för båda språken är att de är serverbaserade och de fungerar därför som en server. Besökarens dator belastas på så vis inte när allt på hemsidan sker via en server.

Trots att PHP inte hade stöd för objektorientering från början har språket detta stöd idag. C# i .Net hade däremot stödet redan när språket lanserades. I vilket fall har PHP och C# i .Net objektorienteringsstödet idag vilket stärker de båda språken då mängden av kod reduceras.

Det finns som synes flera skillnader och likheter mellan de båda språken, vilket gör det svårt att veta vilket programmeringsspråk som är att föredra.

4.2.1.2 *Programmeringsmässiga skillnader och likheter mellan PHP och C# i .Net*

Vi har skapat en hemsida i respektive språk som innehåller samma funktioner och grafiska gränssnitt. Här nedan kommer vi att redovisa de olika funktionerna. För varje funktion redogör vi hur koden skrivs, först för språket PHP och därefter för C# i .Net. Då hemsidorna är enkla utseendemässigt och inte säger något om arbetet bakom, har vi valt att inte redovisa hur dessa ser ut då vi finner det orelevant.

1 Ansluta till databasen

Anslutningen är det första steget till alla händelser som ska kommunicera med en databas; som att hämta, lägga till eller radera data. För språket PHP använde vi oss av en MySQL-databas som är den rekommenderade databasen till det språket. Vid programmeringen med C# i .Net använde vi oss av en Microsoft Access-databas på grund av att vi är vana att arbeta med denna kombination då vi fått lära oss den i våra olika programmeringskurser.

I de kommande exempelkoderna kommer vi inte att inkludera koden för hur man ansluter i början av alla funktioner då vi inte vill upprepa alldeles för mycket kod.

PHP

```
$anslutning = mysql_connect("localhost", "louise");
```

`$anslutning` är namnet på vår anslutning till servern, varje gång namnet anropas körs koden den innehåller. I koden används den inbyggda funktionen `mysql_connect()` som öppnar servern `localhost` där användarnamnet är `louise`.

```
mysql_select_db("exjobb_db", $anslutning);
```

Här hämtar vi databasen `exjobb_db` via vår anslutning `$anslutning` med hjälp av den inbyggda funktionen `mysql_select_db()`.

Sammanfattning av kod:

```
$anslutning = mysql_connect("localhost", "louise");
mysql_select_db("exjobb_db", $anslutning);
```

C# i .Net

```
String databas = Server.MapPath("./App_Data/Exjobb.mdb");
```

För att kunna lokalisera databasen behöver vi dess sökväg, vilken skrivs inom parenteserna i `Server.MapPath()`. Vi har i detta fall valt att lägga sökvägen i variabeln `databas` för att lättare anropa den.

```
OleDbConnection anslutning = new OleDbConnection();
```

```
anslutning.ConnectionString = "Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + databas;
```

Här skapar vi en anslutning till databasen och den döps till anslutning. Genom anslutningen anropas sedan sökvägen databas, som vi tidigare definierat, tillsammans med Provider som alltid krävs för att kunna ansluta till en databas i C# i .Net.

```
anslutning.Open();
anslutning.Close();
```

Ovanstående funktioner används för att öppna och stänga anslutningen. Kod som ska köras medan anslutningen är öppen ska skrivas mellan dessa två rader. För att databasen ska hinna utföra många processer ska man ha den öppen så kort tid som möjligt.

Sammanfattning av kod:

```
String databas = Server.MapPath("./App_Data/Exjobb.mdb");
OleDbConnection anslutning = new OleDbConnection();
anslutning.ConnectionString = "Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + databas;
anslutning.Open();
anslutning.Close();
```

Analys:
Som synes används det här mycket mer kod i C# i .Net jämfört med PHP, för att skapa en anslutning till databasen. En skillnad är att man i C# i .Net måste öppna och stänga anslutningen manuellt vilket man inte behöver göra i PHP. Dessutom är sökvägen till databasen betydligt kortare i PHP än C# i .Net.

2 Hämta data från databasen

PHP

```
$resultat = mysql_query("SELECT id, rubrik, innehall, datum FROM Nyheter");
```

För att hämta ut data ur databasen använder vi oss av en SQL-fråga i funktionen mysql_query(). I SQL-frågan hämtar vi ut kolumnerna id, rubrik, innehall, datum från tabellen Nyheter (SELECT kolumn FROM tabell). Resultatet av SQL-frågan, i detta fall alla nyheter som finns i databasen, läggs i variabeln $resultat.

```
while($rad = mysql_fetch_array($resultat))
{
    echo . $rad['rubrik'] . ” ” . $rad['datum'] . ” ” . $rad['innehall'];
}
```

Vi börjar med att poängtera måsvingarna { } och dess funktion. De öppnar och stänger funktioner. Det är väldigt viktigt att det alltid finns en vinge för att öppna och en för att stänga funktionen, då fel annars kan uppstå om man glömmer det. Hittills har vi hämtat data från databasen. Nästa steg är att få ut datan ur $resultat och att visa nyheterna på hemsidan. mysql_fetch_array() lägger den information som finns i $resultat i en lista (array) som skrivs ut i olika rader, med hjälp av variabeln $rad. while-funktionen upprepas så länge det finns poster kvar i $resultat.

Vi har på hemsidan skapat tre fält; ett fält för varje kolumn som vi vill visa. Informationen i kolumnerna sorterar sig efter fälten. På hemsidan ser det exempelvis ut så här:

Ny hemsida 2009-04-14 Nu lanserar vår nya hemsida med nyhetsfunktion.
[rubrik] [datum] [innehall]

Sammanfattning av kod:

```
$resultat = mysql_query("SELECT id, rubrik, innehall, datum FROM Nyheter");
while($rad = mysql_fetch_array($resultat))
{
    echo . $rad['rubrik'] . ” ” . $rad['datum'] . ” ” . $rad['innehall'];
}
```

C# i .Net

```
string Hamta = "SELECT id, rubrik, innehall, datum FROM Nyheter ";
```

För att hämta ut posterna från databasen används SQL-frågor även i C#. På grund av att vi använder samma sätt för att hämta ut datan från databasen, ser SQL-frågan likadan ut. Resultatet av frågan sparas däremot i variabeln med namnet Hamta.

```
OleDbDataAdapter adapter = new OleDbDataAdapter(Hamta, anslutning);
```

När vi vill koppla ihop vår SQL-fråga med vår anslutning använder vi oss av en adapter som sammanför de värden som anges inom dess parenteser, i vårt fall; Hamta och anslutning. Informationen från databasen sparas i adapter och kan nås på olika sätt.

```
anslutning.Open();
DataSet ds = new DataSet();
adapter.Fill(ds);
```

Vi väljer här att läsa informationen som finns i adapter med hjälp av en DataSet. Adaptern sköter all kommunikation mellan databasen och SQL-frågan, men som en del i processen, att kunna läsa ut datan på hemsidan, använder vi alltså en DataSet. I koden ovan döper vi vår DataSet till ds. För att föra över informationen från adapter till ds används metoden Fill().

```
DataLista.DataSource = ds;
DataLista.DataBind();
anslutning.Close();
```

DataLista är den kontroll som vi valt för att läsa ut datan på hemsidan. Vi kopplar ihop DataLista med datasetet ds så att informationen som finns i ds läses ut i datalisten. Vi har sedan tidigare skapat DataLista som ett objekt i vår HTML-kod innan vi anropar den här i koden. Vi avslutar med att stänga databasen genom anslutning.Close();.

Sammanfattning av kod:

```
string Hamta = "SELECT id, rubrik, innehall, datum FROM Nyheter ";
OleDbDataAdapter adapter = new OleDbDataAdapter(Hamta, anslutning);

anslutning.Open();

DataSet ds = new DataSet();
adapter.Fill(ds);
DataLista.DataSource = ds;
DataLista.DataBind();

anslutning.Close();
```

Analys:
Här ser vi en tydlig likhet i de både språken då SQL-frågor används i de båda språkens koder och de skrivs exakt likadant också. Det är bara sättet att få ut datan på hemsidan som skiljer språken åt.

3 Lägga till data i databasen

PHP

```
mysql_query("INSERT INTO Nyheter (rubrik, innehall, datum)
VALUES ('$_POST[r]','$_POST[i]','$_POST[d]')");
```

Även här använder man sig av en SQL-fråga när man vill kommunicera med databasen. I HTML-koden har vi skapat tre textboxar; en för rubrik som heter r, en för innehåll som döpts till i och en för datum med namnet d. När man fyllt i textboxarna på hemsidan och sedan klickar på Spara-knappen, utförs en funktionen mysql_query(). Informationen i textboxarna skickas med $_POST-variabler och läggs i respektive kolumn i databasen där de sparas. Ordningen i SQL-frågan är viktig för att rätt information ska sparas i rätt kolumn. (INSERT INTO tabell (kolumn1, kolumn2) VALUES värde1, värde2). Textboxen för rubrik (r) står först då rubrik står först av kolumnerna. För varje nyhet som sparas i databasen ges ett unikt id. Detta id är som personnummer är för oss människor.

Sammanfattning av kod:

```
mysql_query("INSERT INTO Nyheter (rubrik, innehall, datum)
VALUES ('$_POST[r]','$_POST[i]','$_POST[d]')");
```

C# i .Net

```
OleDbCommand kommando = new OleDbCommand();
kommando.Connection = anslutning;
```

I OleDbCommand() kan man skapa ett kommando som ska utföra en eller flera uppgifter. Här ovan skapar vi vårt kommando kommando och definierar även vilken anslutning kommando ska använda, alltså vår tidigare skapta anslutning.

```
OleDbParameter p1 = new OleDbParameter("@rubrik", TextBox1.Text);
OleDbParameter p2 = new OleDbParameter("@innehall", TextBox2.Text);
OleDbParameter p3 = new OleDbParameter("@datum", TextBox3.Text);
```

På vår hemsida har vi tre fält (TextBox1, TextBox2, TextBox3) där man skriver in valfri text; en för rubrik, en för nyhetens innehåll och en för dagens datum. För att föra över all text till databasen använder vi oss av parametrar. I koden ovan skapar vi parametrarna, döper dem (p1, p2, p3) och definerar vilket fält som varje parameter ska hämta innehållet från. @rubrik, @innehall och @datum används som ett mellansteg i överföringen från hemsidan till databasen. Deras funktion är att meddela systemet att informationen som ska in i databasen kommer från en parameter.

```
kommando.Parameters.Add(p1);
kommando.Parameters.Add(p2);
kommando.Parameters.Add(p3);
```

Här lägger vi in våra tre parametrar (p1, p2, p3) i kommando.

```
string Tillagga = "INSERT INTO Nyheter(rubrik,innehall,datum)
VALUES (@rubrik, @innehall, @datum)";
kommando.CommandText = Tillagga;
```

Nu är det dags att använda sig av en SQL-fråga till databasen där vi kopplar ihop parametrarna med databasens kolumner. Även här är det viktigt med ordningen i SQL-frågan. I koden använder vi oss av mellanstegen (@rubrik, @innehall, @datum) för att SQL-frågan ska kunna hämta informationen från parametrarna. Resultatet av SQL-frågan sparas i variabeln Tillagga och läggs sedan in i kommando med hjälp av CommandText som är den del av OleDbCommand() som handhar SQL-frågor.

```
anslutning.Open();
kommando.ExecuteNonQuery();
anslutning.Close();
```

Vid det här skedet innehåller kommando en anslutning, tre parametrar och en SQL-fråga. Här öppnar vi anslutningen till databasen och med ExecuteNonQuery() utförs alla uppgifter som kommando innehåller. ExecuteNonQuery() används bland annat för att uppdatera och lägga till data, då inga värden behöver returneras. Slutligen stänger vi anslutningen.

Sammanfattning av kod:

```
OleDbCommand kommando = new OleDbCommand();
kommando.Connection = anslutning;

OleDbParameter p1 = new OleDbParameter("@rubrik", TextBox1.Text);
OleDbParameter p2 = new OleDbParameter("@innehall", TextBox2.Text);
OleDbParameter p3 = new OleDbParameter("@datum", TextBox3.Text);

kommando.Parameters.Add(p1);
kommando.Parameters.Add(p2);
kommando.Parameters.Add(p3);

string Tillagga = "INSERT INTO Nyheter(rubrik,innehall,datum)
VALUES (@rubrik, @innehall, @datum)";
kommando.CommandText = Tillagga;
anslutning.Open();
kommando.ExecuteNonQuery();
anslutning.Close();
```

Analys:
Återigen är SQL-frågan språken använder sig av exakt likadana, men nu krävs det betydligt mycket mer kod med C# i .Net för att kunna lägga till ny data. Jämfört med PHP-koden på två rader behöver man totalt 14 rader kod med C# i .Net. Detta beror mest på att man måste skapa och lägga till parametrar och mellansteg i kommandot, för att binda datan från textboxarna med databasen.

4 Radera data från databasen

PHP

För att kunna radera en nyhet måste vi få tag på den specifika nyhetens unika id. Detta id skickar vi med i radera-länken från sidan Index.php. Denna radera-länk är skapad i HTML-koden och i följande kod har vi gjort den fetstilt.

```
while($rad = mysql_fetch_array($resultat))
{
    echo . $rad['rubrik'] . ” ” . $rad['datum'] . ” ” . $rad['innehall'] . ” “ .
    "<a href='radera.php?id=".$rad['id']."'>Radera</a>";
}
```

När man klickar på radera-länken navigeras man till radera.php och det unika id:t skickas med genom $rad['id'].

```
$unikt_id = $_GET['id'];
```

Med hjälp av $_GET kan vi hämta det id som vi skickade med i radera-länken och detta värde sparas i variabeln $unikt_id.

```
mysql_query("DELETE FROM Nyheter WHERE id='$unikt_id' ");
```

Återigen använder vi oss av en SQL-fråga för att kommunicera med databasen. Med denna kodrad raderas nyheten som har det unika id:t, från databasen. WHERE jämför $unikt_id med alla id i databasen och DELETE FROM raderar posten i tabellen Nyheter där de två värdena stämmer överens (DELETE FROM tabell WHERE kolumn = värde).

Sammanfattning av kod:
Index.php:

```
while($rad = mysql_fetch_array($resultat))
{
    echo . $rad['rubrik'] . ” ” . $rad['datum'] . ” ” . $rad['innehall'] . ” “ .
    "<a href='radera.php?id=".$rad['id']."'>Radera</a>" ;
}
```

Radera.php:

```
$unikt_id = $_GET['id'];
mysql_query("DELETE FROM Nyheter WHERE id='$unikt_id' ");
```

C# i .Net

```
OleDbCommand kommando = new OleDbCommand();
kommando.Connection = anslutning;
```

Vi skapar ett kommando för radera-funktionen och definierar anslutningen, hittills ser koden likadan ut som i funktionen för att lägga till data (*se 3 Lägga till data*).

```
HiddenField radera_id = (HiddenField)e.Item.FindControl("dolt_id");
```

I HTML-koden har vi i vår datalist skapat ett dolt fält (HiddenField). Vi har döpt fältet till dolt_id och för att kunna nå fältet i vår C#-kod måste e.Item.FindControl() användas. Detta dolda fält är tillagt för att kunna komma åt det unika id som varje post har. Det unika värdet hämtas ut och sparas i variabeln radera_id.

```
int idt = int.Parse(radera_id.Value);
```

För att kunna jämföra det unika id:t som radera_id innehåller med alla de id som finns i databasen måste det unika id:t omvandlas till en siffra. Int.Parse() betyder på grov svenska siffra.Tolka() och tar då värdet som finns i radera_id och tolkar det till en siffra. Denna siffra sparas sedan i vår variabel idt.

```
OleDbParameter p1 = new OleDbParameter("@id", idt);
kommando.Parameters.Add(p1);
```

Som i ”3 Lägga till data” skapar vi en parameter för att kunna föra över det unika id:t till databasen. Koderna ser identiska ut förutom att vi här får värdet på vår parameter via variabeln idt istället för objektet TextBox. Vi döper vårt mellansteg till @id och lägger in parametern i kommando.

```
string Radera = "DELETE FROM Nyheter WHERE id=@id";
kommando.CommandText = Radera;
```

I denna SQL-fråga, som vi döper till Radera, raderar vi posten ur tabellen Nyheter där postens id stämmer överens med det unika id som skickats med från parametern. Till sist läggs SQL-frågan in i vårt kommando.

```
anslutning.Open();
kommando.ExecuteNonQuery();
anslutning.Close();
```

Återigen är koden likadan som ”3 Lägga till data” då vi med ExecuteNonQuery() utför de uppgifter som kommando innehåller; anslutningen, en parameter och SQL-frågan.

Sammanfattning av kod:

```
OleDbCommand kommando = new OleDbCommand();
kommando.Connection = anslutning;

HiddenField radera_id = (HiddenField)e.Item.FindControl("dolt_id");
int idt = int.Parse(radera_id.Value);

OleDbParameter p1 = new OleDbParameter("@id", idt);

kommando.Parameters.Add(p1);

string Radera = "DELETE FROM Nyheter WHERE id=@id";
kommando.CommandText = Radera;

anslutning.Open();
kommando.ExecuteNonQuery();
anslutning.Close();
```

Analys:
Precis som i avsnittet innan skiljer sig koderna väldigt mycket åt radmässigt. Med C# i .Net används ofta flera olika steg för att åstadkomma en enkel funktion, jämfört med PHP där allt oftast sker i ett eller få steg. En stor skillnad är att vi här i PHP tvingades att navigera om användaren när funktionen skulle utföras eftersom vi inte fann en annan lösning på det, som dock borde finnas.

5 Logga in på hemsidan

PHP

```
<form action="index.php" method="post">
Användarnamn:<input type="text" name="anvandarnamn"/>
Lösenord:<input type="password" name="losenord"/>
<input type="submit" name="skicka" value="Logga in"/>
</form>
```

Ovanstående text är hämtad från vår HTML-kod till Index.php som är vår inloggningssida. Det vi ser ovan är ett slags formulär innehållande två textboxar (input type=”text”); en för användarnamn och en för lösenord. Till sist har vi en knapp (input type=”submit”) som vi döper till skicka och som har etiketten Logga in. Alltså är det Logga in som står på knappen när hemsidan besöks, men i koden anropar vi knappen med namnet skicka. Att knappen fått typen submit beror på att den ska kunna skicka med värden från textboxarna till en annan sida. Överst i koden definierar vi vart man ska navigeras (action=”index.php”) och att värdena ska skickas iväg med metoden post (method=”post”).

```
if (isset($_POST['skicka']))
```

Om man har fyllt i textboxarna på hemsidan och klickat på Logga in-knappen kontrollerar isset om knappen har skickat med några värden. Skriver man alltså inte in någon information i textboxarna får knappen heller inga värden att skicka med. För att ta emot värdena från skicka som använde sig av post-metoden för att sända informationen måste vi även använda oss av $_POST[] för att ta emot den sända informationen. Den information vi mottager är alltså i detta fall ett användarnamn och ett lösenord.

```
$resultat = mysql_query("SELECT id FROM anvandare
WHERE namn='{$_POST['anvandarnamn']}' AND losenord='{$_POST['losenord']}'";);
```

Om isset mottager värden (användarnamn och lösenord) från knappen skicka vill vi kontrollera att dessa värden stämmer överens med våra värden i databasen. Informationen skickad i textboxen anvandarnamn jämförs med kolumnen namn i tabellen anvandare samtidigt som värdet i textboxen för lösenord jämförs med kolumnen losenord. Vi hämtar alltså id:t från anvandare där (WHERE) namn och (AND) losenord stämmer överens med de skickade värdena. Resultatet av SQL-frågan läggs i varibeln $resultat. Stämmer inte jämförelsen mot databasen får $resultat inga rader.

```
if (mysql_num_rows($resultat) == 0){
    header("Location: misslyckades.php");
}
```

mysql_num_rows() används för att undersöka hur många rader $resultat innehåller.

Med if-satsen bestämmer vi sedan att om $resultat inte innehåller några rader (== 0) ska man navigeras till undersidan misslyckades.php. Hit navigeras man med andra ord då användarnamn och lösenord inte stämmer överens med databasen.

```
else {
    header("Location: nyheter.php");
}
```

Om $resultat däremot innehåller rader betyder det att jämförelsen mot databasen lyckades och då kommer ovanstående kod att köras, som navigerar en till nyheter.php.

Sammanfattning av kod:

HTML-koden:

```
<form action="index.php" method="post">
Användarnamn:<input type="text" name="anvandarnamn"/>
Lösenord:<input type="password" name="losenord"/>
<input type="submit" name="skicka" value="Logga in"/>
</form>
```

PHP-koden:

```
if (isset($_POST['skicka']))
{
    $resultat = mysql_query("SELECT id FROM anvandare
    WHERE namn='{$_POST['anvandarnamn']}' AND losenord='{$_POST['losenord']}'");

    if (mysql_num_rows($resultat) == 0){
        header("Location: misslyckades.php");
    }

    else
    {
    header("Location: nyheter.php");
    }
}
```

C# i .Net

```
string LoggaIn = "SELECT ID,Namn,Losenord FROM Anvandare
WHERE Namn='" + TextBox1.Text + "' AND Losenord='" + TextBox2.Text + "'";
```

I HTML-koden på vår hemsida har vi två textboxar (TextBox1, TextBox2) där man kan fylla i sitt användarnamn och lösenord, exempelvis ”Sara” med lösenordet ”Stjärna”. Texten man skrivit in i textboxarna måste sedan jämföras med användarna som finns registrerade i databasen. Detta görs med hjälp av SQL-frågan LoggaIn. Våra två kolumner (Namn, Losenord) i tabellen Anvandare, som finns i databasen, jämförs med textboxarnas innehåll (Sara och Stjärna). Databasen söks igenom och letar efter en användare med namnet ”Sara” som har lösenordet ”Stjärna” Det räcker inte med att exempelvis Namn är rätt utan med AND i SQL-frågan krävs det att båda dessa villkor är uppfyllda.

```
anslutning.Open();
OleDbCommand kommando = new OleDbCommand(LoggaIn, anslutning);
```

I koden ovan öppnas vår anslutning och ett nytt kommando skapas som innehåller vår SQL-fråga LoggaIn samt vår anslutning. När vi inte använder oss utav några parametrar utan endast vill använda en anslutning och en SQL-fråga kan man definiera detta

direkt när kommandot skapas istället för att lägga in det var för sig (*se 4 Radera data*). Det är väldigt viktigt att SQL-frågans namn anges innan anslutningens namn inom parenteserna för OleDbCommand(), annars fungerar inte kommandot.

```
OleDbDataReader dr = kommando.ExecuteReader();
```

Nu vill vi ha ett retunerat värde från kommando och då använder vi oss av funktionen ExecuteReader(). Denna ExecuteReader() returnerar värdet från vårt kommando kommando i OleDbDataReader som vi här döper till dr. Om användarnamnet och lösenordet inte överenskommer med posterna i databasen, returneras inget värde till OleDbDataReader.

```
if (dr.Read())
{
    Response.Redirect("Nyheter.aspx");
}
else
{
    Response.Redirect("Index.aspx");
}
```

Om ett värde kan läsas ur dr betyder det att användarnamnet och lösenordet stämde överens med varandra. Vi har nu bestämt att om (if) värden kan läsas ut (dr.Read()) så ska man navigeras till sidan Nyheter.aspx. Misslyckas inloggningen ska man istället (else) navigeras om till Index.aspx där man får försöka logga in igen. Sidan man ska navigeras till skrivs inom parenteserna till Response.Redirect().

```
dr.Close();
anslutning.Close();
```

Slutligen stängs vår dr och vår anslutning.

Sammanfattning av kod:

```
string LoggaIn = "SELECT ID,Namn,Losenord FROM Anvandare
WHERE Namn='" + TextBox1.Text + "' AND Losenord='" + TextBox2.Text + "'";

anslutning.Open();
OleDbCommand kommando = new OleDbCommand(LoggaIn, anslutning);
OleDbDataReader dr = kommando.ExecuteReader();

if (dr.Read())
{
    Response.Redirect("Nyheter.aspx");
}
else
{
    Response.Redirect("Index.aspx");
}
dr.Close();
anslutning.Close();
```

Analys:
Största skillnaden mellan koderna är att i PHP används ett formulär i HTML-koden som man kan anropa med hjälp av post-metoden vilket inte behövs i C# i .Net.

Överlag är koderna likadant uppbyggda förutom att de använder sig av olika namn för samma funktion, exempelvis i PHP navigerar man om användaren med hjälp av header(Location: Index.php) medan man med C# i .Net istället använder sig av funktionen Response.Redirect(”Index.aspx”).

6 Sessions

För att man inte ska ha tillgång till alla delar av en hemsida när man inte är inloggad, måste sessions användas. Man ska till exempel inte kunna se radera-knappar i ett forum om man inte är inloggad. När man loggar in ska man därför få ett värde som tillåter användaren att se exempelvis radera-knapparna. Detta värde får man genom att använda sig av sessions på hemsidan.

PHP

```
if (mysql_num_rows($resultat) == 0)
{
    $_SESSION['Inloggad']=false;
    header("Location: misslyckades.php");
}
else
{
    $_SESSION['Inloggad']=true;
    header("Location: nyheter.php");
}
```

Observera att ovanstående kod är samma som i ”5 Logga in på hemsidan” förutom att vi nu har lagt till vår session, som vi har gjort felstilt. Om inloggningen inte lyckas ska heller inte sessionen $_SESSION[’Inloggad’] fungera och därför sätts värdet false på sessionen. Lyckas däremot inloggningen får sessionen värdet true och man har då tillgång till alla delar av hemsidan.

```
session_start();
$Inloggad=$_SESSION['Inloggad'];
```

På varje sida som ska kontrollera om man är inloggad eller inte, behövs de två övre kodraderna. session_start() i kombination med $Inloggad=$_SESSION['Inloggad'] gör att den inloggades session fortsätter vara aktiv även på den nuvarande sidan du besöker. Dessa koder ska skrivas in överst på varje sida innan HTML-koden.

```
if($Inloggad==true)
{
    echo “Du är inloggad”;
}
else
{
    echo ”Du är inte inloggad”
}
```

Är man inloggad har variabeln $inloggad värdet true, är man inte inloggad har den istället värdet false. På de olika sidorna kan man med hjälp av några enkla kodrader visa olika innehåll på hemsidan beroende på om man är inloggad eller inte. Dessa ovanstående kodrader skrivs i HTML-koden efter <body>-taggen. Har variabeln $Inloggad värdet true kommer texten ”Du är inloggad” att visas på hemsidan.

```
session_start();
session_destroy();
```

För att avsluta sessionen navigeras man om till sidan Loggaut.php där dessa kodrader står. Man måste även här använda session_start() för att kontrollera att man innehar en session och sedan avslutas den med funktionen session_destroy().

Sammanfattning av kod:

Index.php:

```
if (mysql_num_rows($resultat) == 0)
{
    $_SESSION['Inloggad']=false;
    header("Location: misslyckades.php");
}
else
{
    $_SESSION['Inloggad']=true;
    header("Location: nyheter.php");
}
```

Varje sida man vill kontrollera sessionen på (ovanför HTML-koden):

```
session_start();
$Inloggad=$_SESSION['Inloggad'];
```

Efter <body>-taggen i HTML-koden:

```
if($Inloggad==true)
{
    echo “Du är inloggad”;
}
else
{
    echo ”Du är inte inloggad”
}
```

Loggaut.php:

```
session_start();
session_destroy();
```

C# i .Net

```
if (dr.Read())
{
    Session["anvandare"] = TextBox1.Text;
    Response.Redirect("Nyheter.aspx");
}
else
{
    Response.Redirect("Index.aspx");
}
dr.Close();
```

Observera att ovanstående kod är samma som i ”5 Logga in på hemsidan” förutom att om inlogg-ningen lyckas, skapas en Session[]. Vi döper vår session till anvandare och den hämtar den inloggades namn från TextBox1, det vill säga det användarnamn man angav vid inloggningen. Detta namn blir användarens sessionsnamn.

```
if (Session["anvandare"] == null)
{
Response.Write("Du är inte inloggad");
}
else
{
Response.Write("Du är inloggad");
}
```

På alla sidor behövs ovanstående kod som kontrollerar om man är inloggad eller inte. Är man inte inloggad har man inget värde och det kallas för null. Då kan vi alltså jämföra sessionen med operatorn == om den är null, är den null visas texten ”Du är inte inloggad”. Om sessionen däremot har ett värde visas texten ”Du är inloggad”.

```
Session["anvandare"] = null;
```

Att avsluta sessionen är betydligt lättare. Endast en kodrad behövs, då vi definierar med operatorn = att sessionen ska vara null. Kodraden körs först när man klickar på vår logga ut-knapp.

Sammanfattning av kod:

Index.aspx:

```
if (dr.Read())
{
    Session["anvandare"] = TextBox1.Text;
    Response.Redirect("Nyheter.aspx");
}
else
{
    Response.Redirect("Index.aspx");
}
dr.Close();
```

Nyheter.aspx:

```
if (Session["anvandare"] == null)
{
Response.Write("Du är inte inloggad");
}
else
{
Response.Write("Du är inloggad");
}
```

Logga ut-knapp:

```
Session["anvandare"] = null;
```

Analys:

Att skapa sessioner i de både språken skiljer sig inte mycket kodmässigt åt då det bara behövs några kodrader i respektive språk. Skillnaderna ligger i att med PHP måste man starta sessionen på varje undersida och även kontrollera den, med C# i .Net räcker det med att man kontrollerar om sessionen är aktiv. Detta görs också på olika sätt då man med C# i .Net använder null och i PHP kontrollerar man mot variabeln $inloggad.

4.2.2 Frågeställning 7

Vilka fördelar och nackdelar finns med att programmera i C# i .Net respektive PHP enligt våra egna bedömningar?

Nedanstående fördelar och nackdelar är helt baserade på våra egna uppfattningar vi har skapat oss, både under våra litteraturstudier och programmeringslaborationer.

4.2.2.1 *PHP*

Fördelar

- *Open source (öppen källkod)*
  Öppen källkod betyder att koden är tillgänglig för alla att använda sig av och modifiera för sitt eget ändamål. Man får även distribuera koden i den utsträckning man själv känner för och vill.
- *Lätt att få hjälp på Internet*
  Det finns många bra forum med mycket kunniga skribenter på Internet där man lätt kan få hjälp med programmeringen. Dessa forum delar också med sig av exempelkod till olika funktioner. Den officiella hemsidan för PHP är ”http://www.php.net” och är även en bra källa att förlita sig på om problem uppstår när man skapar sin dynamiska hemsida.
- *Mindre kod*
  Överlag behövs det betydligt mindre kod att skriva i språket PHP än med C# i .Net. Se exempelvis vår PHP-kod till att skapa en anslutning till en databas och jämför den med koden för att ansluta sig till en databas med C# i .Net.
- *Lätt att lära sig*
  Inlärningskurvan är låg för PHP jämfört med C# i .Net. Strukturen i PHP är lätt att ta till sig och att lägga på minnet. Internet och mängden kod är förmodligen två faktorer till varför PHP inte är så komplicerat att lära sig.
- *Gratis*
  Då PHP kan skrivas i standardprogram som ingår i datorköpet, som till exempel Anteckningar eller WordPad, behöver man inte kosta på sig licenser för avancerade program. Detta är istället valbart för användaren. För att PHP ska fungera måste man först installera språket och en server. Båda dessa är gratis och går att ladda ner från Internet.
- *Plattformsoberoende*
  PHP fungerar på alla plattformar (operativsystem) och fungerar exempelvis lika bra till datortyperna PC och Macintosh. Påbörjar man exempelvis att skriva sin PHP-sida på en Macintosh-dator med Linux som operativsystem kan man utan problem fortsätta programmera på en PC-dator med operativsystemet Windows.
- *Anpassat för dynamiska hemsidor*
  PHP är, som tidigare nämnt, skräddarsytt för skapandet av dynamiska hemsidor, vilket har medfört att koden inte känns lika komplicerad att skriva.
- *Objektorienterat*
  Sedan version PHP4 lanserades har språket blivit mer objektorienterat. Detta är något som har dragit till sig fler användare till språket. Att ett språk är objektorienterat underlättar mycket vid programmeringen då man inte behöver skriva lika mycket kod utan att man istället kan anropa koderna vid behov.

- *Kontinuerlig utveckling*
  Språket utvecklas hela tiden av olika utvecklare världen över, vilket gör att språket blir bättre och bättre hela tiden. Eftersom språket är av öppen källkod bidrar det till den ständiga utvecklingen, då vem som helst kan hjälpa till att föra språket framåt i utvecklingskurvan. Trots att alla kan utveckla språket lokalt är det fortfarande ändå under PHP-ledningens ansvar att bestämma vilka nya funktioner som ska ingå i nästa version av PHP.
- *Populärt*
  På grund av språkets framfart har PHP blivit väldigt populärt i både amatör-kretsar och i de mer professionella kretsarna. Dess popularitet är vad som gör språket så starkt på marknaden.

Nackdelar

- *HTML-inbäddat*
  PHP-koden skrivs in direkt i HTML-koden. Förvisso skrivs koden in med PHP-taggar (*<?php...?>*) men det kan ändå bli svårt att hålla isär de båda delarna; PHP och HTML. Att hitta en viss kodrad, som man exempelvis vill ändra på eller radera, i en rätt omfattande och/eller avancerad hemsida kan bli väldigt tidskrävande. Detta beror på att hemside-filerna kan bli väldigt stora med många inskrivna rader, vilket gör att det kan bli många rader att leta sig genom innan man hittar den kod man söker.
- *Svårt att kontrollera händelser med knapptryckning*
  Exempelvis; hur får man en rad med text att synas först när man har tryckt på en knapp? Det kan vara svårt att förstå förhållandena mellan olika händelser och i vilken ordning de kommer att utföras. Det finns inget bra generellt sätt att få en händelse att starta enbart efter att man tryckt på knappen.
- *Servern*
  För att PHP ska kunna läsas i en webbläsare måste en server installeras på datorn. Datorn måste göras om till en server för att PHP ska kunna visas i webbläsaren. Den WAMP-server som vi valde att installera på vår dator var lite svår att just installera men även lite svår att förstå. För en datorovan användare kan det säkert bli ännu svårare. Det finns inga programvaror för PHP som installerar den nödvändiga servern automatiskt i installationen av programmet, utan detta måste man själv göra manuellt.
- *Bristfällig felsökning*
  När det har uppstått ett fel i koden kan man inte alltid få reda på vad felet är eller var felet befinner sig i koden. När man då öppnar hemsidan i webbläsaren visas ibland inget felmeddelande över huvud taget. Dock visas oftast ett felmeddelande om vilken kodrad felet bör befinna sig på, men det är däremot inte alltid att denna information stämmer.
- *Ej bakåtkompatibelt*
  Att de olika versionerna inte klarar av att läsa all kod från äldre versioner är mycket dåligt. Det kan bli tidskrävande att skriva om koden bara för att man vill ha eller behöver den senaste versionen av PHP. Med tanke på att det kommer nya versioner då och då, kan det bli väldigt tidskrävande att hålla sig uppdaterad i den alltid nya versionen.

Då ovanstående är egna åsikter kan andra ha helt olika åsikter om vad som är fördelar respektive nackdelar. Till exempel kan någon tycka att det är en fördel att PHP är HTML-inbäddat, som nu står som nackdel.

4.2.2.2 *C# i .Net*

Fördelar

- *C#-kod och HTML-kod separerade*
  C#-koden skrivs i separata .cs-filer och behöver på så vis aldrig beblanda sig med HTML-koden. Detta är görs för att hålla isär dessa två olika sorters koder, vilket resulterar i bättre kontroll över koderna.
- *Sessions är förvånansvärt lätt*
  Att skapa en session med C# i .Net är väldigt lätt att åstadkomma (*se 6 Sessions*). Väldigt få kodrader behövs för att både skapa en session och för att använda sig av den på flera av hemsidans undersidor.
- *Kontroll över händelser med knapptryckning*
  Till skillnad från PHP är det lätt för C#-användare att veta exakt när en händelse ska utföras, exempelvis vad som ska hända om man trycker på en knapp på hemsidan. Egenskapen och funktionen ”OnButtonClick” definierar vad som ska hända när man trycker på just den knappen som koden är kopplad till. Trycker man inte på knappen körs heller inte koden.
- *Inbyggd kodhjälp i programmet*
  Visual Studio, som är den mest använda programvaran för programmering av C# i .Net, innehåller en inbyggd kodhjälp. Denna kodhjälp tipsar om vad nästa ord i kodraden kan eller bör vara. Detta underlättar programmeringen avsevärt då bland annat programmeringstiden reduceras kraftigt.
- *Modernt*
  Språket var modernt redan från starten och det håller fortfarande måttet idag, nio år senare. Fortfarande har inte PHP uppnått samma grad av modernitet som C# i .Net har, som ändå inte har fått sitt ordentliga genomslag än. Det bör dock vara på gång snart då fler utbildas i språket på olika skolor runt om i Sverige.
- *Objektorienterat*
  Till skillnad från PHP var C# i .Net objektorienterat redan från lanseringen år 2000. Det är väldigt lätt att anropa funktioner utifrån, med C# i .Net.
- *Det bästa från flera språk*
  När C# i .Net utvecklades togs de bästa delarna av andra populära programmeringsspråk ut för att göra C# till ett starkare och bättre alternativ än de språk som redan fanns på marknaden. Det språk som utvecklarna lånade mest egenskaper från vid skapandet av C# var C, C++, Java och Visual Basic.
- *Genomtänkt*
  När man programmerar i språket får man en känsla av att det är ett väldigt genomtänkt språk. För en riktig programmerare borde språket näst intill kännas optimalt då varje liten del verkar har tagits i åtanke vid skapandet.
- *Felsökning*
  C# i .Net har en väldigt bra felsökning där det tydligt står vad felet är och på vilken rad felet uppstått. Ibland får man även ett förslag på hur kodraden istället bör skrivas, givetvis står det inte rakt ut vad som ska skrivas utan snarare att ”metoden saknar ett värde”. Detta gör det lättare att veta vad som saknas för att göra kodraden komplett.
- *Installation*
  För att vi skulle kunna använda oss av C# i .Net behövde vi endast installera programvaran vi valde att arbeta med, Visual Studio 2008.

Nackdelar

- *Komplicerad kod*
  Oftast behövs det enligt vår uppfattning, väldigt mycket kod för att utföra relativt små funktioner. Med så många och långa kodrader är det lätt hänt att man glömmer bort hur koderna ska skrivas och kan resultera i att man skriver koderna i fel ordning. De orden man använder sig av och den ordning man skriver dem i, kan också kännas förvirrande. Det kan till och med ibland kännas ologiskt trots att språket övergripande känns genomtänkt.
- *Licens behövs*
  För att använda sig av språket C# i .Net med hjälp av programvaran Visual Studio behöver man införskaffa en licens för programmet. Som privatperson kan det vara svårt att lägga ut dessa utgifter, till skillnad från ett företag som ofta investerar i olika programvaror. Är man dessutom en nybörjare i programmeringsvärlden är det sannerligen ingen summa man vill betala när det finns andra gratisalternativ.
- *Endast för Windows*
  C# i .Net är ett plattformsberoende språk och är endast skapat för att fungera på datorer med Windows som operativsystem. Detta låser många möjligheter för flera parter; språket självt då det har svårare att expandera, samt för företag och privatpersoner som använder sig av ett annat operativsystem än Windows.

Som synes har vi inte funnit många nackdelar med C# i .Net, men de är väldigt stora nackdelar. Pengar är alltid en viktig faktor och vi har tidigare konstaterat att med C# i .Net är man även låst i Microsofts miljöer.

5 Diskussion och slutsatser

Vi har valt att disponera detta kapitel på följande sätt: metoddiskussion, resultatdiskussion samt slutsatser. I metoddiskussionen diskuterar vi våra val av metoder och reflekterar över om det var bra metodval eller inte. Resultatdiskussionen är uppbyggd på samma sätt som i kapitel fyra. Med andra ord kommer vi att reflektera över varje frågeställning i den ordning vi angivit dem. Vi avslutar med att dra våra slutsatser av examensarbetet.

5.1 Metoddiskussion

Vi känner att de metoder vi valde var lämpade för vårt examensarbete och för det vi ville uppnå med arbetet. Att vi utförde intervjuer som grund för enkäten anser vi lyfte kvaliteten på enkäten. Det var också ett bra tillfälle för oss att besöka olika byråer och få en bättre uppfattning om intervjupersonernas åsikter då intervjuerna var mer öppna och frispråkiga än enkäten. Vi fick mycket tips och idéer från intervjuerna till utformningen av enkäten.

Metodvalet vi känner är mest relevant för hela examensarbetet är enkätundersökningen. Med hjälp av denna metod fick vi själva en större inblick hur situationen på de olika byråerna såg ut och vi fick en uppfattning om byråernas åsikter om de båda språken. Dock är inte svarsfrekvensen tillräckligt hög för att vi ska kunna generalisera resultaten från enkäten. Svarsfrekvensen uppnådde 58 procent men då fler än hälften av de byråer som svarade angav att de inte programmerar, sänks reliabiliteten ytterligare. Vi är missnöjda med vår svarsfrekvens då den blir väldigt låg med bortfallen, vi känner att vi skulle vilja ha fler byråer som svarade på hela enkäten. Nu kan vi inte generalisera något resultat i den utsträckning vi hade hoppats på. Det var synd att vi fick kontakt med så pass många byråer som inte programmerar då resultatet påverkades mycket av detta.

För att få en högre svarsfrekvens hade vi kunnat genomföra enkäten på ett annorlunda vis. Vi hade exempelvis kunnat ringa till byråerna och ställa de relevanta korta frågorna som vilken typ av byrå de tillhörde och vilket eller vilka programmeringsspråk de använder sig av. Under samtalet skulle man kunna be dem svara på de tre öppna frågorna (sämre och bra egenskaper för respektive språk samt vilket språk de tror kommer vara störst år 2015 och varför) genom att gå in på hemsidan där de frågorna finns. Vi kan tänka oss att det skulle höja svarsfrekvensen något om man kontaktat dem på ett mer personligt vis än bara via ett mejl.

Vi känner att den kodhjälp vi trodde att vi skulle få genom våra litteraturstudier var långt ifrån tillfredsställande, när det gäller båda språken. Då det finns flera olika sätt att utföra en funktion på, belyser de böcker vi läst endast ett av dessa sätt. Tyvärr är de sätten som är redovisade i de olika böckerna inte alltid eller sällan de sätt vi har lärt oss under våra olika programmeringskurser inom C# i .Net, vilket gjorde det lättare för oss att istället söka kodhjälp till våra funktioner på Internet. Det faktum att det var lättare att finna bra kodhjälp på Internet stämmer även för programmeringsspråket PHP då vi även i detta fall ansåg att Internet gav oss mer relevant kodhjälp än vad böckerna gjorde.

Programmeringslaborationerna känner vi var välbehövliga, men nu i efterhand ser vi att laborationerna mest var till för vår egen skull. De var av relevans för frågeställningarna kopplade till det andra huvudsyftet, men kanske inte i så stor utsträckning.

I det område vi nu efteråt känner att vi skulle kunna göra flest ändringar, om vi fick chansen att utföra examensarbetet igen, är i hur enkäten blev uppbyggd. Vi har nu i efterhand insett att vi hade med för många frågor i enkäten som inte var av relevans för våra frågeställningar. Vi hade bland annat en fråga gällande efterfrågan av programmerinsspråk på byråerna och anledningar till varför byråerna skulle kunna byta programmeringsspråk. Ser man till vårt syfte och våra frågeställningar hade vi egentligen ingen anledning att ställa dessa frågor i enkäten. Detta kan tyvärr sänka validiteten på vårt arbete då vi ställde frågor som var irrelevanta för vårt syfte.

Att vi valde att skriva bra respektive sämre egenskaper i enkäten istället för att använda oss av orden fördelar och nackdelar har sin förklaring. Detta beror helt enkelt på att vi under våra intervjuer fick en uppfattning om att flera av personerna vi intervjuade låste sig vid frågan och drog sig för att svara, bara för att vi använde oss av just fördelar och nackdelar som ordval i intervjuguiden.

Som vi tidigare nämnt blev vi förvånade över att så många reklambyråer svarade att de inte programmerar över huvud taget på företaget. Vi har däremot nu i efterhand insett att en anledning till detta beror på att de möjligtvis använder sig av en eller flera samarbetspartners som handhar all deras programmering. Det vi skulle ha kunnat ändra på i enkäten är med andra ord att när de byråer som bockade för rutan att de inte programmerar alls på företaget, skulle de ha fått en följdfråga om de istället använde sig av samarbetspartners för den tjänsten. Som enkäten såg ut vid undersökningstillfället kom frågan om samarbetspartners först efter frågan med svarsalternativet ”Vi programmerar inte alls”. Bockade de i den rutan fick de meddelandet ”Tack för din medverkan. Du behöver inte svara på fler frågor”, vilket medförde att de som icke-programmerande byrå aldrig nådde fram till frågan om samarbetspartners. Vi tror därför att några av dessa reklambyråer faktiskt erbjuder programmering som en tjänst, men att de i sin tur lejer ut det arbetet till samarbetspartners.

Överlag känner vi att reliabiliteten på vårt examensarbete är bra då vi har kombinerat olika metoder för att svara på våra frågeställningar. Vi ser nu i efterhand ingen anledning till varför vi skulle använda oss av några andra metoder för våra syften. Validiteten tycker vi är tillfredsställande men den sänks dock av att vi använde oss av för många frågor i enkäten som inte var av relevans för frågeställningarna.

5.2 Resultatdiskussion

Här diskuterar vi respektive frågeställning i tur och ordning. Vi inflikar också våra egna synpunkter på några av de frågor som gäller de byråerna vi undersökt.

5.2.1 Huvudsyfte 1

Att jämföra användningen av de båda programmeringsspråken C# i .Net och PHP på webbyråer, reklambyråer samt kombinerade byråer i Sverige idag.

Frågeställning 1

Vilket är i dagsläget det mest förekommande språket av C# i .Net och PHP på webbyråer, reklambyråer och kombinerade byråer som ingår i studien?

C# i .Net ligger lite i skuggan av språket PHP som situationen ser ur idag. Däremot kommer nog denna skugga att kunna minska inom ett par år eftersom C# i .Net säkerligen kommer att används mer i framtiden än vad det görs i dagsläget. Vi är dock inte förvånade över det faktum att resultatet pekade på att språket PHP används mer idag än vad just C# i .Net används. Däremot trodde vi, innan vi började med examensarbetet, att programmeringsspråket C# i .Net användes mer än vad det faktiskt verkar göras på byråerna. Detta hade vi nämligen fått en uppfattning om under våra tidigare kurser inom programmering. Vi är väldigt förvånade att det endast var två byråer som svarade att de använder C# i .Net, ett väldigt litet antal i förhållande till de totalt 22 byråer som angav att de idag använder sig av PHP.

Den stora skillnaden, trots den låga svarsfrekvensen, tycker vi ger oss anledning att anta att PHP är det mest använda programmeringsspråket på de olika byråtyperna i dagens Sverige.

Frågeställning 2

Finns det någon skillnad i val av programmeringsspråk beroende på vilken av de utvalda storstadskommunerna byråerna ligger i?

Det fanns inga större tydliga skillnader mellan vilket eller vilka programmeringsspråk som används i de olika kommunerna som vi undersökte (*se figur 5*). En sak vi noterade var däremot att de svarande byråerna i kommunen Linköping endast använder sig av PHP eller inget programmeringsspråk alls. Resultatet i de övriga kommunerna visar att mer än ett programmeringsspråk används per kommun. Vi trodde heller inte att vi skulle få ett annorlunda resultat än detta då vi inte ser någon anledningen till varför det skulle kunna finnas någon lokal skillnad, angående vilket eller vilka programmeringsspråk som används.

Att vi inte ser någon lokal skillnad tycker vi är bra då det borde betyda att man har möjlighet att söka arbete i vilken kommun som helst utan att bli begränsad av sina programmeringskunskaper. Det vore förargligt om man blev begränsad för sina kunskaper i endast ett språk.

Frågeställning 3

Finns det någon skillnad på vilket språk som används beroende på om det är en webbyrå, reklambyrå eller en kombinerad byrå?

Av vårt diagram över vilket eller vilka språk som används på de olika byråerna (*se figur 6*) kan vi konstatera att det finns endast små synliga skillnader mellan de olika typerna av byråer i Sverige. På webbyråer används mestadels språket PHP och vi ser en markant skillnad på användningen jämfört med C# i .Net. Reklambyråer programmerar tydligen inte alls mycket, vilket vi har förvånats över. Vi trodde och antog att fler reklambyråer erbjöd programmeringstjänsten i sitt företag. Många av dessa byråer kanske använder sig av samarbetspartners som sköter all programmering åt dem men det är inget vi vet någonting om idag.

Något som vi tycker är konstigt är att en webbyrå i Stockholm angav att de inte programmerar alls på företaget. Innan vår undersökning gjordes var det en självklarhet för oss att en webbyrå erbjuder tjänsten programmering, men uppenbarligen har inte alla webbyråer den tjänsten att erbjuda.

Skulle vi få ett arbete på en reklambyrå verkar chansen väldigt stor att vi inte kommer behöva programmera någonting alls på företaget utan istället stå för det grafiska. För oss skulle det kännas som bortkastade kunskaper om detta skulle inträffa, det skulle även kännas tråkigt när vi nu har studerat programmering i två år.

Frågeställning 4

Vilka egenskaper i språken anser byråerna vara bättre respektive sämre?

Vi fick in mycket mer svar på de bra respektive sämre egenskaperna för språket PHP än för C# i .Net. Främst för C# i .Net fick vi ibland in svar som talade emot varandra, exempelvis ansåg många att det var en bra egenskap att C# i .Net är ägt av Microsoft medan nästan lika många ansåg att det var en sämre egenskap med språket. Alltså är både den bästa och sämsta egenskapen med C# i .Net det faktum att det är ägt av Microsoft. Om vi talar för oss själva tycker vi att det är en nackdel att det är ägt av Microsoft och det är låst till dess miljö. För PHP rankas språkets enkelhet som dess styrka medan dess svagheter toppas av att språket ibland kan upplevas lite komplext och att lösningarna inte alltid blir så snygga. Att PHP skulle kunna upplevas som lite komplext är inget vi har upplevt under våra programmeringslaborationer.

Överlag har byråerna, som svarat på enkäten, få uppfattningar om språket C# i .Net vilket man bland annat kan se när man studerar synpunkterna över de bra egenskaperna och de sämre egenskaperna i språket (*se avsnitt 4.1.4.2*). I jämförelse med synpunkterna över PHP:s egenskaper (*se avsnitt 4.1.4.1*) är synpunkterna på det sistnämnda språket betydligt mer utförliga. Detta antar vi har att göra med att PHP är mer känt och välspritt bland de olika typerna av byråer i Sverige.

Frågeställning 5

Vilket av dessa språk tros vara det mest använda språket år 2015 och varför?

Språket PHP tros kunna vara det mest använda språket i Sverige år 2015. Även vi tror att detta kommer vara fallet men att C# i .Net har vid den tidpunkten växt ytterligare och gapet mellan de båda språken kommer inte att vara lika stort som det är idag. Om

det blir som vi och byråerna tror skulle det betyda att PHP kommer att hålla sin position och även då vara det ledande programmeringsspråk gentemot C# i .Net.

Programmeringsspråket C# i .Net är i dagsläget inte lika känt och välspritt som PHP, däremot anser vi dock inte att C# i .Net är ett sämre språk utan att det snarare ligger före sin tid. Vi känner att språket inte har växt färdigt än på långa vägar, men eftersom fler personer utbildas i språket tror vi att C# i .Net kommer att ha sin storhetstid inom några år. Vi säger däremot inte att språket kommer att vara större än PHP vid den tiden, men däremot att konkurrensen mellan de båda språken nog kommer ha hårdnat.

Om det skulle vara så att C# i .Net inte växer och blir mer använt på de olika byråtyperna i framtiden skulle det medföra negativa konsekvenser för oss och våra studiekamrater då det skulle betyda att vi utbildats i fel programmeringsspråk.

5.2.2 Huvudsyfte 2

Att få en bättre förståelse för de båda språken dels teoretiskt och dels genom att göra en praktisk jämförelse mellan hur man programmerar i de båda språken.

Frågeställning 6

Vilka är skillnaderna och likheterna mellan de båda språken; PHP och C# i .Net, teoretiskt och programmeringsmässigt?

Vi har märkt under arbetets gång att programmeringsspråken PHP och C# i .Net är mer olika än vad vi förutspådde. Bland annat kodskrivningen och hur exekveringen fungerar är två faktorer som skiljer sig åt i de båda språken. Däremot påverkades vi inte av skillnaden med exekveringen. När man ser hur exekveringen fungerar för C# i .Net rent teoretiskt anser vi att den verkar onödigt komplicerad, men rent programmeringsmässigt är det inget vi märker av, varken med C# i .Net eller PHP.

Att all C#-kod och all HTML-kod är separerade och skiljda från varandra med hjälp av olika dokument är något vi själva personligen föredrar framför att skriva programmeringskoden direkt i HTML-koden, som man gör med språket PHP. Med separerade dokument tycker vi det är lättare att hitta och ändra i koden när de är skiljda åt.

Hemsidorna vi skapade under våra programmeringslaborationer är inte så avancerat uppbyggda gällande de programmerade funktionerna samt det grafiska gränssnittet, vilket gör att PHP passar bättre till vårt ändamål. Fördelen att PHP är anpassat för dynamiska hemsidor spelar en väldigt stor roll enligt vår uppfattning om vilket språk som är lämpligast för vårt ändamål och liknande projekt. Däremot var det inga större svårigheter att få de två olika hemsidorna att både utseendemässigt och funktionsmässigt fungera identiskt. Detta tycker vi är ganska fascinerande eftersom det ligger två helt olika programmeringsspråk bakom sidorna.

Inlärningskurvan för språket PHP är inte lika hög som kurvan är för C# i .Net. Att själva kunna lära oss grunderna i PHP var inte alls svårt och det tog heller inte alls lång tid. Att språket är lätt att lära sig, kan bero på att det finns mycket hjälp att få via Internet. Man behöver inte kunna särskilt mycket om språket för att skapa en fungerande hemsida med tanke på att all kod finns tillgänglig på Internet. I olika forum och andra PHP-relaterade hemsidor kan man enkelt kopiera den kod man vill använda i sitt eget projekt, rakt av.

Däremot finns det delar av PHP som vi ännu inte riktigt har förstått oss på. PHP sägs vara ett objektorienterat språk då man kan samla alla funktioners kod på endast ett ställe i projektet och sedan ska man kunna anropa dessa funktioner från andra ställen av hemsidan genom endast några få kodrader. Just objektorienteringen har vi inte lyckats få till. Det vi kan göra är att anropa funktioner som vi skapat på samma sida där anropets koder finns, men vi har inte lyckats anropa funktionerna utifrån, alltså från andra sidor på hemsidan. För att försöka lösa detta problem vi stötte på, har vi använt oss av flera olika varianter av kod och även sökt efter kodlösningar både i böcker och på olika forum på Internet, dock utan resultat. Vi anser att detta inte borde vara så svårt att få till, med tanke på att det mesta i PHP, som vi har använt oss av i vårt projekt, har varit relativt lätt att åstadkomma.

Frågeställning 7

Vilka fördelar och nackdelar finns med att programmera i C# i .Net respektive PHP enligt våra egna bedömningar?

En intressant iakttagelse vi har gjort, efter att vi har sammanställt alla de svar vi fick in från enkäten, är att flera av våra egna framtagna fördelar och nackdelar för respektive språk stämmer överens med de bra samt sämre egenskaperna som byråerna angav om språken i enkäten. När vi insåg detta kändes det skönt att få en slags bekräftelse på att vi har kunnat skaffa oss rätt bra uppfattningar om språken under arbetets gång, då även andra har liknande åsikter om språken som vi har.

Trots att vi bara ser tre nackdelar för C# i .Net är dessa så stora för oss att de väger upp alla nackdelar som vi funnit för PHP. Vi tycker om båda språken, men en del fördelar med C# i .Net känner vi borde finnas i PHP och tvärtom. I dagsläget favoritiserar vi inget av dessa två språk, men istället favoritiserar vi egenskaper i språken. Det ultimata programmeringsspråket vore en kombinering mellan PHP och C# i .Net då nackdelarna i det ena språket är bättre och satta som fördelar i det andra. Ser man det på detta vis är språken varandras motsatser och vi efterfrågar en sammanslagning av språkens bästa delar. Vi inser att detta kanske inte är tekniskt möjligt, men det vore verkligen det ultimata.

5.3 Slutsatser

- Vår studie tyder på att:
  - PHP är det mest använda språket i dagsläget
  - C# i .Net inte är så spritt i Sverige än
  - Val av programmeringsspråk beror varken på byråtyp eller geografisk plats
  - PHP kommer vara det mest använda språket i Sverige år 2015

- Programmeringsspråken skiljer sig mer från varandra än vad vi tidigare förutspått, både teoretiskt och programmeringsmässigt.

- PHP är det programmeringsspråk vi själva föredrar framför C# i .Net. Till en programmeringssugen person som aldrig tidigare har programmerat skulle vi definitivt rekommendera personen att börja använda PHP.

- Med tanke på att vi känner att vår utbildning är anpassad för arbete på antingen reklambyråer, webbyråer eller kombinerade byråer borde man få utbildning i språket PHP också, inte enbart C# i .Net då det verkar vara det språk som är mest aktuellt.

- För att summera drar vi slutsatsen att PHP passar bättre för de mindre komplicerade hemsidorna och funktionerna medan språket C# i .Net istället är mer lämpat för de större och avancerade lösningarna.

- C# i .Net är skapat för fler användningsområden, till exempel programmering av mjukvaror och applikationer till mobiltelefoner.

Vi konstaterar därför att det språk som vi anser att man bör använda sig av beror helt enkelt på vad man vill åstadkomma med sin hemsida, men överlag anser vi att PHP är det bättre språket att skapa sina dynamiska hemsidor i, med C# i .Net som jämförelse.

Referenser

[1] Statistiska Centralbyrån (2008). http://www.scb.se/Pages/TableAndChart____228197.aspx (2009-03-10)

[2] Ullman, C., Kauffman, J., Hart, C., Sussman, D., Maharry, D. (2004). *Beginning ASP.Net 1.1 with visual C# i .Net 2003*. Indianapolis: Wiley Publishing, Inc., ISBN 0-7645-5708-4.

[3] Duthie, G. Andrew (2002). *Microsoft ASP.Net – Steg för steg*. Svensk översättning. Sundbyberg: Pagina Förlags AB,. ISBN 9163607204.

[4] Pohl, I. (2003). *C# by Dissection: The Essentials of C# Programming*. USA: Pearson Education, Inc., ISBN 0-201-87667-1.

[5] Forsberg, A. (2003). *Programmering i C#.* Lund: Studentlitteratur, ISBN 91-44-02656-0.

[6] Ullman, L. (2002). *Visuell snabbguide - PHP*. England: Pearson Education Limited, ISBN 0-201-74934-3.

[7] W3Schools (2009). http://www.w3schools.com/aspnet/aspnet_intro.asp (2009-03-13).

[8] Medinets, D. (2000). *PHP 3 Programming Browser-based Applications*. New York: The McGraw – Hill Companies, Inc., ISBN 0-07-135342-9.

[9] Overgaard, J., Eriksson, U., Ek, J. (2004). *PHP 5 programmering.* Sundbyberg: Pagina Förlags AB, ISBN 91-636-0800-6.

[10] Lerdorf, R.,Tatroe, K. (2002). *Programming PHP.* Sebastopol: O'Reilly Media, Inc., ISBN 1-56592-610-2.

[11] Eiderbäck, B., Hägglund, P., Bälter, O. (1995). *Objektorienterad programmering i Smalltalk.* Studentlitteratur: Lund. ISBN 91-44-49591-9.

[12] Meloni, J.C., (2000). *PHP Essentials – A better way to learn PHP*. Kalifornien: Prima Publishing, ISBN 0-7615-2729-X.

[13] Gilmore, W. J.(2008). *Beginning PHP and MySQL: From Novice to Professional, Third Edition.* Berkeley: Apress, ISBN 978-1-59059-862-7.

[14] PHP6, (2008). http://www.php6.se (2009-03-19).

[15] PHP, (2009) . http://se.php.net/archive/2004.php (2009-03-20).

Sökord

Bilaga 1 - Enkätformulär

Företagsnamn:

1. Vilken typ av byrå är ni?

- ○ Webbyrå
- ○ Reklambyrå
- ○ Kombinerad webb- och reklambyrå

2. Vilket/Vilka programmeringsspråk använder ni er av idag?

- ☐ PHP
- ☐ C# i .Net
- ☐ Annat språk:
- ☐ Vi programmerar inte alls i företaget.

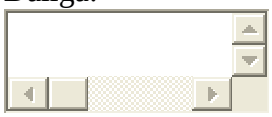
3. Sköts all programmering på företaget?

- ☐ Ja
- ☐ Nej, bara delvis. Vi använder oss också av en/flera samarbetspartners.

4. Enligt er uppfattning; Vilka egenskaper i PHP är bra respektive dåliga?

Bra:

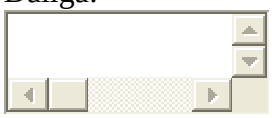
Dåliga:



5. Enligt er uppfattning; Vilka egenskaper i C# i .Net är bra respektive dåliga?

Bra:

Dåliga:



6. Vilken/vilka programvaror används vid skapandet av hemsidor?

7. Vilket av följande språk tror ni kommer vara störst år 2015 i Sverige?

- ☐ PHP
- ☐ C# i .Net

Varför tror ni det?

8. Händer det att kunder har efterfrågat ett visst programmeringsspråk?

- ☐ Ja - Vilket/Vilka?
- ☐ Nej

9. Har ni använt något annat programmeringsspråk tidigare?

- ☐ Ja
- ☐ Nej - *Gå vidare till fråga 12.*

10. Vilket/Vilka språk använde ni er av då?

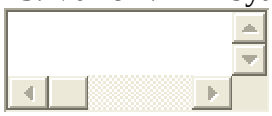
11. Vad fick er att byta?

- ☐ Kostnad
- ☐ Bättre möjligheter med det nya språket
- ☐ Kunden efterfrågade ett visst programmeringsspråk
- ☐ Annat- Vad?

12. Finns funderingar på att byta programmeringsspråk?

- ☐ Ja - Till språket:
- ☐ Nej - *Gå vidare till fråga 14.*

13. Varför vill ni byta?



14. Vad skulle få er att byta programmeringsspråk?

Bilaga 2 – Intervjufrågor

Vilket programmeringsspråk använder ni er av idag?
- Varför?
- Är pengar en faktor?

Nämn fördelar samt nackdelar med det nuvarande programmeringsspråket.

Vad är din uppfattning av programmeringsspråken PHP samt C# i .Net?

Vilket av dessa programmeringsspråk tros vara det största i 2015?

Har ni använt något annat programmeringsspråk tidigare?
- Om ni bytt, varför bytte ni?

Finns funderingar på att byta programmeringsspråk?
- Vad skulle få er att byta?
- Varför?

Hur är arbetet fördelat på företaget angående programmering och design?
- Procentiellt?
- Används ”mallar” för era produktioner?

Vilken programvara används för skapandet av hemsidor?

Kund
Händer det att kunder efterfrågar något visst programmeringsspråk?

Vilka är de vanligaste funktionerna ni får skapa till era kunder?

Hur avancerade funktioner kräver kunderna?

Finns det något samband mellan storlek på kund och svårighetsgrad på funktion/funktioner?