HÖGSKOLAN DALARNA

EXAMENSARBETE

Entity Framework 4.0, en utvärdering av ett ORM-ramverk

Entity Framework 4.0, an evaluation of an ORM-framework

Andreas Hall

Daniel Hindrikes

Akademin industri och samhälle

Nr: IKA052010

HÖGSKOLAN Dalarna

EXAMENSARBETE, Grundnivå 2 Informatik

Ämne	Reg nr	Omfattning
Informatik, Grundnivå 2	IKA052010	15 hp
Namn Andreas Hall Daniel Hindrikes	Månad/År Juni 2010	
	Handledare: Pär Eriksson Examinator: Mark Dougherty	
Företag/Institution Sogeti Sverige AB	Handledare vid företaget/institutionen Martin Olsén	
Titel Entity Framework 4.0, en utvärdering av ett ORM-ramverk		
Nyckelord Entity framework, ORM-ramverk, object-relational mismatch, objektorientering, relationsdatabaser, Visual studio		

Sammanfattning

När man kombinerar ett objektorienterat programmeringsspråk och en relationsdatabas uppstår en del problem för utvecklare eftersom objektorienterade programmeringsspråk och relationsdatabaser har olika fokus, objektorienterade programmeringsspråk fokuserar på att avbilda verkliga objekt och relationsdatabaser fokuserar på data. De problem som uppstår kallas med ett samlingsnamn för object-relational mismatch. Det finns flertalet ramverk för att hantera dessa problem. Ett av dem är Entity Framework.

Syftet med detta projekt var att utvärdera hur utvecklare tycker att Entity Framework fungerar för att lösa problematiken runt object-relational mismatch, hur det är för utvecklare att lära sig använda Entity Framework samt hur tillgången på inlärningsmaterial är.

Under vår studie har vi lärt oss använda Entity Framework samtidigt som vi gjort en studie av tillgången på inlärningsmaterial. Vi har också byggt om en applikation så att den använder Entity

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Framework. Vi har jämfört den ombyggda applikationen med den gamla applikationen för att kunna se vilken skillnad som Entity Framework bidrog till.

Vi kom fram till att Entity Framework hanterar object-relational mismatch på ett bra sätt som bland annat gör att utvecklingsprocessen kortas ner då inte lika mycket kod behöver skrivas. Utvecklare med tidigare kunskaper i .NET-programmering upplever att det är lätt att lära sig Entity Framework. Att det upplevs lätt att lära sig Entity Framework hänger förmodligen ihop med att tillgången på inlärningsmaterial är god.

DALARNA
University College

DEGREE PROJECT, Undergraduate level 2 in Informatics

Subject	Reg number	Extent
Informatics, Undergraduate level 2	IKA052010	15 ects
Names Andreas Hall Daniel Hindrikes	Month/Year June 2010	
	Supervisor: Pär Eriksson Examiner: Mark Dougherty	
Company/Department Sogeti Sverige AB	Supervisor at the Company/Department Martin Olsén	
Title Entity Framework 4.0, an evaluation of an ORM-framework		
Keywords Entity Framework, object oriented, object-relational mismatch, relational databases, ORM-framework, visual studio		

Summary

When you combine an object-oriented programming language and a relational database some problems occur for developers because object-oriented programming languages and relational databases have different focus, object-oriented programming focuses on the imaging of real objects and relational databases focusing on data. The problem that arises is called object-relational mismatch. There are several frameworks for dealing with these problems. One of them is the Entity Framework.

The purpose of this study was to evaluate how developers think the Entity Framework works to solve the problems around the object-relational mismatch and how easy it is for developers to learn to use the Entity Framework as well as the availability of learning materials.

In our study, we have learned to use Entity Framework while we carried out a study on the availability of learning materials. We have also rebuilt an application to use the Entity Framework.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

We have compared the application against the old application to be able to see what differences the Entity Framework did make.

We concluded that the Entity Framework handles object-relational mismatch in a good way for instance makes the development process shorter then you don’t need to write as much code as you will need to without the Entity Framework. Developers that already have knowledge of .NET programming finds it easy to learn the Entity Framework. That it is seen easy to learn the Entity Framework is probably linked to the availability of learning materials is good.

Innehållsförteckning

Figurförteckning

1 Inledning

I inledningskapitlet kommer vi att beskriva bakgrund, information kring uppdragsgivare, information gällande uppdraget, vårt vetenskapliga syfte samt vilken avgränsning och vilka frågeställningar vi har.

1.1 Bakgrund

Den mest utbredda och populäraste tekniken för att lagra data i databaser är såkallade relationsdatabaser. Samtidigt är objektorienterade programmeringsspråk väldigt populära bland systemutvecklare. I många applikationer används ett objektorienterat programmeringsspråk samtidigt som man lagrar data i en relationsdatabas.

Eftersom objektorienterade programmeringsspråk och relationsdatabaser har olika fokuseringar uppstår problemet att de passar dåligt ihop. Relationsdatabaser har data i fokus (se 3.2), medan objektorienterade språk fokuserar på att avbilda verkliga objekt (se 3.1). Därmed måste objekt mappas mot tabeller i databasen. (Ireland, Bowers, Newton, Waugh 2009) Varchar måste översättas till string, integer till int och så vidare (Neward 2006). Det gör att det behöver skrivas väldigt mycket kod. Under 90-talet kunde enligt Pettersson (2009) 20-30% av utvecklingstiden gå till att lösa glappen mellan objektorienterade programmeringsspråk och relationsdatabaser. Han skriver också att ett decennium senare ska den siffran ha ökat till 30-40%.

Object-Relational Mapping eller Object Relational Model (ORM) är en programmeringsteknik för att konvertera data mellan de oförenliga systemen relationsdatabaser och objektorienterade programmeringsspråk (Ubaid,Atique, Begun 2009). Det problemet brukar benämnas object-relational mismatch eller objekt-relationsproblemet på svenska, men i denna rapport kommer vi använda det engelska begreppet då det är ett vedertaget begrepp. I artikeln ”A classification of object-relational impedance mismatch” av Ireland, Bowers, Newton och Waugh från 2009 nämns det Hibernate och Oracle TopLink som exempel på ramverk för ORM. ORM beskrivs mer ingående i kapitel 4.

Microsoft Corporation har också ett ramverk för att hantera ORM. Detta ramverk kallar de för Entity Framework. Microsoft beskriver Entity Framework som ett ramverk för att undvika problemen som uppstår mellan datamodeller och programmeringsspråk (MSDN 2007), det vill säga ett ramverk för att hantera object-relational mismatch.

1.1.1 Uppdragsgivare

Sogeti Sverige AB är ett stort IT-konsultföretag som ingår i Cap Geminikoncernen. Sogeti har ungefär 20 000 anställda i 15 länder varav ungefär 1 000 är positionerade i Sverige. Moderbolaget Cap Gemini bildade år 2002 ett dotterbolag med namnet Sogeti som existerar i sex olika länder. År 2003 bildades Sogeti Sverige AB där deras fokus ligger på den växande lokala IT-marknaden. Sogeti fokuserar på att ha ett nära och långvarigt samarbete med kunden. (Sogeti.se)

1.1.2 Uppdragsbeskrivning

Sogeti vill ha en utvärdering av Entity Framework 4.0 som kom med .NET Framework 4.0. .NET Framework 4.0 är det nyaste .NET-ramverket från Microsoft Corporation. Målet med utvärderingen är att Sogeti vill veta om de ska börja använda Entity Framework i applikationer som de utvecklar.

Sogeti vill även ha en uppgradering av en applikation som de använder för att administrera golftävlingar som anordnas inom företaget och med kunder. Denna applikation som idag bygger på deras egenutvecklade Sogeti .NET Pattern som är baserad på COM+, vill de ha uppgraderad till Entity Framework 4.0.

1.2 Frågeställning

Utifrån bakgrunden och syftet har vi formulerat den frågeställning som presenteras nedan.

1.2.1 Underlättar användandet av Entity Framework 4.0 och Visual Studio 2010 det för systemutvecklare?

Tanken med ett ORM-ramverk som Entity Framework är att det ska underlätta vid utvecklandet av applikationer genom att det hanterar object-relational mismatch, som uppstår vid användning av objektorienterat programmeringsspråk och relationsdatabaser. Vi vill med denna frågeställning veta om Entity Framework 4.0 och Visual Studio 2010 underlättar för systemutvecklare så att det inte behövs skrivas lika mycket kod för att mappa tabeller mot objekt.

1.2.2 Hur upplevs Entity Framework att lära sig och hur är tillgången på inlärningsmaterial?

Om ett ORM-ramverk upplevs svårt att lära sig och är dåligt dokumenterat, kommer systemutvecklare få svårt att börja använda det. Därför vill vi ta reda på hur systemutvecklare med befintliga kunskaper i .NET programmering upplever det att lära sig Entity Framework och om det finns bra material tillgängligt för inlärning.

1.3 Syfte

Eftersom det kan behövas skriva väldigt mycket för att komma förbi glappet mellan objektorienterade programmeringsspråk och relationsdatabaser är det viktigt att använda ett bra ORM-ramverk för att hantera problemet. Därför är vårt syfte med detta projekt att undersöka om Entity Framework 4.0 är ett ramverk som hanterar object-relational mismatch så att det bidrar till att det inte behöver skrivas lika mycket kod för mappning av tabeller mot objekt. Vidare kommer vi att undersöka hur tillgången på material för inlärning av Entity Framework är och hur systemutvecklare upplever det att lära sig Entity Framework.

1.4 Avgränsning

Vi kommer endast att undersöka om Entity Framework underlättar för systemutvecklare att hantera object-relational mismatch-problematiken och hur tillgången på inlärningsmaterial är för att kunna bedöma om vi tycker att Entity Framework är något som systemutvecklare bör använda sig av. Vi har valt att avgränsa oss till just Entity Framework för att det är nytt och därmed intressant att ta reda på hur detta specifika ramverk hanterar problematiken. Intressant just för att företaget Microsoft Corporation är skaparna av ramverket och många använder deras övriga produkter.

1.5 Klargörande av begreppet systemutvecklare/utvecklare

När vi använder begreppen systemutvecklare/utvecklare menar vi personer som har ett yrke i form av programmerare eller av personer som praktiskt kan komma att stöta på object-relational mismatch och kan ha nytta av att använda sig av ett ORM-ramverk som Entity Framework. Eftersom Entity Framework är ett ORM-ramverk för att hantera object-relational mismatch under utvecklingen av .NET-applikationer vänder sig rapporten i första hand till systemutvecklare som utvecklar applikationer för .NET-plattformen.

2 Metod

I detta kapitel klargör vi vår kunskapssyn, beskriver vilken typ av kunskap rapporten bidrar med och beskriver de faser som vår studie har haft.

2.1 Metodik

Här kommer en presentation på de metoder som vi använt under vår studie, vilken kunskapssyn vi har och vad rapporten bidrar med.

2.1.1 Ontologi och epistemologi

Vi ser på kunskap på det sättet att den ska vara till nytta för praktiker och inte bara för de som forskar inom ämnet informatik. Vår rapport riktar sig i första hand till praktiker.

2.1.2 Rapportens kunskapsformer

Rapporten bidrar med normativ kunskap. Vilket innebär att den bidrar med vägledande kunskap. Rapportens kunskapsbidrag är normativt i det avseendet att den kan hjälpa utvecklare i valet om Entity Framework ska användas eller inte.

2.1.3 Studiens olika faser

Studien delas in i tre olika faser, en fas med teoristudier, en med inlärning och studie av tillgången på inlärningsmaterial och en fallstudie av Entity Framework.

Teoristudier
Under den första fasen kommer vi att läsa in oss på problematiken med Object-Relational Mapping för att en klar bild av vad det innebär för att kunna göra en utvärdering av Entity Framework som är ett ORM-ramverk.

Inlärning och studie av tillgång på inlärningsmaterial
Under denna fas kommer vi att lära oss att använda Entity Framework. Vi kommer även att göra en studie av tillgången på material som man kan använda vid inlärandet av Entity Framework. Det känns naturligt att vi genomför dessa två faser tillsammans, då vi ändå kommer att behöva söka efter material för att lära oss Entity Framework.

Fallstudie
Under denna fas har vi gjort en fallstudie på Entity Framework 4.0. Med en fallstudie menas att man undersöker något specifikt i sin realistiska miljö (Backman 1998). I vårt fall gör vi en undersökande fallstudie som kan ha vissa beskrivande infall. Att vi valt att göra en fallstudie beror på att vi tycker att det är det bästa sättet för oss att få en bra bild om hur Entity Framework upplevs vara att utveckla en applikation med.

2.2 Tillvägagångssätt

Här kommer en utförlig genomgång av tillvägagångssättet under studien.

2.2.1 Litteraturstudie ORM

Under litteraturstudien har vi använt oss av artikelsökmotorn Elin, som är en sökmotor för vetenskapliga artiklar. Att vi valt att använda oss av Elin beror på att vi i första hand ville använda oss av vetenskapliga artiklar då det ger mer tyngd åt rapporten och att det är den artikelsökmotor som vi har tillgång till via Högskolan Dalarna. Vi har även använt oss av andra artiklar som publicerats på webben.

2.2.2 Inlärning och studie av tillgång på inlärningsmaterial

Inlärningen och studien med tillgången på inlärningsmaterial för Entity Framework har gått hand i hand. Det blev naturligt att det blev så då vi själva behövde material för att lära oss Entity Framework.

Första steget i denna process var att ta reda på vilka böcker som fanns utgivna om ämnet. För att göra det tittade vi på de största webbplatserna för att köpa böcker online plus att vi använde Googles sökmotor för att söka efter böcker. Anledningen till att vi använde Googles sökmotor är att vi är vana att använda den och känner därför att vi når bäst resultat med hjälp av den.

Vi använde sedan Google för att hitta annat material som kan hjälpa till med inlärning av Entity Framework. Vi sökte också på Microsofts webbplats eftersom det är de som ligger bakom Entity Framework.

2.2.3 Fallstudie

Under fas tre har vi att genomföra en fallstudie av Entity Framework där vi byggde om en befintlig applikation till att använda sig av Entity Framework. Den nya applikationen använder samma databas som den gamla. Det programmeringsspråk som vi har använt oss av är det objektorienterade språket C# som är en del av .NET-plattformen. Vi har även att titta på om det finns några speciella krav på hur en databas ska vara designad för att det ska fungera på ett bra sätt att använda Entity Framework.

Det första steget i denna fallstudie var att analysera den befintliga applikationen, dels för att veta hur vi skulle bygga upp den nya, men också för att kunna se vilka skillnader som implementationen av Entity Framework leder till.

När vi sedan började utveckla den nya applikationen utgick vi från den befintliga och skapade samma metoder i den nya.

Vi gjorde även en kort intervju med en nyanställd person (se bilaga 1) på Sogeti för att få hans bild av Sogeti .NET Pattern som den gamla applikationen byggde på. Det för att få fler reflektioner än våra egna och därmed ett mer trovärdigt resultat.

Vi provade också att generera en databas utifrån en modell som vi skapat i Visual Studio då vi tyckte att vi behövde göra det för att kunna utvärdera Entity Framework på ett bra sätt.

Förutom den intervju vi gjorde om Sogeti .NET Pattern har vi endast använt våra egen observationer och upplevelser när vi utvärderat Entity Framework, detta då vi inte har möjligheten att ta del av andras observationer och upplevelser. Hade man haft möjligheten hade man kunnat låta ett flertal personer använda Entity Framework i ett projekt och sedan föra intervjuer med dem om deras upplevelser med Entity Framework. Vi anser dock att vi genomfört en trovärdig utvärdering ändå, eftersom vi ser oss själva som systemutvecklare även om det såklart hade varit bättre att kunna ta del av fler personers reflektioner.

När vi jämförde den gamla applikationen med den nya tittade vi på den kod som skrivits. Vi jämförde antalet kodfiler, kodrader och metoder för att kunna se vad skillnaden blivit med Entity Framework. Detta för att kunna se om användandet av Entity Framework bidrog till att mindre kod behövde skrivas eller inte för att hantera Object-Relational Mismatch.

2.3 Sammanställning av resultat

Resultatet av vår fallstudie presenteras i kapitel 5 i textform där vi går igenom hur utvecklare upplever att arbeta med Entity Framework 4.0 i Visual Studio 2010, hur svårt det upplevs att lära sig Entity Framework 4.0 samt tillgången på inlärningsmaterial.

I kapitel 6 presenterare vi resultatet av utvärderingsfasen i textform och i kapitel 7 redovisar vi våra slutsatser.

3 Centrala begrepp

För att öka förståelsen för det som skrivs i rapporten, kommer här en presentation av de mest centrala begreppen i rapporten.

3.1 Objektorientering

Överallt i det verkliga livet ser vi objekt, det kan vara personer, bilar, bord, stolar och så vidare. Människan har en förmåga att tänka i objekt. När man skapade objektorienterade programmeringsspråk som exempelvis Java och C# (C Sharp) överförde man det tankesättet till programmeringen. (Deitel, Deitel, 2007)

3.1.1 Begrepp inom objektorienterad programmering

Nedan beskrivs några av de grundläggande begreppen inom objektorienterad programmering.

Objekt

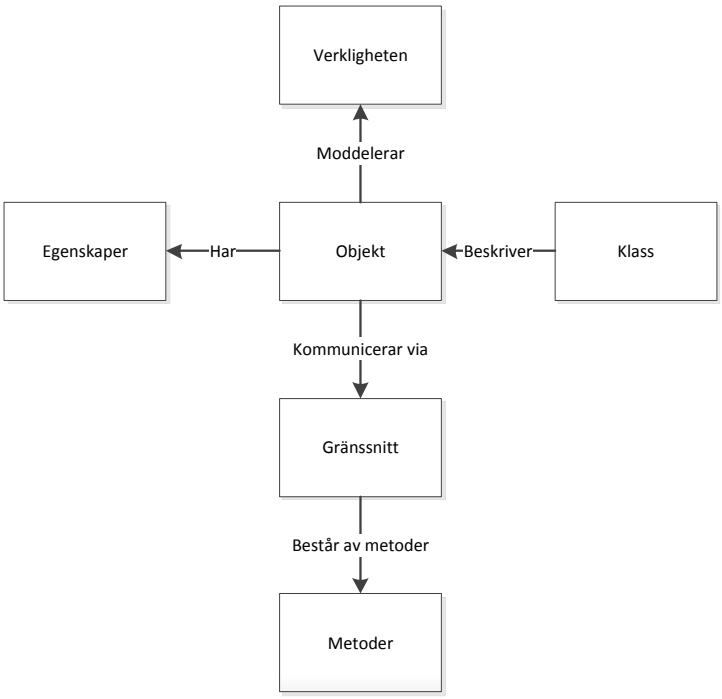
Ett objekt har ett tillstånd, ett beteende och en unik identitet. Med tillstånd menas de statiska och dynamiska egenskaper som ett objekt har. Med beteende menas att ett objekt både kan reagera på handlingar samt utföra handlingar. Ett objekts identitet är det som skiljer det från andra objekt. (Halilovic, 2006)

Klasser

En klass anger en beskrivning för antal objekt. Klassen beskriver de egenskaper och beteenden som ett objekt av en klass har. Alla objekt hör till en klass. Inom objektorienterad programmering är klassbegreppet fundamentalt, utan det kan man inte bedriva objektorienterad programmering. (Halilovic, 2006)

Gränssnitt

Inom objektorienterad programmering är det olika objekt som kommunicerar med varandra genom att objekten skickar meddelanden till varandra. Man säger att objekten använder ett annat objekts metod. Gränssnittet beskriver vilka metoder som objektet har. (Halilovic, 2006)



Figur 3-1: Begreppsgraf över objektorientering

3.1.2 Arv och polymorfism

Idén bakom arv är simpel men väldigt användbar. Tanken är att om du vill skapa en ny klass men redan har en klass som innehåller en del av den kod som du vill använda i den nya klassen. Då kan du låta den nya klassen ärva från den gamla klassen (basklass). Den nya klassen får då tillgång till den gamla klassens metoder och egenskaper. (The Java Tutorials)

Ordet polymorfism kommer från biologin där en organism kan ha flera olika former eller faser. Det kan också appliceras på objektorienterade programmeringsspråk. Det genom att man kan skriva om metoder som finns i en basklass till att ha ett beteende som passar bättre för den berörda klassen. (The Java Tutorials)

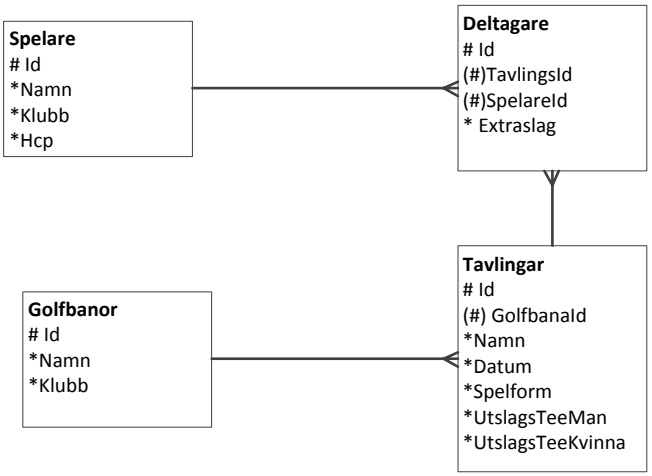
3.2 Databaser

Databaser består av data där data är organiserad i form av rader som beskriver ett fysiskt eller konceptuellt objekt. Relaterade databasrader är grupperade tillsammans i tabeller. (King, Konrad & Jamsa, Kris A., 2002) För att förtydliga detta kommer här ett kortare exempel. Om man exempelvis har en tabell som beskriver medlemmar innehåller denna tabell flera dataelement eller attribut som exempelvis användarnamn, förnamn, efternamn, adress och postnummer som unikt representerar en person som är medlem. Detta betyder då att tabellen medlemmar innehåller många unika medlemmar som är personer i verkligheten.

3.2.1 Relationsdatabaser

En relationsdatabas är en databas där tabeller är relaterade till varandra genom att relationen använder gemensamma kolumner innehållandes datavärden. Denna koppling mellan datavärden i olika tabeller representerar förälder/barn relationer i form av pekare för att etablera sig. (King, Konrad & Jamsa, Kris A., 2002)

Här följer ett exempel för att förtydliga bilden av hur det kan se ut när tabeller är relaterade mellan varandra med gemensamma kolumner. Som vi kan se i figuren nedan innehåller tabellerna Spelare, Deltagare, Tavlingar och Golfbanor varsin kolumn vilket unikt representerar en rad i respektive tabell. Till exempel är kolumnen id i tabellen Golfbanor ett nyckelvärde som markerats med symbolen # i figuren nedan vilket återfinns i tabellen Tavlingar som kolumnnamn Golfbanaid markerat med symbolen (#). Detta symboliserar en relation mellan tabellerna Golfbanor och Tavlingar i form av pekare.



Figur 3-2: En del av golfapplikationens datamodell.

3.2.2 SQL

SQL står för Structured Query Language och är ett standardiserat språk som används i samband med databaser. SQL har tre huvudsakliga roller vilket är skapande av en databas och definiera dess struktur, ställa frågor mot databasen för att få tillgång till det data som är svaret utifrån din fråga samt att kontrollera databassäkerhet. (Wilton, Paul & Colby, John, 2005)

4 Object-Relational Mapping (ORM)

I ett objektorienterat programmeringsspråk ligger fokus på objekt som ska representera verkliga objekt (se 3.1) till skillnad från relationsdatabaser vilket har data i fokus (se 3.2). Det är populärt att använda ett objektorienterat programmeringsspråk samtidigt som data lagras i en relationsdatabas (Ireland, Bowers, Newton, Waugh 2009).

Att fokus hos de två teknikerna skiljer sig skapar ett problem hos dem som vill använda sig av ett objektorienterat programmeringsspråk tillsammans med en relationsdatabas. Tabellerna i databasen måste då mappas till objekt (Ireland, Bowers, Newton, Waugh 2009). Då relationsdatabaser och objektorienterade programmeringsspråk använder olika datatyper måste de översättas, tillexempel varchar till string, integer till int och så vidare (Neward 2006). När denna mappning ska genomföras riskerar man att behöva få skriva väldigt mycket kod för att komma över glappet mellan de båda teknikerna. (Pettersson 2009) Glappet mellan ett objektorienterat programmeringsspråk kallas för object-reational mismatch.

Ett exempel på skillnaden att hantera data i ett objektorienterat programmeringsspråk är om man har användare och användargrupper. I en relationsdatabas har man tre olika tabeller för detta. En som lagrar information om användaren, en som lagrar information om gruppen och en kopplingstabell som talar om vilka användare som är med i vilken grupp. Om man ska modellera samma sak på ett objektorienterat sätt hade man skapat en klass för användare och en för grupper. En instans av ett gruppobjekt hade sedan innehållit referenser till alla objekt som tillhör de användare som är med i gruppen.

I ett objektorienterat programmeringsspråk kan man använda sig utav arv och polymorfism (se 3.1.2), detta är något som man inte har stöd för i relationsdatabaser (Neward 2006).

4.1 ORM-ramverk

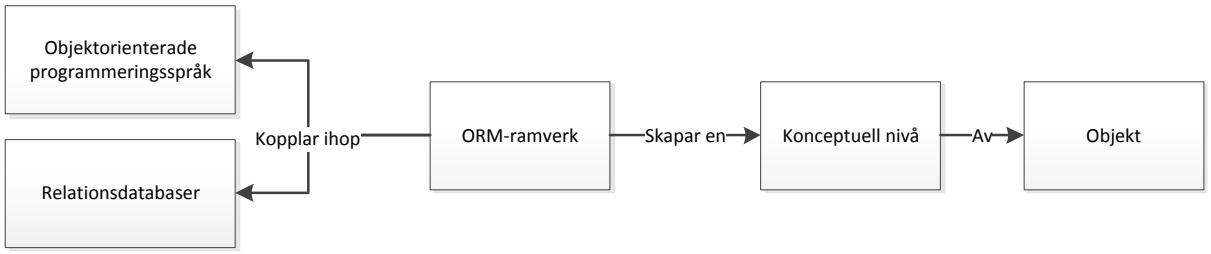
För att hantera object-relational mismatch under applikationsutveckling finns det ett flertal ramverk för att hantera det. Ireland, Bowers, Newton, Waugh nämner Oracle TopLink och Hibernate som exempel på ORM-ramverk. Pettersson (2009) tar upp Microsoft ADO.NET Entity Framework som ett ORM-ramverk, vilket endast benämns som Entity Framework i denna rapport och är det ramverk som vi riktar in oss på.

Det som ORM-ramverket har som syfte är att vara en metod för modellering och frågeställningar mot ett informationssystem på den konceptuella nivån. Eftersom informationssystem i vanliga fall är implementerat på ett databassystem som är baserad på en logisk datamodell som exempelvis grundas på relationer eller i form av objektrelationer, gör

ORM-ramverk det möjligt att mappa mellan de konceptuella och logiska nivåerna. (Halpin 2006)

Halpin (2006) beskriver att ORM kallas som det gör för att ramverket avbildar världen i termer av objekt (entiteter eller värden) som spelar roll (delar i relationer).

Fördelarna som blir med att använda sig utav ett ORM-ramverk enligt Halpin (2006) är för det första att ORM-modeller och frågor är mer stabila genom att attribut kan utvecklas till entiteter eller relationer. För det andra kan ORM-modeller fyllas med flera instanser och för det tredje är ORM mer enhetligt. Med enhetligt menas att man inte behöver en separat notering för att tillämpa samma hinder för ett attribut snarare än en relation.



Figur 4-1: Begreppsgraf ORM

4.1.1 Entity Framework

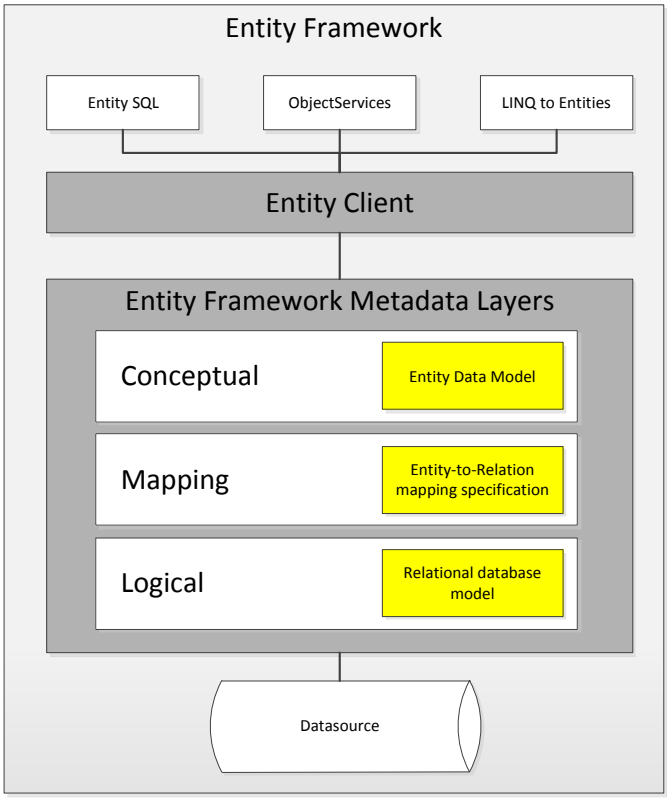
Entity Framework är som tidigare nämnt ett ORM-ramverk som är utvecklat och framtaget av Microsoft för .NET-plattformen. Ubaid, Atique, Begun (2009) tycker dock att det inte är helt korrekt att klassificera Entity Framework som endast ett ORM-ramverk då Entity Framework är mycket mer avancerat än så. Ubaid, Atique, Begun (2009) skriver att Entity Framework strävar efter något betydligt mer sofistikerat till skillnad från vanliga ORM verktyg då Entity Framework skapar den konceptuella modellen som något konkret.

Modellen kan genereras från en befintlig databas eller genom att man skapar modellen först och generar en databas utifrån entitetsdatamodellen (MSDN Channel 9).

Detta åstadkommer Entity Framework genom att tillhandahålla ett ramverk för att skapa upp en abstrakt modell ovanpå relationsdatabasen för att komma över problemet gällande object-relational mismatch. Kärnan i Entity Framework är dess olika lager av abstraktion vilket är uppdelat enligt konceptuella, mappade och logiska skikt som ingår i Entitetsmodellen. (Ubaid, Atique, Begun, 2009)

Entity Framework använder sig dessutom av två API:er, object services och entity client för att kunna arbeta med entitetsdatamodellen samt användning av två konstruktorer för

datamanipulation. De som används är Entity SQL (ESQL) och LINQ to Entitites. Ubaid, Atique, Begun (2009)



Figur 4-2: Entity Framework architecture, Yemelyanov A (2008)

Entity Client

Entity Client är en dataaccessleverantör för Entity Framework även kallat *mapping provider*. Det som Entity Client gör är att den kapslar in och hanterar databas- och entitetsdatamodellanslutningar och liknar den vanliga SQL Client i ADO.NET som tillåter applikationer att ansluta till relationella datakällor. Till skillnad från SQL Client ger Entity Client tillgång till data inom entitetsdatamodellens villkor. (Yemelyanov, 2008)

Object Services

Detta är ett verktyg inom Entity Framework som genererar ett domänobjektslager för användning i applikationen. Genom att man använder *Object Services* är det möjligt att ställa frågor mot entitetsdatamodellen med hjälp av *Entity SQL* eller *LINQ*. (Yemelyanov, 2008)

Entity SQL

Frågespråket Entity SQL är ett frågespråk skapat för entitetsdatamodellen i Entity Framework. Entity SQL går att dra liknelse med traditionell SQL för databaser som utvecklare som använt

det tidigare känner igen sig i. Det Entity SQL bidrar med är att ge Entity Framework dynamisk frågekapacitet där frågor formuleras statiskt exempelvis under designtillfället. (MSDN 2010) Nedan följer ett litet exempel på hur en Entity SQL fråga kan se ut:

```
List<Golfbana> golfbanor = new List<Golfbana>();
using (GolfDomain ctx = new GolfDomain())
{
    Query<Golfbana> banor =
        db.GetQuery<Golfbana>("SELECT * FROM Golfbanor");

    foreach (Golfbana bana in banor)
    {
        Golfbanor.Add(bana)
    }
}
```

Figur 4-3: Exempel på Entity SQL

LINQ

Language-Integrated Query (LINQ) är ett frågespråk som används för att ställa frågor mot olika datakällor som tillexempel datastrukturer i minnet, XML-dokument och även genom ADO.NET mot databaser, entitetsdatamodeller och dataset. Några av dessa använder sig av olika implementationer, vilket är dolt, men alla använder sig av samma syntax och språkkonstruktioner. (MSDN 2010) Nedan följer ett exempel på hur en LINQ-fråga ser ut när den ställs mot en entitetsdatamodell.

```
List<Golfbana> golfbanor = new List<Golfbana>();


var banor = from b in ctx.Golfbana orderby b.Namn select b;

foreach (var bana in banor)
{
    golfbanor.Add(bana);
}
```

Figur 4-4: Exempel på LINQ to entities

5 Implementation av ORM-ramverk

Detta kapitel beskriver vad undersökningen har visat med implementationen av Entity Framework samt en presentation av material som finns tillgängligt för att kunna få kunskap om att jobba med ramverket.

5.1 Lösning i den gamla applikationen

I den gamla applikationen hanteras object-relational mismatch med hjälp av Sogetis egenutvecklade designmönster Sogeti .NET Pattern. Detta ramverk läser in data som finns i databasen och lägger den i dataset. Denna lösning innehåller många olika lager. De nedersta lagren som hanterar den direkta kopplingen mot databasen får man genom att lägga till dll-filer till det aktuella projektet. All kommunikation med databasen sker via lagrade procedurer. Det gör att utvecklaren inte behöver skriva någon kod själv för att hantera kopplingen med databasen då detta redan är gjort.

När man vill spara eller uppdatera data skickar man data i dataset. Detta sker genom att man använder en metod i ett objekt för att lägga till data. Dataseten skickas sedan tillsammans med ett antal parametrar som talar om vad man vill göra med datasetet till det nedre lagret som sköter databaskopplingen.

För att ta reda på vad de anställda på Sogeti tycker om .NET Pattern pratade vi med Karl-Henrik Nilsson som nyligen har blivit anställd på Sogeti och därmed nyligen kommit i kontakt med .NET Pattern. Hans kommentar om Sogeti .NET Pattern var att det är bra om man jämförde det med att inte använda något ramverk överhuvudtaget för att hantera object-relational mismatch. Däremot tyckte han att det var krångligt att lära sig det och att det tog ungefär en vecka för honom att sätta sig in i det. Han tyckte också att det kändes krångligt att använda om man jämförde med Entity Framework.

Det vi upplevt då vi själva har tittat på den gamla applikationen är att .NET Pattern känns rörigt och krångligt att sätta sig in i och att det är väldigt mycket kod som är skriven.

5.2 Entity Framework 4.0 i Visual Studio 2010

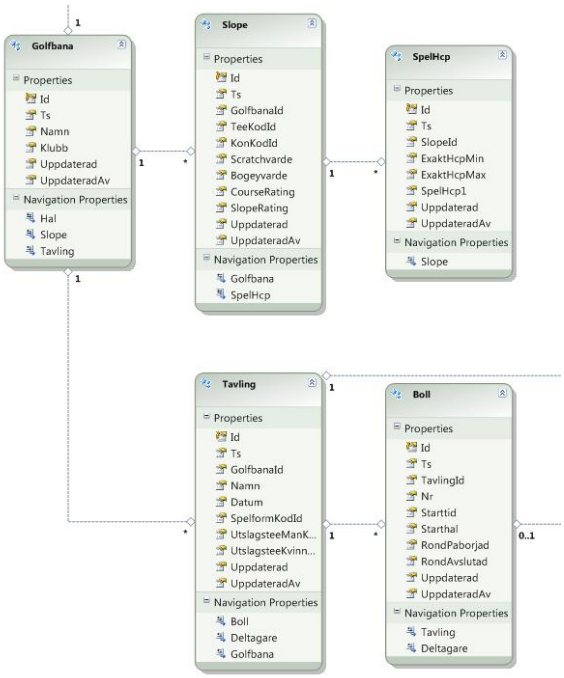
Visual Studio är den utvecklingsmiljö som används för utveckling av .NET applikationer med Entity Framework. Här beskrivs det vi upplevt under applikationsutvecklingen med Entity Framework i Visual Studio.

5.2.1 Generera modeller

Som beskrivs i kapitel 4.1.1 kan man generera en entitetsdatamodell med Entity Framework i Visual Studio 2010. Som det beskrivs i det kapitlet kan man både generera en databas utifrån en färdig modell och generera en modell utifrån en befintlig databas.

Vi har undersökt vad som krävs av en relationsdatabas för att det ska kunna generera en bra modell. Observationer som vi gjorde var att relationer mellan tabellerna måste vara definierade med foreign-keys (pekare) för att man ska få relationer mellan objekten på ett sätt som beskrivs i kapitel 5.2.2 nedan. Vid de tillfällen som vi inte hade tabeller med relationer i databasen genererades det inte heller några relationer mellan entiteterna i entitetsdatamodellen, fast de egentligen borde ha haft det.

I entitetsdatamodellen kan man även ändra namn på entiteter om man vill. Detta påverkar inga namn i databasen utan bara på konceptuell nivå.



Figur 5-1: En del av den genererade entitetsdatamodellen

När man genererar en databas utifrån den modell man skapat i Visual Studio kan man använda sig av arv mellan entiteterna i modellen.

5.2.2 Punktnotation

Om man har flera tabeller i en databas och vill ha ut data från flera tabeller måste man använda sig av såkallade join-satser i SQL-frågor om man inte använder sig av ett ORM-ramverk. Dessa SQL-frågor kan bli ganska stora om man har många relationer.

I Entity Framework behöver man inte skriva några join-satser. Det räcker med att man hämtar huvudobjektet. Utifrån huvudobjektet kan man sedan hämta ut relaterande data. Detta sker med punktnotation precis som när man hanterar vanliga objekt, man skriver referensen till det aktuella objektet och sedan namnet på det relaterade objektet och sedan namnet på den egenskapen man vill hämta.

Ett exempel ur den kod vi har skrivit under utvecklingen av applikationen är när vi har hämtat ett tävlingsobjekt **t** och vill ha ut namnet på den golfbana som tävlingen utspelar sig på. Detta görs genom att skriva **t.Golfbana.Namn.**

5.2.3 Spara, uppdatera, radera och hämta data

I Entity Framework behöver inte utvecklaren skriva någon SQL-kod för att spara, uppdatera, radera och hämta data från databasen.

Spara data
När man vill spara data till databasen med Entity Framework skapar man upp ett objekt och lägger till det till ett containerobjekt som är det objekt som representerar entitetsdatamodellen och anropar en metod för att spara.

Uppdatera data
När man vill uppdatera är det precis samma princip, man ändrar bara de egenskaper hos objektet man vill förändra och anropar metoden för att spara.

Radera data
För att radera data med Entity Framework krävs det att man har läst in aktuellt objekt som man vill ta bort för att sedan radera objektet med en radera metod.

Lagrade procedurer
Det finns stöd för att använda sig av proceduranrop om man har sådana eller vill använda sig av dem ur exempelvis optimeringssynpunkt. Det finns stöd för att använda procedurer till insert, update och delete.

5.2.4 Mindre kod i den nya applikationen

När vi jämförde den gamla applikationen med den nya och kollade på kodfiler, kodrader och metoder visade det sig att det inte alls behövde skrivas lika mycket kod när man använde sig av Entity Framework. Eftersom vi byggde den nya applikationen efter den gamlas metoder har vi alla de metoder som fanns i koden för den gamla applikationen med i den nya också. Dock behövde vi inte använda oss av alla när vi använde Entity Framework.

5.3 Att lära sig Entity Framework 4.0

Något som påverkar hur man uppfattar enkelheten av att lära sig hantera ett nytt ramverk är tillgången på dokumentation och material för inlärning. Därför har vi tittat på tillgången på sådant. Nedan presenteras hur tillgången på media, som kan användas vid inlärningen av Entity Framework. Exempel på media kan vara både skriven text och video, är.

Det mesta av det materialet som finns tillgängligt är på engelska. Men det finns även en del på svenska.

5.3.1 Microsofts egen webbplats

Microsoft Developer Network (MSDN) är den portal som Microsoft har till stöd för utvecklare. Portalen tillhandahåller utvecklare information som exempelvis guider, undersidan channel 9 som används för videoföreläsningar, API:er, community, bloggar, hämtbara filer m.m. Den största delen av materialet som finns på MSDN är skrivet på engelska, men det finns även en svensk version av MSDN där det finns visst material på svenska.

Det vi lärde oss bäst av var en video på channel 9 på svenska av Dag König där han gick genom grunderna i Entity Framework.

5.3.2 Övriga webbplatser

Förutom den ovan nämnda webbplatsen av Microsoft Corporation finns det andra webbplatser där man kan söka sig fram till via Internet som tillhandahåller mängder med information om .NET med Entity Framework. Det finns både bloggar, guider och liknande webbplatser som behandlar Entity Framework.

5.3.3 Böcker

I skrivande stund finns det inga böcker om Entity Framework 4.0. Dock är det på gång med flertalet böcker som handlar om Entity Framework 4.0. En bok som man får många träffar på om man söker på Google är ”Programming Entity Framework” skriven av Julie Lerman.

Om den föregående versionen av Entity Framework finns det ett större sortiment av böcker att få tag på.

6 Analys av Entity Framework 4.0

Nedan presenteras vår analys som baseras på det som skrivits i de tidigare kapitlen och utifrån syftet och frågeställningen.

6.1 Använda Entity Framework

Eftersom det blev mindre kod att skriva när vi använde Entity Framework jämfört med vad det blev med .NET Pattern och mot vad det blivit om man inte hade använt något ramverk för att hantera object-relational mismatch problematiken, medför det att problematiken med att det är tidsödande att komma runt glappet mellan objektorienterade programmeringsspråk och relationsdatabaser minskar markant.

Att vi inte behövde använda alla de metoder som vi skrivit beror på Entity Frameworks sätt att hantera relationer mellan objekten (se 5.2.2). Detta anser vi vara ett starkt argument för att implementera Entity Framework.

Vi tycker också att koden blir mer tydlig och lättläst om man jämför med hur det såg ut i den applikationen som byggde på Sogeti .NET Pattern. Hade man gjort mappningen helt manuellt utan att använda ett ORM-ramverk hade man förmodligen kunnat uppnå samma resultat. Men erfarenheter vi har sedan tidigare säger oss att det hade tagit betydligt längre tid.

Entity Framework känns som något vi saknat alla de gånger vi byggt applikationer och hanterat mappningen mellan objektorienterade programmeringsspråk och relationsdatabaser manuellt. Vi har då upplevt den manuella mappningen som väldigt tidsödande, precis som det beskrivits i flertalet av de artiklar som vi läst.

Den största anledningen till att Entity Framework upplevs så smidigt som det gör, skulle vi vilja säga beror på att Visual Studio 2010 hanterar det på ett väldigt enkelt sätt. Att man bara med några knapptryckningar kan skapa upp en entitetsdatamodell utifrån en befintlig databas gör att det går väldigt snabbt att komma igång med applikationsutvecklingen. Det grafiska verktyget i Visual Studio som man använder vid skapandet av en ny modell tycker vi också fungerar på ett smidigt och bra sätt.

Möjligheten att ändra namn i entitetsmodellen gör att även om man har en databas med otydlig namngivning kan man få en tydlig konceptuell databasmodell att programmera mot.

Bäst resultat av användningen av Entity Framework får man om man väljer att skapa en ny modell som man genererar en databas utifrån. Detta då man först då kan använda alla funktioner i Entity Framework. Tillexempel kan man då skapa arv mellan entiteter.

Att man kan använda lagrade procedurer för insert, update och delete är något som vi ser som positivt då man kan skapa procedurer om man tillexempel vill att data ska hanteras på något sätt innan den sparas i databasen. Det kan ju vara så att man vill använda sig av lagrade procedurer för att kopplingarna mot databasen ska bli mer optimerade.

Att hantera objekt för att skapa, ändra och ta bort data ur databasen tycker vi bidrar till att man bibehåller ett objektorienterat arbetssätt mot om man inte skulle använda ett ORM-ramverk och därmed vore tvungen att skriva SQL-frågor för att göra detta.

6.2 Inlärning av Entity Framework

Inlärningen av Entity Framework var betydligt lättare än vad vi hade förväntat oss, speciellt efter att vi hade tittat på Sogeti .NET Pattern, som vid första anblicken kändes väldigt svårt och rörigt. Att en nyanställd på Sogeti som precis börjat arbeta med .NET Pattern har samma uppfattning som oss tyder på att det är ett krångligt ramverk att lära sig.

Att Entity Framework känns så pass enkelt att lära sig beror förmodligen på tillgången på inlärningsmaterial som vi upplever som väldigt bra. Med väldigt bra avser vi i det här avseendet att vi inte upplevt att vi saknat något. Att det både finns tillgång till video och text tycker vi är bra då olika personer lär sig bäst på olika sätt. De som vill läsa text kan göra det, de som vill titta på kodexempel kan göra det och de som vill se en videobaserad föreläsning kan göra det, eller som vi gjorde, att vi kombinerade dessa tre inlärningssätt.

Att det i undersökande stund finns dåligt med böcker som tar upp Entity Framework 4.0 känns inte som något problem. Anledningen till det är att det finns gott om material tillgängligt på webben och att det planeras att släppa böcker om Entity Framework.

Inlärningen av Entity Framework gick för oss väldigt fort om man jämför med Sogeti .NET Pattern, som för den person på Sogeti som vi pratade med tog en vecka att lära sig. För oss tog det bara några timmar att komma igång och kunna använda Entity Framework. Att börja använda Entity Framework känns inte som något stort steg då vi redan var vana att programmera i C# och programmeringstekniken med Entity Framework är väldigt likt det arbetssätt man har när man jobbar i en ren objektorienterad miljö utan att koppla till en relationsdatabas.

Att man kan komma åt objekten i entitetsdatamodellen på två olika sätt tycker vi är en styrka hos Entity Framework. Det finns LINQ för att ställa frågor mot entitetsdatamodellen för de som inte är vana att använda SQL, men kanske har använt LINQ för att ställa frågor mot tillexempel

arrayer eller XML-dokument, vilket betyder att de inte behöver lära sig något nytt för att komma år objekten.

De som är vana att använda frågespråket SQL och som känner sig bekväma med det kan använda sig av Entity SQL för att ställa frågor mot entitetsdatamodellen. Här kan man tycka att en liten del av vitsen med Entity Framework försvinner då man ändå måste använda ett SQL-liknande språk. Men man behöver inte alls skriva lika avancerade frågor som man skulle få göra om man skrev SQL-frågor direkt mot en databas, eftersom man bara behöver hämta ett objekt och använda dess relationer för att komma åt relaterade data på det sättet som beskrivs i kapitel 5.2.2.

7 Slutsatser

Vi har kommit fram till att användandet av Entity Framework 4.0 i Visual Studio 2010 underlättar för systemutvecklare vid utvecklandet av applikationer som använder sig av .NET plattformens objektorienterade programmeringsspråk tillsammans med en relationsdatabas. Detta då Entity Framework hanterar object-relational mismatch på ett sätt som gör att det underlättar för systemutvecklare. Detta medför att utvecklaren sparar tid då mappningen mellan det objektorienterade programmeringsspråket och relationsdatabasen sköts automatiskt av Entity Framework. Därför tycker vi att Entity Framework i Visual Studio 2010 är lämpligt att använda av systemutvecklare som utvecklar applikationer i något av .NET-plattformens objektorienterade programmeringsspråk och lagrar data i en relationsdatabas. Dock krävs det att databasen är designad med relationer mellan tabeller för att man ska få full effekt av Entity Framework.

Entity Framework upplevs enkelt att lära sig för systemutvecklare som har kunskaper i något av programmeringsspråken som kan användas tillsammans med .NET-plattformen, eftersom det finns mycket inlärningsmaterial att tillgå för Entity Framework på Internet. Att arbetssättet blir mer objektorienterat är också bidragande att det inte upplevs som något stort steg att börja använda Entity Framework.

8 Författarnas reflektioner

Denna studie har gett oss ord på problem som vi har upplevt när vi har utvecklat applikationer i tidigare projekt då vi inte har använt oss av ORM-ramverk. Vi har fått förståelse för varför vi har haft de problemen vi har haft och hur vi ska hantera dem. Vi tror att denna rapport ska kunna vara till nytta för andra som vill använda sig av Entity Framework.

8.1 Rapportens relevans

Rapporten känns väldigt relevant då object-relational mismatch är något som alla utvecklare som använder objektorienterade programmeringsspråk och relationsdatabaser kommer i kontakt med. Att Microsoft nyligen lanserat Entity Framework gör också att vi tycker det är relevant att undersöka just det ORM-ramverket. Vi upplever också att .NET-plattformen är den dominerande, åtminstone är den det på Sogeti och känslan är att det är så i hela Sverige.

Frågeställningen tycker vi känns relevant relaterat till syfte och bakgrund eftersom frågorna ger möjlighet till svar som bidrar till att syftet uppnås.

8.2 Resultatets kvalité

Att vi har använt oss av flera olika källor under vår litteraturstudie av object-relational mapping bidrar till en högre kvalité på det vi skrivit. Även att vi har använt oss av vetenskapliga artiklar gör att kvalitén ökar. Det då de är granskade innan de publicerats.

Eftersom vi endast är två personer som undersökt Entity Framework skulle man kunna se det som en svaghet. Hade vi låtit fler personer utvecklat med Entity Framework hade vi kunnat jämföra deras upplevelser med våra egna upplevelser och därmed fått ett tyngre resultat. Men vi anser ändå att vi har fått en bra bild om Entity Framework, då vi har utvecklat applikationer i flera år både på utbildningen på högskolan, i privata projekt och har erfarenheter av att arbeta med ett objektorienterat programmeringsspråk och relationsdatabaser. Dock så har vi inte använt oss av några ORM-ramverk innan denna studie påbörjades. Detta gör att vi inte har haft någon bild av hur ORM-ramverk är att använda vid utveckling av applikationer.

Det resultat vi kommit fram till angående inlärningen av Entity Framework anser vi vara av hög kvalitet. Detta då vi befann oss i det stadiet innan vi började inlärningen av Entity Framework som de som vi riktar rapporten till gör. Nämligen till personer som redan innehar kunskaper i .NET-plattformens objektorienterad programmering.

8.3 Förslag till vidare studier

Då vi endast undersökt Entity Framework ur ett perspektiv som fokuserar på hur det är för utvecklare att utveckla applikationer med hjälp av Entity Framework har vi utelämnat flera perspektiv. En sak som man skulle kunna undersöka vidare är hur lösningen att använda Entity Framework för att hantera object-relational mismatch påverkar prestanda på applikationer. Man skulle kunna studera hur det ur en prestanda synpunkt skiljer sig mot att inte använda något ORM-ramverk eller att jämföra mot andra ORM-ramverk.

9 Referenser/Källförteckning

9.1 Litteratur

Deitel P.J, Deitel H.M (2007)
Java, How to program, seventh edition. Deitel & Deitel

Backman J (1998)
Rapporter och uppsatser. Studentlitteratur, Lund

Halilovic A (2006)
Ett praktikperspektiv på hantering av mjukvarukomponenter. Linköpings universitet

I. Letovsky S (1999)
Bioinformatics: Databases and Systems. Kluwer Academic publicher group

King K, A. Jamsa, K
SQL Tips and Techniques. Premier Press Inc

Ireland C, Bowers D, Newton M, Waugh K (2009)
A Classification of Object-Relational Impedance Mismatch. The Open University, Milton Keynes, UK

Ubaid M, Atique N, Begum S (2009)
A pattern for the effective use of object oriented databases.

Pettersson T (2009)
Utvärdering av ADO.NET Entity Framework. KTH Stockholm

Yemelyanov A (2008)
Using ADO.NET Entity Framework in Domain-Driven Design: A Pattern Approach, ISSN: 1651-4769, Chalmers Universitet Göteborg

Halpin T (2006)
Handbook on Architectures of Information Systems, Springer Berlin Heidelberg

9.2 Elektroniska källor

Microsoft MSDN - 2010-03-30

http://msdn.microsoft.com/en-us/library/aa697427%28VS.80%29.aspx

2010-04-26

http://msdn.microsoft.com/en-us/library/w0x726c2%28v=VS.100%29.aspx

2010-05-05

http://msdn.microsoft.com/library/aa697427%28VS.80%29.aspx#ado.netenfrmovw_topic4

2010-05-24

http://msdn.microsoft.com/en-us/library/aa697427%28VS.80%29.aspx

Sogeti - 2010-04-01

http://www.sogeti.se/Om-Sogeti/Historik/

http://www.sogeti.se/Om-Sogeti/Foretagspresentationer/

MSDN Channel 9 - 2010-05-04

http://channel9.msdn.com/posts/johanlindfors/MSDN-TV-Introduktion-till-Entity-Framework/

Neward T (2006) - 2010-05-22

http://blogs.tedneward.com/2006/06/26/The+Vietnam+Of+Computer+Science.aspx

The Java Tutorials – 2010-05-23

http://java.sun.com/docs/books/tutorial/java/IandI/subclasses.html

http://java.sun.com/docs/books/tutorial/java/IandI/polymorphism.html

9.3 Personer

Nilsson, Karl-Henrik – 2010-05-24
IT-konsult, Sogeti Sverige AB

Bilaga 1

Den 24/5 – 2010 genomförde vi en intervju med Karl-Henrik Nilsson, IT-konsult på Sogeti Sverige AB.

Vad tycker du om Sogeti .NET Pattern?
Om man jämför med att skriva allt själv så är det bra. Men om man jämför med det jag sett av Entity Framework så är det krångligare.

Var det svårt att lära sig att använda Sogeti .NET Pattern?
Det var inte direkt svårt men det tog rätt långt tid.

Hur lång tid tog det?
Det tog ungefär en vecka att lära sig .NET Pattern. Vilket jag tycker är lite för mycket för att lära sig ett ramverk.