Institutionen för
Tillämpad fysik och elektronik
Stefan Berglund

2014-10-31

Laboration:
Datahantering

Målsättning

Syftet med denna laboration är att ge grundläggande kunskap och erfarenhet runt olika tekniker för att hantera data som lagrats i SQL Server. Centrala moment i laborationen är lagrade procedurer, ADO.NET samt Transaktionshantering.

Teori

Undervisningsfilmer och eget kunskapsinhämtande på Internet.

Material

Visual Studio och SQL Server.

Genomförande

Laborationen innehåller fyra grunduppgifter som kan genomföras individuellt eller i grupper om två studenter och vi förutsätter att de flesta av er baserar arbetet på ASP.NET och MVC. Det finns möjlighet att ta ut svängarna och utveckla i MVC men i alternativa miljöer, förankra i sådana fall detta hos handledande lärare.

För plus (VG) på denna laboration krävs det dessutom att du även löser de två fördjupningsuppgifterna som återfinns.

Tips:
Användbarhet och gränssnittsdesign är både viktiga ingredienser vid systemutveckling, men detta är inte fokus för denna laboration. Här ska ni istället hitta lösningar som klart och tydligt visar att ni förstått hur olika tekniker fungerar och hur man kan använda dem.

Redovisning

Redovisning av laboration sker individuellt. Laborationen redovisas dels genom en individuell rapport som beskriver din/er lösning, men även genom uppvisande av en väl fungerande applikation. I rapporten ska det klart och tydligt framgå om man löst laborationen på G- eller VG-nivå. Man ska kunna utläsa att och hur du löst ingående uppgifter och man ska hitta svar på de reflektionsuppgifter som återfinns. Rapporten ska i den mån det går följa den rapportmall som återfinns på kurshemsidan. Muntliga redovisningen sker antingen i laborationssal, via Skype eller via inspelade Jing-filmer.

Uppgift 1

I denna uppgift ska du skapa en enkel relationsdatabas bestående av minst två relaterade tabeller. Har du en passande databas sedan tidigare så är det helt okej att använda sig av denna.

Mot databasen utvecklar du sedan ett enkelt webbaserat gränssnitt som baserar sig på ASP.NET MVC och Entity Framework (EF), du förväntas med andra ord arbeta enligt metoden Database First. Gränssnittet ska ha enkel CRUD-funktionalitet, det vill säga man ska kunna utföra Create, Read, Update och Delete från gränssnittet mot i alla fall en av tabellerna i databasen.

Uppgift 2

Du ska nu bygga ut din applikation genom att skapa en vy där data från två relaterade tabeller i databasen visas i en och samma tabell i webbsidan. I vyn ska man kunna filtrera det data som visas (filtrering sker förslagsvis med hjälp av en textbox och en knapp eller med hjälp av en dropplista).

Den utökade funktionaliteten ska basera sig på egenkonstruerade Data Access Object (DAO) och ADO.NET (direkt, inte via EF) och den ska komplettera den lösning du utvecklat i Uppgift 1.

Uppgift 3

Databasen du gjorde i Uppgift 1 kompletterar du i denna uppgift med minst en lagrad procedur (Stored Procedure). Den lagrade proceduren ska ta emot en in-parameter som ska användas till filtrering och den ska likt fallet i Uppgift 2 hämta data från två tabeller. Du skapar en likadan vy i denna uppgift som i den du gjorde i Uppgift 2.

Reflektionsuppgift:
Är lösningen med en lagrad procedur att föredra mot Uppgift 2?
Fördelar? Nackdelar?

Det finns något som heter Vy som man kan skapa i SQL Server vilka för- respektive nackdelar har dessa mot Lagrade Procedurer?

Fördjupningsuppgift 1

Om du, som i som i denna laboration, arbetar med metoden Database First så automatgenererar Entity Framework data modeller (Models) utifrån en databas.

Om databasen har konstiga tabellnamn erhåller dina modeller lika konstiga namn och kanske vill du i din Vy vill visa för och efternamn i en och samma kolumn men de är i olika kolumner i databasen.

Det man brukar göra då är att skapa View Models som man kan säga fugnerar som gränssnitt mellan dina vyer och de modeller som EF genererat.

I denna uppgift ska du skapa sådana View Models för din databas. Du behöver inte ha något meningsfullt syfte med detta, det viktiga är att du skapar dem, att du visar att du förstått hur man gör.

I väldigt omfattande system kan detta arbete vara tidskrävande, då finns det verktyg som kan hjälpa en med detta. Exempelvis hittar ni verktyget Automapping via Nuget. Anser du dig ha lite tid över rekommenderas det att du tittar på Automapping eller liknande för denna uppgift.

Reflektionsuppgift:
Vad händer om någon andrar utseendet på databasen? Modellerna kan uppdateras automatiskt med hjälp av EF, vad händer med mappningen?

Uppgift 4

Transaktionshantering kan ske hos databashanteraren och det kan ske i er applikation. I denna uppgift ska ni skapa en lagrad procedur som utför minst två transaktioner mot er databas, ingen av transaktionerna ska gå igenom om den andra fallerar. Transaktionshanteringen ska i denna uppgift hanteras av SQL Server.

Ni ska konstruera denna uppgift så att det lätt går att visa att er transaktionshantering fungerar.

Reflektionsuppgift:
Transaktionshantering kan också ske i din applikation.
Vart ska man lägga sin transaktionshantering? Finns det för- respektive nackdelar?

Fördjupningsuppgift 2

Till denna uppgift behöver du först utveckla ytterligare en databas (den behöver inte innehålla mer än en enkel tabell).

Skapa sedan en applikation (eller bygg ut den du redan har) som utför transaktioner (insert eller update) mot de två databaserna och säkerställ att båda transaktionerna går igenom, eller ingen. I denna uppgift hanteras transaktionerna av din applikation.

Även denna uppgift konstrueras så att det lätt går att visa att transaktionshanteringen fungerar som den ska.