# WEBBUTIK BACKEND

Entity Framework For The Win!

MARS 2021

NET20D

Inlämning 2, Marcus Medina

# Inlämning 2 – Entity Framework FTW!

## Innehåll

## Beskrivning av projektet

En kund som säljer begagnade böcker behöver en webbshop. För att snabba upp skapandet av projektet har hon outsourcat projektet i två delar. En back-end och en front-end.

Du ska ta hand om backend-delen. Vad innebär detta då? Det innebär att du ska skapa en databas med Entity Framework Code First och skapa en del sökningar som ska göras i denna, baserat på input från användare från Front-end delen. Font end programmeraren kan inget om databaser och behöver därför specialfunktioner som kommer att anropas via C#.

Vad behövs för projektet ska fungera? Här kommer en beskrivning på projektet.

APIn ska heta **WebbShopAPI**
Databasen ska heta **WebbShop + DittNamn,** exempelvis WebbShopMarcusMedina

## Databas

Grund för tabeller och kolumner, fler kolumner läggas till om man vill.

| Tabell | Förklaring | Kolumner |
|---|---|---|
| **Users** | Kundtabell | ID<br>Name<br>Password, default: Codic2021<br>LastLogIn (dateTime)<br>SessionTimer<br>IsActive (default true)<br>IsAdmin (default false) |
| **BookCategory** | Kategorier för böckerna | Id<br>Name |
| **Books** | Alla böcker, dessa kopplas till kategorin | Id<br>Title<br>Author<br>Price<br>Amount<br>CategoryId |
| **SoldBooks** | Kopia från Books men kopplade till kund | Id<br>Title<br>Author<br>CategoryId<br>Price<br>PurchasedDate<br>UserId |

## Grundmaterial

Vid första programstart ska databasen skapas och följande kunder och böcker ska ska genereras (Id finns inte med i tabellerna för det får EF fixa automagiskt till oss)

| Namr | Password | IsAdmin |
|---|---|---|
| **Administrator** | CodicRulez | True |
| **TestCustomer** | Codic2021 | False |

| Name |
|---|
| **Horror** |
| **Humor** |
| **Science Fiction** |
| **Romance** |

| Title | Author | Price | Amount | CategoryId |
|---|---|---|---|---|
| **Cabal (Nightbreed)** | Clive Barker | 250:- | 3 | Horror |
| **The Shinng** | Stephen King | 200:- | 2 | Horror |
| **Doctor Sleep** | Stephen King | 200:- | 1 | Horror |
| **I Robot** | Isaac Asimov | 150:- | 4 | Science Fiction |

Man kan leta upp kategorins ID genom exempelvis såhär: `Categories.FirstOrDefault(c=>c.Name=='Horror')`

När en kund vill köpa en bok måste denne först ha ett konto och vara inloggad. Kunden kan köpa varor (om de är tillgängliga). När köp görs (vi kontrollerar inte betalning just nu) så kommer böckerna att flyttas till SoldBooks, dit kopieras all information om boken plus att datum och kundnummer läggs till. Sedan minskas antalet tillgängliga böcker.

# API

API ska förse användarna med följande funktioner

| Function | Parameter | Returns | Action |
|---|---|---|---|
| **Login** | Username, password | User Id if success<br>Nothing if fail<br>Set SessionTimer | Loggar in användare |
| **Logout** | User Id | Sets SessionTimer to DateTime.Default | |
| **GetCategories** | | List all categories | |
| **GetCategories** | Keyword | List of categories matching keyword | |
| **GetCategory** | Category Id | List of books in category | |
| **GetAvailableBooks** | Category Id | List of books with amount>0 | |
| **GetBook** | Book Id | Information about book | |
| **GetBooks** | Keyword | List of matching books | |
| **GetAuthors** | Keyword | List of matching books | |
| **BuyBook** | User Id<br>Book Id | True if book purchase is OK | Check Session Timer<br>Fail om user inte finns<br>Kopierar bokdata till soldbooks, fyller på med datum och användarId |
| **Ping** | User Id | "Pong" if customer is online<br>string.empty is customer is offline | Ping för att hålla anslutningen vid liv |
| **Register** | Name<br>Password<br>PasswordVerify | True if user is created<br>False is user already exist<br>False is password != verify | Skapar en kund |

Vid varje anrop uppdateras den inloggade användarens SessionTimer till DateTime.Now, men om den inte uppdaterats på över 15 minuter kommer användaren att anses som utloggad och då kommer denne inte längre att kunna handla böcker förrän inloggning har gjorts igen.

## Administrator access to API

Inloggad administratör (User.IsAdmin==true) får också tillgång till andra funktioner

*AdminID är den inloggade admins User Id egentligen*

| Function | Parameter | Returns | Explanation |
|---|---|---|---|
| **AddBook** | Admin Id<br>Id<br>Title<br>Author<br>Price<br>Amount | True if success<br>False if fail | Öka book.amount om boken redan finns, annars sätt book.amount till antal som skickades in<br><br>adminId är den inloggade användarens Id |
| **SetAmount** | Admin Id<br>Book Id | Sets amount of available books | |
| **ListUsers** | Admin Id | List of users | |
| **FindUser** | Admin Id<br>Keyword | List of matching users | |
| **UpdateBook** | Admin Id<br>Id<br>Title<br>Author<br>Price | True if success<br>False if fail | |
| **DeleteBook** | Admin Id<br>Book Id | True if success<br>False if fail | Minska book.amount, om den blir 0 radera boken |
| **AddCategory** | Admin Id<br>Name | True if success<br>False if fail | |
| **AddBookToCategory** | Admin Id<br>Book Id<br>Category Id | True if success<br>False if fail | |
| **UpdateCategory** | Admin Id<br>CategoryId,<br>Name | True if success<br>False if fail | |
| **DeleteCategory** | Admin Id<br>CategoryId | True if success<br>False if fail | Fails om kategorin inte är tom |
| **AddUser** | Admin Id<br>Name<br>Password | True if success<br>False if fail | Fails om user redan finns<br>Fails om lösenord är tom |

## VG Funktionalitet

| Function | Parameter | Returns | Explanation |
|---|---|---|---|
| **SoldItems** | Admin Id | Lists all sold books | |
| **MoneyEarned** | Admin Id | How much money was earned by sold books | |
| **BestCustomer** | Admin Id | Customer that bought most books | |
| **Promote** | Admin Id<br>User Id | True if success<br>False if fail | Fail om user inte finns |
| **Demote** | Admin Id<br>User Id | True if success<br>False if fail | Fail om user inte finns |
| **ActivateUser** | Admin Id<br>User Id | True if success<br>False if fail | |
| **InactivateUser** | Admin Id<br>User Id | True if success<br>False if fail | |

Programmet ska inte ha någon meny eller gränssnitt, dock ska man kunna testa flödet från Program.Main(), det räcker gott med bara metodanrop utan inputs från användaren

**Exempel 1:**

1. Logga in Testuser
2. Fråga efter kategorier
3. Välj kategori Horror
4. Lista böcker i kategorin
5. Välj boken Dr Sleep
6. Hur många finns tillgängliga?
7. Köp bok
8. Kontrollera antal böcker som finns tillgängliga
9. UML Diagram som visar dina klasser
10. ER Diagram som visar dina tabeller

**Exempel 2:**

1. Logga in som Admin
2. Skapa kategori
3. Flytta en bok dit

**Exempel 3:**

1. Logga in som Admin
2. Skapa en användare

## Inlämningskrav

# Krav för godkänd

1. Clean Code
2. Databasen ska göras med Entity FrameWork och Linq ska användas för sökningar
3. Mappar för klasser i MVC stil (Database, Models, Helpers etc), API klassen ska ligga i roten
4. Alla metoder implementeras och returnerar det de ska
5. Metoderna får inte orsaka NULL errors (använd try catch)
6. Alla tabeller skapas
7. Grunddata skapas
8. XML kommentarer i APIn
9. UML och ER Diagram

# Krav för väl godkänd

1. Att alla krav för godkänd uppfylls
2. XML kommentarer i alla klasser
3. VG funktionalitet implementeras och returnerar det de ska

# Disclaimer

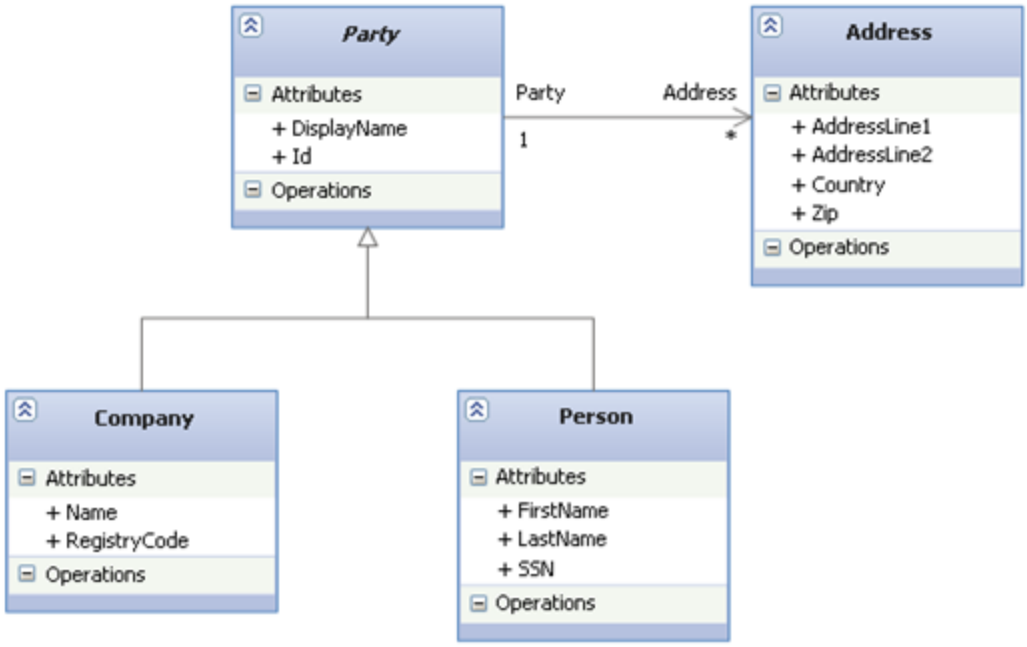
Även om projektet är tänkt att vara realistisk har jag minskat ner funktionaliteten, så att lösenord inte behöver krypteras, sessionsvariabler skippas, kundvagn och betalning hoppas över och böckerna har bara en kategori.

Dessa funktioner kan ni själva bygga in efter inlämning om ni känner att det ger en bra grund för ett riktigt projekt.

Fokus här ligger på en bra databas och Linq sökningar.

## Exempel på UML Diagram

(inte relaterat till projektet)




## Exempel på ER diagram

(inte relaterat till projektet)