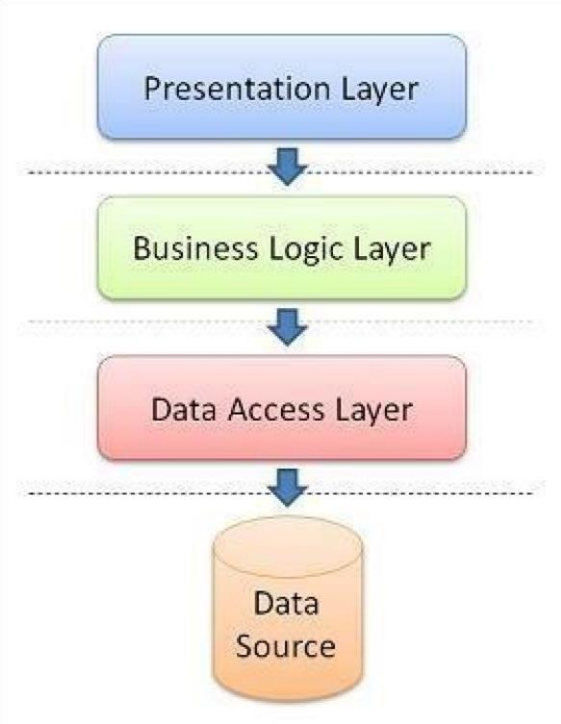
Av **Erik ”Zkall” Svensson**
@esdev.se

3 Lagers Arkitektur

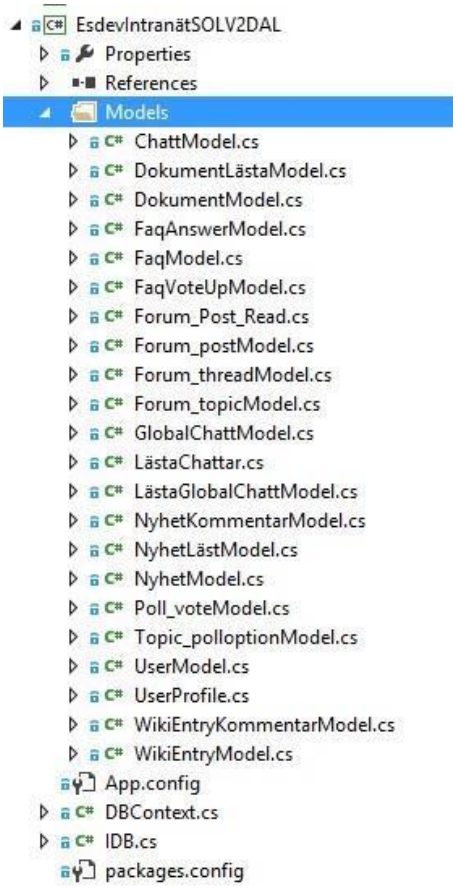


3 lagers arkitektur innebär att du delar upp koden i 3-4 lager. Dessa lager är:

1. Presentations lagret som kallar på funktioner från
2. Affärslagret där all logik sker med hjälp utav
3. Data åtkomst lagret. Där man ibland på en annan server använder sig av
4. Databas server.

Observera att det här EJ BETYDER att du delar upp koden på 4 olika maskiner, du kan göra det men häng inte upp dig på det. Det här är en arkitektur som handlar om att få mer ren kod. För nu så antar vi att alla 4 ligger på samma server. Hur ser då detta ut i verkligheten?

Här har jag delat upp i 3 lager, går in lite djupare på dom i underrubriker. För att göra det enkelt, du skapar ett projekt precis som vanligt när du jobbar med 3 lager. Du använder dig utav Class librarys som vid kompilering blir dll filer, du får då en dll för varje lager i presentations lagret (en hemsida



eller wcf). När du kompilerar så kommer du enbart att få filer för det presentations lager du har. Så det är inget avancerat i det här.

DAL (Data Access Layer)

Här sker alla databas kopplingar, här lägger du allt som har med databasen att göra det vill säga databas modelerna, databas contexten etc.

Så här kan en DAL se ut, en mapp för alla models en context med ett interface skapat utav db classen för att göra den möjlig för dependency injection. Om du inte läst om dependency injection rekommenderar jag er att kolla på den pdfen.

Även fast du har en app.config fil här så använder du aldrig connection string i datalagret utan det sker i presentations lagret.

TIPS: När du skapaat alla modeler för databasen kopiera och klistra in dom i affärslagret och ändra Model till VM. Då har du sparat lite tid på att knappra in classer igen.

OBS! DAL ska refererras ifrån affärslagret.

BLL (Business Logic Layer)

Det är här som all logik sker, aka det roliga lagret.

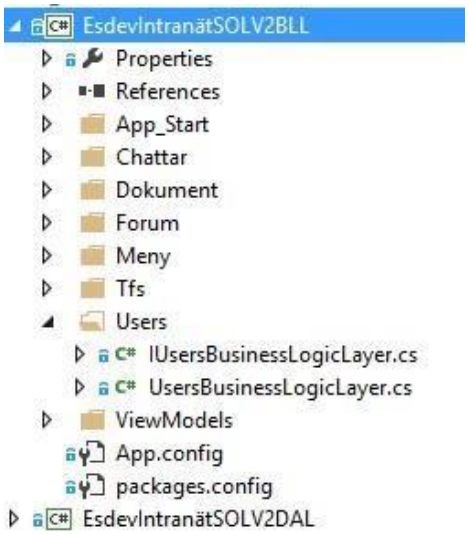
Många if, mycket spaghetti.

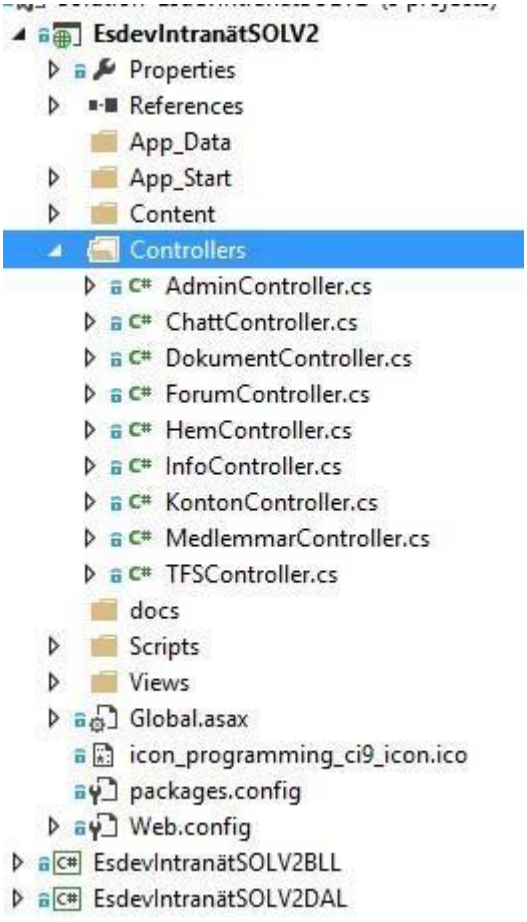
Använd er av många metoder och splittra upp i olika classer om det är ett större projekt.

Samma här som i DAL så använd er utav interfaces för varje class så kan ni injecta dom som dependencys, jag kör med UnityContainer och MVC Bootstrap nugget paketen här för att kunna injecta dependencys i affärslogiken redan.

För mig så börjar jag jämt med DAL efter att ha gjort en databas skiss på behov av databasobjekt. Efter det går jag till Affärslogiken och lägger till en funktion för varje
sak jag kan tänkas vilja göra med mina databas objekt, t.ex. updatera en viewmodel till en databasmodel och spara i contextet.

Ta bort, Lägg till etc. När väll allt det är klart går jag till:





Presentationslagret (MVC app t.ex.)

Här så presenterar du allt ditt arbete och nu är den tuffa delen klar, allt du behöver göra nu är att använda dina metoder från BLL. Kom ihåg att lägga till BLL som referens i presentations referenslistan.

I controllerna så använder du dig bara av att implementa interfacet för varje klass på rätt ställen och kallar sedan bara på funktionera du skapade i fas 2 av denna pdf.

Om du jobbat med större projekt tidigare kommer du märka att du får grymt mycket renare kod, 2-3 rader per action som är grymt bra.

Avslut

Kom ihåg att 3 lagers arkitekturen kan vara svår att koppla i början så blir inte alltför besvikna om ni inte fattar på igen gång. Men om ni fortsätter banka huvudet mot det så fattar man tillslut.