Den første rudimentære udgave af modellen for tabellen for brugere: Users.

Der er to forskellige udgaver af prim og erAdmin, da datatypen i databasen er tekst, men rent logisk er det en boolsk værdi. C# kan ikke blande datatyper ved get/set, så derfor er der to properties for hver værdi. Disse properties bliver synkroniseret ved set.

Der er oprettet en metode, hentBruger, men som det fremgår, er den aldeles triviel og ganske værdiløs; bortset fra, at den er nødvendig for at kunne konstruere en test.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace VoresJazzklubSomMVC.Models {

public class UsersTableModel {

private bool \_prim, \_erAdmin;

private char \_primChar, \_erAdminChar;

public string brugerId { get; set; }

public DateTime createTs { get; set; }

public string password { get; set; }

public string salt { get; set; }

public string email { get; set; }

public bool prim {

get { return \_prim; }

set { \_prim = value; \_primChar=(\_prim)?'T':'F'; }

}

public char primChar {

get { return \_primChar; }

set { \_primChar = value; \_prim = \_primChar == 'T'; }

}

public bool erAdmin {

get { return \_erAdmin; }

set { \_erAdmin = value; \_erAdminChar = (\_erAdmin)? 'T':'F'; }

}

public char erAdminChar {

get { return \_erAdminChar; }

set { \_erAdminChar = value; \_erAdmin = \_erAdminChar=='T'; }

}

public void hentBruger(string brugerId) {

}

}

}

Tanken med klassen er, at når metoden hentBruger bliver kaldt, skal den finde brugeren på MySQL-databasen i tabellen voresjazzklub.Users og sætte klassens properties i overensstemmelse med de fundne data.

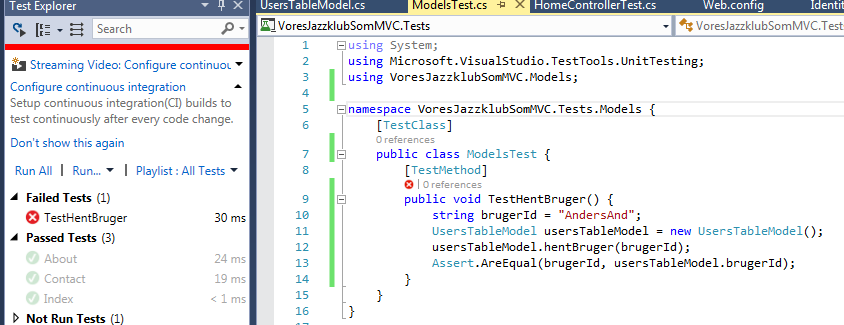
Følgende script køres i phpmyadmin web-siden i xampp mod MySQL for at have noget testdata. Senere vil det komme til at indgå i et test-script, således, at tabellen tømmes før testserierne og undervejs vil de enkelte rækker blive lagt ind, ændre og slettet for at gennemføre test af controllers og models.

insert into users(`brugerId`, `createTs`, `password`, `salt`, `email`, `prim`, `erAdmin`)

values('AndersAnd', CURRENT\_TIMESTAMP, 'PasswordDerSkalHashes', 'medNogetSalt', 'a.and@andeby.utopia', 'T', 'F');

En testklasse oprettes til at undersøge, om en given person rent faktisk kan findes. I metoden TestHentBruger skrives navnet på en bruger, der skal kunne findes. Der oprettes en instans af klassen, der skal testes, og herefter kaldes, den metode, der skal testes. Til sidst sammenlignes brugerId i testklassen med brugerId i den testede klasse.

I nedenstående skærmdump ses tydeligt, at testen fejlede. Ikke at det kommer som en overraskelse, da der jo netop ikke er nogen kode til at udføre det ønskede arbejde. Hvis det havde været en stor, kompleks applikation med dependency injection i configuration filer og andre sløringer af, hvor data kommer fra, ville situationen kunne se anderledes ud.



Man starter med testmetoden. IDE’et vil ikke kunne kompilere testmetoden, da der hverken er en klasse eller metode IDE’et kan finde referencer til. Så laves det minimum, der skal til at kunne kompilere, hvorefter testen køres.

De næste trin er at tilføre og ændre koden, i den testede metode, indtil testen bliver grøn. Alle test, til de berørte komponenter, skal også køres igen, for at se, om man har fået ødelagt noget.

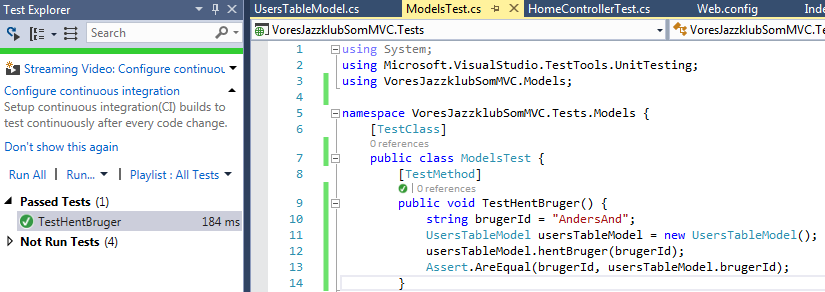
Sidste step er at checke sourcen incl test ind i ens versionsstyringssystem, hvortil der knyttet et Continuous Integration (CI) set-up [CI], hvilket betyder, at der bliver kørt et komplet build og alle test bliver kørt. Testene skal så gerne være grønne alle sammen.

Til et miniature skole/hobby-projekt, virker CI som overkill, men tanken om at indlede med en test ,og herefter ændre koden og teste indtil tingen virker, kan man sagtens gennemføre på udviklingsmaskinen, og så en gang imellem køre samtlige test.

Efter lidt tilføjelse af kode og variable, blev testen grøn og glad. Resultatet er ganske vist ikke kønt at se på, der mangler noget kodning, og fejlhåndteringen skal ændres. Det skal også overvejes, hvor connection.Close(); skal indsættes; men hovedformålet, at få en grøn test, blev nået.

I koden refereres klassens properties prefixed med this. for ikke at ramme metodens variable i stedet. Ser man godt efter, konstateres to forskellige farver på this. Det er fordi det kun er ved this.brugerId, hvor det er nødvendigt, men hellere skrive det nogle flere gange fremfor at buge tid på fejlfinding.





[CI] https://martinfowler.com/articles/continuousIntegration.html