Eksamensopgave 2016

Noter til programmet

Generics; Indeholder interfaces og klasser der definerer generic Collections. Som kan give mulighed for at lave stærkere type Collections der giver bedre type sikkerhed og performance i modsætning til ikke generic-collections.

Delegate; Is a type-safe function pointer. Er det kommunikerbare led mellem two parties. Leddet kommunikerer gennem callbacks. Man kan sagtens bare kalde en funktion direkte men det giver ikke samme fleksibilitet. Kan bruges til at håndtere genbrugelig kode ved at bruge en delegate som en funktionsparameter.

DynamicInvoke; Forskellen mellem dynamic invoke og invoke er at Invoke er meget hurtigere og bruges hver gang du kender typen af delegaten du arbejder på. Hvis man ikke kender typen bruger man DynamicInvoke der skal udpakke alt information selv og basically udføre mange flere operationer for det samme resultat.

```
private Dictionary<string, Delegate> adminCommand;
```

I controller klassen ser vi en dictionary med en string og en ukendt delegate som parametre. Derfor bruges dynamicinvoke i stedet for invoke, hvis vi havde lavet 3 delegates med forskellige parametre og brugt i switchen i stedet og brugt invoke ville det have været bedre optimerings mæssigt.

Regex; Regex er en RegularExpression. Regex har en række af metoder med forskellige parametere der håndterer typer, som regel strings. Ligesom Method.Assert i testmethod. Regex.Replace i mit program har syntaxen: Regex.Replace(name [(**en string**)], ***"<.*?>"*** [(**en string også, altså checker min name string for disse tegn**)], string.Empty [(**en tom string**)].

Regex på mail adressen kunne bruge System.Net.Mail.MailAddress.

218

TLD's like .museum aren't matched this way, and there are a few other long TLD's. Also, you can validate email addresses using the MailAddress class as Microsoft explains here in a note:

> Instead of using a regular expression to validate an email address, you can use the System.Net.Mail.MailAddress class. To determine whether an email address is valid, pass the email address to the MailAddress.MailAddress(String) class constructor.

```
public bool IsValid(string emailaddress)
{
    try
    {
        MailAddress m = new MailAddress(emailaddress);

        return true;
    }
    catch (FormatException)
    {
        return false;
    }
}
```

This saves you a lot af headaches because you don't have to write (or try to understand someone else's) regex.

Get Set; Get aflæser noget, set writer noget. En property med kun en **Get** er read-only. En property med kun en **Set** er write only. Jeg bruger en private variabel og en public Get Setter. Dvs jeg kan acces den private gennem en get setter.

```
private Product productitem;
public Product productItem
{
    get
    {
        return productitem;
    }
    set
    {
        productitem = value;
    }
}
```