Dagens emner

- Custom objecter og MVC
  - Bruge klasser fra ASP.NET framework og bruge dem i MVC
  - Skrive vores egne klasser og vise custom klasser i MVC
  - Mere razor træning

I dag handler mest om ”hands-on” træning for at få rutinen med Asp.Net MVC – ikke så mange nye ting (dog lidt i øvelserne)

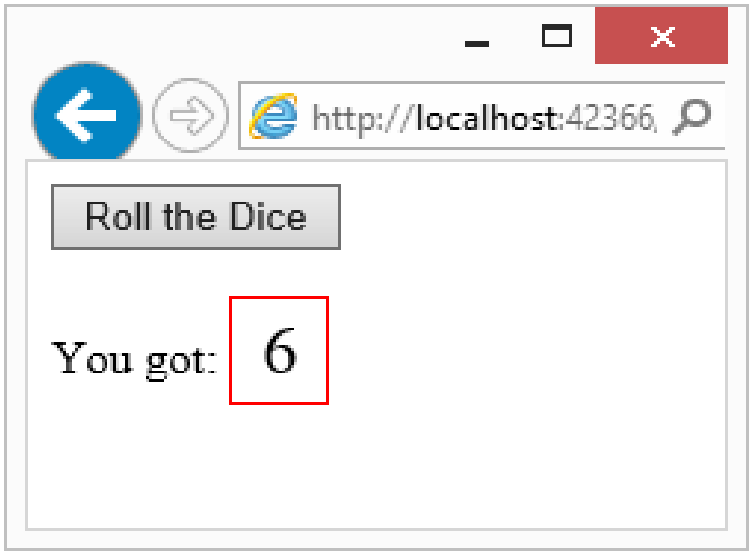
Random class

- Brug af **`Random`** class (kan bruges til en terning)

```
// Create a new Random obj.
Random rnd = new Random();

// Returns a random number between 1 and 7
(exclusive)
int i = rnd.Next(1, 7);

// Returns a random number between 0.0 and 1.0
(exclusive).

double d = rnd.NextDouble();
```

En opvarmningsøvelse!

1. Byg en controller (**`DiceRoller`**) og et simpelt view med en html submit button
2. Knappen skal emulere en dice roller. Hver gang man trykker skal et tilfældigt tal mellem 1 og 6 vises
3. Skriv og test programmet i browseren.



Brug af custom klasser og razor in MVC:

Product eksempel

```
public class Product
{
   private string name;
   private double price;
   private string imageUrl;
}
```

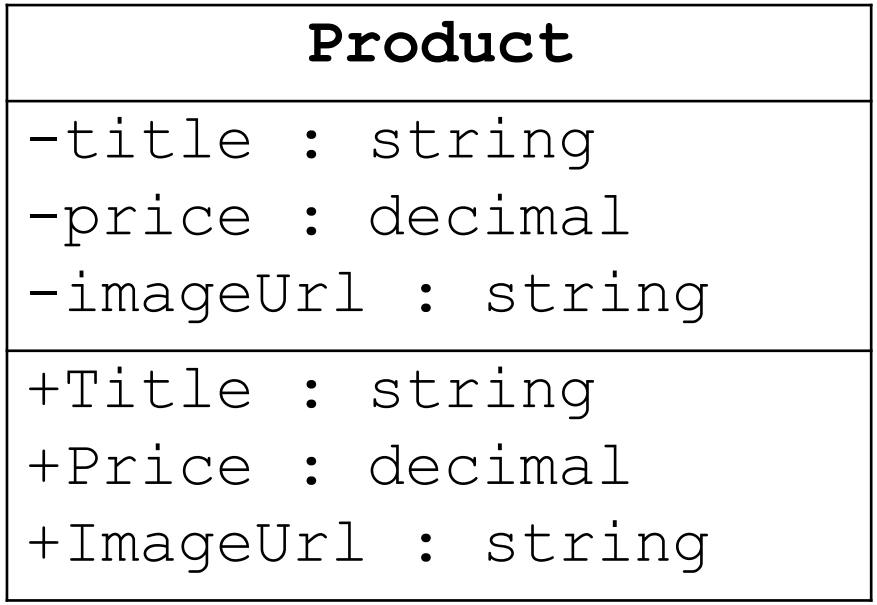
Product

- Vi tilføjer Properties til at få vores fields (eller brug auto-implemented properties):

```
private string name;

public string Name
{
  get { return name; } // get accessor method
  set { name = value; } // set accessor method
}
```

- Og det samme for **`price`** **`,imageUrl`**

UML class diagram



Tjek om prisen giver mening ved setter

```
public double Price
{
   set {
        if (price <= 0)
        {
               throw new Exception("Price is not accepted");
        }
        else
        {
               price = value;
        }
   }
   get { return price; }
}
```

Constructor

- Vi tilføjer også en constructor

```
public class Product
{ ...
   public Product(string name, double price, string
  imageUrl)
    {
        this.name = name;
        this.price = price;
        this.imageUrl = imageUrl;
    }
}
```

Brug af en metode til at genere HTML for Product klassen

- En måde at gøre det på er at tilføje en metode til produkt klassen, sådan at vi kan vise produktklassen i browseren. En slags toString(), som laver HTML i stedet for og som kan vises inden i et **<body>** tag
- Kald metoden for GetAsHtml() og vi vil gerne kunne kalde den sådan her:

```
string html = knife.GetAsHtml();
ltrProduct.Text = html;
```

Erklæring af metoden:

```
public class Product
{ ...
  public string GetAsHtml()
  {
    string html;
    html = "<h1>" +name+ "</h1>";
    html+= "<h3>Cost: " +price+ "</h3>";
    html+= "<img src=\""+imageUrl+"\" />";
    return html;
  }
```

Prøv det selv i opgave 1!

Product
-title: string -price: decimal -imageUrl: string +Title: string +Price: decimal +ImageUrl: string
+Product(title:string,price:decimal) +Product(title:string,price:decimal,imageUrl:string) +GetAsHtml(): string

Brug af lister og Razor (recap fra sidst)

Eksempel på en liste

```
using System.Collections.Generic;
...
// instantiates a new List object
List<string> myFriends = new List<string>();

// Use the method Add to add new elements to
  the list
myFriends.Add("Reno");
myFriends.Add("Lisa");
myFriends.Add("Michael");
myFriends.Add("Susan");
myFriends.Add("Peter");
```

Vise en liste i Razor

```
string s = "";

for (int i = 0; i < myFriends.Count; i++)
{
   @myFriends[i] + "<br />";
}
```

Vise en liste i Razor med foreach

```
string s = "";

foreach (string friend in myFriends)
{
      @friend + "<br />";
}
```

Opgaver 1 – 6 (virkelig god træning af MVC, Razor og objekter – det skal vi bruge meget senere!)