Упражнение: Реализиране на MVC приложение

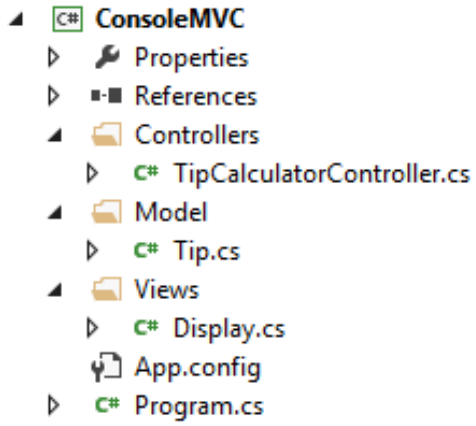
В рамките на това приложение ще създадем малко примерно конзолно MVC приложение за изчисление на сметката в ресторант.

Структура

Създайте проект за конзолно приложение **ConsoleMVC**. В него добавете следните директории:

- **Program.cs –** Главна програма, нищо по-различно от обичайното ☺
- **Controllers** – Папка, в която ще се съхраняват **контролер класове**
  - Създайте **TipCalculatorController.cs**
- **Model** – Папка, в която ще се съхраняват **моделите**
  - Създайте **Tip.cs**
- **Views** – Папка, в която ще се съхраняват **изгледите**
  - Създайте **Display.cs**

В случай, че сте работили правилно резултатът би трябвало да изглежда по сходен начин:



Модел

Тук ще опишем **данните**, с които ще боравим в рамките на това приложение. Това са **цената на поръчката** и **процента бакшиш**, който желаем да оставим. Ще създадем полета, свойства и конструктор, които да обслужват тези данни, а също така и два публични метода – **CalculateTip()** и **CalculateTotal()**, които изчисляват съответно стойността на бакшиша и общата сума за плащане, заедно с бакшиша.

Полетата и свойствата тук са по-скоро стандартни. Единствената особеност е свойството за процент – то се грижи да превърне въведената информация в процент от 0 до 1, в случай че е въведено по-голямо число. Кодът е както следва:

```
    private double amount;
    private double percent;
    public double Amount
    {
        get { return this.amount; }
        set { this.amount = value; }
    }
    public double Percent
    {
        get { return this.percent; }
        set
        {
            if (value > 1)
            {
                this.percent = value / 100.0;
            }
            else
            {
                this.percent = value;
            }

        }
    }
```

Следват конструкторите. Тук правим и празен конструктор, който извиква верижно другия със стойности 0 за двата параметъра. Кодът е както следва:

```
public Tip(double amount, double percent)
{
    Amount = amount;
    Percent = percent;
}
public Tip() : this(0, 0) { }
```

Накрая реализираме двата метода за изчисления, които са по-скоро тривиални:

```
public double CalculateTip()
{
    return Amount * Percent;
}

public double CalculateTotal()
{
    return CalculateTip() + Amount;
}
```

Изглед

В рамките на този клас трябва да създадем **свойства** за **процент**, **стойност на поръчката**, **бакшиш** и **обща сума** за плащане с включения бакшиш. **Процентът** и **стойността** ще получим при въвеждането им от

потребителя през **конзолата**, а другите две стойности се изчисляват от **модела**. В рамките на класа ние ще създадем два метода **GetValues()** и **ShowTipAndTotal()**.

Кодът реализиращ свойствата е както следва:

```
public double Percent { get; set; }
public double Amount { get; set; }
public double Total { get; set; }
public double TipAmount { get; set; }
```

Сега трябва да реализираме конструктора. Ще използваме този конструктор, за да дадем стойности по подразбиране на всичките свойства, но и за да извикаме метод **GetValues()**, чиято цел е да въведе стойности от конзолата за процента и сумата. Кодът е както следва:

```
public Display()
{
    Percent = 0;
    Amount = 0;
    Total = 0;
    TipAmount = 0;
    GetValues();
}
```

Сега нека минем към реализацията на **GetValues()**. Ще използваме обикновени входно/изходни операции:

```
private void GetValues()
{
    Console.WriteLine("Enter the amount of the meal:");
    Amount = double.Parse(Console.ReadLine());
    Console.WriteLine("Enter the percent you want to tip: ");
    Percent = double.Parse(Console.ReadLine());
}
```

Реализацията на **`ShowTipAndTotal()`** не е нищо по-сложно:

```
public void ShowTipAndTotal()
{
    Console.WriteLine("Your tip is: {0:C}", TipAmount);
    Console.WriteLine("The total will be {0:C}", Total);
}
```

Тук използваме **{0:C}**, защото си имаме работа с пари в определена валута.

Контролер

Остана да съберем отделните парченца заедно – това се случва в контролера. Първата ни задача е да направим полета от типове съответстващи на моделите и изгледите, които желаем да ползваме. Тук ще използваме Tip и Display. След това в конструктора ще създадем и обекти от въпросните класове. Важно уточнение е, че тук при създаването на обекта от Tip вече ще разполагаме с въведени от потребителя данни, тъй като извикването на конструктора Display води до въвеждането на информацията. В този смисъл тази информация може да бъде подадена на Tip като параметри.

Тук е и мястото, където ще зададем стойности на TipAmount и Total полетата от Display, извиквайки методи от модела. По този начин в контролера извършваме връзка между двата класа и предаваме данни.

Накрая извикваме метод ShowTipAndTotal(), за да се визуализира резултат от програмата.

Ако сте работили правилно кодът ще е както следва:

```
    private Tip tip;
    private Display display;

    public TipCalculatorController()
    {
        display = new Display();
        tip = new Tip(display.Amount, display.Percent);
        display.TipAmount = tip.CalculateTip();
        display.Total = tip.CalculateTotal();
        display.ShowTipAndTotal();
    }
```

Тук трябва да се добавят и следните using директиви:

```
using ConsoleMVC.Model;
using ConsoleMVC.Views;
```

Главна програма

Добавете следната using директива: **using ConsoleMVC.Controllers;**

След това трябва да създадем обект от желания контролер клас:

```
TipCalculatorController tipCalculatorController = new TipCalculatorController();
```

С това нашето приложение е готово ☺

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).