GeekBrains

ASP.NET MVC и углубление в изучение C#

Шаблонизатор

На этом уроке

1. Познакомимся с понятием шаблонизатор.
2. Узнаем как им пользоваться в формировании Word документов.
3. Рассмотрим то, как нужно создавать шаблоны в Word.
4. Научимся формировать документы и обычный текст.
5. Познакомимся с Razor шаблонами.

Оглавление

Введение в шаблонизаторы

Общее о шаблонизаторах

Документы Microsoft Word

Шаблон документа для шаблонизатора TemplateEngine.docx

Подключаем TemplateEngine.docx

Усложняем задачу с формирование DOCX документа

Формирование базового HTML или текста

Практическое задание

Дополнительные материалы

Используемые источники

Введение в шаблонизаторы

Допустим по требованиям от аналитиков в программном обеспечение, которое передается на разработку, требуется создать модуль отчетов, модуль отправки писем и, в добавок, еще и модуль формирования документов. При этом нужно предусмотреть, чтобы шаблоны можно было легко менять и дополнять. А иногда на одни и те же данные требуется по несколько разных документов и отчетов.

Что объединяет эти три модуля? Ответ прост – шаблонизация. Шаблонизация берет за основу какой-либо шаблон документа Microsoft Word/Microsoft Excel или HTML кода, и подставляет данные из базы либо из какой-либо выборки, формируя конечный документ или страницу.

Написание своих шаблонизаторов дело непростое, это может занять продолжительное время и грозит, как минимум недочетами и ошибками.

Но, как можно догадаться, на рынке уже есть nuget решения, которые можно взять за основу, избавив себя от головной боли написания большого количества кода, и сосредоточиться уже непосредственно на самом шаблоне.

Общее о шаблонизаторах

Шаблонизаторы часто используют для формирования любых больших документов либо страниц. В самом .net core есть уже встроенные шаблонизаторы – Razor, Blazor, которые на лету создают HTML-страницы. Использовать уже доступные решения легче и проще, чем использовать StringBuilder. Хотя, последний не исключается при шаблонизации маленьких и не ресурсоемких шаблонов, к примеру сообщения для пользователей.

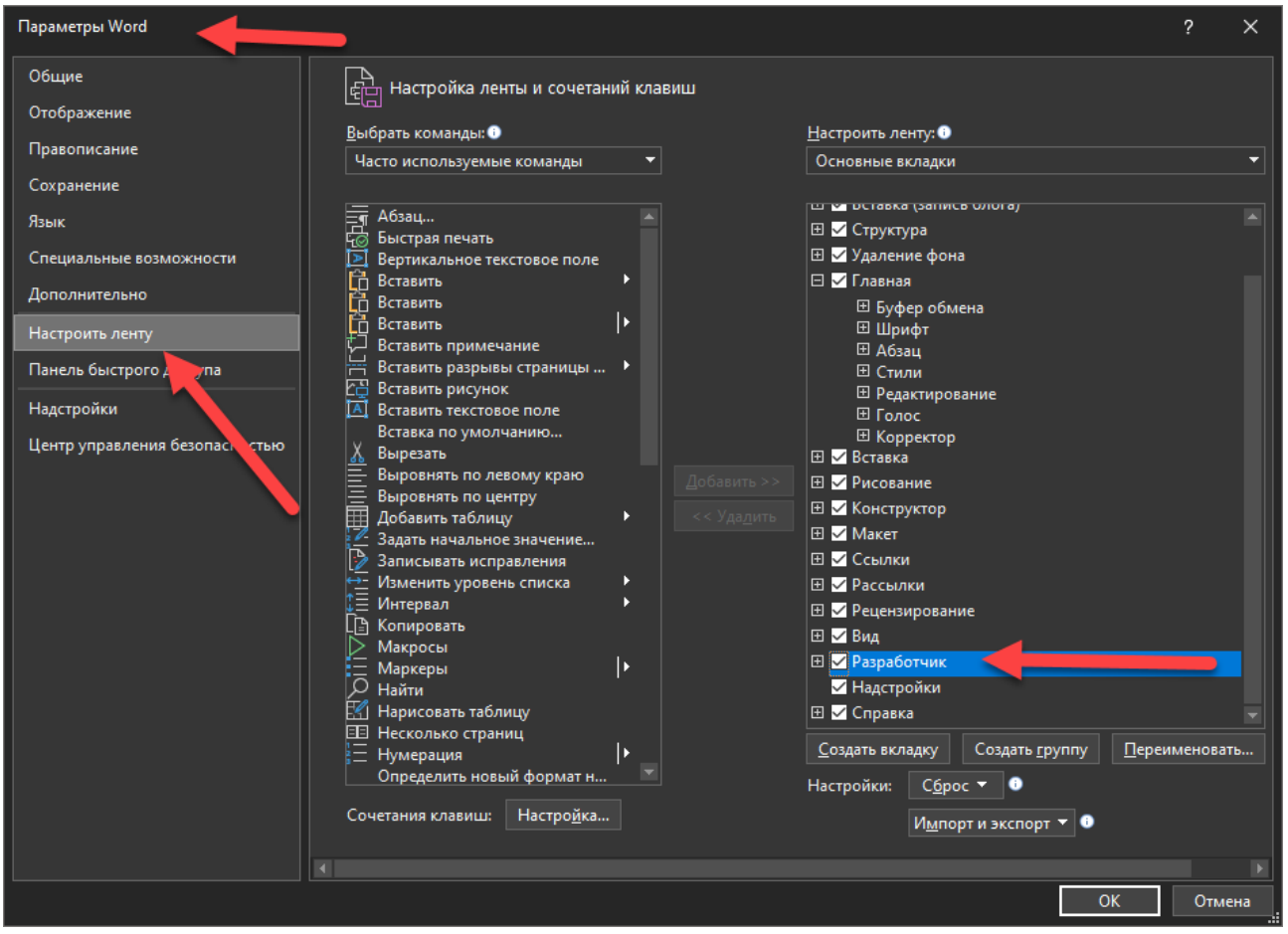
Документы Microsoft Word

Формирование документов на базе DOCX от Microsoft является одной из самых популярных задач в любой отчетной системе. Это универсальный формат, который поддерживается практически на любой операционной системе. Его структура вполне ясна и поддается шаблонизации. Шаблонизация базируется на внутренних «маркерах» в самом документе, по которым программа должна ориентироваться при формировании документа.

Шаблон документа для шаблонизатора TemplateEngine.docx

Очевидно, что чтобы сформировать документ по шаблону – нужно иметь сам шаблон. Для начала создадим простой шаблон, который будет выводить данные, напрмер, о какой-либо компании.

Откроем Microsoft Word, и добавим в Ribbon ленту вкладку «Разработчик». Делается это через «Файл»-«Параметры»-«Настроить Ленту»

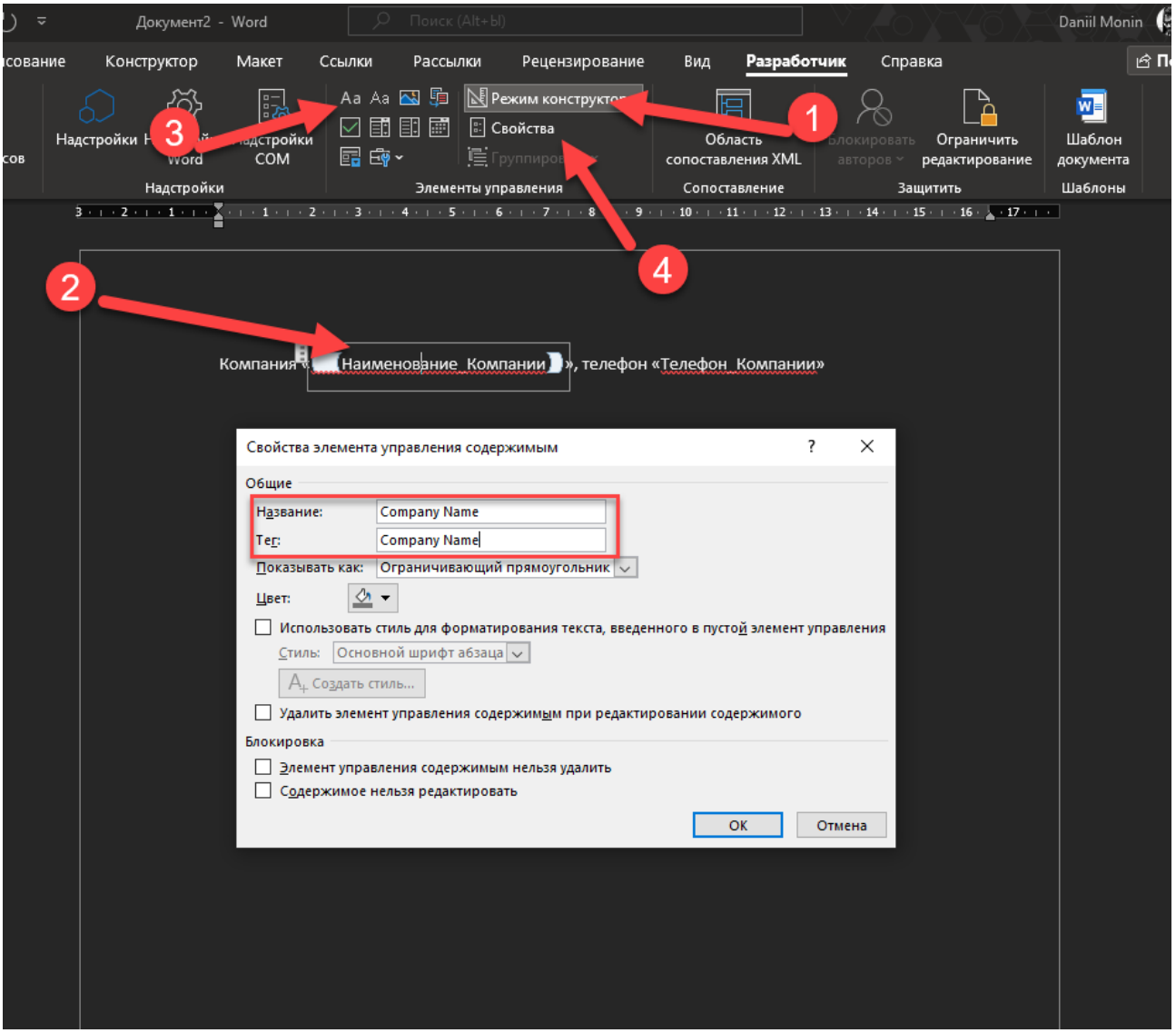


После нам уже доступна вкладка «Разработчик» и можно приступить к написанию нашего первого шаблона. Перейдите на вкладку «Разработчик» и включите режим «Конструктора» (пункт 1 на снимке снизу). В теле документа напишите схожий текст:

```
Компания «Наименование_Компании», телефон «Телефон_Компании»
```

Затем нужно выделить слова Наименование_Компании (пункт 2 на снимке снизу) и нажать на «Элемент управления содержимом Форматированный текст» (пункт 3 на снимке снизу). Затем нужно вызвать свойства элемента (пункт 4 на снимке снизу).

В окне нужно задать название и тег. В нашем случае в полях зададим Company Name. Проделайте так же и с телефоном компании, главное, чтобы тег и имя были уникальны в разрезе данного шаблона.



После завершения внесения тегов и названий документ стоит сохранить рядом с вашим проектом, либо там, где, есть возможность достучаться до него из кода.

Наш шаблон готов. Осталось только запустить и проверить шаблонизацию.

Подключаем TemplateEngine.docx

Нам нужен nuget пакет с одноименным названием - TemplateEngine.Docx. На написании методического пособия версия данного пакета была 1.1.5

После подключения пакета уже ничего не мешает, чтобы написать свой первый шаблонизированный отчет.

```
using System;
using System.IO;
using TemplateEngine.Docx;

namespace Interview.Sample
{

      public sealed class CompanyInfo
      {
      public string CompanyName { get; set; }

      public string CompanyPhone { get; set; }
      }

      public sealed class ReportService
      {
      public void GenerateReport(CompanyInfo companyInfo, string output =
"")
      {
         if (companyInfo is null)
            {
                return;
            }

            if (string.IsNullOrWhiteSpace(output))
            {
                output = Path.Combine(Directory.GetCurrentDirectory(),
"CompanyReport.docx");
            }

            if (File.Exists(output))
            {
                File.Delete(output);
            }

            File.Copy("E:\\Templates\\Простой_договор.docx", output);

            var valuesToFill = new Content(
                new FieldContent("Company Name", companyInfo.CompanyName),
                new FieldContent("Company Phone",
companyInfo.CompanyPhone)
            );

            using(var outputDocument =
                new TemplateProcessor(output)
                .SetRemoveContentControls(true))
            {
                outputDocument.FillContent(valuesToFill);
                outputDocument.SaveChanges();
            }
      }
      }

      class Program
      {
      static void Main(string[] args)
      {
            CompanyInfo companyInfo = new CompanyInfo()
            {
```

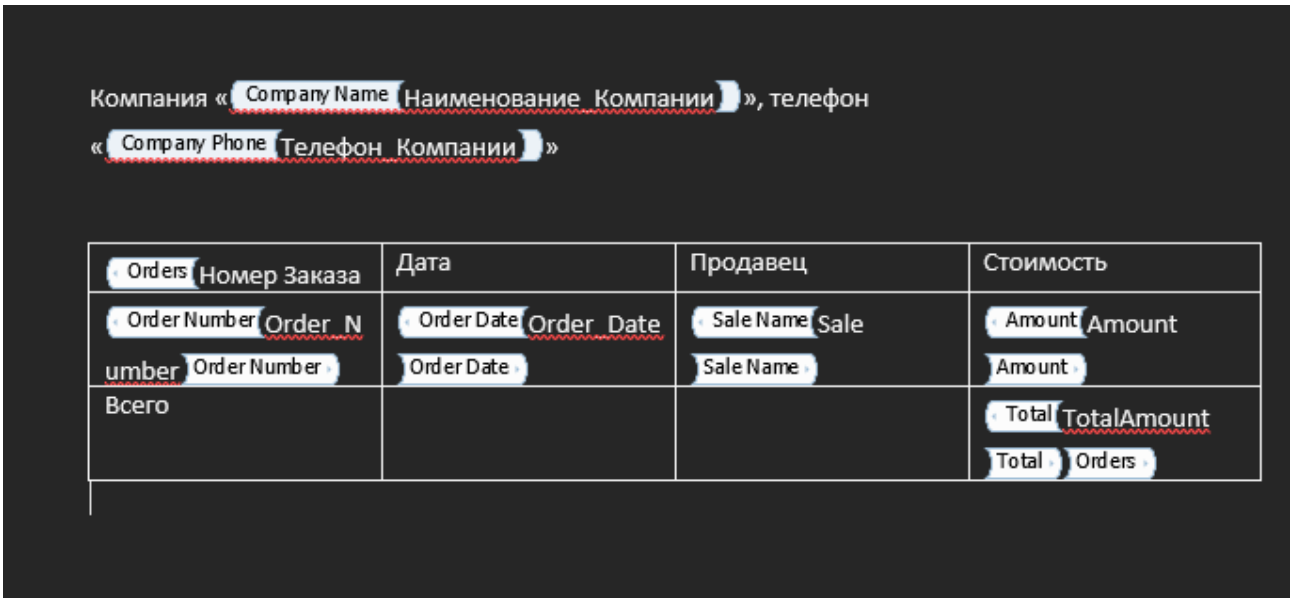
```
                CompanyName = "ООО Моя супер компания",
                CompanyPhone = "88005553535"
            };

            ReportService reportService = new ReportService();

            reportService.GenerateReport(companyInfo);
        }
        }
}
```

Алгоритм работы прост. Сначала заполняется структура о компании CompanyInfo, затем передается в ReportService, который в свою очередь проверяет ранее созданный документ, удаляет его и создает новый. После подгружает в шаблонизатор, и по имени тегов уже проставляет значения из структуры CompanyInfo. И на выходе у вас уже рабочий документ, который содержит нужные записи. Все достаточно просто, и пользователь будет в восторге.

Усложняем задачу с формирование DOCX документа

Данный шаблонизатор позволяет не только вставлять одиночные данные, но и таблицы и изображения. К примеру, требуется вывести список заказов. Для формирования таблицы нужно добавить саму таблицу, так же дать ей имя и тэг, к каждой ячейки тоже дать теги и названия.

Вот так должна быть выглядеть таблица:

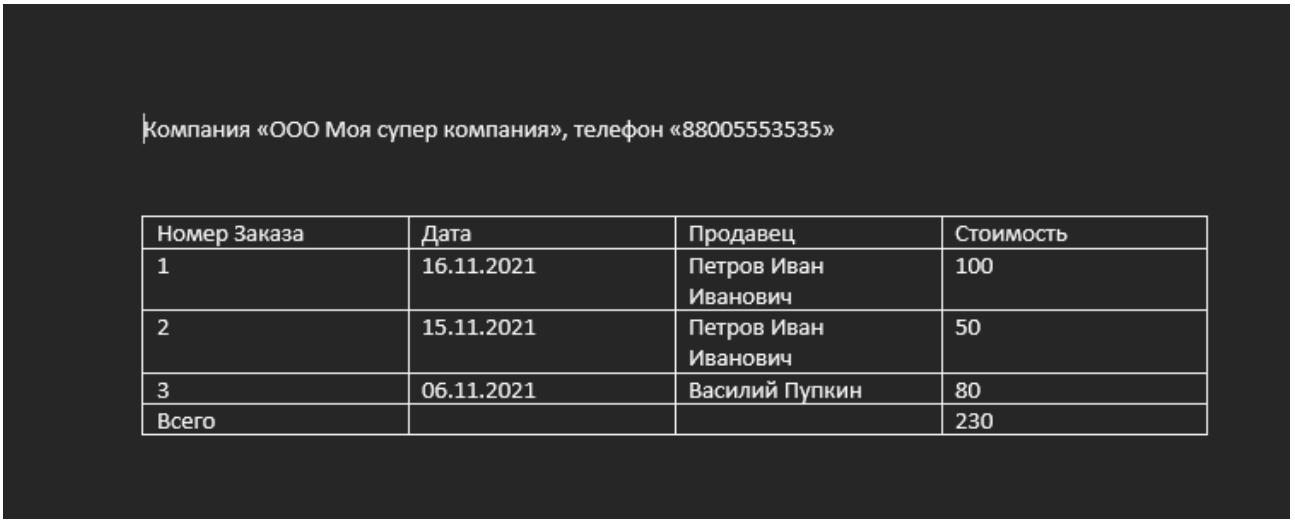


```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using TemplateEngine.Docx;

namespace Interview.Sample
{

      public sealed class CompanyInfo
      {
      public string CompanyName { get; set; }

      public string CompanyPhone { get; set; }

      public List<Order> Orders { get; set; }
      }

      public sealed class Order
      {
      public DateTime Date { get; set; }

      public int Id { get; set; }

      public string Sale { get; set; }

      public decimal Amount { get; set; }
      }

      public sealed class ReportService
      {
      public void GenerateReport(CompanyInfo companyInfo, string output =
"")
      {
            if (companyInfo is null)
            {
            return;
            }

            if (string.IsNullOrWhiteSpace(output))
            {
            output = Path.Combine(Directory.GetCurrentDirectory(),
"CompanyReport.docx");
            }

            if (File.Exists(output))
            {
              File.Delete(output);
            }

           File.Copy("E:\\Templates\\Простой_договор.docx", output);

            List<TableRowContent> rows = new List<TableRowContent>();

            foreach (Order order in companyInfo.Orders)
            {
            rows.Add(new TableRowContent(new List<FieldContent>()
            {
                  new FieldContent("Order Number", order.Id.ToString()),
```

```
                    new FieldContent("Order Date",
order.Date.ToShortDateString()),
                    new FieldContent("Sale Name", order.Sale),
                    new FieldContent("Amount", order.Amount.ToString()),
              }));
              }


              var valuesToFill = new Content(
              new FieldContent("Company Name", companyInfo.CompanyName),
              new FieldContent("Company Phone", companyInfo.CompanyPhone),
                  TableContent.Create("Orders", rows),
              new FieldContent("Total", companyInfo.Orders.Sum(x =>
x.Amount).ToString())
              );

              using(var outputDocument =
              new TemplateProcessor(output)
                  .SetRemoveContentControls(true))
              {
                  outputDocument.FillContent(valuesToFill);
              outputDocument.SaveChanges();
              }
        }
        }

        class Program
        {
        static void Main(string[] args)
        {
              CompanyInfo companyInfo = new CompanyInfo()
              {
              CompanyName = "ООО Моя супер компания",
              CompanyPhone = "88005553535",
              Orders = new List<Order>()
              {
                    new Order()
                    {
                          Amount = 100,
                          Date = DateTime.Now,
                          Id = 1,
                          Sale = "Петров Иван Иванович"
                      },
                    new Order()
                    {
                          Amount = 50,
                          Date = DateTime.Now.AddDays(-1),
                          Id = 2,
                          Sale = "Петров Иван Иванович"
                       },
                    new Order()
                    {
                          Amount = 80,
                          Date = DateTime.Now.AddDays(-10),
                          Id = 3,
                          Sale = "Василий Пупкин"
                    },
              }
              };
```

```
            ReportService reportService = new ReportService();

            reportService.GenerateReport(companyInfo);
        }
        }
}
```

Общий итог документ, который содержит и информацию о компании и перечисление с заказами.



Компания «ООО Моя супер компания», телефон «88005553535»

Номер Заказа	Дата	Продавец	Стоимость
1	16.11.2021	Петров Иван Иванович	100
2	15.11.2021	Петров Иван Иванович	50
3	06.11.2021	Василий Пупкин	80
Всего			230

Формирование базового HTML или текста

Формирование HTML удобно, к примеру, для просмотра онлайн отчетов, либо формирования письма HTML, который приходит на электронную почту пользователя. Это самый распространенное применение данного вида шаблонизатора. Стоит отметить, что формировать можно не только HTML, но и обычный текст, что намного расширяет возможности выдачи документов или отчетов. Стоит помнить, что данный шаблонизатор — это плод MVC фреймворка, который будет рассмотрен далее по курсу.

Для подключения шаблонизатора нужно добавить nuget пакет под названием RazorEngine.NetCore

Простейшее применение можно уже создать на базе предыдущего примера. Удалите код формирования DOCX документа и замените его на новый сервис.

```
      public sealed class ReportService

      {

      public void GenerateReport(CompanyInfo companyInfo, string output =
"")

      {

            string template = "Компания @Model.CompanyName, телефон
@Model.CompanyPhone";



            string     report     =     Engine.Razor.RunCompile(template,
"myUniqueReport", null, companyInfo);



            Console.WriteLine(report);

      }

      }
```

Как можно увидеть, формирование происходит через символ собачки + Model и нужного нам свойства. На выходе в консоле должно быть строка с данными о компании.

```
Компания ООО Моя супер компания, телефон 88005553535
```

На что в данном случае стоит обратить внимание? На то, что компиляция данного шаблона происходит автоматически и в последствии он доступен по уникальному ключу. То есть, если грубо описать процесс – сформируется сборка, которая будет автоматически подгружаться в приложении при запросе на формирования отчета. Осталось только добавить хештеги HTML, что бы это была обычная HTML-разметка.

Практическое задание

1. Создайте отчетный модуль для вашего файлового менеджера по использованию диска (сколько свободного места и т.д.) .

Дополнительные материалы

1. Статья «Razor».

Используемые источники

1. Статья «Razor Engine».
2. Статья Razor Engine