ОСОБЕННОСТИ РАЗРАБОТКИ СЛОЖНОГО ИНТЕРФЕЙСА НА ASP.NET MVC

Сорокина С.В., студентка гр. ИТб-161, IV курс
Научный руководитель: Сахопотинов Г. А., ассистент
Кузбасский государственный технический университет имени Т.Ф. Горбачева,
г. Кемерово

Прежде чем погрузиться в тему данной статьи вкратце рассмотрим основные положения ASP.NET для того чтобы понимать в дальнейшем суть проблемы и варианты ее решений.

Платформа ASP.NET MVC представляет собой фреймворк для создания сайтов и веб-приложений с помощью реализации паттерна MVC. Сам паттерн MVC предполагает разделение приложения на три компонента:

- **Модель** (model) представляет класс, описывающий логику используемых данных.
- **Контроллер** (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных.
- **Представление** (view) - это собственно визуальная часть или пользовательский интерфейс приложения, использующий модель для вывода некоторых данных. Как правило, html-страница, которую пользователь видит, зайдя на сайт. В ASP.NET оно реализовано с помощью движка представлений Razor. Razor — это синтаксис разметки для внедрения в веб-страницы серверного кода, он позволяет сделать переход от разметки html к коду C#.

Моя дипломная работа представляет собой веб-сайт, созданный с помощью фреймворка ASP.NET и паттерна MVC. Проект разработан для сервисного отдела АО «СУЭК-Кузбасс», который содержит в себе функции учета и создания графика дежурств, функции рассылки графика по почте и создания отчета о дежурствах, а также списки сотрудников и аварий, на которые они выезжали. По большей части это были несложные страницы с простым списком сотрудников или аварий. Но на главной странице сайта заказчик требовал представить график дежурств отдела в виде календаря на месяц, в каждой ячейке которого должно было находиться скрытое поле. Поле открывается нажатием на число, и там в выпадающем списке можно выбрать дежурного на данный день и после сохранения изменений его имя остается в данной ячейке. Таким образом, каждый сотрудник может зайти на сайт и увидеть, когда он дежурит, и кто дежурит в другие дни. Соответственно, основной алгоритм построения календаря находится в представлении и реализован с помощью движка Razor.

Однако, спустя какое-то время заказчик также пожелал сделать еще и график отпусков, с подобным интерфейсом - в виде календаря.

Таким образом, передо мной встала проблема, как разработать календарь максимально лаконичным и понятным образом, если заказчику в будущем захочется самому что-то добавить или переделать? И как избежать повторения одного и того же кода?

Первоначально для решения проблемы на ум сразу же приходит решение - создать два представления календаря, для графика дежурств и для графиков отпусков, отличающихся лишь в деталях. Однако, такой способ хоть и не требует дополнительного анализа технологии разработки, но имеет два существенных минуса. Во-первых, это довольно банальное копирование кода, что совершенно не подходит под решение заявленной проблемы. Во-вторых, если придется редактировать календарь или добавлять в него дополнительный функционал, то работу придется проводить в дважды, что не исключает человеческий фактор. В результате, можно сделать вывод что такой способ не годится.

Таким образом, я начала усердные поиски другого более гибкого решения и выяснила, что фреймворк ASP.NET MVC обладает также таким мощным инструментом, как HTML-хелперы, позволяющие генерировать html-код и вызывать их как обычную функцию. Существует два вида таких хелперов. Рассмотрим каждый из них.

Строчный хелпер. Строчные хелперы представляют собой обычное представление, а это, как мы уже знаем, «симбиоз» html и C#. Но отличительная особенность их в том, что они начинаются с тега @helper и работают подобно методам, т.е. их можно вызвать в нужном представлении и передать туда некоторые аргументы.

Пример:

```
@helper BookList(IEnumerable<BookStore.Models.Book> books)
{
    <ul>
        @foreach (BookStore.Models.Book b in books)
            {
                <li>@b.Name</li>
            }
    </ul>
}
```

Листинг 1. Пример строчного хелпера.

Хелпер в виде C# класса. В таком классе не имеется html разметки. Вся разметка создаются с помощью объекта TagBuilder, который имеет ряд свойств и методов, задающих значения для html-тегов:

- Свойство **InnerHtml** позволяет установить или получить содержимое тега в виде строки.
- Метод **MergeAttribute (string, string, bool)** позволяет добавить к элементу один атрибут. Для получения всех атрибутов можно использовать коллекцию Attributes.
- Метод **SetInnerText(string)** устанавливает текстовое содержимое внутри элемента.
- Метод **AddCssClass(string)** добавляет класс css к элементу.

Пример:

```
public static class ListHelper
   {
      public static MvcHtmlString CreateList(this HtmlHelper html, string[]
items)
      {
            TagBuilder ul = new TagBuilder("ul");
            foreach (string item in items)
            {
                  TagBuilder li = new TagBuilder("li");
                  li.SetInnerText(item);
                  ul.InnerHtml += li.ToString();
            }
             return new MvcHtmlString(ul.ToString());
      }
   }
```

Листинг 2. Пример хелпера в виде C# класса.

В своем проекте я протестировала оба вида хелперов. Каждый удобен тем, что их можно вызвать в любом представлении и в любом месте, однако я также выявила ряд недостатков:

- Оба типа хелперов не создаются в одном каталоге с представлениями, их обязательно необходимо помещать в каталог в корне проекта - App_Code.
- Оба типа хелперов не имеют поддержки специальных встроенных хелперов (чаще всего они создают простую часто используемую разметку, например, ссылки), т.к. каталог, в котором они содержатся, предназначен для динамически собираемого кода.

- Хелпер в виде C# класса из-за значительного количества объектов TagBuilder и его методов, даже для небольшой разметки, для разметки, в которой необходимо представить целый календарь, получается практически невозможным для составления и чрезвычайно обширным и, соответственно, плохо читаемым и менее понятным.

Исходя из изученных и проанализированных мною способов календарь графика дежурств и отпусков я построила в виде строчного хелпера. Он подошел мне в наибольшей степени, по нескольким причинам. Во - первых, прежде у меня уже была написана разметка календаря, и мне не пришлось менять ее полностью. Во - вторых, теперь обе страницы с календарями вызывают один и тот же метод который выдаёт им разметку, и если у меня или у заказчика в дальнейшем возникнет необходимость редактировать функционал календаря – это не нужно будет делать в нескольких местах, но при этом изменения будут распространяться на оба календаря. Также, это будет очень полезно при создании других календарей.

Однако, если представление для календарей всего одно, а хелпер и является представлением, то оно должно работать с конкретной моделью. Тогда каким образом удается работать в одном календаре с дежурстве, а в другом с отпусками? Действительно, в моём проекте имеются модели дежурств и отпусков, но хелпер принимает в себя третью модель – модель «Calendar».

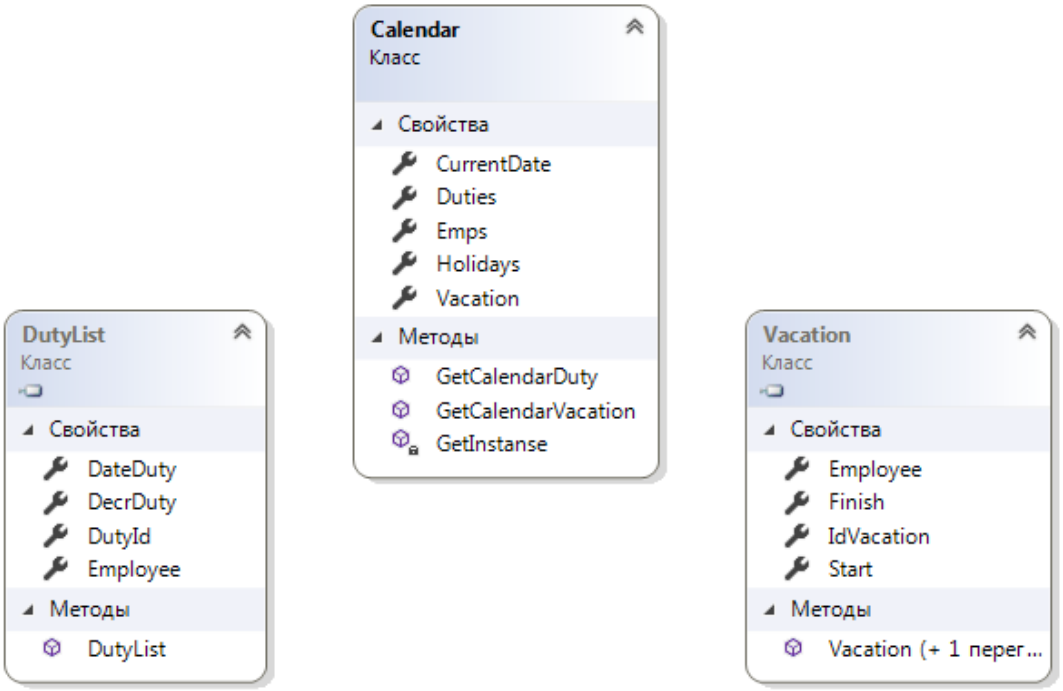

Рис. 1. Диаграмма классов моделей.

Тут на помощь приходит контроллер, его обязанность заключается в том, чтобы выбрать данные из дежурств и отпусков, и в зависимости от вызываемой страницы сайта, создавать модель календаря с теми или иными данными. Таким образом, при вызове страницы графика дежурств, контроллер выбирает данные из модели «DutyList» и записывает в экземпляр модели «Calendar» в поле Duties, представляющее из себя массив пар дат и дежурных сотрудников. Затем контроллер цельно передает модель календаря в представление. Аналогично происходит и при вызове страницы сайта отпусков.

Соответственно, два разных календаря реализуются одним и тем же методом, но за счёт передаваемых данных имеют разный вид:

График дежурств

Разослать график

<< Февраль - 2020 >>

База данных выходных и праздников на данный месяц не обновлена!

ПН	ВТ	СР	ЧТ	ПТ	СБ	ВС
27	28	29	30	31	1 Сидоров Николай	2
3	4 Сидоров Николай	5	6	7	8 Сидоров Николай	9
10	11 Сидоров Николай	12	13	14	15 Сорокина Светлана	16
17 Петров Петр	18 Петров Петр	19	20	21	22	23
24	25	26	27	28	29	1

Рис. 2. Календарь графика дежурств.

<< Февраль - 2020 >>

База данных выходных и праздников на данный месяц не обновлена!

ПН	ВТ	СР	ЧТ	ПТ	СБ	ВС
27	28	29	30	31	1 Сорокина Светлана Иванов Иван Петров Петр	2 Сорокина Светлана Иванов Иван Петров Петр
3 Сорокина Светлана Иванов Иван Петров Петр	4 Сорокина Светлана Иванов Иван Петров Петр	5 Сорокина Светлана Иванов Иван Петров Петр	6 Сорокина Светлана Иванов Иван Петров Петр	7 Сорокина Светлана Иванов Иван Петров Петр	8 Сорокина Светлана Иванов Иван Петров Петр	9 Иванов Иван Петров Петр
10 Иванов Иван Петров Петр	11 Иванов Иван Петров Петр	12 Иванов Иван Петров Петр	13 Иванов Иван Петров Петр	14 Иванов Иван Петров Петр	15 Иванов Иван Петров Петр	16 Иванов Иван
17 Иванов Иван	18 Иванов Иван	19 Иванов Иван	20 Иванов Иван	21 Иванов Иван	22 Иванов Иван	23 Иванов Иван
24 Иванов Иван	25 Иванов Иван	26	27	28	29	1

Рис. 3. Календарь отпусков.

В заключении хочу уточнить, что хоть метод для решения заявленной проблемы и был найден, он, так или иначе, имеет свои недостатки. И я могу лишь надеяться, что компания Microsoft в выпуске новых версий фреймворка обратит на это внимание и устранит эти недостатки.

Список литературы:

1. Основы ASP.NET [Электронный ресурс] URL: https://professorweb.ru/my/ASP_NET/base/level1/base_aspnet_index.php (дата обращения 13.02.2020).
2. Руководство по ASP.NET MVC 5 [Электронный ресурс] URL: https://metanit.com/sharp/mvc5/ (дата обращения 12.02.2020).
3. Введение в Тэг-хелперы (Tag Helpers) [Электронный ресурс] URL: https://dotnet.today/ru/aspnet5-vnext/mvc/views/tag-helpers/intro.html (дата обращения 13.02.2020).