Лабораторная работа №3

Частичные представления и компоненты представления. Передача данных представлению

1. Цель работы.

Изучение возможностей получения частичной разметки страницы. Знакомство с механизмом передачи данных представлению.

Время выполнения работы: 2 часа

2. Общие сведения.

Для передачи данных между контроллером и представлением используются объекты ViewData и ViewBag. Они представляют собой словари, формируемые динамически, и доступные как свойства в контроллере и в представлении.

Пример использования ViewData:

```
ViewData["Text"] = "Лабораторная работа 2";
```

или

```
[ViewData]
public string Text { get; set; }
```

ViewData также предоставляет свойство **Model**, которое может представлять собой объект класса. В представлениях Razor свойство Model доступно динамически. Это значит, что можно получить доступ к свойствам модели, не зная типа класса модели. Однако, для использования механизма IntelliSense желательно указать, к какому классу относится модель. Для этого используется ключевое слово **@model.** Такое представление называется строго типизированным. Модель передается в представление как параметр.

Частичное представление – это разметка, которая помещается внутри другой разметки.

Для вызова частичного представления можно использовать вспомогательный метод **@Html.Partial()** или **@Html.RenderPartial()**. Однако, предпочтительнее использовать тэг <partial>, например:

```
<partial name="_UserPartial" />
```

Компоненты представления (ViewComponent), как и частичные представления, предназначены для создания части разметки, которая затем помещается на страницу. Однако, компонент позволяет реализовать сложную бизнес-логику. В этом работа компонента похожа на работу контроллера, но в отличие от контроллера класс компонента содержит один публичный метод

```
public IViewComponentResult Invoke()
```

Вызов компонента на странице осуществляется с помощью команды:

```
@await Component.InvokeAsync("Имя компонента")
```

Компоненты принято размещать в папке Components (необязательно).

Представления размещаются по пути:

*/Views/**контроллер**/Components/**ИмяКомпонента**/Default.cshtml*

или

*/Views/Shared/Components/**ИмяКомпонента**/Default.cshtml*

3. Выполнение работы

Используйте проект из лабораторной работы №2.

3.1. Задание №1

Оформите меню сайта (см. рис. 7, 8 лабораторной работы 1) в виде компонента (View Component) «Menu».

Для активного пункта меню предусмотреть свой CSS-стиль.

На странице макета (_Layout.cshtml) вместо разметки меню подключите созданный компонент.

3.1.1. Рекомендации к заданию №1

В папке Models создайте класс MenuItem, описывающий параметры элементов меню сайта:

- IsPage - является ли вызываемая ссылка страницей (Razor Page) или методом контроллера;

- Text - текст надписи;

- Controller - имя контроллера;

- Action - имя метода;

- Page - имя страницы;

- Area - имя области (Area);

- Active - имя класса CSS для текущего (активного) пункта меню.

В классе компонента меню создайте коллекцию исходных данных главного меню: `List<MenuItem> items = new List<MenuItem> { . . .}`

Пример:

```
// Инициализация списка элементов меню
private List<MenuItem> _menuItems = new List<MenuItem>
{
      new MenuItem{ Controller="Home", Action="Index", Text="Lab 2"},
      new MenuItem{ Controller="Product", Action="Index",
      Text="Каталог"},
      new MenuItem{ IsPage=true, Area="Admin", Page="/Index",
      Text="Администрирование"}
};
```

Внутри метода Invoke() нужно обойти коллекцию _menuItems и, если имя контроллера или имя области совпадает с текущим, то в данном пункте меню свойству Active присвоить значение «active».

Для определения текущего пункта меню (для назначения соответствующего стиля кнопке меню) можно использовать свойство ViewContext:

```
ViewContext.RouteData.Values["controller"];
ViewContext.RouteData.Values["page"];
ViewContext.RouteData.Values["area"];
```

Полученную коллекцию нужно передать представлению в качестве модели

В представлении компонента эта коллекция обходится в цикле «foreach» для создания элементов навигационной панели.

Пример:

```
<div class="navbar-nav">
    @foreach (var item in Model)
    {
        @if (item.IsPage)
        {
            <a class="nav-item nav-link @item.Active"
               asp-area="@item.Area"
              asp-page="@item.Page">
                @item.Text
            </a>
        }
        else
        {
            <a class="nav-item nav-link @item.Active"
               asp-controller="@item.Controller"
              asp-action="@item.Action">
                @item.Text
            </a>
        }
    }
</div>
```

3.2. Задание №2

Информацию пользователя (см. рис. 7, 8 лабораторной работы 1) оформить в виде частичного представления _UserPartial.cshtml.

Информацию о корзине заказа оформить в виде компонента «Cart»

На странице макета (_Layout.cshtml) вместо разметки меню пользователя подключите созданное частичное представление.

3.3. Пример окончательной разметки заголовка страницы:

```
<header>
        <div class="container">
            <!-- Панель навигации -->
            <nav class="navbar navbar-expand-md navbar-dark bg-dark">
                <!-- меню сайта -->
                <a class="navbar-brand" asp-action="Index" asp-
controller="Home">WebLabsV06</a>
                <div class="navbar-nav">
                    @await Component.InvokeAsync("Menu")
                </div>
                <!-- меню сайта - конец -->
                <!-- Информация пользователя -->
                <partial name="_UserPartial"/>
                <!-- Информация пользователя - конец -->
            </nav> <!-- Панель навигации - конец -->
        </div><!-- container - конец -->
```

```
</header><!-- header - конец -->
```

4. Контрольные вопросы

Как передать в представление одновременно несколько объектов разных классов?

Что такое строго типизированное представление?

Чем отличается компонент представления от частичного представления?

Как разместить частичное представление на странице?

Как разместить компонент представления на странице?