Лабораторная работа №7

Вспомогательные классы тэгов, Ajax

1. Цель работы.

Знакомство с тэг-хелперами. Знакомство с механизмом асинхронных запросов Http.

2. Задача работы

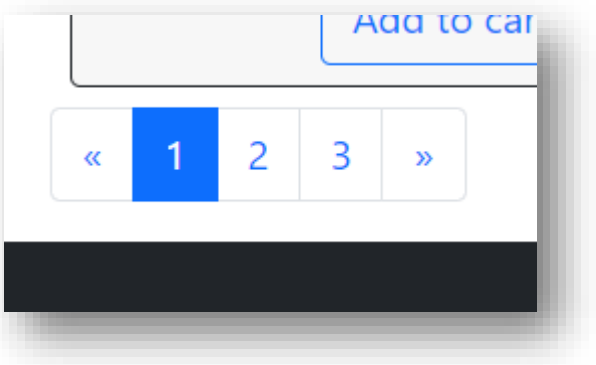
Научиться выполнять создавать и регистрировать тэг-хелперы. Научиться посылать и обрабатывать асинхронные запросы.

Время выполнения работы: 4 часа

3. Выполнение работы.

3.1. Задание 1

Требуется создать тэг-хелпер, который будет выводить кнопки пейджера:



Пример использования такого тэг-хелпера:

```
<Pager current-page="@Model.CurrentPage"
       total-pages="@Model.TotalPages"
       category="@category"></Pager>

     или

<Pager current-page="@Model.PageNo"
       total-pages="@Model.TotalPages"
       admin="true" ></Pager>
```

Обязательно: для создания тэгов использовать методы класса **TagBuilder**

3.1.1. Рекомендации к заданию 1

Для получения адресов страниц (для указания атрибута *href* тэга *a*) внедрите конструктор тэг-хелпера объект LinkGenerator.

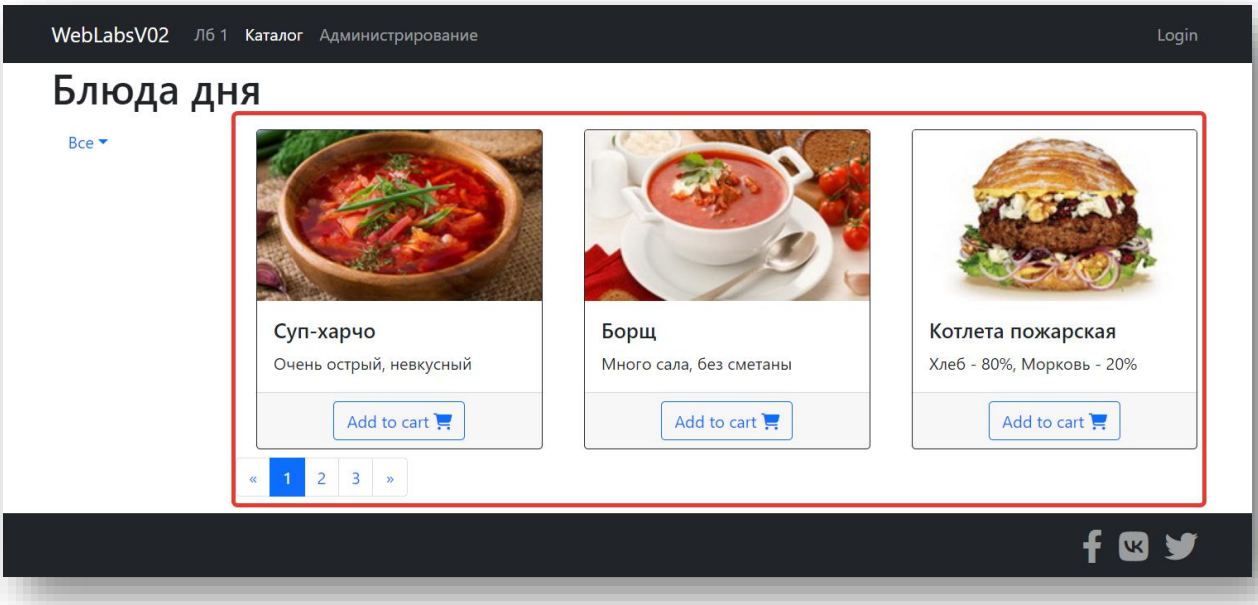
Атрибут «admin» нужен, так как при использовании пейджера на странице Admin/Index нужно будет генерировать ссылки на страницы, а не на методы контроллеров.

Ссылка на страницу формируется методом `_linkGenerator.GetPathByPage`, который принимает объект HttpContext. Контекст можно получить из объекта IHttpContextAccessor.

Не забудьте зарегистрировать тэг-хелпер в файле _ViewImports.cshtml

3.2. Задание 2

Переключение страниц выполнить с помощью запросов Ajax. При выполнении асинхронного запроса должен подгружаться только список объектов и пейджер (на рисунке – участок, выделенный красным квадратом)



3.2.1. Рекомендации к заданию 2

Оформите выделенный фрагмент в виде частичного представления.

В контроллере **Product** нужно проверить, был ли запрос асинхронным. Если да, то нужно вернуть **частичное** представление. Если запрос не был асинхронным, то нужно вернуть **полное** представление.

Асинхронный запрос имеет заголовок «`x-requested-with`» со значением «`xmlhttprequest`»

В папке wwwroot/scripts есть пустой файл site.js. Он подключен на странице макета. Можно использовать его для написания кода асинхронных запросов.

В коде js нужно подписаться на событие «click» всех тэгов «a» в разметке пейджера (при использовании классов bootstrap – это элементы с классом «page-link»).

Используйте расширяющий метод jquery $(“xxx”).load() (см. https://api.jquery.com/load/)

3.3. Задание 3

Опишите расширяющий метод для класса HttpRequest, проверяющий, был ли запрос асинхронным. Используйте его в контроллере Product, например:

```
public async Task<IActionResult> Index(
     [FromServices]IConfiguration config,
     int pageNo=1,
     int category=0)
{
            . . .

     var page = ProductPageViewModel.GetModel(data, pageNo,
pageSize);

     if (Request.IsAjaxRequest())
          return PartialView("_ListPartial", page);

     return View(page);
}
```

3.3.1. Рекомендации к заданию 3

Назовите класс HttpRequestExtensions. Поместите файл с расширяющим методом в папку Extensions.

4. Контрольные вопросы

1. Что такое tag-helper?
2. Какие tag-helpers тэга <form> вы знаете?
3. Какие tag-helpers тэга <a> вы знаете?
4. Как зарегистрировать использование Tag-helper в проекте?
5. Как определить, что запрос был выполнен по технологии Ajax?
6. Может ли MVC контроллер вернут частичное представление?
7. Может ли MVC контроллер вернуть ответ без тела сообщения? Если да, приведите пример.
8. Приведите пример, когда контроллер возвращает ответ, не являющийся представлением.
9. Можно ли с помощью tag-helper поменять тэг элемента?
10. Как в tag-helper получить Url конечной точки?