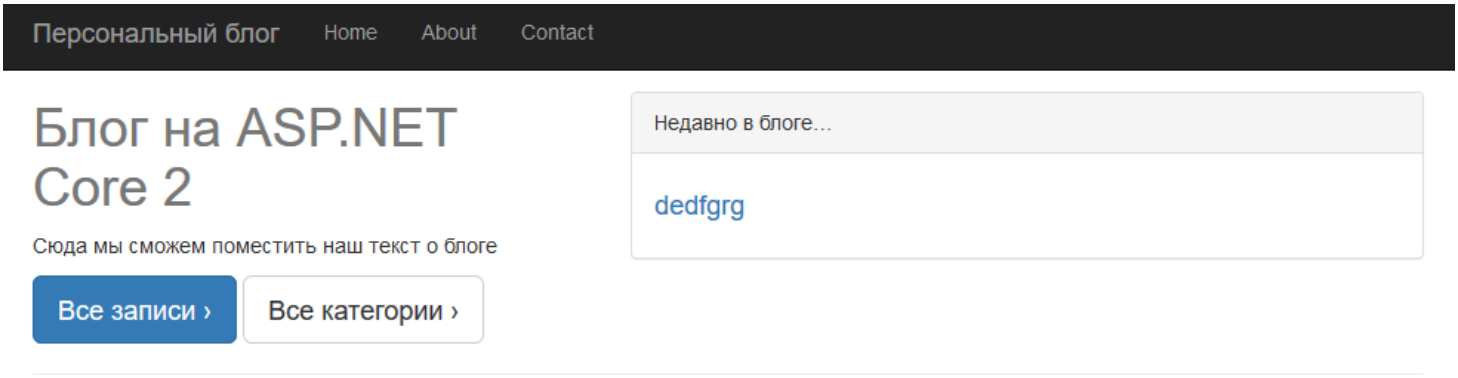
Лабораторная работа №3

Тема: разработка блога на ASP.NET Core 2 и Microsoft SQL Server 2014

Часть четвертая: Разработка административного интерфейса

Предыдущий результат

На главной странице вашего блога уже должен отображаться последний пост, должна быть ссылка на отображение списка записей и ссылка на отображение категорий. Так же должен быть создан контроллер `Admin`, плюс к этому созданы **пустые** действия `Write`, `Edit`, `Delete`, `Index`. Пример преподавателя:



Добавление ссылки на административный интерфейс

Ссылка на административный интерфейс должна быть либо убрана со страниц приложения в целях безопасности, либо помещена в «подвал» сайта. Найдите файл `~Views/Shared/_Layout.cshtml`. В теге `<footer>` измените содержимое на следующее:

```
<p class="pull-left">
    &copy; @DateTime.Now.Year &mdash; Персональный блог
</p>
<p class="pull-right"><a asp-controller="Admin">admin</a></p>
```

Здесь мы выводим текущий год, добавляем длинное тире и название блога. И делаем ссылку на административный интерфейс.

В файл `Startup.cs` метод `ConfigureServices` (куда мы добавляли подключение к БД) также необходимо внести пару изменений:

<u>ПОЖАЛУЙСТА, НЕ НАДО СПИСЫВАТЬ КОММЕНТАРИИ!</u>

```
// чтобы все строки, уходящие в веб-разметку из программного кода имели верную
// кодировку. Не забудьте про подсказки VS!
services.Configure<WebEncoderOptions>(options =>
{
    options.TextEncoderSettings = new TextEncoderSettings(UnicodeRanges.All);
});
// Сделаем так, чтобы ВСЕ ссылки имели вид blog/, т.е. в нижнем регистре
services.AddRouting(options =>
{
    options.LowercaseUrls = true;
});
```

Перезапустите приложение (`Ctrl+F5`) чтобы увидеть результат. Обратите внимание, что все ссылки, генерируемые через asp-… теперь имеют нижний регистр. Так же, если ранее вы смотрели исходный код

страницы, вы замечали символы из базы данных, которые имели вид `#x0…` - это символы, преобразованные в коды Юникода. Сейчас их нет.

Интересная информация. Этот подход увеличивает в разы размер страницы. Он хорош только тогда, когда мы пользуемся сразу несколькими языками. Например, английский (English), иврит (Hebrew), японский (Japanese), русский (Russian). Эти языки радикально отличаются друг от друга не только символами, но и направлением письма (writing direction). Японский имеет 3 версии написания иероглифов – кандзи, хирагана и катакана.

Вывод списка записей в административном интерфейсе

Работаем с контроллером Admin действием Index. Пропишем код, который получит все записи + сделает JOIN на категории. Добавьте переменную `db` по примеру `HomeController` и `BlogController`.

```
public async Task<IActionResult> Index()
{
    var posts = await db.Posts.Include(c => c.Category).ToListAsync();
    return View(posts);
}
```

И сделаем вывод записей, а также ссылки на редактирование и удаление:

```
<table class="table table-striped table-hover">
    <thead>
        <tr>
            <th width="50px"> </th>
            <th>Заголовок</th>
            <th>Категория</th>
            <th>Дата</th>
            <th>Действия</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var p in Model)
        {
            <tr>
                <td>
                    <form method="post" asp-action="Delete" class="pull-right">
                        <input type="hidden" name="id" value="@p.Id" />
                        <button type="submit" class="btn btn-sm btn-danger">
                            &times;
                        </button>
                    </form>
                </td>
                <td>@p.Title</td>
                <td>@p.Category.Title</td>
                <td>@p.PublishDate.ToShortDateString()</td>
                <td>
                    <a class="btn-sm btn-primary" asp-action="Edit" asp-route-
id="@p.Id">Изменить</a>
                </td>
            </tr>
        }
    </tbody>
</table>
```

Персональный блог Главная О программе Контакты

Все записи

	Заголовок	Категория	Дата	Действия
×	dedfgrg	Разработка	07.11.2017	Изменить
×	ded	Рецепты	07.11.2017	Изменить
×	kjd	Мысли	07.11.2017	Изменить

В итоге, после нажатия Ctrl+F5 и перехода в административный интерфейс вы должны увидеть примерно такое вот зрелище. Теперь по порядку:

1. Мы используем таблицу для отображения записей.
2. Мы создаем заголовок таблицы, используя тег `<thead>`
3. После этого идет тело (`<tbody>`) таблицы, в котором и будут отображены данные
4. Используя `@foreach` мы проходим по коллекции записей, записывая в итерации результат в `@p`
5. Мы создаем форму, использующую в качестве action URL типа `/delete/<id>`. Метод POST используется для ограничения посещения пользователем этой ссылки в браузере. Помним, что, вводя в адресную строку URL мы обращаемся к странице используя метод GET. Надеюсь, ясно, зачем здесь submit?
6. Выводим категорию и дату в коротком формате (`ToShortDateString()`)
7. Делаем ссылку на изменение конкретной записи (`/Edit/<id>`)

Важная информация: для действий текущего контроллера указывать его имя в `asp-controller` не обязательно. Этот атрибут можно опустить.

Давайте так же над таблицей добавим кнопку на написание поста:

```
<a class="btn btn-lg btn-primary" asp-action="Write">Добавить запись</a>
```

Форма добавления записи

```
@model weblog.Models.Post
<h2>Добавить запись</h2>
<form asp-action="Write" method="post">
    <div class="form-group">
        <input asp-for="Title" type="text" class="form-control" />
    </div>
    <div class="form-group">
        <input asp-for="PublishDate" type="date" class="form-control" />
    </div>
    <div class="checkbox">
        <label>
            <input asp-for="IsDraft" type="checkbox" />
            Это черновик
        </label>
    </div>
    <div class="form-group">
        <textarea asp-for="Content" class="form-control"></textarea>
    </div>
    <button type="submit">Добавить запись</button>
</form>
```

Здесь `asp-for` служит в качестве заменителя для name="", т.к. в первой строчке мы связали эту страницу с моделью `Post`. Это еще один способ связи, но только с пустой моделью, для строгой типизации форм в приложении.

Добавление записи в базу данных

Перейдем в контроллер Admin, и создадим **перегрузку** метода Write, и добавим атрибут `[HttpPost]`:

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Write(
    [Bind("Title,PublishDate,IsDraft,CategoryId,Content")]Post model)
{
    try
    {
        if (ModelState.IsValid)
        {
            await db.AddAsync(model);
            await db.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
    }
    catch(DbUpdateException ex)
    {
        ModelState.AddModelError("", ex.InnerException.Message);
    }
    return View(model);
}
```

Дабы не тратить бумагу на объяснение этого метода, позовите преподавателя, если вам не ясен принцип работы.

Удаление записи

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Delete(Guid id)
{
    var post = await db.Posts.FindAsync(id);
    if(post != null)
    {
        db.Posts.Remove(post);
        await db.SaveChangesAsync();
    }
    return RedirectToAction(nameof(Index));
}
```