Практическая работа №2. Разработка web-сайта с использованием Asp.Net Core.

Задание: реализовать web-приложение, поддерживающее работу с данными:

- добавление новых объектов;

- удаление существующих объектов;

- просмотр всех объектов;

- вывод общей статистики;

- поиск по полю (имени, идентификатору и т.п.)

Рекомендации к выполнению.

1. Создайте в Visual Studio проект типа ASP.NET CORE с использованием шаблона MVC и без протокола HTTPS.
2. В папку Models добавьте классы, отвечающие за работу с данными. В работе можно использовать один класс – сущность и класс с набором данных.
   Класс-сущность должен содержать свойства разных типов (текстовые, числовые, логические, перечисления). Например,

```
public class Country {
      public string Name { get; set; }
      public float Population { get; set; }
      public float Size {get; set; }
      public bool IsG20Member { get; set; }
      public Continent Continent { get; set; }
}
```

Множество сущностей и их обработку предлагается вынести в отдельный класс CountryRepository :

```
      public class CountryRepository {
            List<Country> countries = ..
            public void AddCountry(Country c);
            public IEnumerable<Country> GetAll();
            public Country Get(int id);
            ..
}
```

3. В папку Controllers добавьте классы-контроллеры с методами-действиями.

```
public class CountryController : Controller
{
```

```
      CountryRepository repo;
      public CountryController(CountryRepository _repo) {..}
      public ActionResult GetAll() {..}
      public ActionResult GetByName(string Name) { .. }
      public void Add(Country c) { .. }
      public ActionResult Statistics() { .. }
}
```

Для того чтобы контроллеры всегда работали с одним и тем же объектом CountryRepository, необходимо в файле Startup.cs подключить сервис внедрения зависимостей:

```
public class Startup
{
      public void ConfigureServices(IServiceCollection services)
      {
            ..
            services.AddSingleton<CountryRepository>();

      }
..
}
```

Теперь при обработке запросов инфраструктура будет создавать экземпляры контроллера CountryController, но каждый раз передавать в конструктор один и тот же объект CountryRepository. Его можно сохранить как поле класса.

4. Методы-действия выполняют подготовку данных (н-р, фильтрацию, сортировку) и отправку в представление (View).

```
public ActionResult GetByName(string Name)
{
   Country c = repo.FirstOrDefault( .. );
   ..
   return View("ShowOneCountryView", c);
}
```

5. Каждое представление нужно поместить в папку, соответствующую контроллеру (н-р, для метода действия ShowOneCountry контроллера CountryController необходимо создать подкаталог Country в папке Views; а затем создать файл-представление ShowOneCountry.cshtml.

   Файл представления желательно создавать как «пустой», чтобы минимизировать объем кода, предоставляемого Visual Studio, и добавить только необходимые строки.

   В представлении предлагается не использовать компоновку (компоновка задает единый макет для всего веб-сайта) – для этого параметру Layout приравниваем null.

Для простоты связывания элементов представления со свойствами сущностей используем «строго типизированные представления» (первая строка)

```
@model  Book
@{
  Layout = null;
}
```

Файл представления желательно создавать как «пустой», чтобы минимизировать объем кода, предоставляемого Visual Studio, и добавить только необходимые строки.

Для создания html-элементов, позволяющих вводить и отображать значения сущности (модели), можно использовать вспомогательные методы (html-helpers):

```
<h2> Создание нового элемента </h2>
<div>
      <label>Название</label>
      @Html.TextBoxFor(country => country.Name)
</div>
<div>
      <label>Входит в G20</label>
      @Html.CheckBoxFor(country => country.IsInG20)
</div>
<div>
      <label>Материк</label>
      @Html.DropDownListFor(country => country.Continent,
          new SelectList(new[] { «Африка», «Австралия») }))
</div>
..
```

Для отправки введенных данных контроллеру необходимо использовать вспомогательный метод Html.BeginForm:

```
<h2> Создание нового элемента </h2>
@using (Html.BeginForm()) {
      <div>
          <label>Название</label>
          @Html.TextBoxFor(country => country.Name)
      </div>
      ..
      <input type=”submit” value=”Submit” />
}
```

В этом случае в контроллере Country нужно определить две перегрузки метода NewCountry (одна – для начала работы, вторая – для сохранения данных):

```
public ActionResult NewCountry()
```

```
{
   return View(new Country());
}
[HttpPost]
public ActionResult NewCountry(Country c)
{
   repo.AddCountry(c);
   // Перенаправляем в начало
   return RedirectToAction(“Index”);
}
```

6. Отображение списка элементов можно организовать следующим образом:

```
@model  IEnumerable<Country>
@{
    Layout = null;
}

<!DOCTYPE html>

@foreach(var c in Model)
{
    <div>
        <h2>Страна: @c.Name</h2>
        <p>Континент: @c.Continent</p>
    </div>

}
```