**Вариант 7**

7) Создать web-приложение, содержащее сведения о сотрудниках предприятия. Каждая запись имеет поля: фамилия, имя, отчество, оклад, номер отдела. Вывести на экран фамилии и инициалы сотрудников, имеющих максимальный оклад в отделе. Также для каждого отдела вывести сумму зарплат.

1. Удалите исходный контроллер и папку Home в Views

2. В папке Models создайте файл Employee.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class Employee

{

[Key]

public int EmployeeId { get; set; }

[Required]

public string LastName { get; set; } // Фамилия

[Required]

public string FirstName { get; set; } // Имя

public string MiddleName { get; set; } // Отчество

[Required]

public decimal Salary { get; set; } // Оклад

[Required]

public int DepartmentNumber { get; set; } // Номер отдела

}

}

3. В папке Controllers создайте EmployeeController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class EmployeeController : Controller

{

// Статичный список как временная замена базы данных

private static List<Employee> employees = new List<Employee>

{

new Employee { EmployeeId = 1, LastName = "Иванов", FirstName = "Иван", MiddleName = "Иванович", Salary = 50000, DepartmentNumber = 1 },

new Employee { EmployeeId = 2, LastName = "Петров", FirstName = "Петр", MiddleName = "Петрович", Salary = 60000, DepartmentNumber = 1 },

new Employee { EmployeeId = 3, LastName = "Сидоров", FirstName = "Сидор", MiddleName = "Сидорович", Salary = 55000, DepartmentNumber = 2 },

new Employee { EmployeeId = 4, LastName = "Кузнецов", FirstName = "Алексей", MiddleName = "Николаевич", Salary = 70000, DepartmentNumber = 2 },

new Employee { EmployeeId = 5, LastName = "Смирнова", FirstName = "Ольга", MiddleName = "Владимировна", Salary = 65000, DepartmentNumber = 2 },

new Employee { EmployeeId = 6, LastName = "Васильев", FirstName = "Василий", MiddleName = "Васильевич", Salary = 45000, DepartmentNumber = 3 }

};

public ActionResult Index()

{

return View(employees);

}

public ActionResult DepartmentStats()

{

// Сотрудники с максимальной зарплатой в каждом отделе

var maxSalaryEmployees = employees

.GroupBy(e => e.DepartmentNumber)

.Select(g => new

{

Department = g.Key,

Employees = g.Where(e => e.Salary == g.Max(emp => emp.Salary))

.Select(e => $"{e.LastName} {e.FirstName[0]}.{e.MiddleName[0]}.")

})

.ToList();

// Сумма зарплат по отделам

var departmentSalaries = employees

.GroupBy(e => e.DepartmentNumber)

.Select(g => new

{

Department = g.Key,

TotalSalary = g.Sum(e => e.Salary)

})

.ToList();

// Передаем оба результата в View

ViewBag.MaxSalaryEmployees = maxSalaryEmployees;

ViewBag.DepartmentSalaries = departmentSalaries;

return View();

}

}

}

4. В папке Views создайте папку Employee, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.Employee>

<h2>Все сотрудники</h2>

<table class="table">

<thead>

<tr>

<th>Фамилия</th>

<th>Имя</th>

<th>Отчество</th>

<th>Оклад</th>

<th>Отдел</th>

</tr>

</thead>

<tbody>

@foreach (var employee in Model)

{

<tr>

<td>@employee.LastName</td>

<td>@employee.FirstName</td>

<td>@employee.MiddleName</td>

<td>@employee.Salary.ToString("C")</td>

<td>@employee.DepartmentNumber</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("DepartmentStats", "Employee")" class="btn btn-primary">Показать статистику по отделам</a>

DepartmentStats.cshtml:

<h2>Сотрудники с максимальным окладом в отделах</h2>

<table class="table">

<thead>

<tr>

<th>Отдел</th>

<th>Сотрудник (Фамилия И.О.)</th>

</tr>

</thead>

<tbody>

@foreach (var item in ViewBag.MaxSalaryEmployees)

{

<tr>

<td>@item.Department</td>

<td>@string.Join(", ", item.Employees)</td>

</tr>}</tbody>

</table>

<h2>Сумма зарплат по отделам</h2>

<table class="table">

<thead>

<tr>

<th>Отдел</th>

<th>Сумма зарплат</th>

</tr>

</thead>

<tbody>

@foreach (var item in ViewBag.DepartmentSalaries)

{

<tr>

<td>@item.Department</td>

<td>@item.TotalSalary.ToString("C")</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("Index", "Employee")" class="btn btn-secondary">Назад</a>;

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=Employee}/{action=Index}/{id?}");

**Вариант 8**

8) Создать web-приложение, содержащее сведения о справочной автовокзала, где хранится расписание движения автобусов. Для каждого рейса указаны: номер рейса, пункт назначения, время отправления, время прибытия на конечный пункт. Вывести на экран информацию о рейсах, которыми можно воспользоваться для поездки в указанный пункт назначения раньше заданного времени.

1. Удалите исходный контроллер и папку Home в Views (если они есть)

2. В папке Models создайте файл BusSchedule.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class BusSchedule

{

[Key]

public int ScheduleId { get; set; }

[Required]

public string RouteNumber { get; set; } // Номер рейса

[Required]

public string Destination { get; set; } // Пункт назначения

[Required]

[DataType(DataType.Time)]

public DateTime DepartureTime { get; set; } // Время отправления

[Required]

[DataType(DataType.Time)]

public DateTime ArrivalTime { get; set; } // Время прибытия

}

}

3. В папке Controllers создайте BusScheduleController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class BusScheduleController : Controller

{

// Статичный список как временная замена базы данных

private static List<BusSchedule> schedules = new List<BusSchedule>

{

new BusSchedule { ScheduleId = 1, RouteNumber = "101", Destination = "Минск",

DepartureTime = DateTime.Parse("08:00"), ArrivalTime = DateTime.Parse("10:30") },

new BusSchedule { ScheduleId = 2, RouteNumber = "102", Destination = "Гомель",

DepartureTime = DateTime.Parse("09:15"), ArrivalTime = DateTime.Parse("12:45") },

new BusSchedule { ScheduleId = 3, RouteNumber = "201", Destination = "Минск",

DepartureTime = DateTime.Parse("14:30"), ArrivalTime = DateTime.Parse("17:00") },

new BusSchedule { ScheduleId = 4, RouteNumber = "103", Destination = "Брест",

DepartureTime = DateTime.Parse("10:00"), ArrivalTime = DateTime.Parse("14:20") },

new BusSchedule { ScheduleId = 5, RouteNumber = "202", Destination = "Гомель",

DepartureTime = DateTime.Parse("16:45"), ArrivalTime = DateTime.Parse("20:15") }

};

public ActionResult Index()

{

return View(schedules);

}

public ActionResult Search(string destination, string time)

{

DateTime targetTime;

if (!DateTime.TryParse(time, out targetTime))

{

targetTime = DateTime.MaxValue; // Если время не указано, показываем все рейсы

}

var result = schedules

.Where(s => s.Destination.Equals(destination, StringComparison.OrdinalIgnoreCase)

&& s.DepartureTime <= targetTime)

.OrderBy(s => s.DepartureTime)

.ToList();

ViewBag.Destination = destination;

ViewBag.Time = time;

return View(result);

}

}

}

4. В папке Views создайте папку BusSchedule, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.BusSchedule>

<h2>Расписание автобусов</h2>

<table class="table">

<thead>

<tr>

<th>Номер рейса</th>

<th>Пункт назначения</th>

<th>Время отправления</th>

<th>Время прибытия</th>

</tr>

</thead>

<tbody>

@foreach (var schedule in Model)

{

<tr>

<td>@schedule.RouteNumber</td>

<td>@schedule.Destination</td>

<td>@schedule.DepartureTime.ToString("HH:mm")</td>

<td>@schedule.ArrivalTime.ToString("HH:mm")</td>

</tr>

}

</tbody>

</table>

@using (Html.BeginForm("Search", "BusSchedule", FormMethod.Get))

{

<div class="form-group">

<label>Пункт назначения:</label>

<input type="text" name="destination" class="form-control" required />

</div>

<div class="form-group">

<label>Время не позднее (HH:mm):</label>

<input type="time" name="time" class="form-control" />

</div>

<button type="submit" class="btn btn-primary">Поиск рейсов</button>

}

Search.cshtml:

@model List<YourProjectName.Models.BusSchedule>

<h2>Результаты поиска</h2>

<h4>Рейсы в @ViewBag.Destination @(string.IsNullOrEmpty(ViewBag.Time) ? "" : "до " + ViewBag.Time)</h4>

<table class="table">

<thead>

<tr>

<th>Номер рейса</th>

<th>Время отправления</th>

<th>Время прибытия</th>

</tr>

</thead>

<tbody>

@foreach (var schedule in Model)

{

<tr>

<td>@schedule.RouteNumber</td>

<td>@schedule.DepartureTime.ToString("HH:mm")</td>

<td>@schedule.ArrivalTime.ToString("HH:mm")</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("Index", "BusSchedule")" class="btn btn-secondary">Назад к расписанию</a>;

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=BusSchedule}/{action=Index}/{id?}");

**Вариант 9**

9) Создать web-приложение, содержащее сведения о междугородней АТС. Информация о разговорах содержит: дату разговора; название города, время разговора, тариф. Вывести на экран за каждый месяц по каждому городу общее время разговора с ним и сумму к оплате.

1. Удалите исходный контроллер и папку Home в Views (если они есть)

2. В папке Models создайте файл CallRecord.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class CallRecord

{

[Key]

public int CallId { get; set; }

[Required]

[DataType(DataType.Date)]

public DateTime CallDate { get; set; } // Дата разговора

[Required]

public string City { get; set; } // Название города

[Required]

public int DurationMinutes { get; set; } // Время разговора (в минутах)

[Required]

public decimal RatePerMinute { get; set; } // Тариф за минуту

}

}

3. В папке Controllers создайте CallRecordController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class CallRecordController : Controller

{

// Статичный список как временная замена базы данных

private static List<CallRecord> calls = new List<CallRecord>

{

new CallRecord { CallId = 1, CallDate = new DateTime(2023, 1, 15), City = "Москва", DurationMinutes = 12, RatePerMinute = 2.5m },

new CallRecord { CallId = 2, CallDate = new DateTime(2023, 1, 20), City = "Минск", DurationMinutes = 8, RatePerMinute = 1.8m },

new CallRecord { CallId = 3, CallDate = new DateTime(2023, 2, 5), City = "Москва", DurationMinutes = 25, RatePerMinute = 2.5m },

new CallRecord { CallId = 4, CallDate = new DateTime(2023, 2, 18), City = "Киев", DurationMinutes = 15, RatePerMinute = 2.0m },

new CallRecord { CallId = 5, CallDate = new DateTime(2023, 1, 30), City = "Минск", DurationMinutes = 10, RatePerMinute = 1.8m },

new CallRecord { CallId = 6, CallDate = new DateTime(2023, 2, 28), City = "Москва", DurationMinutes = 7, RatePerMinute = 2.5m }

};

public ActionResult Index()

{

return View(calls);

}

public ActionResult MonthlyReport()

{

var report = calls

.GroupBy(c => new {

Month = c.CallDate.Month,

Year = c.CallDate.Year,

c.City

})

.Select(g => new

{

MonthYear = $"{g.Key.Month}/{g.Key.Year}",

g.Key.City,

TotalMinutes = g.Sum(c => c.DurationMinutes),

TotalCost = g.Sum(c => c.DurationMinutes \* c.RatePerMinute)

})

.OrderBy(r => r.MonthYear)

.ThenBy(r => r.City)

.ToList();

return View(report);

}

}

}

4. В папке Views создайте папку CallRecord, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.CallRecord>

<h2>Все звонки</h2>

<table class="table">

<thead>

<tr>

<th>Дата</th>

<th>Город</th>

<th>Длительность (мин)</th>

<th>Тариф (за минуту)</th>

<th>Стоимость</th>

</tr>

</thead>

<tbody>

@foreach (var call in Model)

{

<tr>

<td>@call.CallDate.ToShortDateString()</td>

<td>@call.City</td>

<td>@call.DurationMinutes</td>

<td>@call.RatePerMinute.ToString("C")</td>

<td>@((call.DurationMinutes \* call.RatePerMinute).ToString("C"))</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("MonthlyReport", "CallRecord")" class="btn btn-primary">Показать месячный отчет</a>

MonthlyReport.cshtml:

@model IEnumerable<dynamic>

<h2>Месячный отчет по междугородним звонкам</h2>

<table class="table">

<thead>

<tr>

<th>Месяц/Год</th>

<th>Город</th>

<th>Общее время (мин)</th>

<th>Сумма к оплате</th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@item.MonthYear</td>

<td>@item.City</td>

<td>@item.TotalMinutes</td>

<td>@item.TotalCost.ToString("C")</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("Index", "CallRecord")" class="btn btn-secondary">Назад к списку звонков</a>

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=CallRecord}/{action=Index}/{id?}");

**Вариант 10**

10) Создать web-приложение, содержащее сведения об участниках спортивных соревнований. Каждая запись содержит: название команды, фамилию, имя, отчество игрока, возраст. Вывести на экран название и средний возраст самой молодой команды, а также фамилию и инициалы ее самого молодого игрока.

1. Удалите исходный контроллер и папку Home в Views (если они есть)

2. В папке Models создайте файл Player.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class Player

{

[Key]

public int PlayerId { get; set; }

[Required]

public string TeamName { get; set; } // Название команды

[Required]

public string LastName { get; set; } // Фамилия

[Required]

public string FirstName { get; set; } // Имя

[Required]

public string MiddleName { get; set; } // Отчество

[Required]

[Range(10, 50)]

public int Age { get; set; } // Возраст

}

}

3. В папке Controllers создайте PlayerController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class PlayerController : Controller

{

// Статичный список как временная замена базы данных

private static List<Player> players = new List<Player>

{

new Player { PlayerId = 1, TeamName = "Торпедо", LastName = "Иванов", FirstName = "Иван", MiddleName = "Иванович", Age = 22 },

new Player { PlayerId = 2, TeamName = "Торпедо", LastName = "Петров", FirstName = "Петр", MiddleName = "Петрович", Age = 24 },

new Player { PlayerId = 3, TeamName = "Торпедо", LastName = "Сидоров", FirstName = "Сидор", MiddleName = "Сидорович", Age = 21 },

new Player { PlayerId = 4, TeamName = "Динамо", LastName = "Кузнецов", FirstName = "Алексей", MiddleName = "Николаевич", Age = 19 },

new Player { PlayerId = 5, TeamName = "Динамо", LastName = "Смирнов", FirstName = "Олег", MiddleName = "Владимирович", Age = 20 },

new Player { PlayerId = 6, TeamName = "Спартак", LastName = "Васильев", FirstName = "Василий", MiddleName = "Васильевич", Age = 23 },

new Player { PlayerId = 7, TeamName = "Спартак", LastName = "Николаев", FirstName = "Николай", MiddleName = "Николаевич", Age = 25 }

};

public ActionResult Index()

{

return View(players);

}

public ActionResult YoungestTeam()

{

// Находим средний возраст по командам

var teamAverages = players

.GroupBy(p => p.TeamName)

.Select(g => new

{

TeamName = g.Key,

AverageAge = g.Average(p => p.Age),

Players = g.OrderBy(p => p.Age).ToList()

})

.OrderBy(t => t.AverageAge)

.ToList();

// Самая молодая команда

var youngestTeam = teamAverages.FirstOrDefault();

if (youngestTeam != null)

{

// Самый молодой игрок в команде

var youngestPlayer = youngestTeam.Players.First();

ViewBag.TeamName = youngestTeam.TeamName;

ViewBag.AverageAge = Math.Round(youngestTeam.AverageAge, 1);

ViewBag.YoungestPlayer = $"{youngestPlayer.LastName} {youngestPlayer.FirstName[0]}.{youngestPlayer.MiddleName[0]}.";

}

return View();

}

}

}

4. В папке Views создайте папку Player, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.Player>

<h2>Все участники соревнований</h2>

<table class="table">

<thead>

<tr>

<th>Команда</th>

<th>Фамилия</th>

<th>Имя</th>

<th>Отчество</th>

<th>Возраст</th>

</tr>

</thead>

<tbody>

@foreach (var player in Model)

{

<tr>

<td>@player.TeamName</td>

<td>@player.LastName</td>

<td>@player.FirstName</td>

<td>@player.MiddleName</td>

<td>@player.Age</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("YoungestTeam", "Player")" class="btn btn-primary">Показать самую молодую команду</a>

YoungestTeam.cshtml:

<h2>Самая молодая команда</h2>

<div class="alert alert-info">

<h4>Название команды: @ViewBag.TeamName</h4>

<p>Средний возраст: @ViewBag.AverageAge лет</p>

<p>Самый молодой игрок: @ViewBag.YoungestPlayer</p>

</div>

<a href="@Url.Action("Index", "Player")" class="btn btn-secondary">Назад к списку игроков</a>

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=Player}/{action=Index}/{id?}");

**Вариант 11**

11) Создать web-приложение, содержащее сведения об участниках спортивных соревнований. Каждая запись содержит: название команды, фамилию, имя, отчество игрока, рост. Вывести на экран название и средний возраст самой рослой команды, а также фамилию и инициалы ее самого высокорослого игрока.

1. Удалите исходный контроллер и папку Home в Views (если они есть)

2. В папке Models создайте файл Player.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class Player

{

[Key]

public int PlayerId { get; set; }

[Required]

public string TeamName { get; set; } // Название команды

[Required]

public string LastName { get; set; } // Фамилия

[Required]

public string FirstName { get; set; } // Имя

[Required]

public string MiddleName { get; set; } // Отчество

[Required]

[Range(150, 250)]

public int Height { get; set; } // Рост в см

}

}

3. В папке Controllers создайте PlayerController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class PlayerController : Controller

{

// Статичный список как временная замена базы данных

private static List<Player> players = new List<Player>

{

new Player { PlayerId = 1, TeamName = "Локомотив", LastName = "Иванов", FirstName = "Иван", MiddleName = "Иванович", Height = 185 },

new Player { PlayerId = 2, TeamName = "Локомотив", LastName = "Петров", FirstName = "Петр", MiddleName = "Петрович", Height = 192 },

new Player { PlayerId = 3, TeamName = "Локомотив", LastName = "Сидоров", FirstName = "Сидор", MiddleName = "Сидорович", Height = 188 },

new Player { PlayerId = 4, TeamName = "Зенит", LastName = "Кузнецов", FirstName = "Алексей", MiddleName = "Николаевич", Height = 195 },

new Player { PlayerId = 5, TeamName = "Зенит", LastName = "Смирнов", FirstName = "Олег", MiddleName = "Владимирович", Height = 198 },

new Player { PlayerId = 6, TeamName = "Спартак", LastName = "Васильев", FirstName = "Василий", MiddleName = "Васильевич", Height = 182 },

new Player { PlayerId = 7, TeamName = "Спартак", LastName = "Николаев", FirstName = "Николай", MiddleName = "Николаевич", Height = 184 }

};

public ActionResult Index()

{

return View(players);

}

public ActionResult TallestTeam()

{

// Находим средний рост по командам

var teamAverages = players

.GroupBy(p => p.TeamName)

.Select(g => new

{

TeamName = g.Key,

AverageHeight = g.Average(p => p.Height),

Players = g.OrderByDescending(p => p.Height).ToList()

})

.OrderByDescending(t => t.AverageHeight)

.ToList();

// Самая рослая команда

var tallestTeam = teamAverages.FirstOrDefault();

if (tallestTeam != null)

{

// Самый высокий игрок в команде

var tallestPlayer = tallestTeam.Players.First();

ViewBag.TeamName = tallestTeam.TeamName;

ViewBag.AverageHeight = Math.Round(tallestTeam.AverageHeight, 1);

ViewBag.TallestPlayer = $"{tallestPlayer.LastName} {tallestPlayer.FirstName[0]}.{tallestPlayer.MiddleName[0]}.";

ViewBag.TallestPlayerHeight = tallestPlayer.Height;

}

return View();

}

}

}

4. В папке Views создайте папку Player, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.Player>

<h2>Все участники соревнований</h2>

<table class="table">

<thead>

<tr>

<th>Команда</th>

<th>Фамилия</th>

<th>Имя</th>

<th>Отчество</th>

<th>Рост (см)</th>

</tr>

</thead>

<tbody>

@foreach (var player in Model)

{

<tr>

<td>@player.TeamName</td>

<td>@player.LastName</td>

<td>@player.FirstName</td>

<td>@player.MiddleName</td>

<td>@player.Height</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("TallestTeam", "Player")" class="btn btn-primary">Показать самую рослую команду</a>

TallestTeam.cshtml:

<h2>Самая рослая команда</h2>

<div class="alert alert-info">

<h4>Название команды: @ViewBag.TeamName</h4>

<p>Средний рост: @ViewBag.AverageHeight см</p>

<p>Самый высокий игрок: @ViewBag.TallestPlayer (@ViewBag.TallestPlayerHeight см)</p>

</div>

<a href="@Url.Action("Index", "Player")" class="btn btn-secondary">Назад к списку игроков</a>

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=Player}/{action=Index}/{id?}");

**Вариант 12**

12) Создать web-приложение, содержащее сведения о книгах, хранящихся в библиотеке. Для каждой записи задаются: автор, название, год издания. Вывести на экран список книг с фамилиями авторов в алфавитном порядке, изданных после 2022 года; список книг заданного автора.

1. Удалите исходный контроллер и папку Home в Views (если они есть)

2. В папке Models создайте файл Book.cs:

using System;

using System.ComponentModel.DataAnnotations;

namespace YourProjectName.Models

{

public class Book

{

[Key]

public int BookId { get; set; }

[Required]

public string AuthorLastName { get; set; } // Фамилия автора

[Required]

public string Title { get; set; } // Название книги

[Required]

[Range(1000, 2100)]

public int YearPublished { get; set; } // Год издания

}

}

3. В папке Controllers создайте BookController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using YourProjectName.Models;

namespace YourProjectName.Controllers

{

public class BookController : Controller

{

// Статичный список как временная замена базы данных

private static List<Book> books = new List<Book>

{

new Book { BookId = 1, AuthorLastName = "Толстой", Title = "Война и мир", YearPublished = 1869 },

new Book { BookId = 2, AuthorLastName = "Достоевский", Title = "Преступление и наказание", YearPublished = 1866 },

new Book { BookId = 3, AuthorLastName = "Пелевин", Title = "Transhumanism Inc.", YearPublished = 2023 },

new Book { BookId = 4, AuthorLastName = "Глуховский", Title = "Пост", YearPublished = 2022 },

new Book { BookId = 5, AuthorLastName = "Сорокин", Title = "Доктор Гарин", YearPublished = 2023 },

new Book { BookId = 6, AuthorLastName = "Пелевин", Title = "KGBT+", YearPublished = 2022 }

};

public ActionResult Index()

{

return View(books);

}

public ActionResult RecentBooks()

{

var recentBooks = books

.Where(b => b.YearPublished > 2022)

.OrderBy(b => b.AuthorLastName)

.ToList();

return View(recentBooks);

}

public ActionResult AuthorBooks(string author)

{

var authorBooks = books

.Where(b => b.AuthorLastName.Equals(author, StringComparison.OrdinalIgnoreCase))

.OrderBy(b => b.YearPublished)

.ToList();

ViewBag.Author = author;

return View(authorBooks);

}

}

}

4. В папке Views создайте папку Book, а в ней:

Index.cshtml:

@model List<YourProjectName.Models.Book>

<h2>Все книги в библиотеке</h2>

@using (Html.BeginForm("AuthorBooks", "Book", FormMethod.Get))

{

<div class="form-group">

<label>Поиск по автору:</label>

<input type="text" name="author" class="form-control" required />

</div>

<button type="submit" class="btn btn-primary">Найти книги автора</button>

}

<table class="table">

<thead>

<tr>

<th>Автор</th>

<th>Название</th>

<th>Год издания</th>

</tr>

</thead>

<tbody>

@foreach (var book in Model)

{

<tr>

<td>@book.AuthorLastName</td>

<td>@book.Title</td>

<td>@book.YearPublished</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("RecentBooks", "Book")" class="btn btn-primary">Книги после 2022 года</a>

RecentBooks.cshtml:

@model List<YourProjectName.Models.Book>

<h2>Книги, изданные после 2022 года</h2>

<table class="table">

<thead>

<tr>

<th>Автор</th>

<th>Название</th>

<th>Год издания</th>

</tr>

</thead>

<tbody>

@foreach (var book in Model)

{

<tr>

<td>@book.AuthorLastName</td>

<td>@book.Title</td>

<td>@book.YearPublished</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("Index", "Book")" class="btn btn-secondary">Назад к списку книг</a>

AuthorBooks.cshtml:

@model List<YourProjectName.Models.Book>

<h2>Книги автора: @ViewBag.Author</h2>

<table class="table">

<thead>

<tr>

<th>Название</th>

<th>Год издания</th>

</tr>

</thead>

<tbody>

@foreach (var book in Model)

{

<tr>

<td>@book.Title</td>

<td>@book.YearPublished</td>

</tr>

}

</tbody>

</table>

<a href="@Url.Action("Index", "Book")" class="btn btn-secondary">Назад к списку книг</a>

5. В Program.cs измените маршрутизацию:

app.MapControllerRoute(

name: "default",

pattern: "{controller=CallRecord}/{action=Index}/{id?}");