API кконтроллерлермен жұмыс

Бұл тарауда мен веб-қосымшаның шаблонын қолдандым ASP.NET ApiControllers деп аталатын жаңа бос жобаны жасау үшін Core (. NET Core).

Модель және репозиторий құру

Мен Models қалтасын құрудан, Reservation деп аталатын сынып файлын қосудан бастадым.cs және оны 20-1 листингінде көрсетілген модель класын анықтау үшін пайдалану.

Листинг 20-1. Reservation файлының мазмұны.модельдер қалтасындағы cs

```
namespace ApiControllers.Models {

public class Reservation {

public int ReservationId { get; set; }

public string ClientName { get; set; }

public string Location { get; set; }

}

}
```

Мен сондай-ақ IRepository файлын қостым.CS models қалтасында және оны 20-2 Листингте көрсетілгендей модель репозиторийіне арналған интерфейсті анықтау үшін қолданды.

Листинг 20-2. IRepository файлының мазмұны.модельдер қалтасындағы cs

```
using System.Collections.Generic;

namespace ApiControllers.Models {

public interface IRepository {

IEnumerable<Reservation> Reservations { get; }

Reservation this[int id] { get; }

Reservation AddReservation(Reservation reservation);

Reservation UpdateReservation(Reservation reservation);

voidDeleteReservation(intid);
```

```
}

}
```

Мен MemoryRepository класс файлын қостым.CS models қалтасында және оны 20-3 Листингте көрсетілгендей IRepository интерфейсінің тұрақты емес орындалуын анықтау үшін қолданды.

Листинг 20-3. MemoryRepository файлының мазмұны.модельдер қалтасындағы cs

```
using System.Collections.Generic;

namespace ApiControllers.Models {

public class MemoryRepository : IRepository {

private Dictionary<int, Reservation> items;

public MemoryRepository() {

items = new Dictionary<int, Reservation>();

new List<Reservation> {

new Reservation { ClientName = "Alice", Location = "Board Room" },

new Reservation { ClientName = "Bob", Location = "Lecture Hall" },

new Reservation { ClientName = "Joe", Location = "Meeting Room 1" } }.ForEach(r =>
AddReservation(r));

}

public Reservation this[int id] => items.ContainsKey(id) ? items[id] : null;

public IEnumerable<Reservation> Reservations => items.Values;

public Reservation AddReservation(Reservation reservation) {

if (reservation.ReservationId == 0) {

int key = items.Count;

while (items.ContainsKey(key)) { key++; };

reservation.ReservationId = key;

}

items[reservation.ReservationId] = reservation;

return reservation;
```

```
}

public void DeleteReservation(int id) => items.Remove(id);

public Reservation UpdateReservation(Reservation reservation)

=>AddReservation(reservation);

}

}
```

Сақтау Қарапайым модель нысандарының жиынтығын жасайды және тұрақты сақтау болмағандықтан, бағдарлама тоқтатылған немесе қайта іске қосылған кезде кез-келген өзгерістер жоғалады. (SportsStore қосымшасының мысалы ретінде тұрақты репозиторийді қалай құруға болатындығы туралы 8-тарауды қараңыз.)

Контроллер мен көріністерді жасау

Кейінірек осы тарауда мен RESTful контроллерлерін жасаймын, бірақ дайындық барысында келесі мысалдарға негіз беру үшін қарапайым контроллер жасауым керек. Мен Controllers қалтасын жасадым, HomeController деп аталатын файлды қостым.cs және оны 20-4 листингінде көрсетілген контроллерді анықтау үшін қолданды.

Листинг 20-4. HomeController файлының мазмұны.контроллер қалтасындағы cs

```
using Microsoft.AspNetCore.Mvc; using ApiControllers.Models;

namespace ApiControllers.Controllers {

public class HomeController : Controller {

private IRepository repository { get; set; }

public HomeController(IRepository repo) => repository = repo;

public ViewResult Index() => View(repository.Reservations);

[HttpPost]

public IActionResult AddReservation(Reservation reservation) { repository.AddReservation(reservation);

returnRedirectToAction("Index");

}

}
```

```
}
```

Бұл контроллер әдепкі бағдарлама болып табылатын және деректер моделін көрсететін Index әрекетін анықтайды. Ол сонымен қатар AddReservation әрекетін анықтайды, ол тек HTTP POST сұраулары үшін қол жетімді және пайдаланушыдан форма деректерін алу үшін қолданылады. Бұл әрекеттер 17-тарауда сипатталған Post / Redirect / Get үлгісіне сәйкес келеді, осылайша веб-бетті қайта жүктеу кезінде қайталанатын пішін көрінісі жасалмайды. Мен HTML мазмұнын құжаттың тақырыбынан бөлу үшін макет жасадым, бұл кейінірек осы тарауда енгізген кейбір өзгертулерді жеңілдетеді. Мен Views / Shared қалтасын жасадым, _Layout файлы деп аталатын орналасуды қостым.cshtml және 20-5 листингінде көрсетілген түзетуді қосты.

Листинг 20-5. _Layout файлының мазмұны.көрініс қалтасындағы cshtml / ортақ

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>RESTful Controllers</title>
<link asp-href-include="lib/bootstrap/dist/css/*.min.css" rel="stylesheet" />
</head>
<body class="m-1 p-1">
@RenderBody()
</body>
</html>
```

Содан кейін мен Views / Home қалтасын жасадым, Index деп аталатын көрініс файлын қостым.cshtml және 20-6 листингінде көрсетілген мазмұнды қосты.

Листинг 20-6. Индекс файлының мазмұны.көрініс / үй қалтасындағы cshtml

```
@model IEnumerable<Reservation>
@{
```

```
Layout = "_Layout";
}
<form id="addform" asp-action="AddReservation" method="post">
<div class="form-group">
<label for="ClientName">Name:</label>
<input class="form-control" name="ClientName" />
</div>
<div class="form-group">
<label for="Location">Location:</label>
<input class="form-control" name="Location" />
</div>
<div class="text-center panel-body">
<button type="submit" class="btn btn-sm btn-primary">Add</button>
</div>
</form>
<table class="table table-sm table-striped table-bordered m-2">
<thead>
<tr>
<th>ID</th>
<th>Client</th>
<th>Location</th>
</tr>
</thead>
<tbody>
@foreach (var r in Model) {
<tr>
<td>@r.ReservationId</td>
<td>@r.ClientName</td>
<td>@r.Location</td>
```

```
</tr>

}

</tbody>

</table>
```

Бұл қатаң терілген көрініс Reservation нысандарының тізбегін өзінің моделі ретінде алады және кестені толтыру үшін Razorforeach циклын қолданады. Сондай-ақ, AddReservation әрекетіне POST сұрауларын жіберу үшін теңшелген форма бар.

Осы тараудағы мысалдар BootstrapCSS пакетіне байланысты. Жобаға Bootstrap қосу үшін мен bower файлын жасау үшін bowerconfigurationfile элементінің шаблонын қолдандым.JSON жобаның түбірінде және пакетті 20-7 Листингте көрсетілгендей тәуелділік бөліміне қосты.

Листинг 20-7. Буманы bower файлына қосу.json

```
{

"name": "asp.net",

"private": true,

"dependencies": {

"bootstrap": "4.0.0-alpha.6"

}

}
```

Содан кейін мен _ViewImports файлын жасадым.Views қалтасындағы cshtml және оны 20-8 Листингте көрсетілгендей, Razor көріністерінде және модель аттар кеңістігін импорттау үшін кірістірілген тег көмекшілерін конфигурациялау үшін қолданды.

Листинг 20-8. _ViewImports файлының мазмұны.көрініс қалтасындағы cshtml

```
@using ApiControllers.Models

@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

MVCFramework және әзірлеу үшін қажет аралық бағдарламалық жасақтама компоненттерін қосу үшін мен Startup класына 20-9 листингінде көрсетілген өзгертулер енгіздім. Мен сондай-ақ модель репозиторийіне қызмет салыстыруын орнату үшін AddSingleton әдісін қолдандым.

Листинг 20-9. Startup файлында аралық бағдарламалық жасақтаманы қосу.ApiControllers қалтасындағы cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.DependencyInjection;

using ApiControllers.Models;

namespace ApiControllers {

public class Startup {

public void ConfigureServices(IServiceCollection services) { services.AddSingleton<IRepository,
MemoryRepository>();

services.AddMvc();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env) { app.UseStatusCodePages();

app.UseDeveloperExceptionPage();

app.UseStaticFiles();

app.UseMvcWithDefaultRoute();

}

}

}
```

HTTP портын орнату осы тараудағы кейбір мысалдар URL мекен-жайларын қолмен енгізу арқылы тексеріледі. Сипаттаманы жеңілдету үшін Мен HTTP сұрауларын алу үшін пайдаланылатын портты сұраймын. VisualStudioProject мәзірінен ApiControllersProperties таңдаңыз, Debug қойындысын ашыңыз және 20-1 суретте көрсетілгендей http: // localhost: 7000 / -де қосымшаның URL өрісінің мәнін өзгертіңіз. Порт нөмірін орнатқаннан кейін өзгертулерді сақтауды ұмытпаңыз.

Сурет 20-1. Қолданбаның URL мекенжайын орнату

Бағдарламаны іске қосыңыз, пішінді толтырыңыз және "Қосу" түймесін басыңыз; қолданба 20-2 суретте көрсетілгендей модельге жаңа тапсырыс қосады. Репозиторийге енгізген өзгертулер тұрақты емес және бағдарлама тоқтаған немесе қайта іске қосылған кезде жоғалады.

Сурет 20-2. Мысал қосымшасын қолдану