# Тәуелділіктерді енгізу

Бұл тарауда мен үлгіні қолдандым ASP.NET DependencyInjection деп аталатын жаңа бос жобаны құру үшін Core Web Application (. NET Core). 18-1-Листингте жоба үшін аралық бағдарламалық жасақтама қызметтері мен компоненттерін конфигурациялайтын Startup класы көрсетілген.

Listing 18-1.Мазмұны: Startup Файлы.cs впапке DependencyInjection

```
используяСистему;


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;


namespace DependencyInjection {

public class Startup {

public void ConfigureServices(IServiceCollection services) {

services.AddMvc();

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env) {

app.UseStatusCodePages();

app.UseDeveloperExceptionPage();

app.UseStaticFiles();

app.UseMvcWithDefaultRoute();

}
```

```
}

}
```

Модель және репозиторий құру

Осы тараудағы мысалдар мен Models қалтасын жасап, product деп аталатын сынып файлын қосу арқылы жасаған қарапайым модельді қажет етеді.мен 18-2 Листингте көрсетілген сыныпты анықтау үшін қолданған cs.

Listing 18-2. Өнім файлының мазмұны.модельдер қалтасындағы cs

```
namespace DependencyInjection.Models {

public class Product {

public string Name { get; set; }

public decimal Price { get; set; }

}

}
```

Модельді басқару үшін мен IRepository деп аталатын сыныпты қостым.CS models қалтасында және оны 18-3 Листингте көрсетілген интерфейсті анықтау үшін қолданды.

Listing 18-3. IRepository файлының мазмұны.модельдер қалтасындағы cs

```
using System.Collections.Generic;

namespace DependencyInjection.Models {

public interface IRepository {

IEnumerable<Product> Products { get; }

Product this[string name] { get; }

void AddProduct(Product product);

void DeleteProduct(Product product);
```

```
}
}
```

Интерфейс өнім объектілерінің жинағында орындалатын әрекеттерді анықтайды. Интерфейстің орындалуын қамтамасыз ету үшін мен MemoryRepository класс файлын қостым.CS models қалтасында және 18-4 Листингте көрсетілген сыныпты анықтады.

Listing 18-4. MemoryRepository файлының мазмұны.модельдер қалтасындағы cs

```
using System.Collections.Generic;

namespace DependencyInjection.Models {
public class MemoryRepository : IRepository {
private Dictionary<string, Product> products;
public MemoryRepository() {
products = new Dictionary<string, Product>();
new List<Product> {
new Product { Name = "Kayak", Price = 275M },
new Product { Name = "Lifejacket", Price = 48.95M },
new Product { Name = "Soccer ball", Price = 19.50M }
}
.ForEach(p => AddProduct(p));
}
public IEnumerable<Product> Products => products.Values;
public Product this[string name] => products[name];
public void AddProduct(Product product) =>
products[product.Name] = product;
public void DeleteProduct(Product product) =>
products.Remove(product.Name);
```

```
}

}
```

MemoryRepository класы жеке объектіні сақтап қалады. Бұл дегеніміз, тұрақты сақтау жоқ және қолданбаны тоқтату немесе қайта іске қосу модельді конструкторда жасалған деректер нысандарының үлгілеріне қалпына келтіреді. Бұл нақты жоба үшін ақылға қонымды тәсіл емес, бірақ бұл қосымшаның жұмысының басқа аспектісіне назар аударатын осы тарау үшін жеткілікті болады.

Контроллер мен көріністі жасау

Мен Controllers қалтасын жасадым, HomeController деп аталатын сынып файлын қостым.cs және оны 18-5 Листингте көрсетілген сыныпты анықтау үшін қолданды.

Listing 18-5. HomeController файлының мазмұны.контроллер қалтасындағы cs

```
using Microsoft.AspNetCore.Mvc;

namespace DependencyInjection.Controllers {

public class HomeController : Controller {

public ViewResult Index() => View();

}

}
```

Контроллерде әдепкі көріністі көрсететін ViewResult жасау үшін View әдісін қолданатын бір ғана әрекет әдісі бар. Әрекет әдісіне қатысты көріністі жасау үшін мен Views / Home қалтасын жасап, Индекс деп аталатын Razor файлын қостым.cshtml. Listing 18-6 мен көрініске қосқан таңбаны көрсетеді.

Listing 18-6. Индекс файлының мазмұны.көрініс / үй қалтасындағы cshtml

```
@model IEnumerable <Product>

@ {Layout = null; }
```

```
<! DOCTYPE html>

< html >

<head >

<meta name = "viewport" content = "width = device-width" />

<title > тәуелділіктерді енгізу< / title>

<link rel = "stylesheet" asp-href-include = "lib / bootstrap / dist / css / *. min.css" />

</ head >

<body class = "m-1 p-1">

@if (ViewData.Count> 0) {

<table class = "table table-border-table-sm table-striped">

@foreach (var kvp ViewData-да) {

<tr>

<td>@kvp.Key</td>

<td>@kvp.Value</td>

</tr>

}

</table>

}

<table class="table table-bordered table-sm table-striped">

<thead>

<tr>

<th>Name</th>

<th>Price</th>

</tr>

</thead>

<tbody>

@if (Model == null) {

<tr><td colspan="3" class="text-center">No Model Data</td></tr>

} else {
```

```
@foreach (var p in Model) {
<tr>
<td>@p.Name</td>
<td>@string.Format("{0:C2}", p.Price)</td>
</tr>
}
}
</tbody>
</table>
</body>
</html>
```

Көрініс өнім нысандарының тізімін қолдана отырып қатаң түрде теріледі, ал көріністің негізгі мазмұны HTML кестесі болып табылады. Егер контроллер модель туралы ешқандай мәлімет бермесе, онда хабарлама кестенің жалғыз мазмұны ретінде көрсетіледі. Егер модельдік деректер болса, онда тізімге әрбір өнім нысаны үшін жол кестеге қосылады. Сондай-ақ, егер бар болса, view bag-де кілттер мен мәндерді тізімдейтін кесте бар, бірақ басқаша жасырылған. Мен бұл кестені кейінірек осы тарауда қолданамын.

Көрініс HTML элементтерін сәндеуге арналған bootstrap CSS пакетіне байланысты. Жобаға Bootstrap қосу үшін мен bower файлын құру үшін Bower конфигурация файлы элементінің шаблонын қолдандым.json және Bootstrap пакетін 18-7 Листингте көрсетілгендей тәуелділік бөліміне қосты.

Listing 18-7. Жүктеу жолағын жүктеу файлына қосу.JSON DependencyInjection қалтасында

```
{
"name": "asp.net",
"private": true,
"dependencies": {
"bootstrap": "4.0.0-alpha.6"
}
}
```

Соңғы дайындық _ViewImports файлын жасау болып табылады.views қалтасындағы cshtml, ол Razor көріністерінде пайдалану үшін кірістірілген тег көмекшілерін орнатады және 18-8 Листингте көрсетілгендей модельдік аттар кеңістігін импорттайды.

Listing 18-8. _ViewImports файлының мазмұны.көрініс қалтасындағы cshtml

```
@using DependencyInjection.Models

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