Create the ASP.NET Core MVC Web App.

> **创建 ASP.NET Core MVC Web 应用。**

When creating a new visual studio project, select ASP.NET Core Web App (Model-View-Controller) project template. For consistency name the project OrgMgmt

> 创建新的 Visual Studio 项目时，选择 ASP.NET Core Web 应用（Model-View-Controller）项目模板。为保持一致性，将项目命名为 OrgMgmt

.

For now, set Authentication Type to none and uncheck everything else.

> 暂时 将身份验证类型设为无，并取消选中其他所有选项。

You can choose the latest .net core version but older ones should work as well.

> 你可以选择最新的 .net core 版本 ，但较旧的版本也应该可用。

Make sure that the template project runs fine. Index action of the Home Controller will run and produce a welcome view.

> Make 确保模板项目可以正常运行。Home 控制器的 Index action 将运行并生成欢迎视图。

Include the following NuGet packages (only versions compatible with your .net core version will install):

> 包括以下 NuGet 包（只有与你的 .net core 版本兼容的版本会被安装）：

Tools -> NuGet Package Manager -> Manage NuGet packages for Solution. Browse for and install the following packages.

> 工具 -> NuGet 包管理器 -> 管理解决方案的 NuGet 包。 浏览并安装 以下包 。

Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore

Microsoft.EntityFrameworkCore.SqlServer

You can check the .net core version installed on computer by typing the following command on the windows command prompt:

> 您可以在 Windows 命令提示符中输入以下命令来检查计算机上安装的 .net core 版本：

dotnet --version

Create a model class Client (In the OrgMgmt project folder of the OrgMgmt solution right click on Models -> Add -> Class.. and then name the class Client.cs)

> 创建一个模型类 Client（在 OrgMgmt 解决方案的 OrgMgmt 项目文件夹中，右键单击 Models -> 添加 -> 类..，然后将类命名为 Client.cs）

.

namespace OrgMgmt.Models

{

public class Client

{

public int ID { get; set; }

public string Name { get; set; }

public string Address { get; set; }

}

}

and the database context class named OrgDbContext (Right click OrgMgmt project folder -> Add -> Class.. and then name the class OrgDbContext.cs)

> 以及名为数据库上下文类 OrgDbContext（在 OrgMgmt 项目文件夹上右键 -> 添加 -> 类..，然后将类命名为 OrgDbContext.cs）

.

```
using Microsoft.EntityFrameworkCore;
using OrgMgmt.Models;
namespace OrgMgmt
{
public class OrgDbContext : DbContext
{
public OrgDbContext(DbContextOptions<OrgDbContext> options) : base(options) { }
public OrgDbContext() { }
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
base.OnModelCreating(modelBuilder);
modelBuilder.Entity<Client>().ToTable("Client");
}
public DbSet<Client> Clients { get; set; }
}
}
```

Refer to the attached OrgMgmt.zip for the reference.

> 请参阅随附的 OrgMgmt.zip 以供参考。

You will need to register the database context (in Program.cs) and provide value for the connection string (in appsettings.json). If you like the database to be created if it does not already exists then you need to specify that by calling dbcontext.Database.EnsureCreated(). Check out the attached MVC project (in particular the Program.cs and appsettings.json)

> 你需要在 Program.cs 中注册数据库上下文，并在 appsettings.json 中提供连接字符串的值。如果你希望在数据库尚不存在时创建数据库，则需要通过调用 dbcontext.Database.EnsureCreated() 来指定。请查看随附的 MVC 项目（特别是 Program.cs 和 appsettings.json）。

.

Program.cs should look like as follows:

> Program.cs 应如下所示：

```
using Microsoft.EntityFrameworkCore;
using OrgMgmt;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
// 将服务添加到容器。

builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<OrgDbContext>(options => options.UseSqlServer(
builder.Configuration.GetConnectionString("DefaultConnection")
));
var app = builder.Build();
using (var scope = app.Services.CreateScope())
{
var dbContext = scope.ServiceProvider.GetRequiredService<OrgDbContext>();
dbContext.Database.EnsureCreated();
}
// Configure the HTTP request pipeline.
// 配置 HTTP 请求管道。

if (!app.Environment.IsDevelopment())
{
app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapControllerRoute(
name: "default",
pattern: "{controller=Home}/{action=Index}/{id?}");
app.Run();
```

appsettings.json should look like as follows:

> appsettings.json 应如下所示：

```
{
"ConnectionStrings": {
"DefaultConnection": "Server=
(localdb)\\mssqllocaldb;Database=OrgMgmt;Trusted_Connection=True;MultipleActiveRe
sultSets=true"
},
"Logging": {
"LogLevel": {
"Default": "Information",
"Microsoft.AspNetCore": "Warning"
}
},
"AllowedHosts": "*"
}
```

Refer to the following link for the above steps.

> 有关上述步骤，请参阅以下链接。

https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro?view=aspnetcore-7.0

Also check out the LaunchSettings.json in the properties folder of the project.

> 另请查看项目属性文件夹中的 LaunchSettings.json。

After the above steps, build the project and generate controller and also the views for the model class as follows:

> 在完成上述步骤后，构建项目并为模型类生成控制器和视图，操作如下：

[right click on controllers folder] -> Add -> New Scaffolded Item -> MVC controller with Views using EntityFramework

> [在 controllers 文件夹上右击] -> 添加 -> 新建脚手架项 -> 使用 EntityFramework 的 MVC 带视图的控制器

.

For model class choose Client and for db context class choose OrgDbContext

> 在模型类中选择 Client，在数据库上下文类中选择 OrgDbContext

Note: There is some possibility that you may be asked to download yet another nuget package at this time to facilitate this auto-generation probably if you have older versions of .net core installed on your computer

> 注意：在此过程中你可能会被要求下载另一个 nuget 包来辅助此自动生成，尤其是在你电脑上安装了较旧版本的 .NET Core 时

.

If you run the project and append “/Clients” to the url on the automatically launched browser instance you will see the Clients index view which would allow you to manage clients. Alternative, add “Clients” tab to the navigation bar by adding the following “nav-item” in the _layout.cshtml file in Views -> Shared folder (see for reference the _layout.cshtml in the attached sample project to make this addition to the navigation bar)

> 如果你运行该项目并在自动启动的浏览器实例的 URL 后追加“/Clients”，你将看到 Clients 索引视图，该视图允许你管理客户。或者通过在 Views -> Shared 文件夹中的 _layout.cshtml 文件中添加以下“nav-item”将“Clients”选项卡添加到导航栏（作为参考，请参阅附带示例项目中的 _layout.cshtml，以便向导航栏添加此项）

.

```
<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Clients" asp-
action="Index">Clients</a>

</li>
```

After you run the app, the database (if does not exist) will be recreated that you can check via Sql Server Management Studio.

> 在你运行应用后，数据库（如果不存在）将被重新创建，你可以通过 SQL Server Management Studio 进行核查。

If you are using JetBrains Rider on MacOS or Linux, the steps are mostly the same.

> 如果您在 MacOS 或 Linux 上使用 JetBrains Rider，步骤大致相同。

Make sure that .Net is installed by typing the following on the terminal:

> 确保已在终端输入以下内容以确认已安装 .Net：

dotnet --version

If you don’t see .net installed then follow online Microsoft tutorials on installing .net on MacOS

> 如果您没有看到已安装 .net，请按照微软的在线教程在 MacOS 上安装 .net

Instead of SQL Server, you can use Sqlite and thus install the following NuGet package instead:

> 您可以使用 Sqlite 替代 SQL Server，因此可以改为安装以下 NuGet 包：

Microsoft.EntityFrameworkCore.Sqlite

Replace `options.UseSqlServer` with `options.UseSqlite` in the `Program.cs` file and the connection string in the `appsettings.json` should look like as follows:

> 在 `Program.cs` 文件中将 `options.UseSqlServer` 替换为 `options.UseSqlite`，并且 `appsettings.json` 中的连接字符串 应如下所示：

```
"ConnectionStrings": {

"DefaultConnection": "Data Source=app.db"

}
```

If you get “unrecognized option –useSqlite” while scaffolding, install the code generator tool by opening the terminal window in Rider:

> 如果在脚手架操作时收到 “unrecognized option –useSqlite” 错误，请在 Rider 中打开终端窗口并安装 代码生成工具：

```
dotnet tool install -g dotnet-aspnet-codegenerator
```

and make sure the following NuGet packages exist:

> 并确保存在以下 NuGet 包：

Microsoft.VisualStudio.Web.CodeGeneration.Design
Microsoft.EntityFrameworkCore.Tools