开场

大家好，本次是进击吧！Blazor！系列课程第三次课程，上一次课程我们完成了ToDo应用的界面制作，这一节课我们要将客户端的数据写入数据库，并从数据库中读物我们需要的数据。

我们先来看一下这个数据交互流程

客户端—》http—》WebAPI—》EF—》DataBase（PPT）

Blazor—》HttpClient—》Asp.net core—》EF—》SQLServer

* Blazor

Blazor客户端就是我们上节课做的ToDo程序。

* HttpClient

HttpClient就是我们完成网络通讯用的组件，对于这类组件我们希望在一个应用中只构造一次，这样避免重复分配资源，所以我们需要在Program.cs中进行注册。（查看代码）

这里就引入了一个“依赖关系注入 (DI)”的概念

应用可通过将内置服务注入组件来使用这些服务。 应用还可定义和注册自定义服务，并通过 DI 使其在整个应用中可用。（PPT）

DI 是一种技术，它用于访问配置在中心位置的服务。 该技术可在 Blazor 应用中用于以下方面：

跨多个组件共享服务类的单个实例，称为“单一实例”服务。（PPT）

通过使用引用抽象将组件与具体服务类分离。（PPT）

服务生存期（PPT）

Scoped：Blazor WebAssembly 应用当前没有 DI 范围的概念。 已注册 Scoped 的服务的行为与 Singleton 服务类似。 但是，Blazor Server 托管模型支持 Scoped 生存期。 在 Blazor Server 应用中，范围内服务注册的范围为“连接”。 因此，即使当前意图是在浏览器中运行客户端，对于范围应限定为当前用户的服务来说，首选使用 Scoped 服务。

Singleton：DI 创建服务的单个实例。 需要 Singleton 服务的所有组件都会接收同一服务的实例。

Transient：每当组件从服务容器获取 Transient 服务的实例时，它都会接收该服务的新实例。

这里的HttpClient使用了AddScoped方法，那么就是当前范围内使用同一个实例，因为项目是WASM的，所以相当于单例了。

因此这里设置的参数在全局生效了。介绍BaseAddress、DefaultRequestHeaders、Timeout

* ASP.Net Core

我用ASP.Net Core项目给Blazor应用提供WebAPI接口，一起看一下项目结构

**Properties\launchSettings.json**

这里配置了我们调试的方式，端口等，这里与普通的Web项目多了"inspectUri"属性

inspectUri 属性具有以下作用：

使 IDE 能够检测到该应用为 Blazor WebAssembly 应用。

指示脚本调试基础结构通过 Blazor 的调试代理连接到浏览器。

已启动的浏览器 (browserInspectUri) 上 WebSocket 协议 (wsProtocol)、主机 (url.hostname)、端口 (url.port) 和检查器 URI 的占位符值由框架提供。

**Controllers**

控制器放在这里，站点的路由表是通过遍历项目中带有ApiControllerAttribute（基类ControllerAttribute）的类，然后寻找里面的方法实现，他和Blazor的路由表创建方法上有点相似。

**Pages**

站点的页面存放位置，我们这个项目并不涉及，今天不做介绍，我们欢迎有同学来Reactor给有兴趣的同学分享这方面知识

**appsettings.json**

站点的配置文件，我们的项目就用到了数据库链接字符串配置

**Program.cs**

应用的Main函数在这里，这里完成了Host的启动

**Startup.cs**

ConfigureServices运行时将调用此方法。使用此方法将服务添加到容器。

Configure运行时将调用此方法。使用此方法来配置HTTP请求管道。

app.UseBlazorFrameworkFiles();配置应用程序以从根路径"/"为 Blazor Web 组装框架文件提供服务。

endpoints.MapFallbackToFile("index.html");默认路由地址是"index.html"

* EF Code

接着我们用EF Code操作数据库

Entity Framework (EF) Core 是轻量化、可扩展、开源和跨平台版的数据访问技术。（PPT）

EF Core可用作对象关系映射程序 (O/RM)，能让我们用对象来处理数据库，使用Linq进行查询，这样我们就可以不用编写大量SQL代码了。

他支持采用Code Firs或者Database First两种模式。（PPT）

Code Firs用代码编写对象关系，然后通过它创建数据库

Database First可以提供现有数据库，反向生成对象映射

鉴于个人比较喜欢先确定数据库，所以这里使用Database First。

* Database

所以我们先来看看我之前创建好的数据库。

用VS的SQL Server对象资源管理器查看我们的数据数据库

我这里用的是(localdb)作为数据库，这个对于本机开发来说非常好用，他拥有SQLServer数据库的基本功能，几乎不占用我们宝贵的硬件资源。打开我们的ToDo数据，打开Task表

接下来我们创建一个ToDo.Entity项目用于存储ORM映射以及EF的Context。

（使用EF Power Tool工具创建代码）

（appsettings.json中添加链接字符串）

（ConfigureServices中添加服务注册）

（配置日志，可以将sql输出到控制台）

* 功能实现（实操）

实现数据交互流程准备工作已经完成，我们现在开始完成我们的ToDo应用

1. 列出当天的所有代办工作
   1. Web接口（Load）
   2. Http请求并显示
2. 添加代办
   1. Web接口（Save）
   2. 回车事件，数据是否为空判断
3. 编辑抽屉
   1. Web接口（Get、Save）
   2. 模型的验证
4. 修改重要程度
   1. Web接口（SetStar）
5. 修改完成与否
   1. Web接口（SetFinish）
6. 删除代办
   1. Web接口（Del）
7. 查询代办
   1. Web接口（Search）
   2. 载入默认检索，条件检索，分页，（排序（待定））
8. 查看详细抽屉
   1. Web接口（Get，Save）
   2. 任务服务（TaskService），注册，及使用
   3. 抽屉服务Dialog模式（先用回调模式，然后说明这种模式可以解决回调海的问题）