Building Product Management Application Using Real-Time Communication with SignalR

Introduction

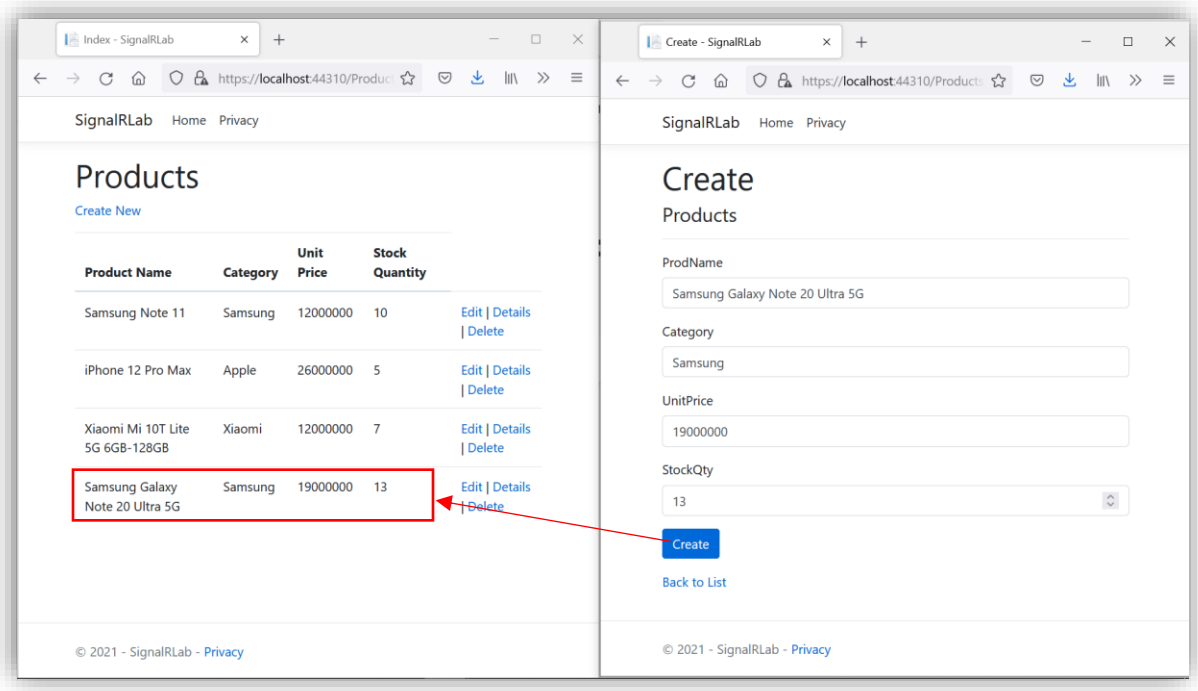
Imagine you're an employee of a product retailer named Product Store. Your manager has asked you to develop a web application for product management Product (ProductID, ProductName, Category, Quantity, Price). The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

This lab explores creating an application using Real-Time Communication with SignalR, ASP.NET Core, and C#. An **SQL Server Database** will be created to persist the product data that will be used for reading and managing product data by **Entity Framework Core.**

Lab Objectives

In this lab, you will:

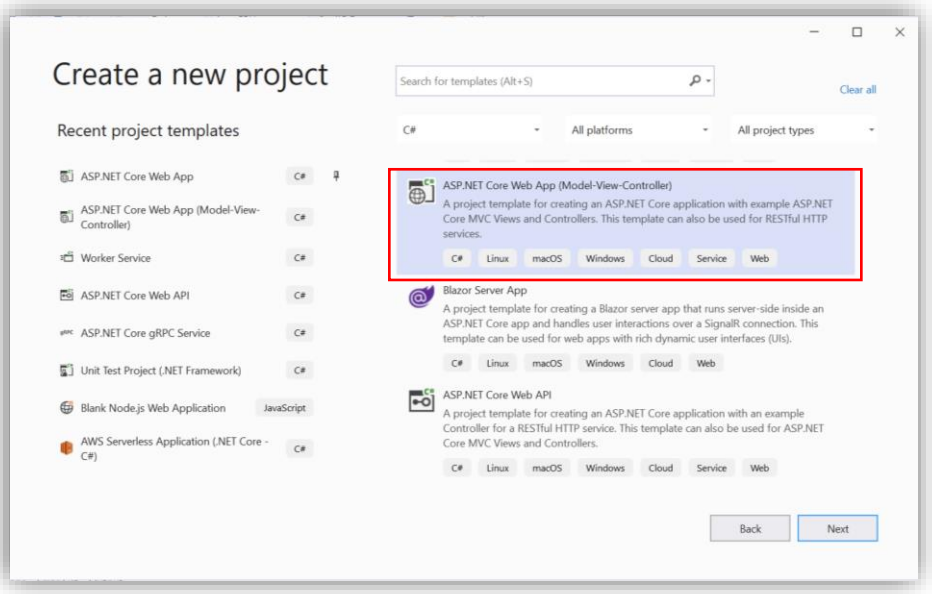
- Use the Visual Studio.NET to create ASP.NET Core Web Application Project.
- Develop application using MVC Pattern.
- Use Entity Framework to Create a SQL Server database named SignalRLab that has a Product table.
- Develop Entity Classes a and DBContext class to perform CRUD actions using Entity Framework Core.
- Apply JavaScript library for Real-Time communication application.
- Run the project and test the application actions.


Index - SignalRLab
SignalRLab Home Privacy
Products
Create New
Product Name Category Unit Price Stock Quantity
Samsung Note 11 Samsung 12000000 10 Edit | Details | Delete
iPhone 12 Pro Max Apple 26000000 5 Edit | Details | Delete
Xiaomi Mi 10T Lite 5G 6GB-128GB Xiaomi 12000000 7 Edit | Details | Delete
Samsung Galaxy Note 20 Ultra 5G Samsung 19000000 13 Edit | Details | Delete
© 2021 - SignalRLab - Privacy
Create - SignalRLab
SignalRLab Home Privacy
Create
Products
ProdName
Samsung Galaxy Note 20 Ultra 5G
Category
Samsung
UnitPrice
19000000
StockQty
13
Create
Back to List
© 2021 - SignalRLab - Privacy

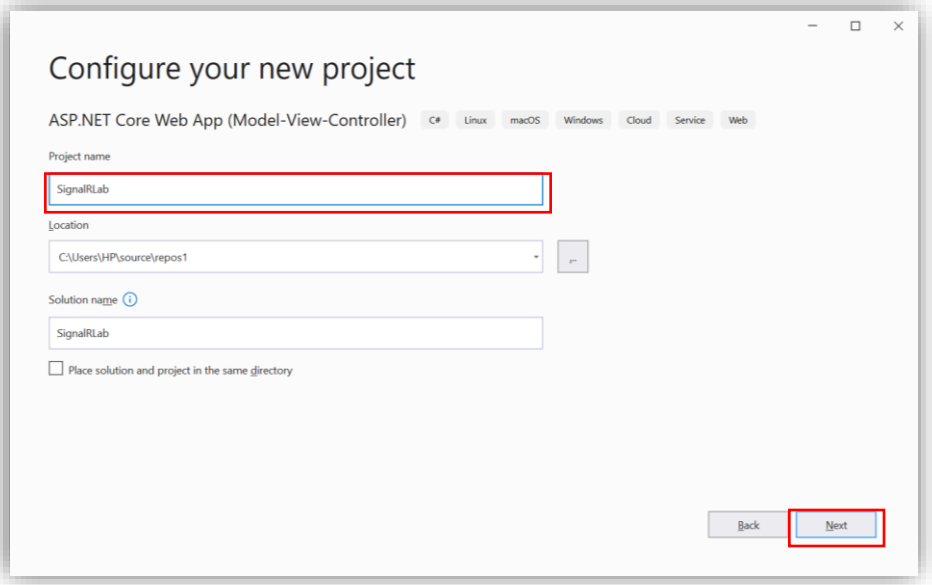
Activity 01: Create Project

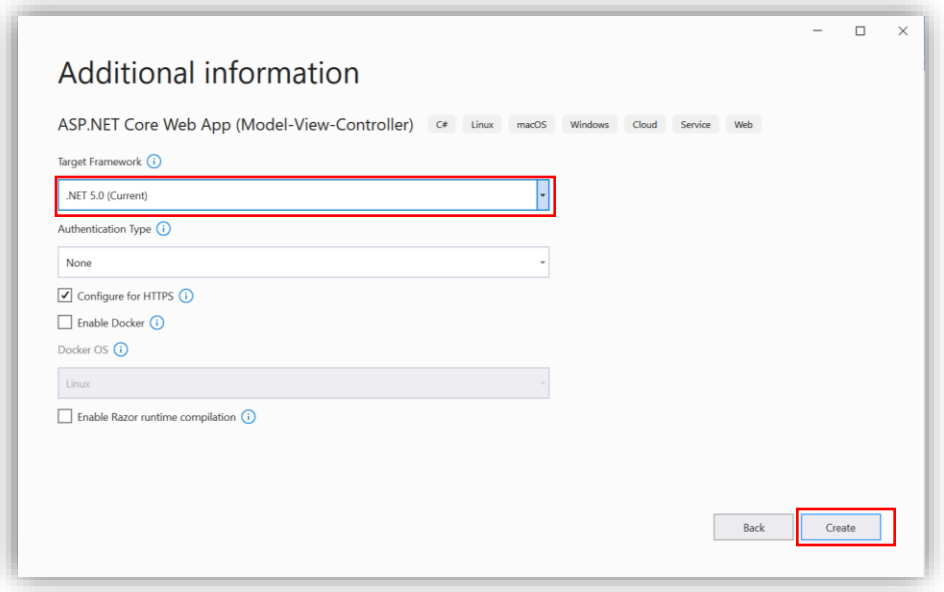
Create a Solution named SignalRLab. Create ASP.NET Core Web Application (Model-View-Controller). Open the Visual Studio .NET application and performs steps as follows:

Create ASP.NET Core Web App (Model-View-Controller)

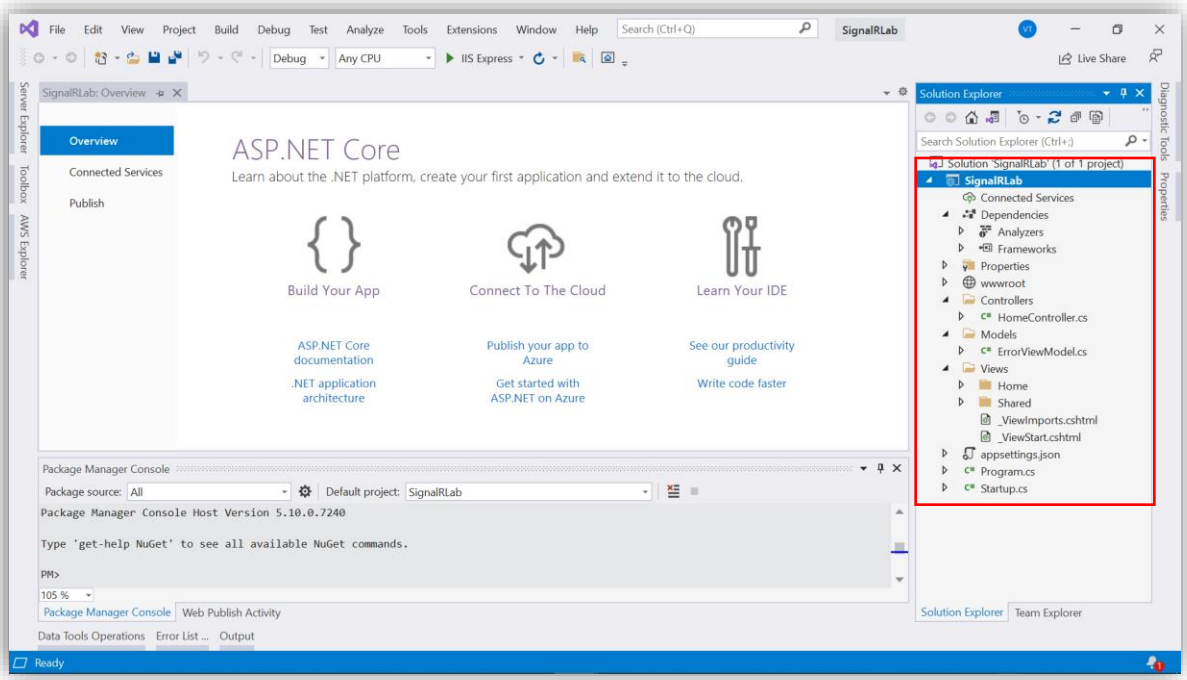


Specify Name and Location of the Project “SignalRLab”



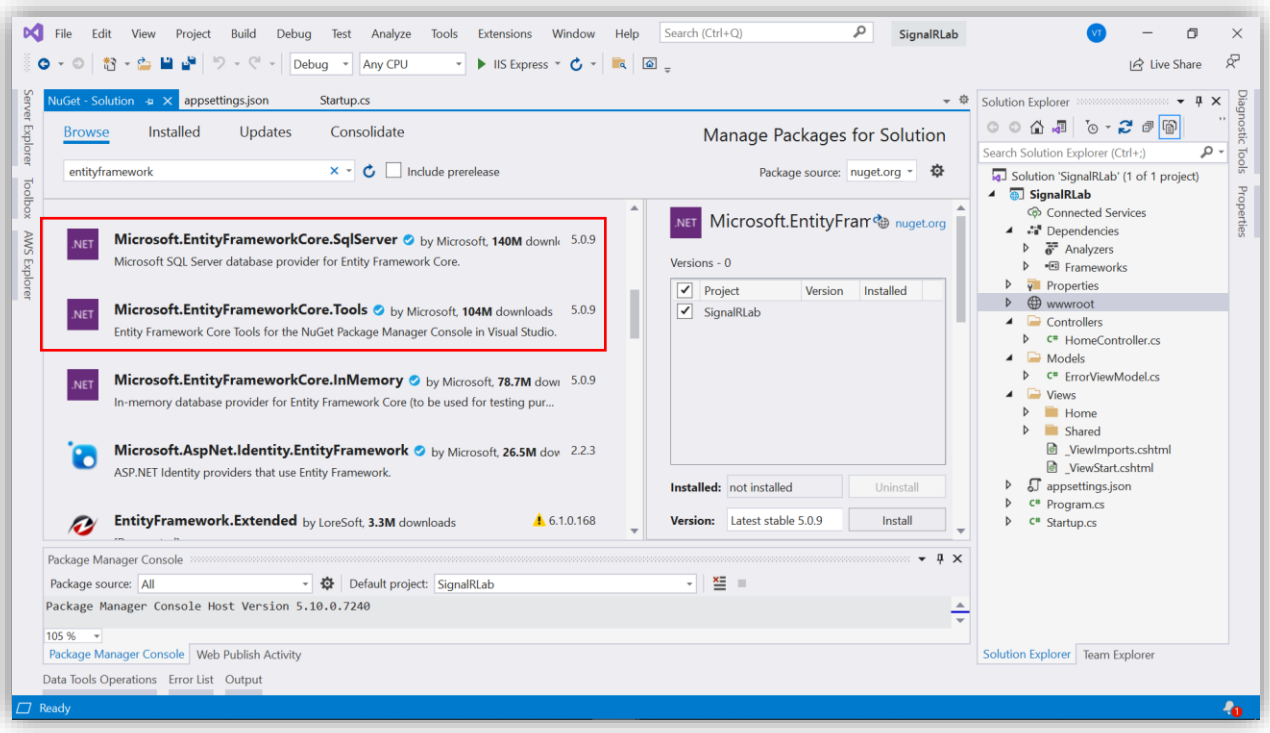


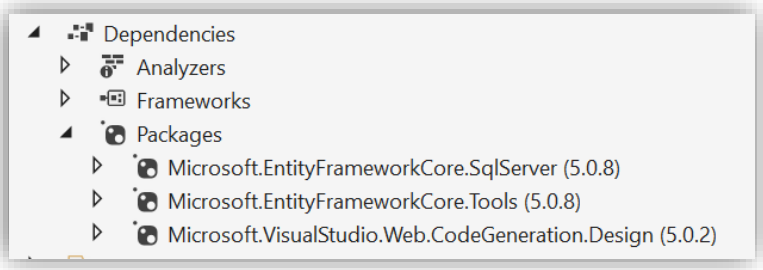
The structure of ASP.NET Core Web Application Project “SignalRLab”.



Activity 02: Work with Entity Framework

Step 01. Install the following packages from NuGet:





Step 02. Add Connection string (appsettings.json file)

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
```

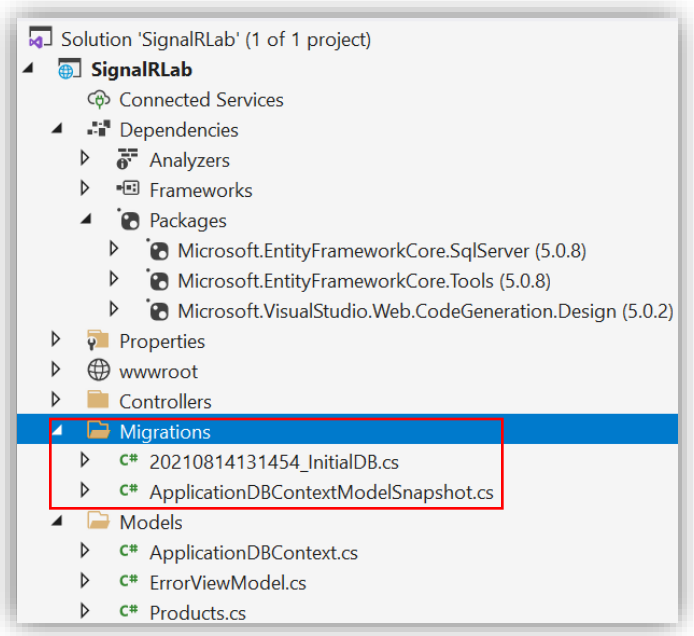
```
    "DefaultConnection": "Persist Security Info=False;User
ID=sa;Password=1234567890;Initial Catalog=SignalRLabDB;Data Source=.;Connection
Timeout=100000"
  }
}
```

Step 03. Add “Products.cs” entity and “ApplicationDBContext.cs” classes

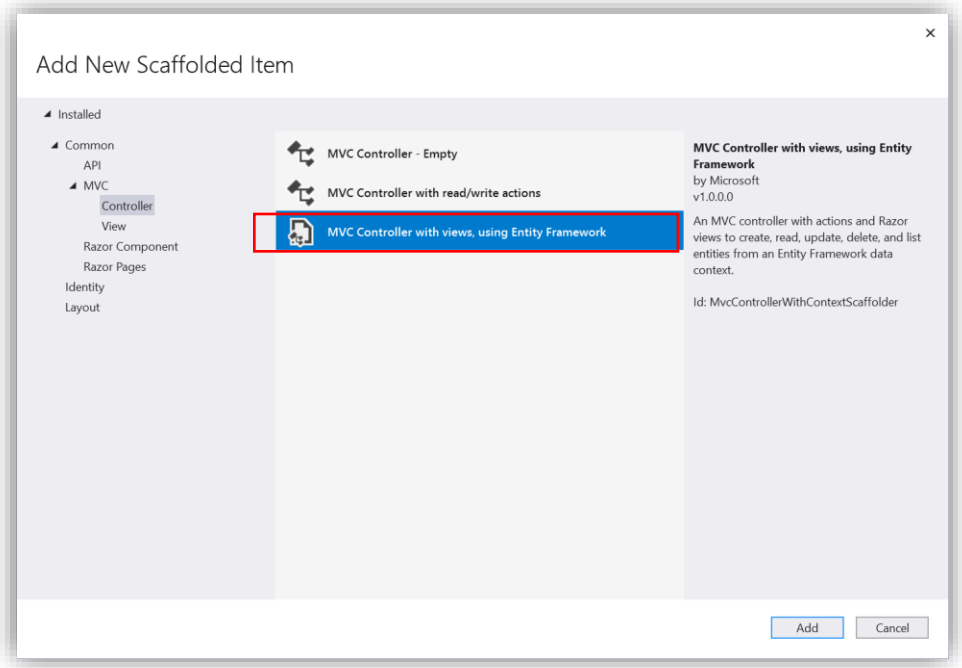
```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace SignalRLab.Models
{
    18 references
    public class Products
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        8 references
        public int ProdId { get; set; }
        10 references
        public string ProdName { get; set; }
        10 references
        public string Category { get; set; }
        10 references
        public decimal UnitPrice { get; set; }
        10 references
        public int StockQty { get; set; }
    }
}
```

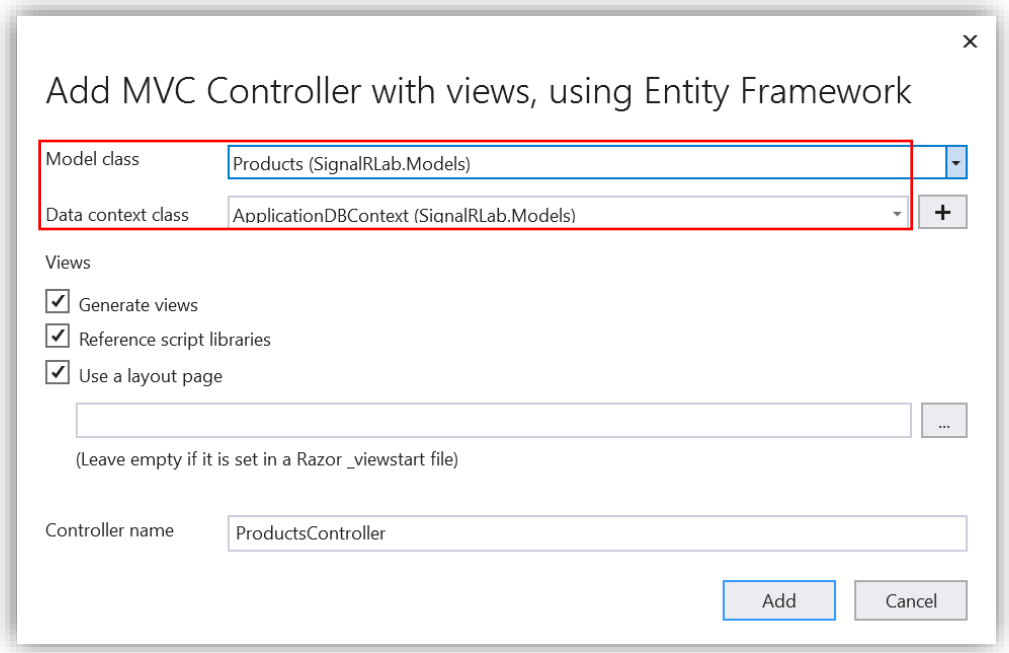
```
using Microsoft.EntityFrameworkCore;

namespace SignalRLab.Models
{
    7 references
    public class ApplicationDBContext : DbContext
    {
        0 references
        public ApplicationDBContext(DbContextOptions<ApplicationDBContext> options) :
            base(options)
        {
        }
        8 references
        public virtual DbSet<Products> Products { get; set; }
    }
}
```

Step 04. Add-Migration and Update-Database



Step 05. Add ProductsController with Scraffolding




Add MVC Controller with views, using Entity Framework
Model class
Products (SignalRLab.Models)
Data context class
ApplicationDBContext (SignalRLab.Models)
Views
Generate views
Reference script libraries
Use a layout page
(Leave empty if it is set in a Razor _viewstart file)
Controller name
ProductsController
Add
Cancel

Activity 03: Create SignalR Hub and configure SignalR

Step 01. Create SignalR Hubs in the SignalrServer.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Threading.Tasks;
5   using Microsoft.AspNetCore.SignalR;
6
7   namespace SignalRLab
8   {
        3 references
9       public class SignalrServer:Hub
10      {
11      }
12  }
```

Step 02. Add SignalR to Startup.cs

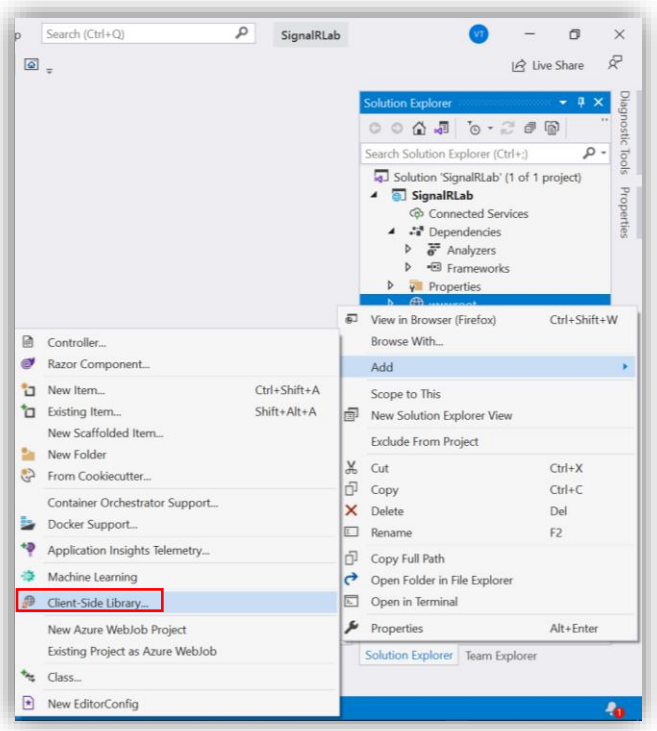
```
1   using Microsoft.AspNetCore.Builder;
2   using Microsoft.AspNetCore.Hosting;
3   using Microsoft.Extensions.Configuration;
4   using Microsoft.Extensions.DependencyInjection;
5   using Microsoft.Extensions.Hosting;
6   using SignalRLab.Models;
7   using Microsoft.EntityFrameworkCore;
8
9
10  namespace SignalRLab
11  {
        2 references
12      public class Startup
13      {
            0 references
14          public Startup(IConfiguration configuration)
15          {
16              Configuration = configuration;
17          }
18
            2 references
19          public IConfiguration Configuration { get; }
20
21          // This method gets called by the runtime. Use this method to add services t
```

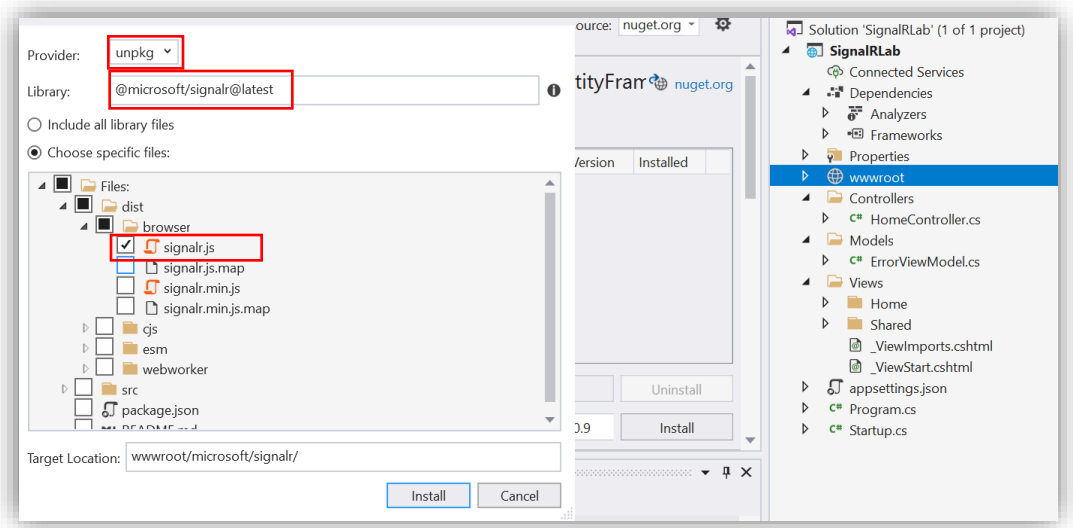
```
22          public void ConfigureServices(IServiceCollection services)
23          {
24              services.AddControllersWithViews();
25              services.AddSignalR();
26              services.AddControllers().AddJsonOptions(options =>
27              {
28                  options.JsonSerializerOptions.PropertyNamingPolicy = null;
29              });
30
31              services.AddDbContext<ApplicationDBContext>(options =>
32                  options.UseSqlServer(Configuration.GetConnectionString("DefaultConnec
33          }
34
35          // This method gets called by the runtime. Use this method to configure the H
            0 references
36          public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
37          {
38              if (env.IsDevelopment())
39              {
40                  app.UseDeveloperExceptionPage();
41              }
42              else
43              {
44                  app.UseExceptionHandler("/Home/Error");
45                  // The default HSTS value is 30 days. You may want to change this for
46                  app.UseHsts();

47              }
48              app.UseHttpsRedirection();
49              app.UseStaticFiles();
50
51              app.UseRouting();
52
53              app.UseAuthorization();
54
55              app.UseEndpoints(endpoints =>
56              {
57                  endpoints.MapControllerRoute(
58                      name: "default",
59                      pattern: "{controller=Home}/{action=Index}/{id?}");
60                  endpoints.MapHub<SignalrServer>("/signalrServer");
61              });
62          }
63      }
64  }
```

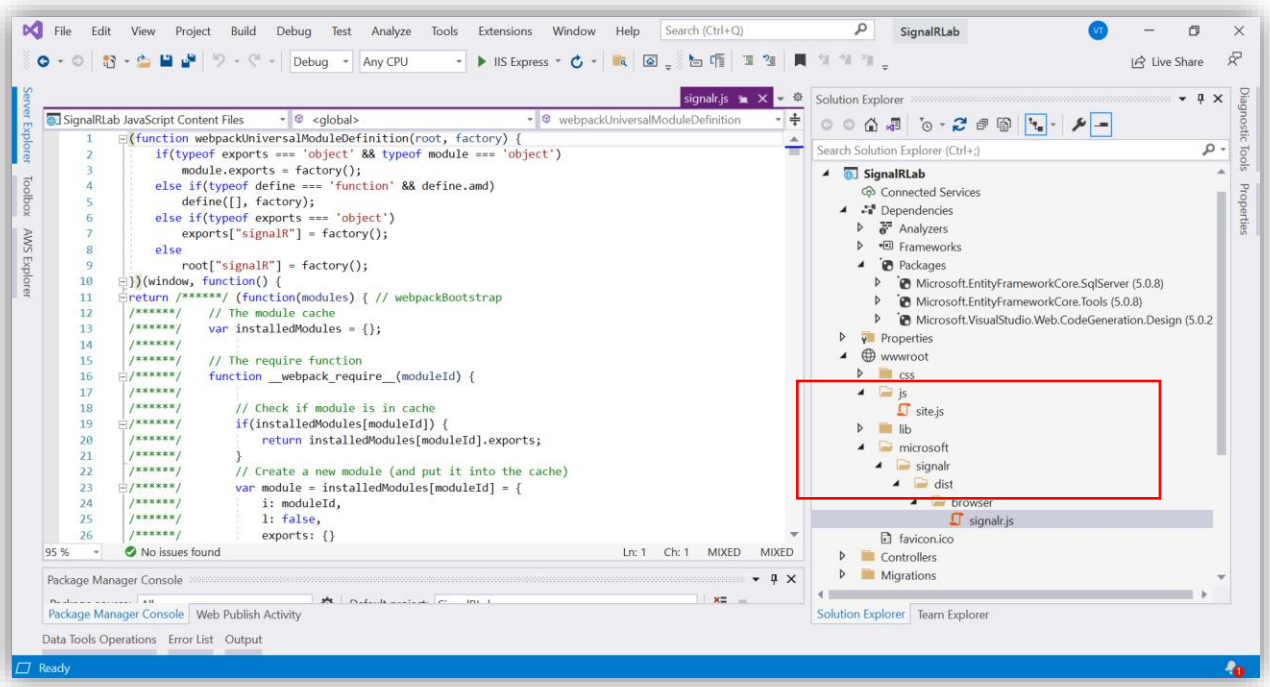
Activity 04: Build CRUD functions with SignalR

Step 01. Add Client-Side Library

- Right-click the project, and select Add > **Client-Side Library**.
- In the Add Client-Side Library dialog, for Provider select **unpkg**.
- For Library, enter **@microsoft/signalr@latest**.
- Select Choose specific files, expand the *dist/browser* folder, and select ***signalr.js* and *signalr.min.js***.
- Set Target Location to ***wwwroot/microsoft/signalr/*** , and select Install.




Provider:
unpkg
Library:
@microsoft/signalr@latest
Include all library files
Choose specific files:
Files:
dist
browser
signalr.js
signalr.js.map
signalr.min.js
signalr.min.js.map
cjs
esm
webworker
src
package.json
Target Location:
wwwroot/microsoft/signalr/
Install
Cancel
Uninstall
Install
Solution 'SignalRLab' (1 of 1 project)
SignalRLab
Connected Services
Dependencies
Analyzers
Frameworks
Properties
wwwroot
Controllers
HomeController.cs
Models
ErrorViewModel.cs
Views
Home
Shared
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs
Startup.cs


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
SignalRLab
Debug
Any CPU
IIS Express
Live Share
signalr.js
SignalRLab JavaScript Content Files
webpackUniversalModuleDefinition
Solution Explorer
Search Solution Explorer (Ctrl+;)
SignalRLab
Connected Services
Dependencies
Analyzers
Frameworks
Packages
Properties
wwwroot
css
js
site.js
lib
microsoft
signalr
dist
browser
signalr.js
favicon.ico
Controllers
Migrations
No issues found
Package Manager Console
Web Publish Activity
Data Tools Operations
Error List
Output
Ready

Step 02. Create a callback function in the script (site.js)

```
$(() => {
    LoadProdData();
    var connection = new signalR.HubConnectionBuilder().withUrl("/signalrServer").build();
    connection.start();

    connection.on("LoadProducts", function () {
        LoadProdData();
    })

    LoadProdData();

    function LoadProdData() {
        var tr = '';
        $.ajax({
            url: '/Products/GetProducts',
            method: 'GET',
            success: (result) => {
                $.each(result, (k, v) => {
                    tr += `<tr>
                        <td> ${v.ProdName} </td>
                        <td> ${v.Category} </td>
                        <td> ${v.UnitPrice} </td>
                        <td> ${v.StockQty} </td>
                        <td>
                            <a href='../Products/Edit?id=${v.ProdId}'> Edit </a> |
                            <a href='../Products/Details?id=${v.ProdId}'> Details </a> |
                            <a href='../Products/Delete?id=${v.ProdId}'> Delete </a>
                        </td>
                        </tr>`
                })

                $("#tableBody").html(tr);
            },
            error: (error) => {
                console.log(error)
            }

        });
    }
})
```

Step 03. Add the notification to CRUD actions

- The SignalR hub is the core abstraction for sending messages to clients connected to the SignalR server.
- Use a SignalR IHubContext to send notifications to clients from outside a hub.

```
private readonly IHubContext<SignalrServer> _signalRHub;
```

- Then use an instance of `IHubContext`, call client methods as if you were in the hub itself

```
await _signalRHub.Clients.All.SendAsync("LoadProducts");
```

```
1   using System.Linq;
2   using System.Threading.Tasks;
3   using Microsoft.AspNetCore.Mvc;
4   using Microsoft.AspNetCore.SignalR;
5   using Microsoft.EntityFrameworkCore;
6   using SignalRLab.Models;
7
8   namespace SignalRLab.Controllers
9   {
        1 reference
10      public class ProductsController : Controller
11      {
12          private readonly ApplicationDBContext _context;
13          private readonly IHubContext<SignalrServer> _signalRHub;
14
            0 references
15          public ProductsController(ApplicationDBContext context, IHubContext<SignalrServer> signalRHub)
16          {
17              _context = context;
18              _signalRHub = signalRHub;
19          }
20
21          // GET: Products
            3 references
22          public async Task<IActionResult> Index()...
26
27          [HttpGet]
            0 references
28          public IActionResult GetProducts()
29          {
30              var res = _context.Products.ToList();
31              return Ok(res);
32          }
33
            0 references
34          public async Task<IActionResult> Details(int? id)...
49
50          // GET: Products/Create
            0 references
51          public IActionResult Create()...
55
58          [HttpPost]
59          [ValidateAntiForgeryToken]
            0 references
60          public async Task<IActionResult> Create([Bind("ProdId,ProdName,Category,UnitPrice,StockQty")]
61                                                  Products products)
62          {
63              if (ModelState.IsValid)
64              {
65                  _context.Add(products);
66                  await _context.SaveChangesAsync();
67                  await _signalRHub.Clients.All.SendAsync("LoadProducts");
68                  return RedirectToAction(nameof(Index));
69              }
70              return View(products);
71          }
72
73          // GET: Products/Edit/5
            0 references
74          public async Task<IActionResult> Edit(int? id)...
88
```

```
91          [HttpPost]
92          [ValidateAntiForgeryToken]
            0 references
93          public async Task<IActionResult> Edit(int ProdId, [Bind("ProdId,ProdName,Category,UnitPrice,Sto
94          {
95              if (ProdId != products.ProdId)
96              {
97                  return NotFound();
98              }
99              if (ModelState.IsValid)
100             {
101                 try
102                 {
103                     _context.Update(products);
104                     await _context.SaveChangesAsync();
105                     await _signalRHub.Clients.All.SendAsync("LoadProducts");
106                 }
107                 catch (DbUpdateConcurrencyException)
108                 {
109                     if (!ProductsExists(products.ProdId))
110                     {
111                         return NotFound();
112                     }
113                     else
114                     {   throw;      }
115                 }
116                 return RedirectToAction(nameof(Index));
117             }
118             return View(products);
119         }

122         public async Task<IActionResult> Delete(int? id)...
138
139         // POST: Products/Delete/5
140         [HttpPost, ActionName("Delete")]
141         [ValidateAntiForgeryToken]
            0 references
142         public async Task<IActionResult> DeleteConfirmed(int ProdId)
143         {
144             var products = await _context.Products.FindAsync(ProdId);
145             _context.Products.Remove(products);
146             await _context.SaveChangesAsync();
147             await _signalRHub.Clients.All.SendAsync("LoadProducts");
148             return RedirectToAction(nameof(Index));
149         }
150
            1 reference
151         private bool ProductsExists(int id)
152         {
153             return _context.Products.Any(e => e.ProdId == id);
154         }
155     }
156 }
```

Step 04. Add SignalR JavaScript client to View

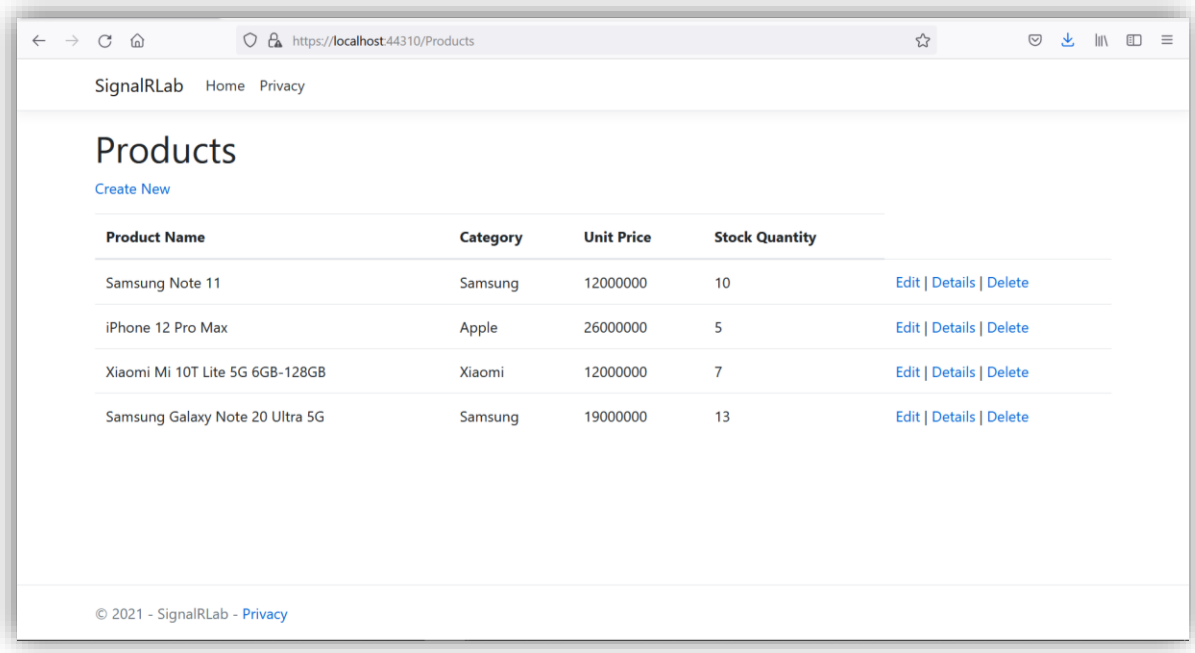
Views/Shared/_Layout.cshtml

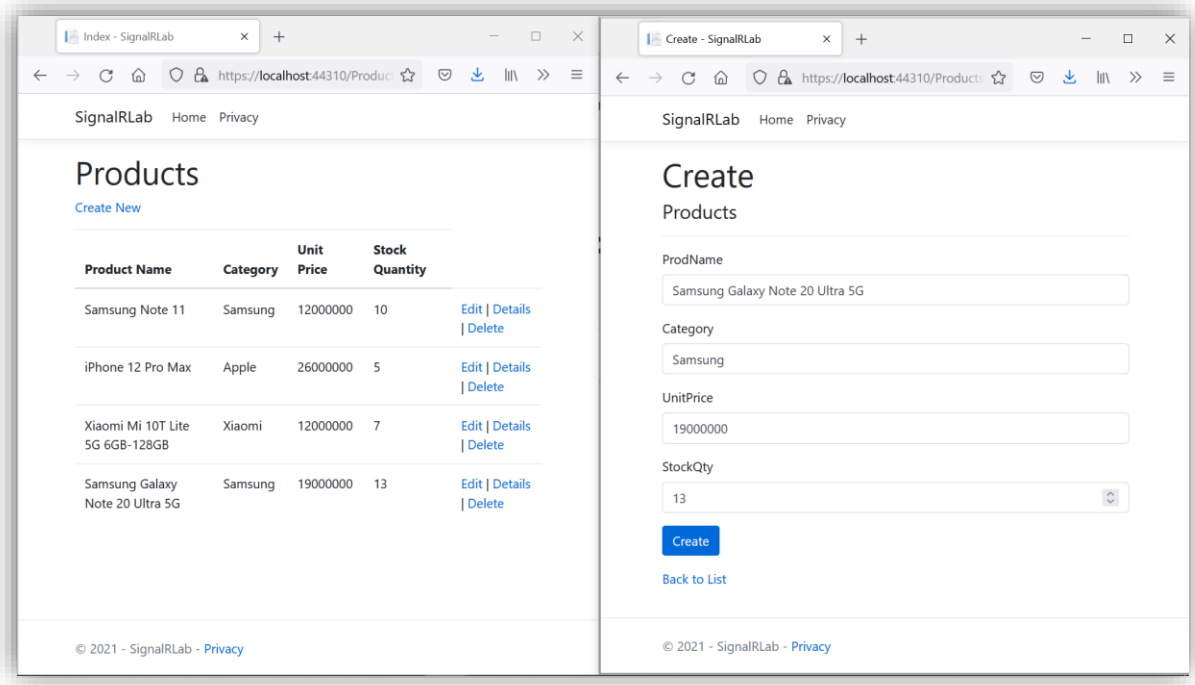
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - SignalRLab</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    <header>…</header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>

    <footer class="border-top footer text-muted">…</footer>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>

    <script src="~/microsoft/signalr/dist/browser/signalr.js"></script>
    <script src="~/js/site.js"></script>

    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

Views/Products/Index.cshtml

```
@model IEnumerable<SignalRLab.Models.Products>

@{
    ViewData["Title"] = "Index";
}

<h1>Products</h1>

<p>
    <a asp-action="Create">Create New</a>

    <table class="table">
        <thead>
            <tr>
                <th>Product Name</th>
                <th>Category</th>
                <th>Unit Price</th>
                <th>Stock Quantity</th>
            </tr>
        </thead>
        <tbody id="tableBody"></tbody>
    </table>
</p>
```

Activity 05: Build and run Project. Test all CRUD actions





Client 1 Client 2