MONASH University

FIT5032 Working with Signal R

Chief Examiner: ABM Russel

POST CLASS ACTIVITIES

TOPICS

- Real-time features of a web application
- Usage of SignalR at a basic level

It is not compulsory to complete this tutorial. It functions as a supplementary material to showcase how to use SignalR to demonstrate a real-time function.

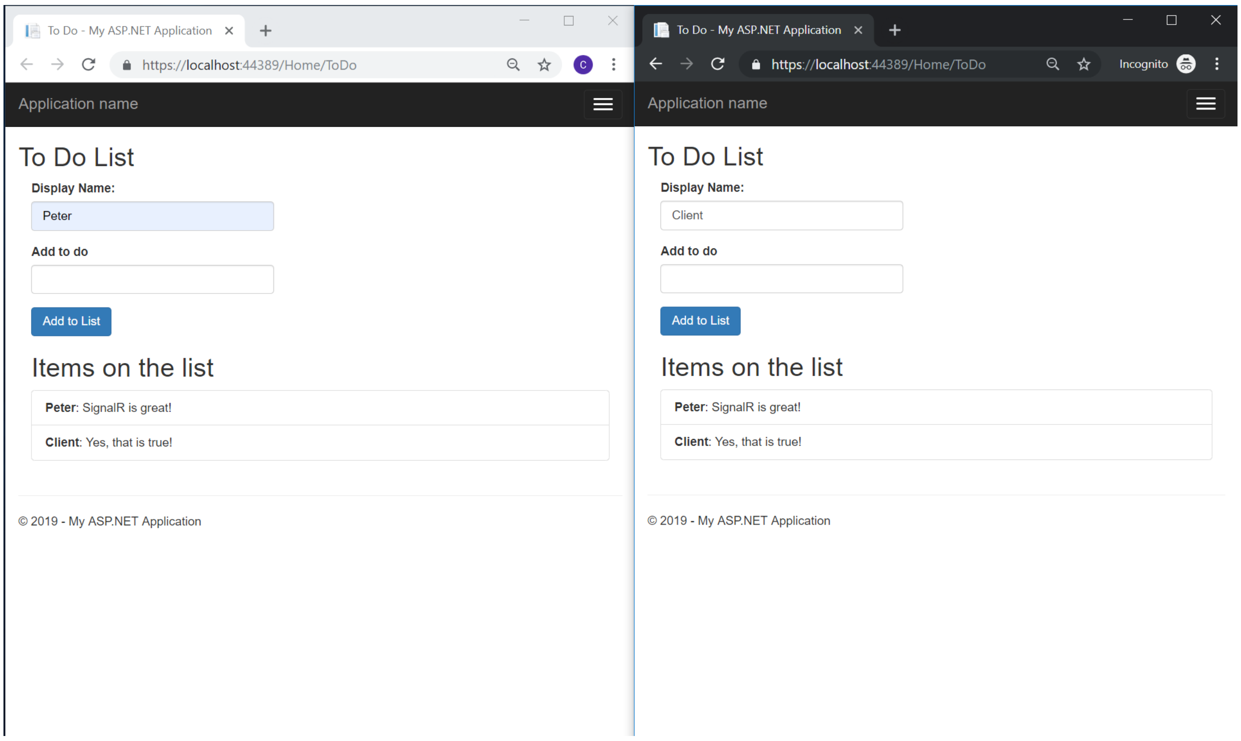
Real-time features are very important for modern websites. This can be accomplished using WebSockets. The most commonly used library these days is Socket.io. However, in the .NET ecosystem, it can be done using SignalR. This supplementary material uses SignalR to accomplish the real-time features, which allows you to deliver information to your user as it happens.

Here are some examples of use cases that require real-time functionalities

- Live chat features
- Updated location of a user (For example when you are using Uber, the location of the driver constantly gets updated)
- Stock market prices (You do not want the user to constantly hit refresh on the page). In this case, whenever the values in the server are updated, it sends an update to each of the clients

By using SignalR it is possible to achieve a Pub/Sub architecture.

Upon the completion of this tutorial, you will have a basic understanding of how SignalR works. In the end product, you should be able to see how SignalR can perform real-time features.


To Do - My ASP.NET Application
https://localhost:44389/Home/ToDo
Application name
Incognito

To Do List

Display Name: Peter

Add to do

Add to List

Items on the list

Peter: SignalR is great!

Client: Yes, that is true!

© 2019 - My ASP.NET Application

To Do List

Display Name: Client

Add to do

Add to List

Items on the list

Peter: SignalR is great!

Client: Yes, that is true!

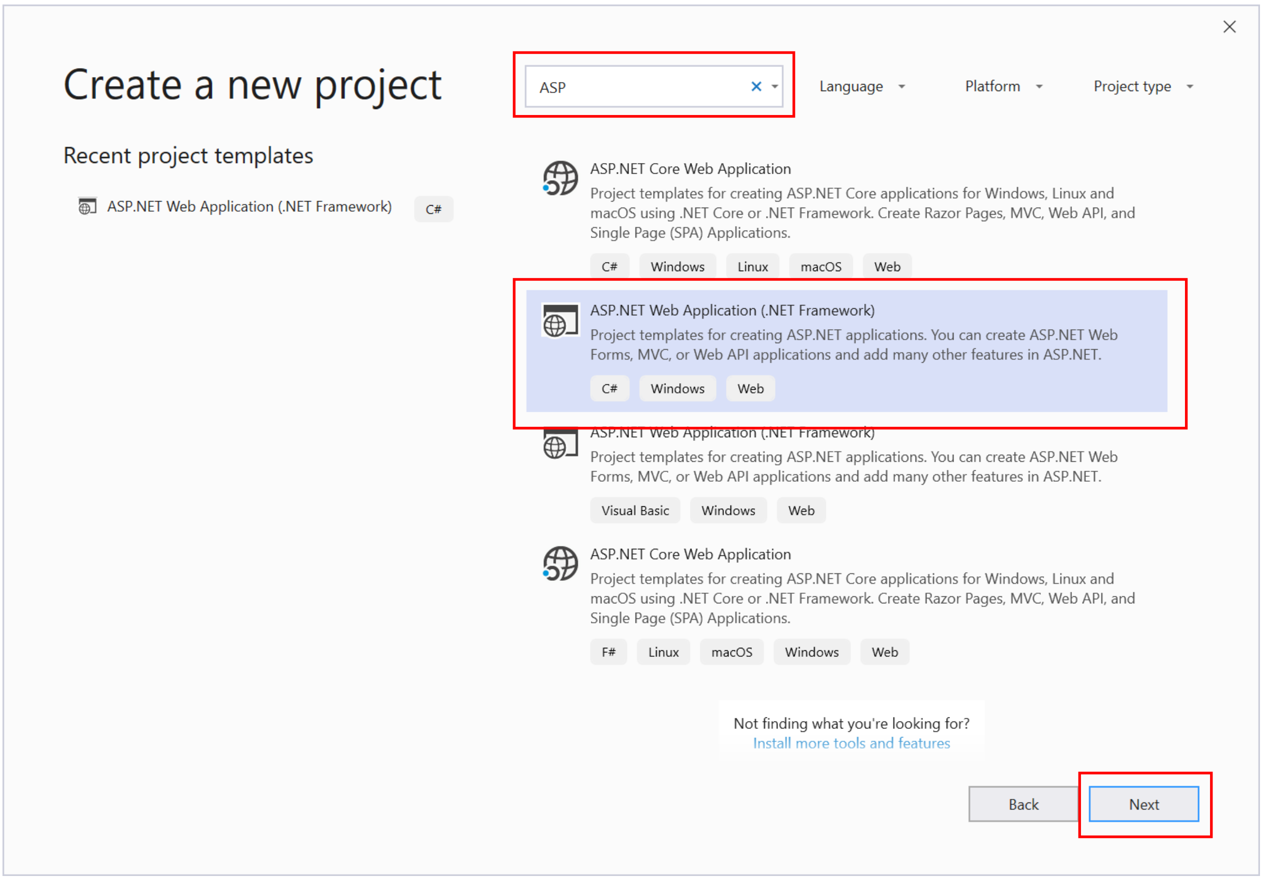
© 2019 - My ASP.NET Application

You will notice that in this screenshot, I have opened two browsers. Both of these are clients. Once the first client creates an item on the to-do list, it will appear on the second client as well. This happens at real-time.

MONASH University

STUDIO ACTIVITIES

Step 1


Create a new project

ASP

Language Platform Project type

Recent project templates

ASP.NET Web Application (.NET Framework) C#

ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.
C# Windows Linux macOS Web

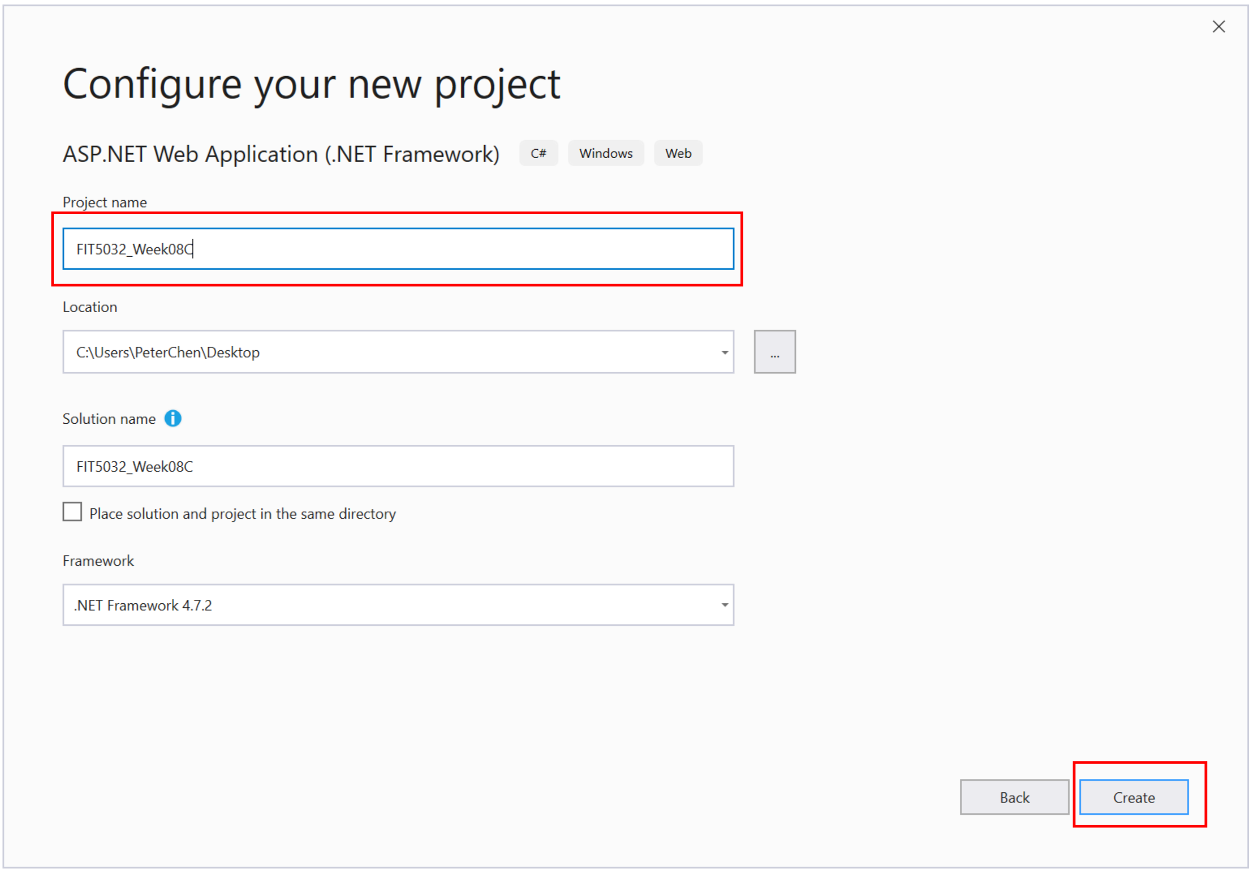
ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# Windows Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Visual Basic Windows Web

ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create Razor Pages, MVC, Web API, and Single Page (SPA) Applications.
F# Linux macOS Windows Web

Not finding what you're looking for?
Install more tools and features

Back Next


Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Web

Project name
FIT5032_Week08C

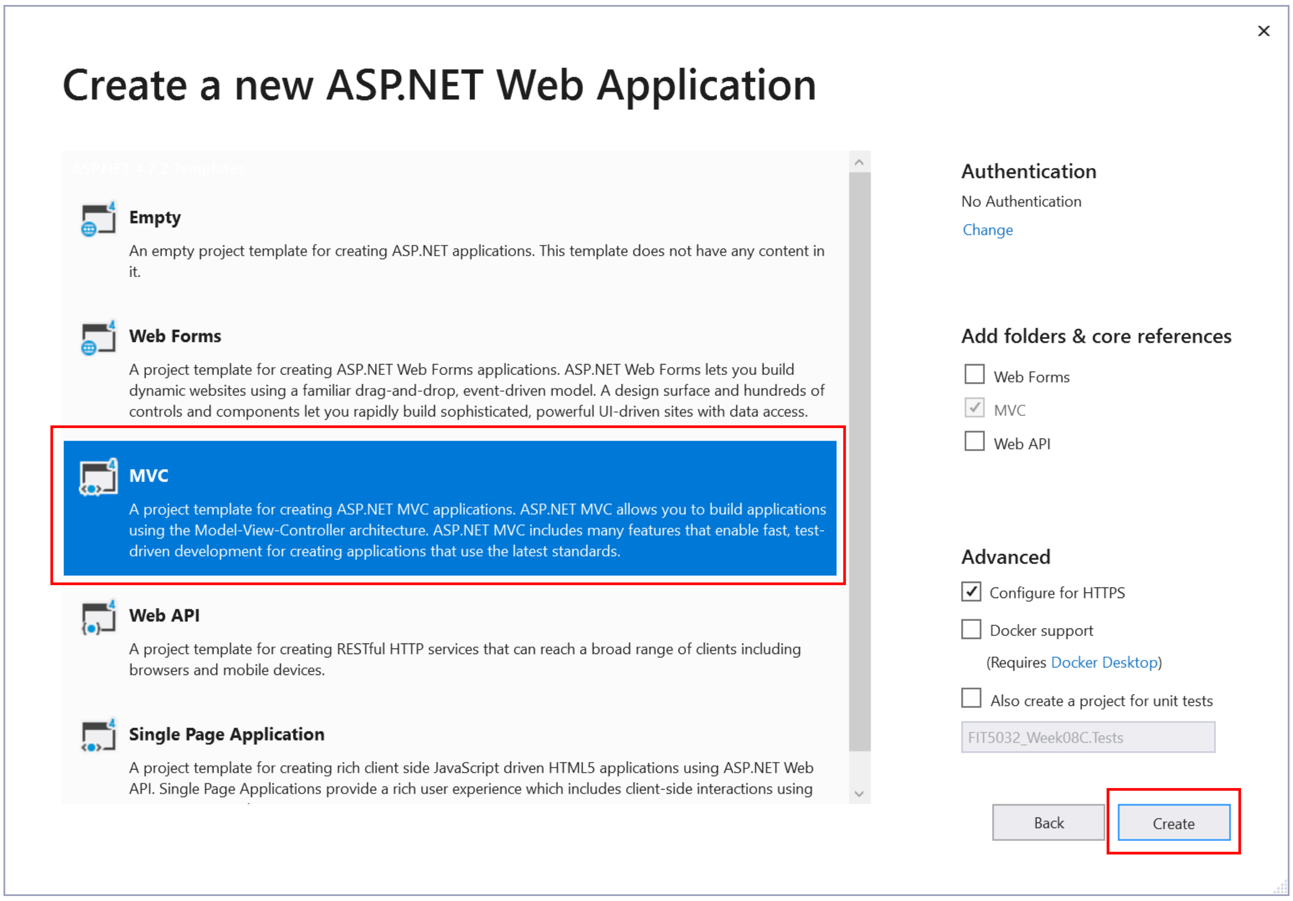
Location
C:\Users\PeterChen\Desktop

Solution name
FIT5032_Week08C

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Back Create



Create a new ASP.NET Web Application

Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application

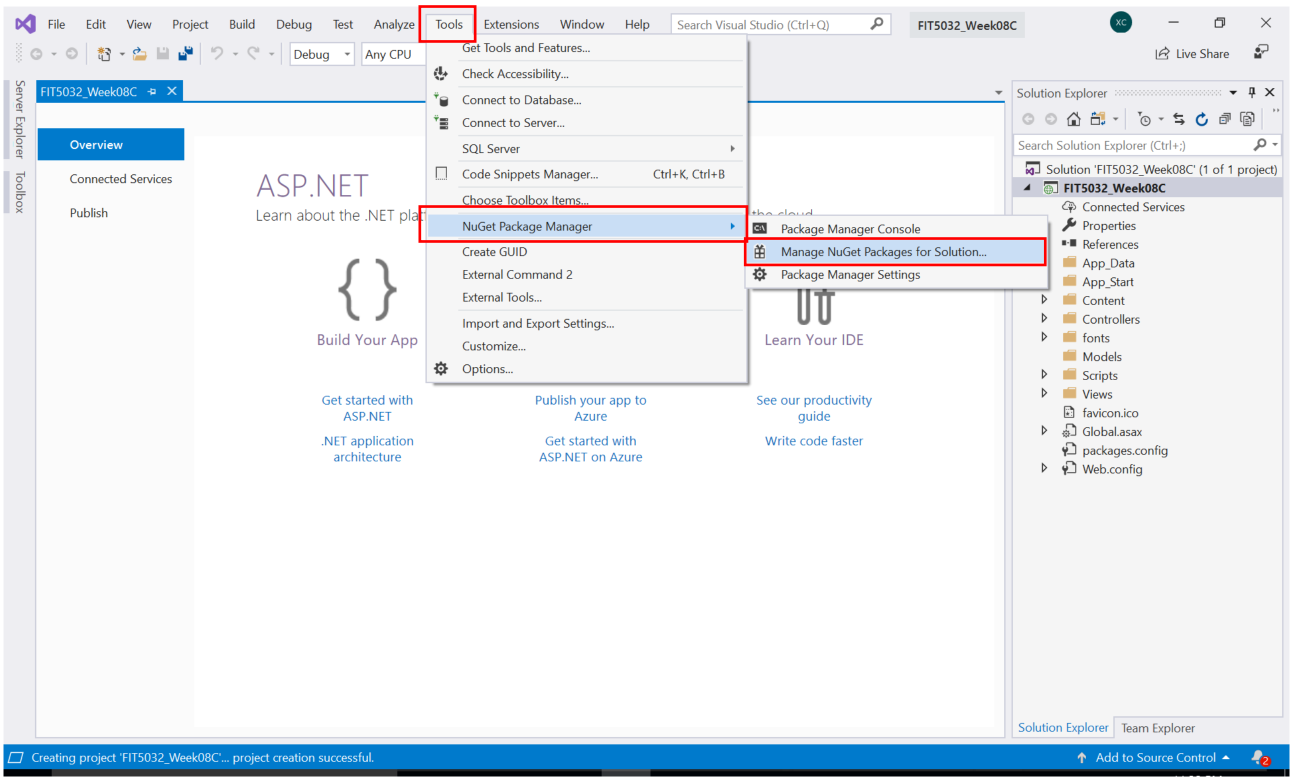
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using

Authentication
No Authentication
Change

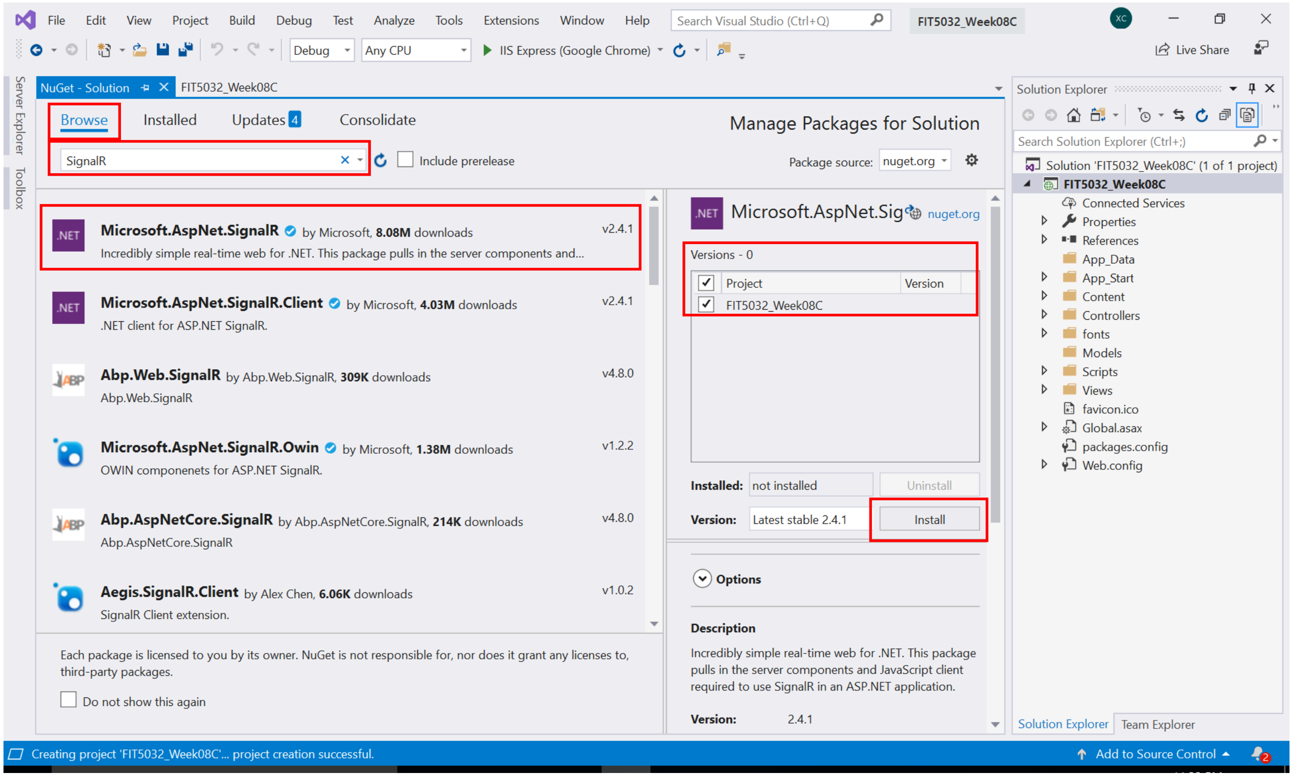
Add folders & core references
☐ Web Forms
☑ MVC
☐ Web API

Advanced
☑ Configure for HTTPS
☐ Docker support
(Requires Docker Desktop)
☐ Also create a project for unit tests
FIT5032_Week08C.Tests

Step 2



Step 3



Microsoft.AspNet.SignalR by Microsoft, **8.08M** downloads v2.4.1
Incredibly simple real-time web for .NET. This package pulls in the server components and...

Microsoft.AspNet.SignalR.Client by Microsoft, **4.03M** downloads v2.4.1
.NET client for ASP.NET SignalR.

Abp.Web.SignalR by Abp.Web.SignalR, **309K** downloads v4.8.0
Abp.Web.SignalR

Microsoft.AspNet.SignalR.Owin by Microsoft, **1.38M** downloads v1.2.2
OWIN componenets for ASP.NET SignalR.

Abp.AspNetCore.SignalR by Abp.AspNetCore.SignalR, **214K** downloads v4.8.0
Abp.AspNetCore.SignalR

Aegis.SignalR.Client by Alex Chen, **6.06K** downloads v1.0.2
SignalR Client extension.

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Do not show this again

Microsoft.AspNet.Sig nuget.org

Versions - 0

Project	Version
✓ FIT5032_Week08C	

Installed: not installed Uninstall

Version: Latest stable 2.4.1 Install

Options

Description

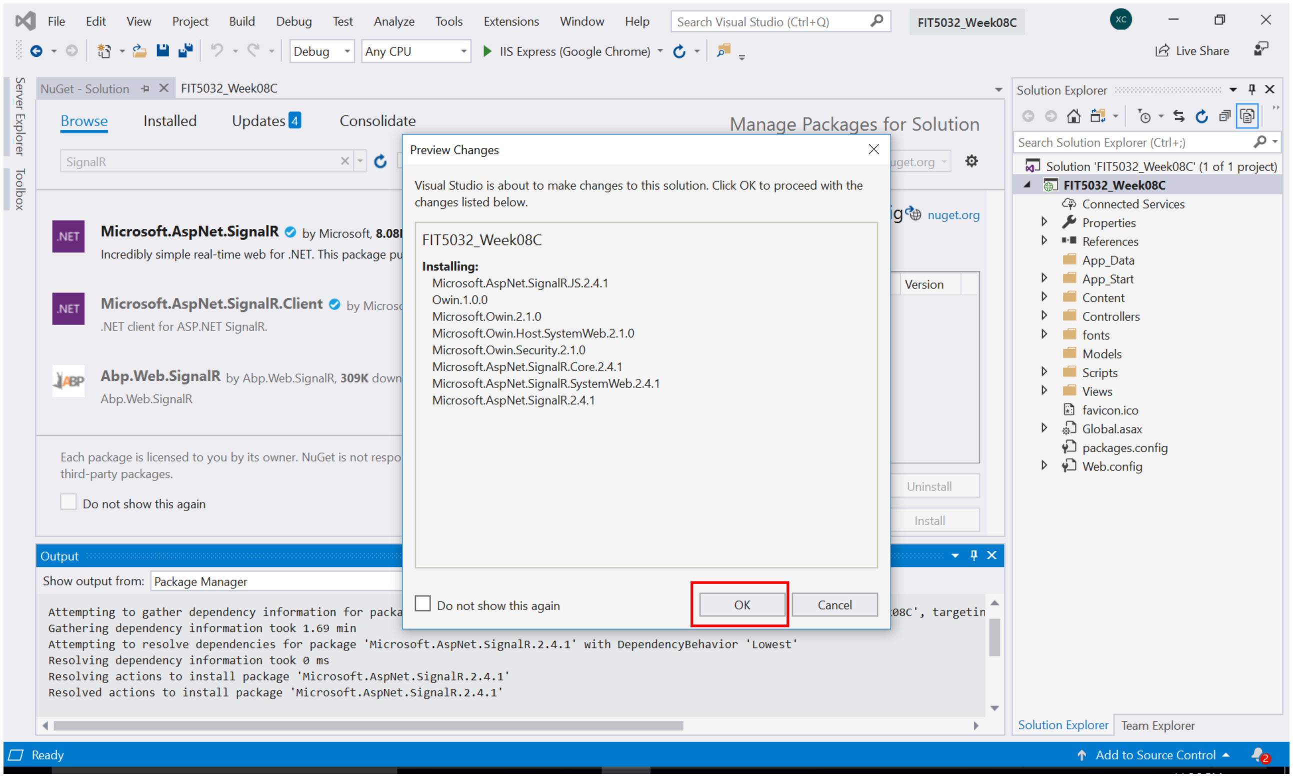
Incredibly simple real-time web for .NET. This package pulls in the server components and JavaScript client required to use SignalR in an ASP.NET application.

Version: 2.4.1

Creating project 'FIT5032_Week08C'... project creation successful.

Add to Source Control

Step 4



Preview Changes

Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.

FIT5032_Week08C

Installing:

Microsoft.AspNet.SignalR.JS.2.4.1
Owin.1.0.0
Microsoft.Owin.2.1.0
Microsoft.Owin.Host.SystemWeb.2.1.0
Microsoft.Owin.Security.2.1.0
Microsoft.AspNet.SignalR.Core.2.4.1
Microsoft.AspNet.SignalR.SystemWeb.2.4.1
Microsoft.AspNet.SignalR.2.4.1

Do not show this again OK Cancel

Output

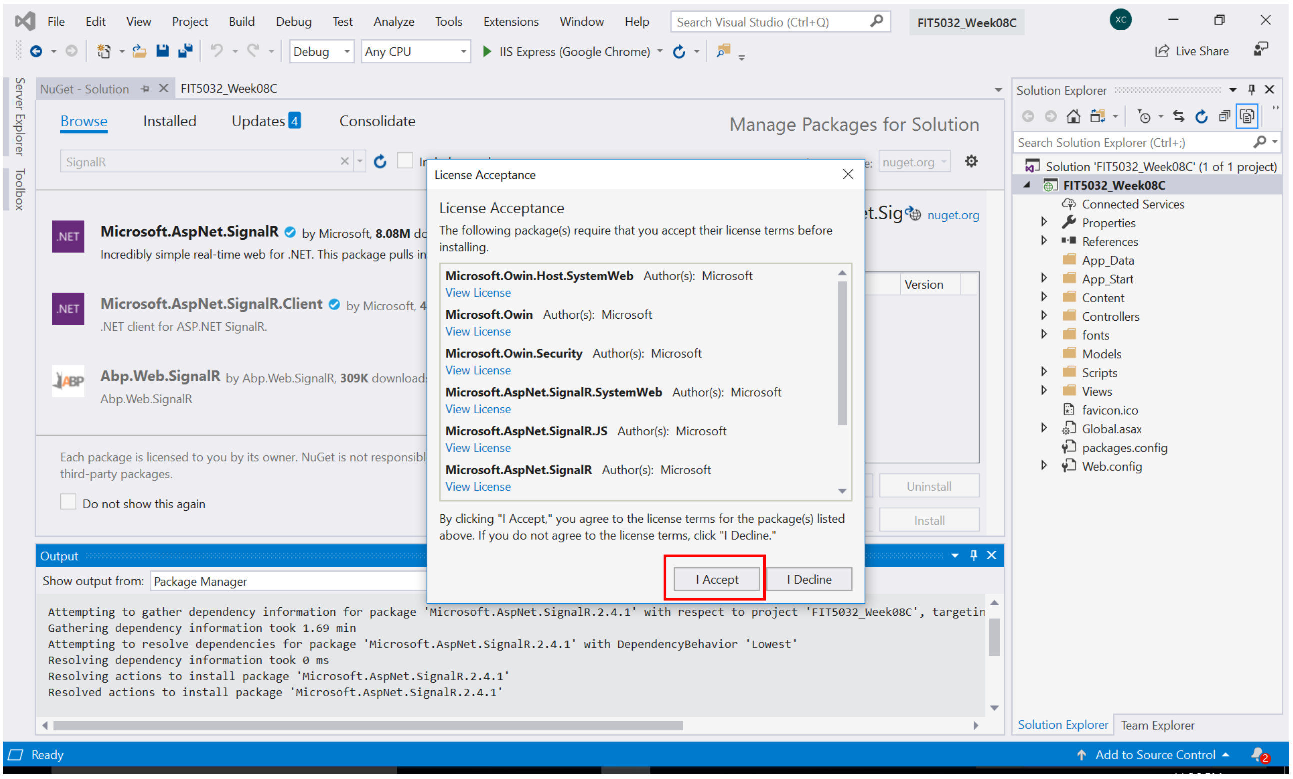
Show output from: Package Manager

```
Attempting to gather dependency information for packa                                          08C', targetin
Gathering dependency information took 1.69 min
Attempting to resolve dependencies for package 'Microsoft.AspNet.SignalR.2.4.1' with DependencyBehavior 'Lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'Microsoft.AspNet.SignalR.2.4.1'
Resolved actions to install package 'Microsoft.AspNet.SignalR.2.4.1'
```

Ready

Add to Source Control

Step 5



License Acceptance

The following package(s) require that you accept their license terms before installing.

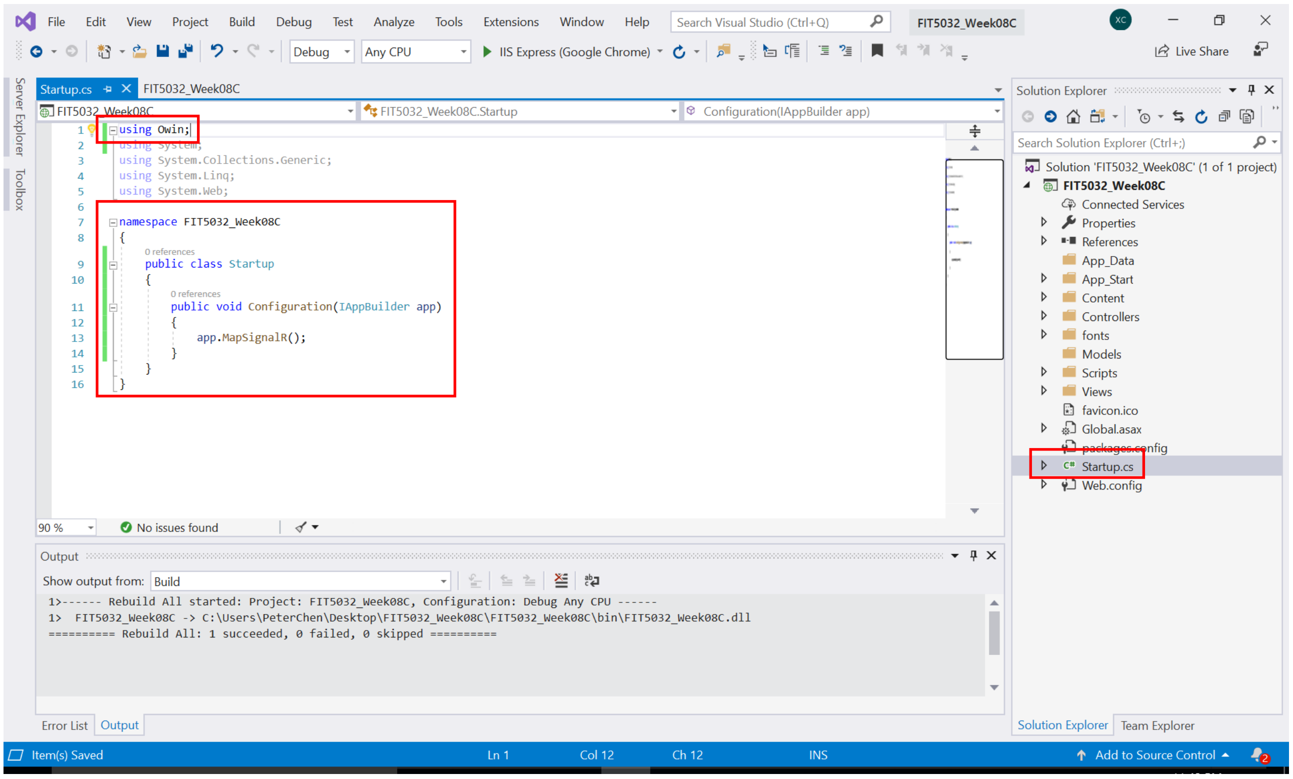
Microsoft.Owin.Host.SystemWeb Author(s): Microsoft
View License
Microsoft.Owin Author(s): Microsoft
View License
Microsoft.Owin.Security Author(s): Microsoft
View License
Microsoft.AspNet.SignalR.SystemWeb Author(s): Microsoft
View License
Microsoft.AspNet.SignalR.JS Author(s): Microsoft
View License
Microsoft.AspNet.SignalR Author(s): Microsoft
View License

By clicking "I Accept," you agree to the license terms for the package(s) listed above. If you do not agree to the license terms, click "I Decline."

```
Attempting to gather dependency information for package 'Microsoft.AspNet.SignalR.2.4.1' with respect to project 'FIT5032_Week08C', targetin
Gathering dependency information took 1.69 min
Attempting to resolve dependencies for package 'Microsoft.AspNet.SignalR.2.4.1' with DependencyBehavior 'Lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'Microsoft.AspNet.SignalR.2.4.1'
Resolved actions to install package 'Microsoft.AspNet.SignalR.2.4.1'
```

Step 6

SignalR needs to be started so there must be a **Startup** configuration created. Create the Startup class using the following code.



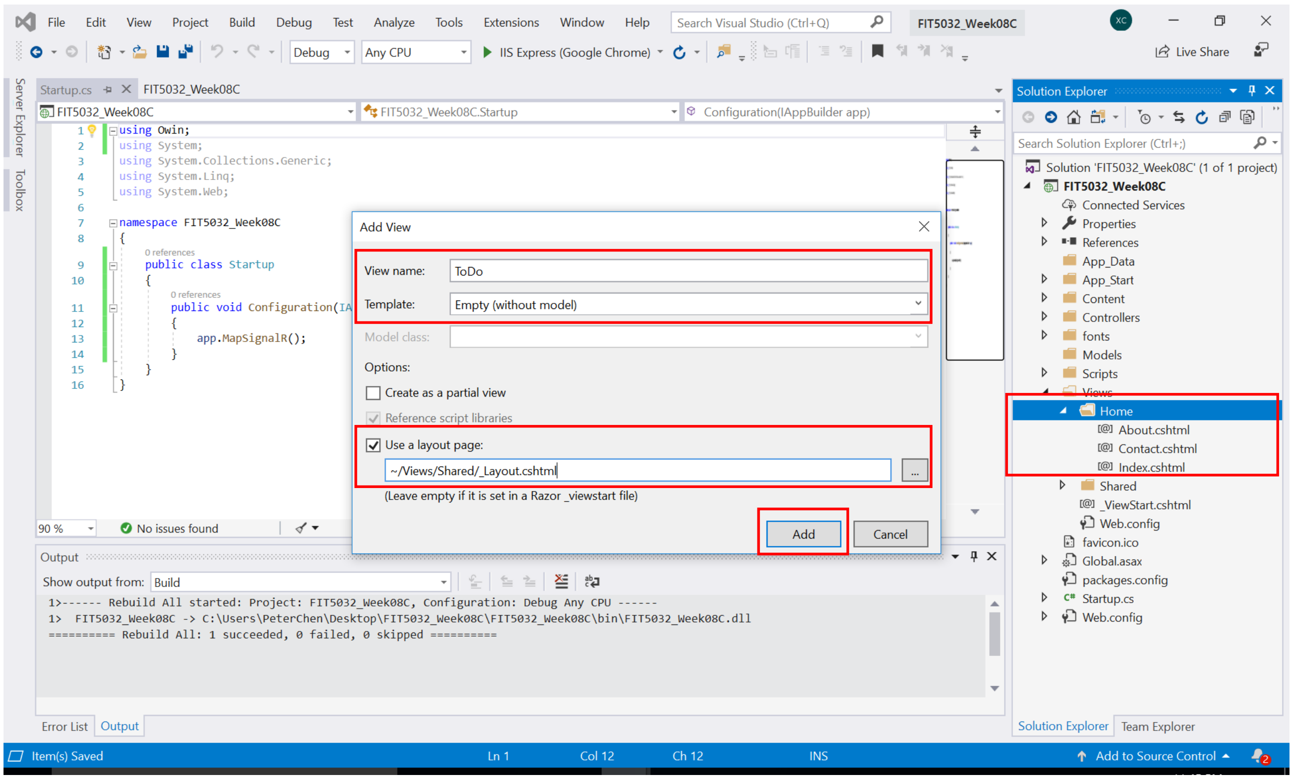
```
using Owin;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace FIT5032_Week08C
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.MapSignalR();
        }
    }
}
```

```
1>------ Rebuild All started: Project: FIT5032_Week08C, Configuration: Debug Any CPU ------
1>  FIT5032_Week08C -> C:\Users\PeterChen\Desktop\FIT5032_Week08C\FIT5032_Week08C\bin\FIT5032_Week08C.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

MONASH University

Step 7

Add a View called **ToDo** (Put it inside of the Views → Home folder).



```
using Owin;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace FIT5032_Week08C
{
    public class Startup
    {
        public void Configuration(IA
        {
            app.MapSignalR();
        }
    }
}
```

View name: ToDo

Template: Empty (without model)

Model class:

Options:

- [ ] Create as a partial view
- [✓] Reference script libraries
- [✓] Use a layout page:

~/Views/Shared/_Layout.cshtml

(Leave empty if it is set in a Razor _viewstart file)

```
1>------ Rebuild All started: Project: FIT5032_Week08C, Configuration: Debug Any CPU ------
1>  FIT5032_Week08C -> C:\Users\PeterChen\Desktop\FIT5032_Week08C\FIT5032_Week08C\bin\FIT5032_Week08C.dll
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

Step 8

The **ToDo.cshtml** is provided as follows.

```
@{
    ViewBag.Title = "To Do";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>To Do List</h2>
<div class="container">
    <form onsubmit="return false">
        <div class="form-group">
            <label>Display Name:</label>
            <input type="text" class="form-control" id="displayname" required />
        </div>
        <div class="form-group">
            <label>Add to do</label>
            <input type="text" class="form-control" id="message" required />
        </div>
        <button type="submit" class="btn btn-primary" id="sendmessage">Add to List</button>
    </form>
    <h2>Items on the list</h2>
    <ul id="discussion" class="list-group"></ul>
</div>
@section scripts {
    <!--Script references. -->
    <!--The jQuery library is required and is referenced by default in _Layout.cshtml. -->
    <!--Reference the SignalR library. -->
    <script src="~/Scripts/jquery.signalR-2.4.1.min.js"></script>
    <!--Reference the autogenerated SignalR hub script. -->
    <script src="~/signalr/hubs"></script>
    <!--SignalR script to update the chat page and send messages.-->
    <script>
        $(function () {
            // Reference the auto-generated proxy for the hub.
            var toDo = $.connection.toDoHub;
            // Create a function that the hub can call back to display messages.
            toDo.client.addNewMessageToPage = function (name, message) {
                // Add the message to the page.
                $('#discussion').append("<li class='list-group-item'><strong>" + htmlEncode(name)
                    + '</strong>: ' + htmlEncode(message) + '</li>');
            };
            // Start the connection.
            $.connection.hub.start().done(function () {
                $('#sendmessage').click(function () {
                    var displayname = $('#displayname').val();
                    var message = $('#message').val();
                    if (displayname.length == 0 || message.length == 0)
                        return;
                    // Call the Send method on the hub.
                    toDo.server.send($('#displayname').val(), $('#message').val());
                    // Clear text box and reset focus for next comment.
                    $('#message').val('');
                });
            });
        });
        // This optional function html-encodes messages for display in the page.
        function htmlEncode(value) {
            var encodedValue = $('<div />').text(value).html();
            return encodedValue;
        }
    </script>
}
```

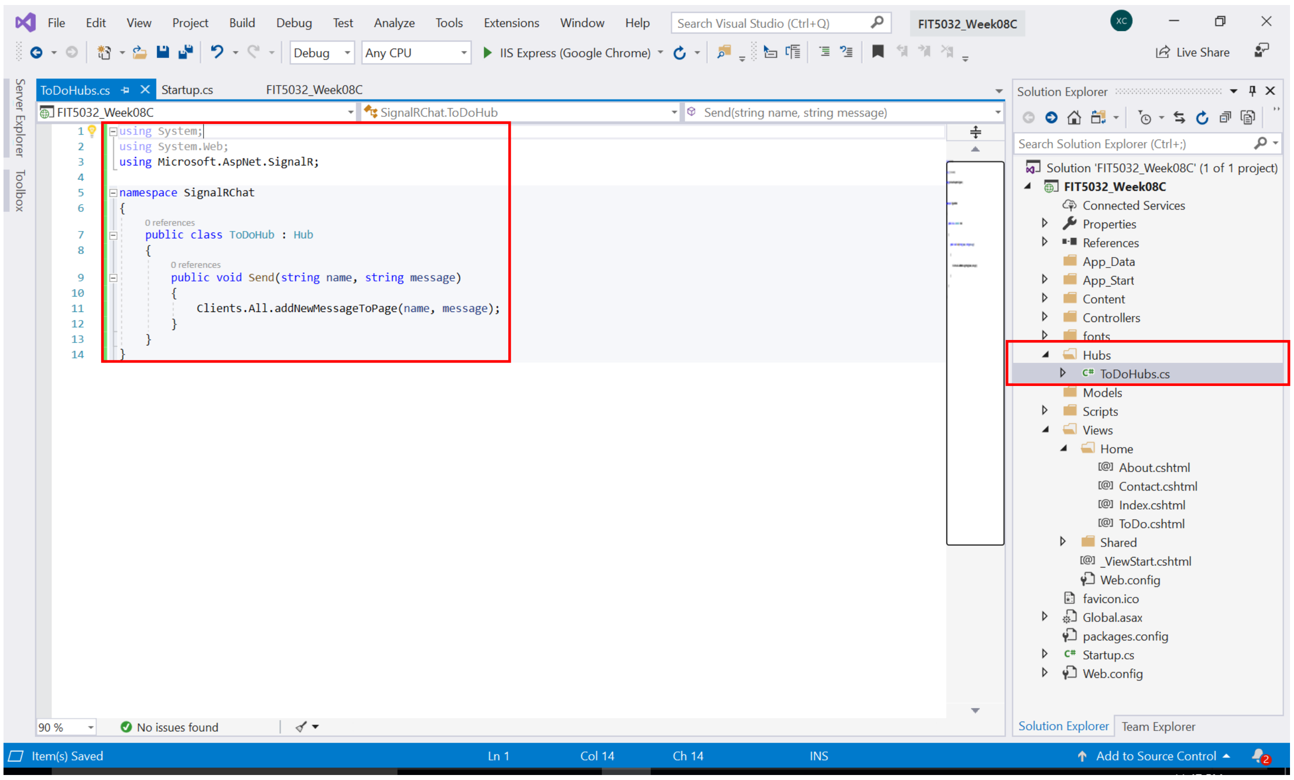
Please ensure that the SignalR version is checked and updated accordingly.

<script src="~/Scripts/**jquery.signalR-2.4.1.min.js**"></script>

Step 9

Create a folder called **Hubs** and create a **ToDoHubs.cs** file with the following code.

```
using System;
using System.Web;
using Microsoft.AspNet.SignalR;


namespace SignalRChat
{
    public class ToDoHub : Hub
    {
        public void Send(string name, string message)
        {
            Clients.All.addNewMessageToPage(name, message);
        }
    }
}
```



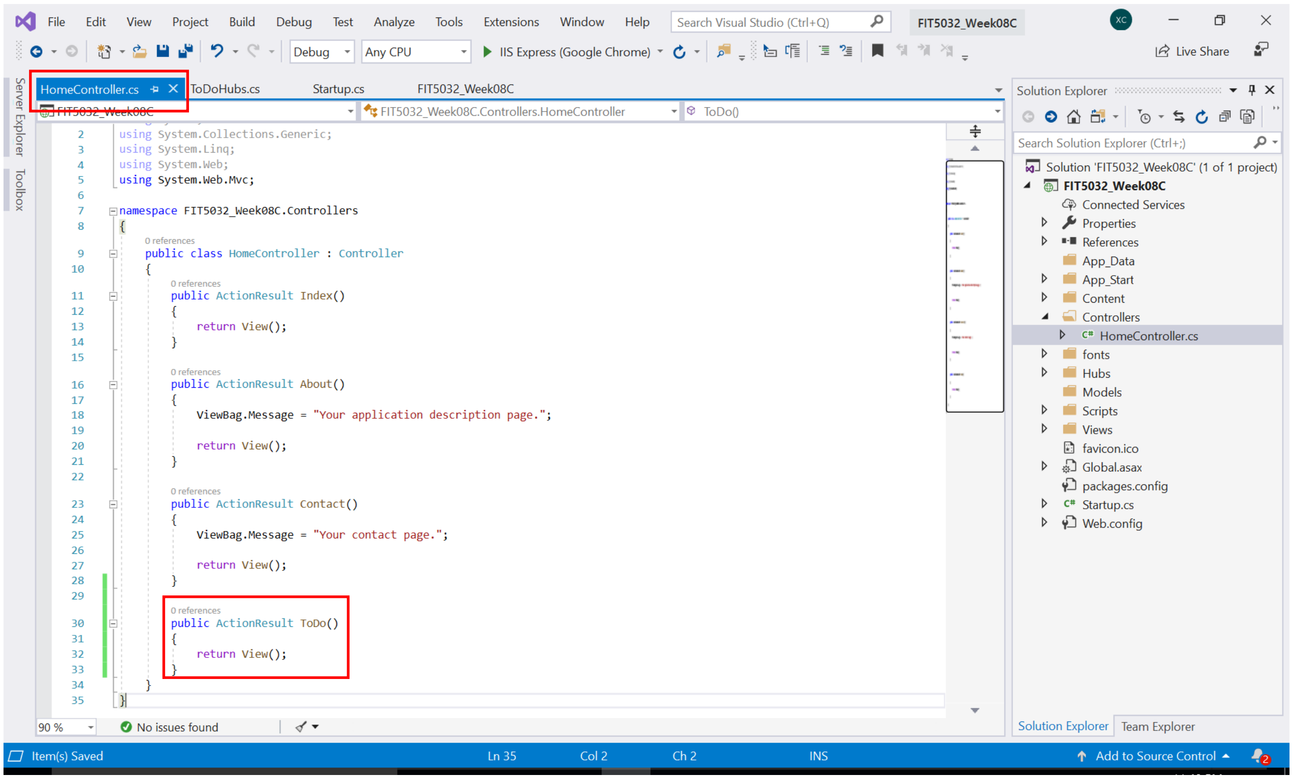
```
using System;
using System.Web;
using Microsoft.AspNet.SignalR;

namespace SignalRChat
{
    0 references
    public class ToDoHub : Hub
    {
        0 references
        public void Send(string name, string message)
        {
            Clients.All.addNewMessageToPage(name, message);
        }
    }
}
```

```
Solution 'FIT5032_Week08C' (1 of 1 project)
  FIT5032_Week08C
    Connected Services
    Properties
    References
    App_Data
    App_Start
    Content
    Controllers
    fonts
    Hubs
      ToDoHubs.cs
    Models
    Scripts
    Views
      Home
        About.cshtml
        Contact.cshtml
        Index.cshtml
        ToDo.cshtml
      Shared
      _ViewStart.cshtml
      Web.config
    favicon.ico
    Global.asax
    packages.config
    Startup.cs
    Web.config
```

Step 10

Remember to also introduce the controller method at the **HomeController.cs**

```
public ActionResult ToDo()
{
    return View();
}
```

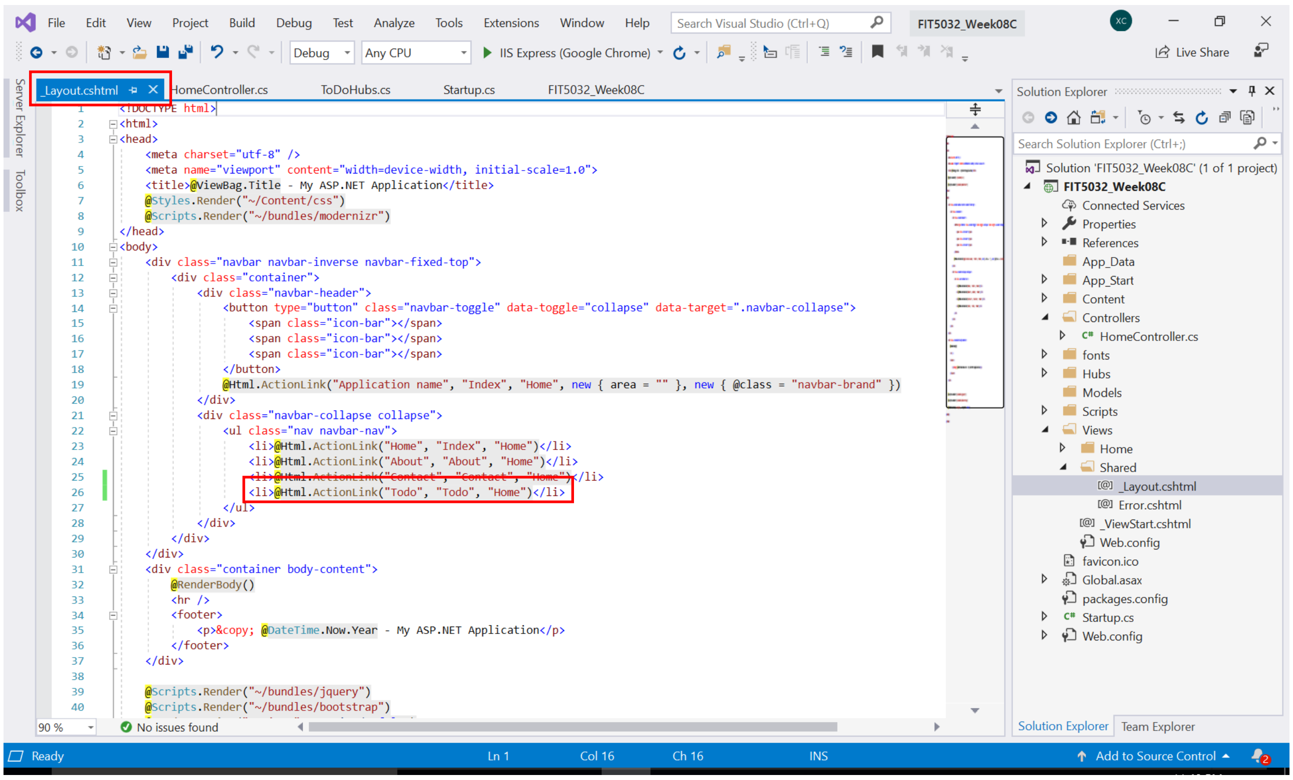


```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace FIT5032_Week08C.Controllers
{
    0 references
    public class HomeController : Controller
    {
        0 references
        public ActionResult Index()
        {
            return View();
        }

        0 references
        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        0 references
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";

            return View();
        }

        0 references
        public ActionResult ToDo()
        {
            return View();
        }
    }
}
```

Step 11

To make things easier, you can also put a hyperlink at the **_Layout.cshtml** file.

```
<div class="navbar-collapse collapse">
    <ul class="nav navbar-nav">
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("About", "About", "Home")</li>
        <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        <li>@Html.ActionLink("Todo", "Todo", "Home")</li>
    </ul>
</div>
```

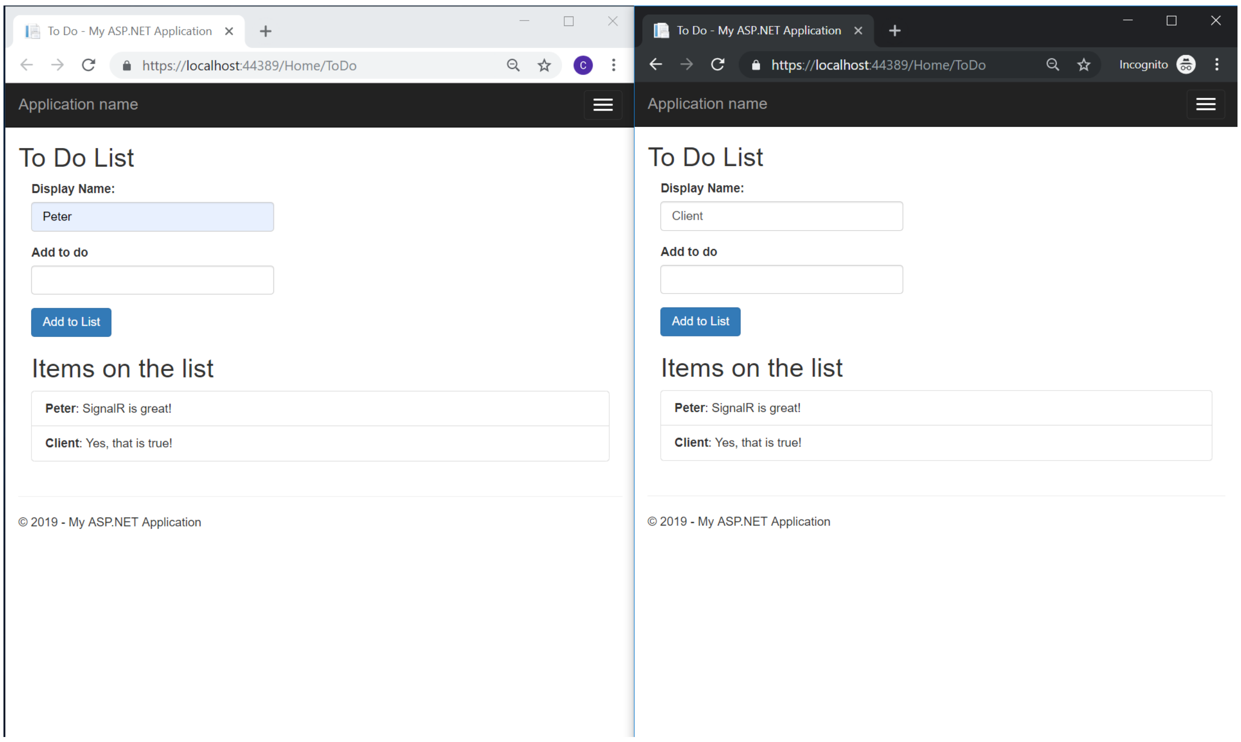


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Application name", "Index", "Home", new { area = "" }, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                    <li>@Html.ActionLink("Todo", "Todo", "Home")</li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
```

Step 12

After you have done so, you can test it out by opening 2 browsers. You can have one of the browsers in InCognito mode (CTRL + SHIFT + N on Google Chrome).

If you add to-dos, you will realize that the other client will receive the item as well. This happens at real-time. So, you do not see the browser being "Refreshed".


To Do - My ASP.NET Application
https://localhost:44389/Home/ToDo
Application name
To Do List
Display Name:
Peter
Add to do
Add to List
Items on the list
Peter: SignalR is great!
Client: Yes, that is true!
© 2019 - My ASP.NET Application
To Do - My ASP.NET Application
https://localhost:44389/Home/ToDo
Incognito
Application name
To Do List
Display Name:
Client
Add to do
Add to List
Items on the list
Peter: SignalR is great!
Client: Yes, that is true!
© 2019 - My ASP.NET Application

CONCLUSION

Upon the completion of this supplementary material, you will gain a basic understanding of how SignalR works. The current project uses a modified version of the tutorial online with some basic Bootstrap classes added.

At the moment, only the clients send messages to each other, but it is possible for the server to send real-time updates as well.