MONASH University

FIT5032 - Internet Applications Development

WEEK 08C - SIGNALR

Last updated: 29th July 2018
Author: Jian Liew

Introduction

There is no need 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 this. Real time features allows you to deliver information to your user as it happens.

Here are some examples of use cases that requires 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 is updated, it sends and update to each of the client.

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

Objectives

Estimated Time To Complete - 1 hour
Upon the completion of this tutorial, you will gain a basic understanding of

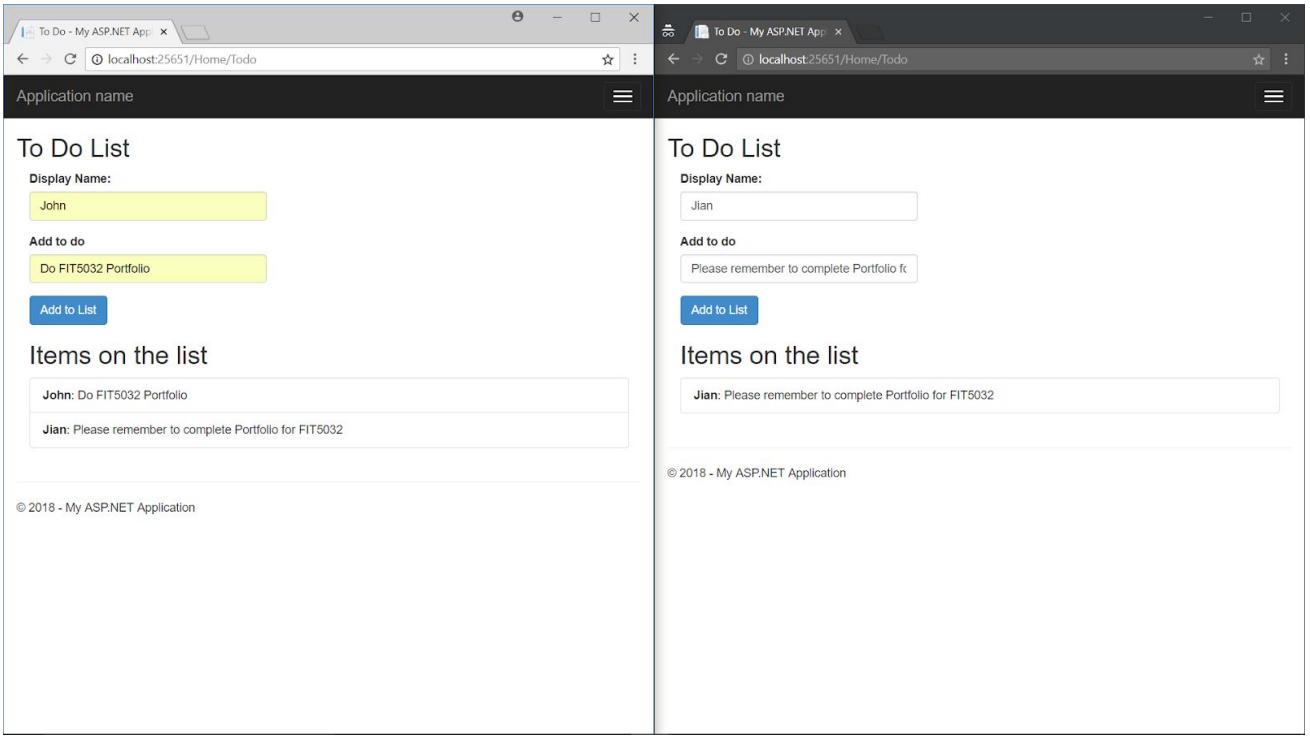
- Real Time Features of a Web Application
- How to use SignalR at a basic level

DoubtFire Submission

- **None**

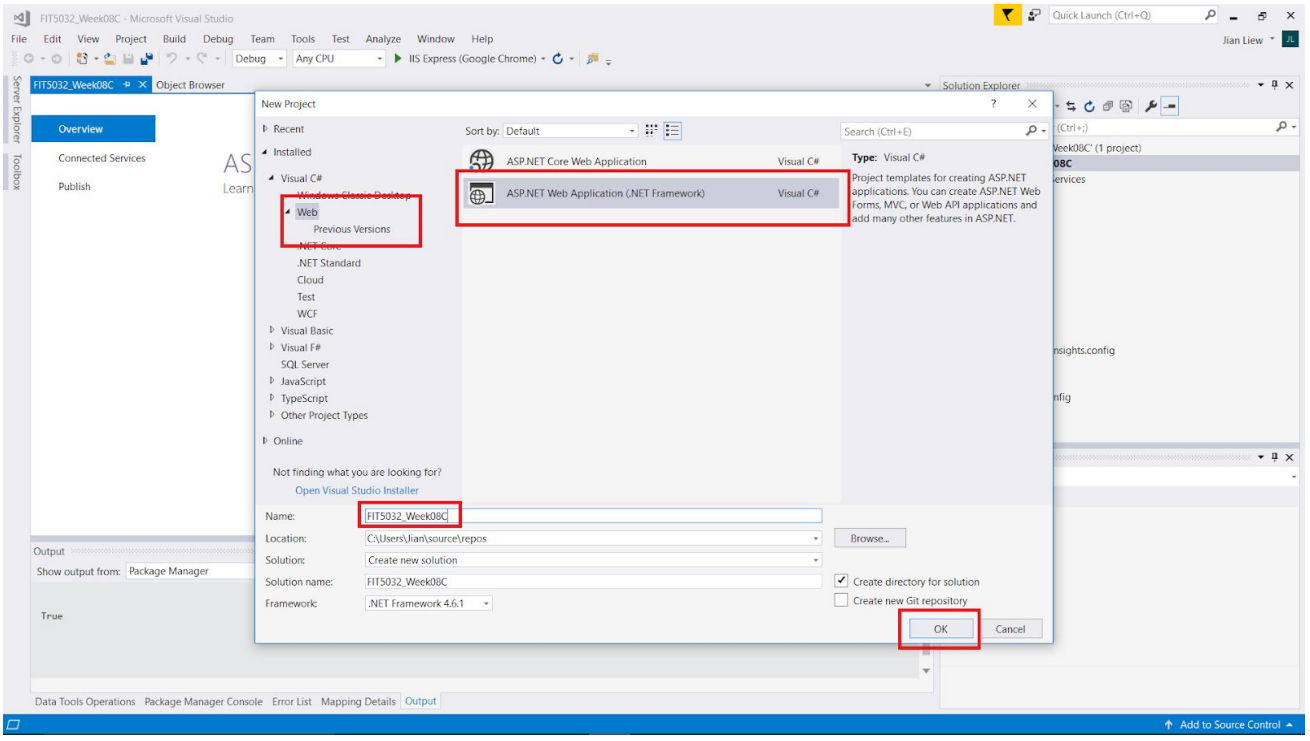
Upon the completion of this tutorial, you will have a basic understanding of how SignalR works.

In the end product, you should be able to see how SignalR is able to perform real time features.

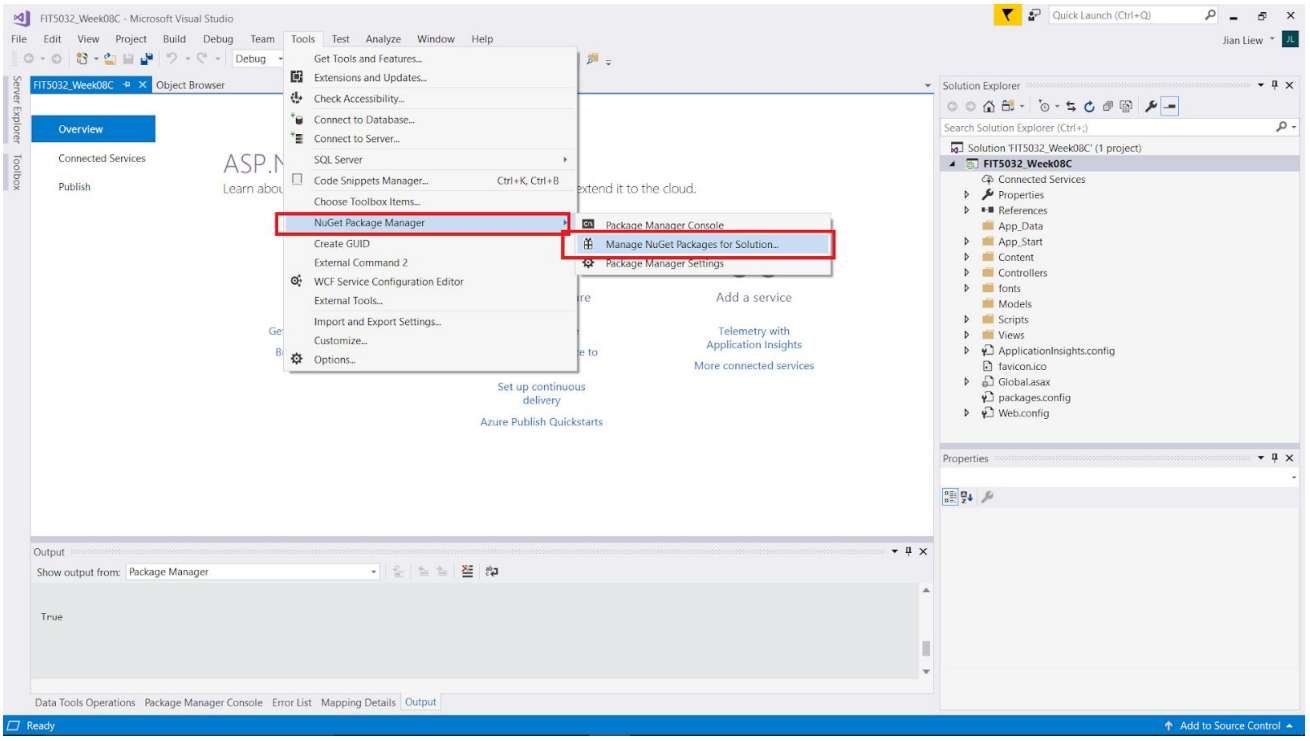


You will notice that in this screenshot, I have opened two browsers. Both of these are clients. Once the first client creates an item on the to do list, it will appear on the second client as well. This happens at real time.

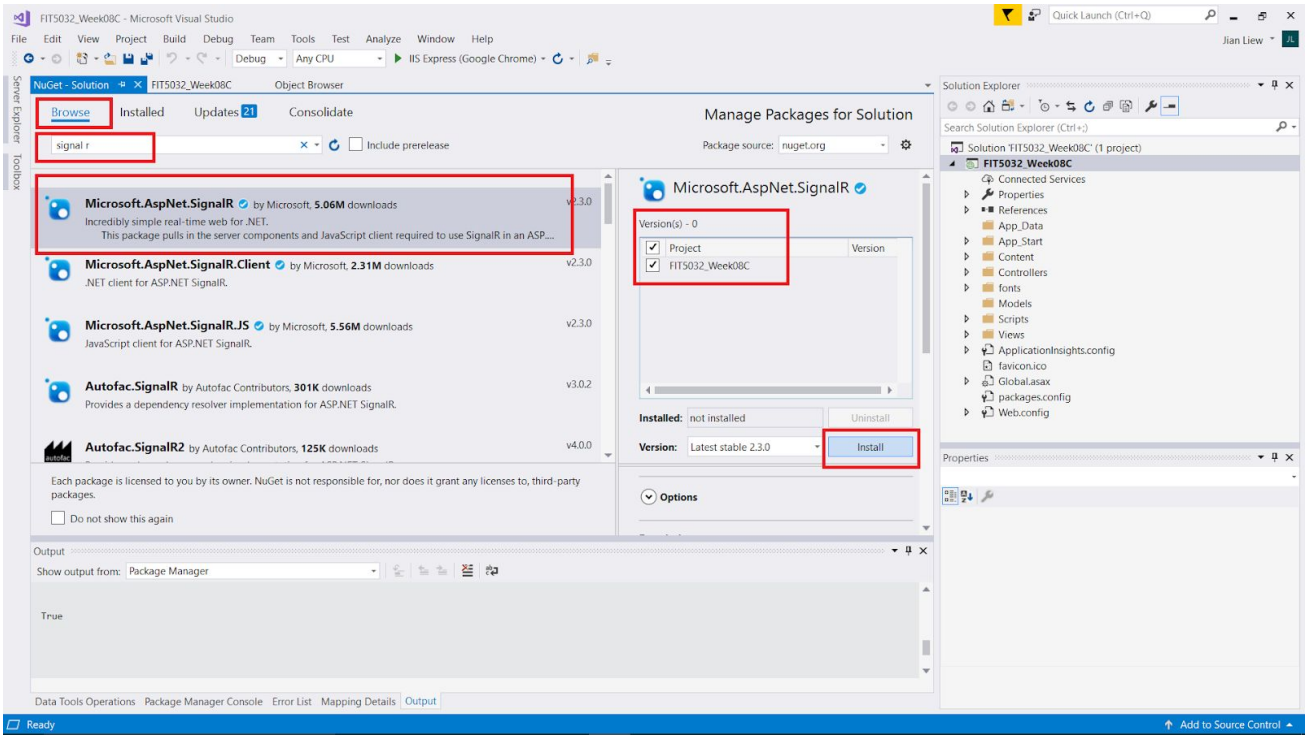
Step 1


FIT5032_Week08C - Microsoft Visual Studio
New Project
Recent
Installed
Visual C#
Web
Previous Versions
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
JavaScript
TypeScript
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
ASP.NET Core Web Application
ASP.NET Web Application (.NET Framework)
Type: Visual C#
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.
Name: FIT5032_Week08C
Location: C:\Users\Jian\source\repos
Solution: Create new solution
Solution name: FIT5032_Week08C
Framework: .NET Framework 4.6.1
Browse...
Create directory for solution
Create new Git repository
OK
Cancel
Overview
Connected Services
Publish
Output
Show output from: Package Manager
True
Data Tools Operations
Package Manager Console
Error List
Mapping Details
Output
Add to Source Control

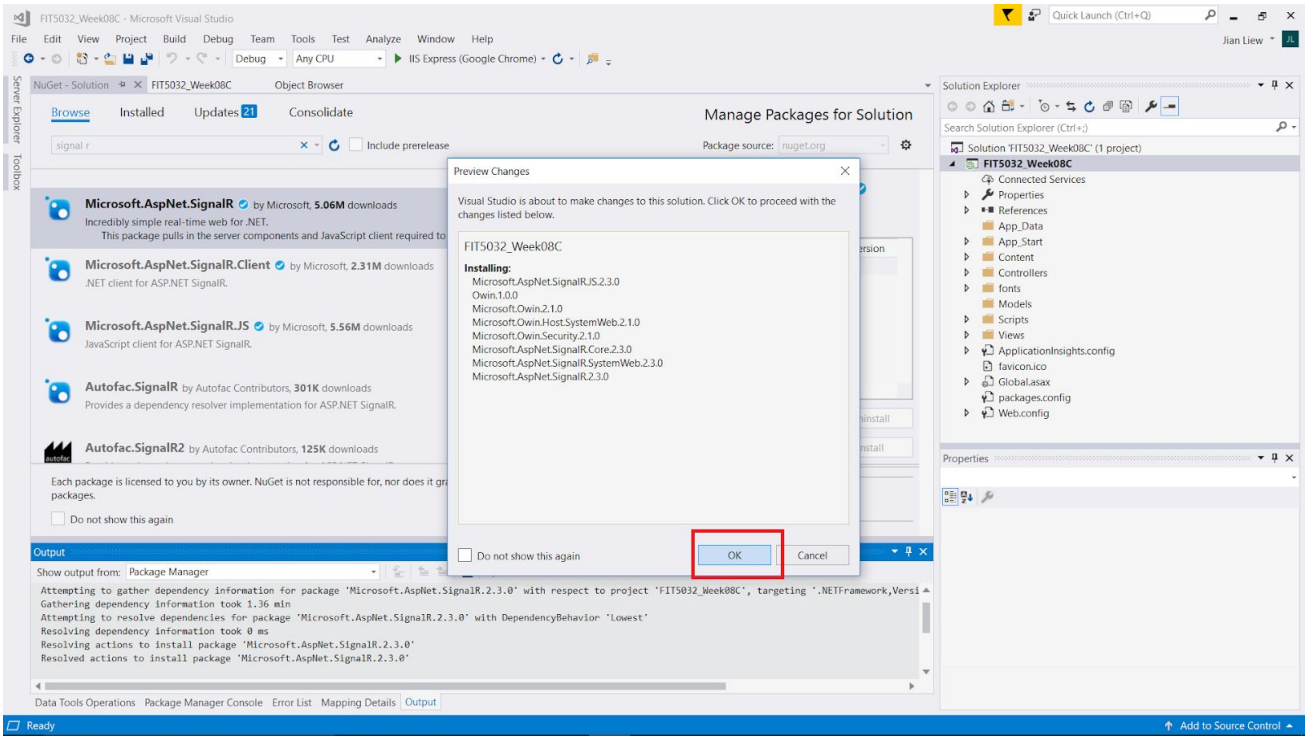
Step 2


Tools
Get Tools and Features...
Extensions and Updates...
Check Accessibility...
Connect to Database...
Connect to Server...
SQL Server
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
NuGet Package Manager
Create GUID
External Command 2
WCF Service Configuration Editor
External Tools...
Import and Export Settings...
Customize...
Options...
Package Manager Console
Manage NuGet Packages for Solution...
Package Manager Settings
Solution Explorer
Solution 'FIT5032_Week08C' (1 project)
FIT5032_Week08C
Connected Services
Properties
References
App_Data
App_Start
Content
Controllers
fonts
Models
Scripts
Views
ApplicationInsights.config
favicon.ico
Global.asax
packages.config
Web.config
Add a service
Telemetry with Application Insights
More connected services
Set up continuous delivery
Azure Publish Quickstarts
Output
Show output from: Package Manager
True
Data Tools Operations
Package Manager Console
Error List
Mapping Details
Output
Ready
Add to Source Control

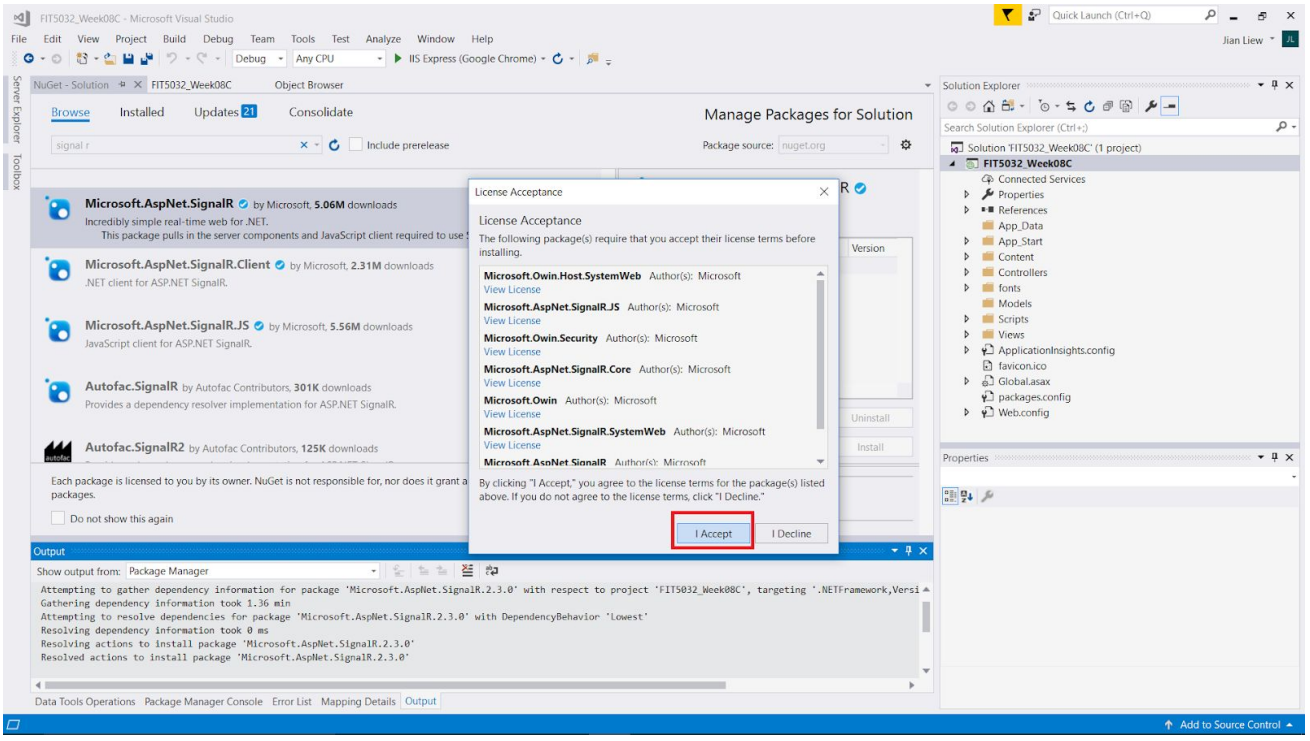
Step 3


FIT5032_Week08C - Microsoft Visual Studio
Manage Packages for Solution
Browse
Installed
Updates 21
Consolidate
signal r
Include prerelease
Package source: nuget.org
Microsoft.AspNet.SignalR by Microsoft, 5.06M downloads
Incredibly simple real-time web for .NET.
This package pulls in the server components and JavaScript client required to use SignalR in an ASP....
Microsoft.AspNet.SignalR.Client by Microsoft, 2.31M downloads
.NET client for ASP.NET SignalR.
Microsoft.AspNet.SignalR.JS by Microsoft, 5.56M downloads
JavaScript client for ASP.NET SignalR.
Autofac.SignalR by Autofac Contributors, 301K downloads
Provides a dependency resolver implementation for ASP.NET SignalR.
Autofac.SignalR2 by Autofac Contributors, 125K downloads
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
Version(s) - 0
Project
Version
FIT5032_Week08C
Installed: not installed
Uninstall
Version: Latest stable 2.3.0
Install
Options
Output
Show output from: Package Manager
True
Solution Explorer
Properties
Ready
Add to Source Control

Step 4

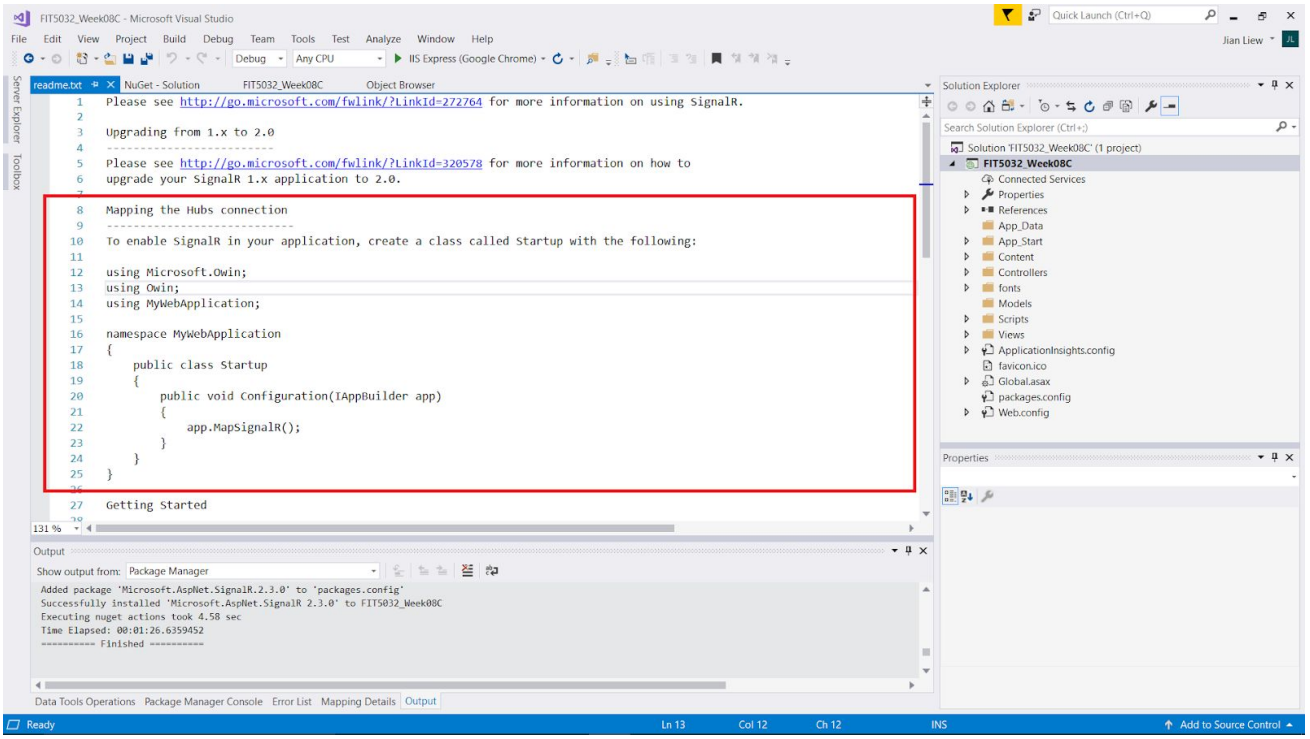

Preview Changes
Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.
FIT5032_Week08C
Installing:
Microsoft.AspNet.SignalR.JS.2.3.0
Owin.1.0.0
Microsoft.Owin.2.1.0
Microsoft.Owin.Host.SystemWeb.2.1.0
Microsoft.Owin.Security.2.1.0
Microsoft.AspNet.SignalR.Core.2.3.0
Microsoft.AspNet.SignalR.SystemWeb.2.3.0
Microsoft.AspNet.SignalR.2.3.0
Do not show this again
OK
Cancel
Attempting to gather dependency information for package 'Microsoft.AspNet.SignalR.2.3.0' with respect to project 'FIT5032_Week08C', targeting '.NETFramework,Versi
Gathering dependency information took 1.36 min
Attempting to resolve dependencies for package 'Microsoft.AspNet.SignalR.2.3.0' with DependencyBehavior 'Lowest'
Resolving dependency information took 0 ms
Resolving actions to install package 'Microsoft.AspNet.SignalR.2.3.0'
Resolved actions to install package 'Microsoft.AspNet.SignalR.2.3.0'

Step 5



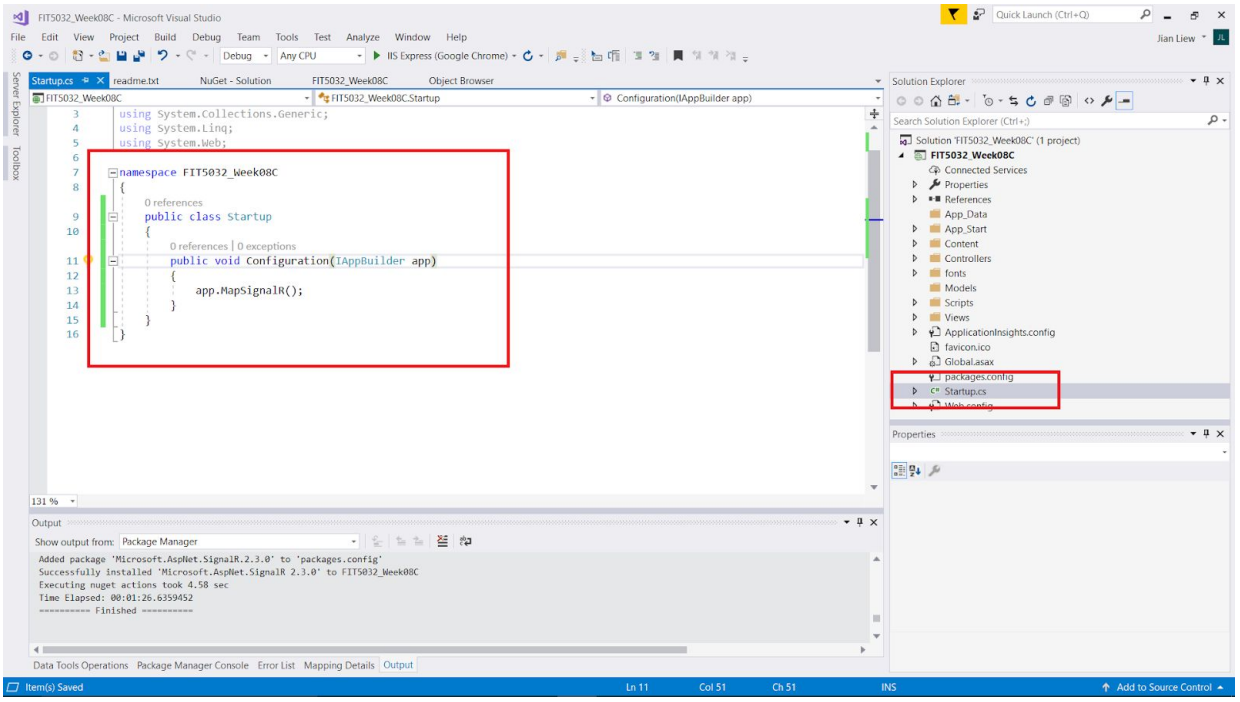
Step 6

You will be given a set of instructions on how to map the hub connections in the readme.txt but you do not need to follow them. Follow this guide instead.



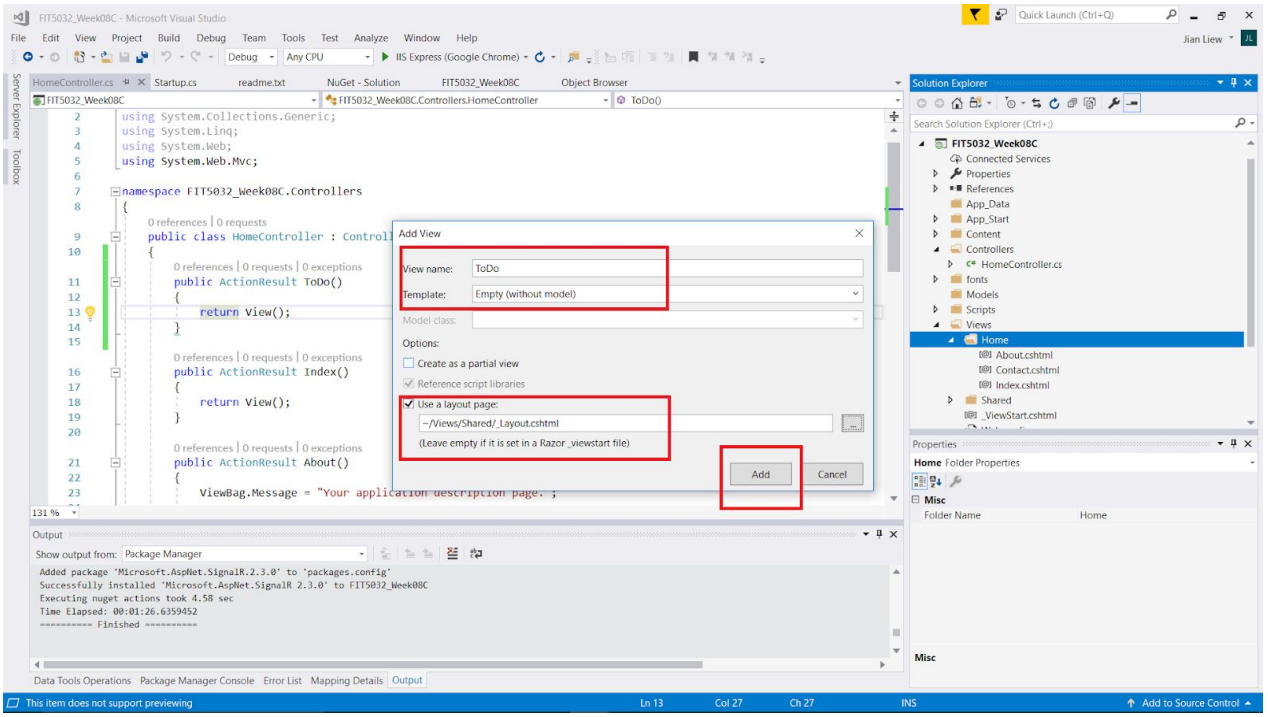
Step 7

SignalR needs to be started so there must be a **StartUp** configuration created. The Codes for the StartUp class can be found at the readme.txt which was seen before.



Step 8

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



Step 9

The ToDo.cshtml is provided as follows. These codes are from the examples with minor modifications.

```
@{
    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-2.3.0.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>
}
```

Step 10

Create a Folder called "Hubs" and create a **"ToDoHubs.cs"**

```
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);
        }
    }
}
```

Step 11

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

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

Step 11

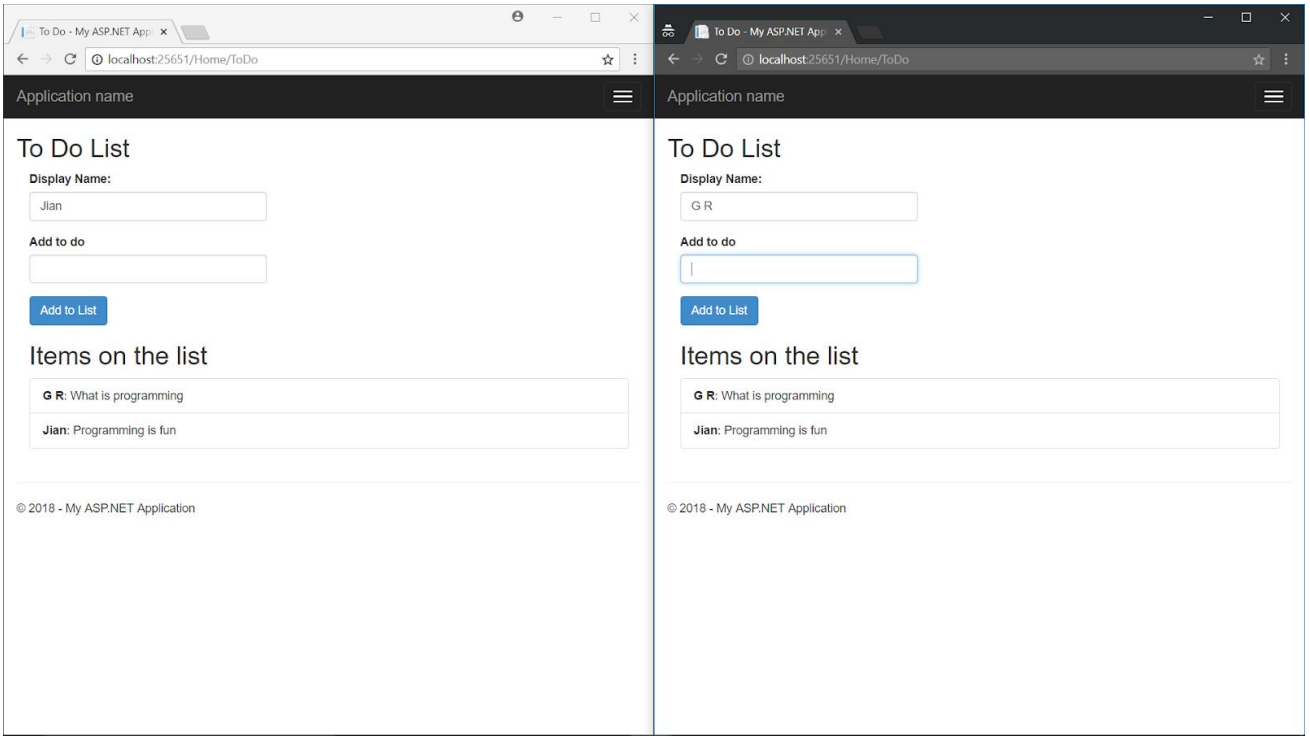
To make things easier, you can also put a hyperlink at the _Layout.cshtml

```
<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>
```

Step 12

After you have done so, you can test it out. Do this, 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 realise that the other client will receive the item as well. This happens at real time. So, you do not see the browser being "Refreshed".



Explanation

What SignalR does, is it automatically creates client side JavaScripts.

Conclusion

Upon the completion of this supplementary tutorial, 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, it only the client sending messages to each other, but it is actually possible for the server to send real time updates as well.