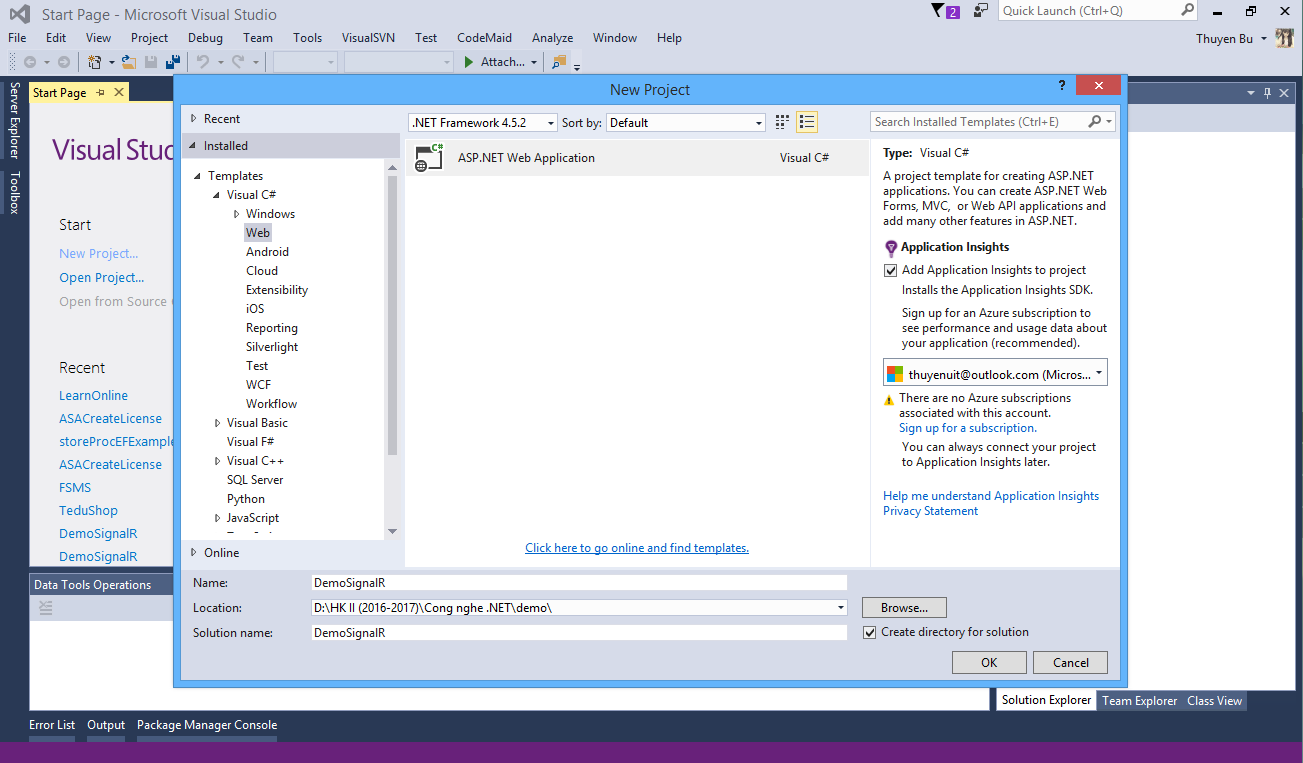
Yêu cầu: Chuẩn bị Visual Studio 2013 trở lên

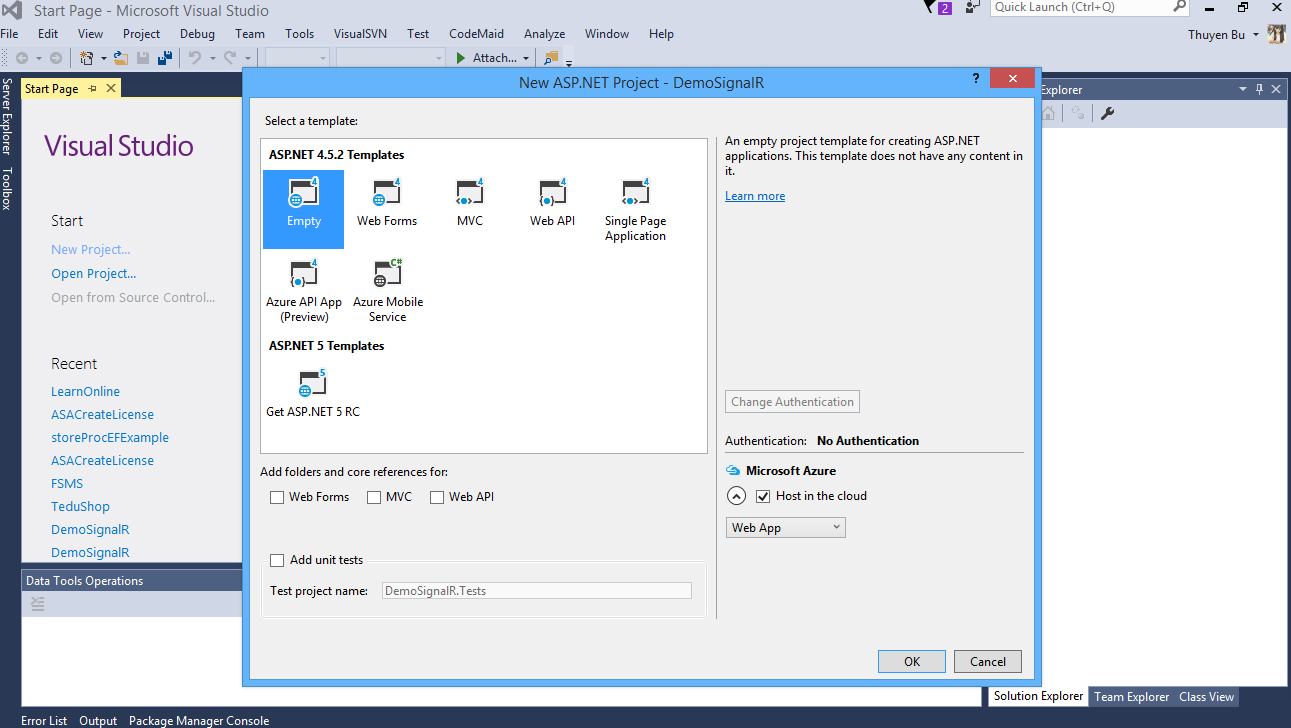
# Bước 1: **New project.**

Chọn **File** -> **New project**

Chọn **Templates** -> **Web** và đặt tên cho project là **DemoSignalR** và click **OK**.

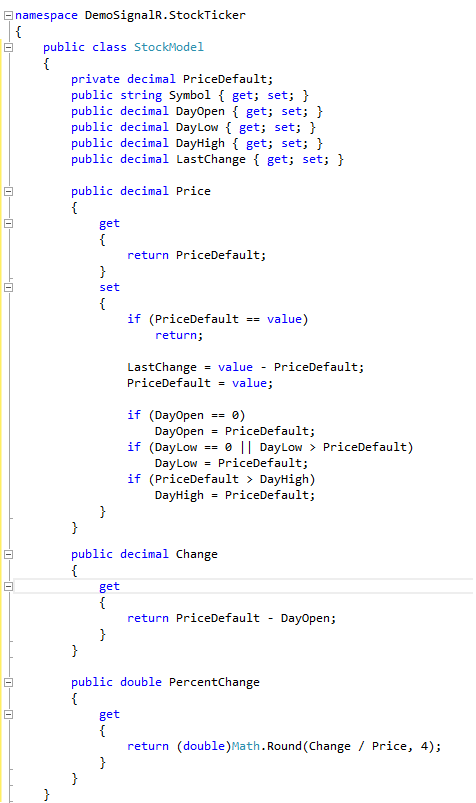


Một của sổ New ASP.NET Project hiện ra, chọn **Empty** và click OK.



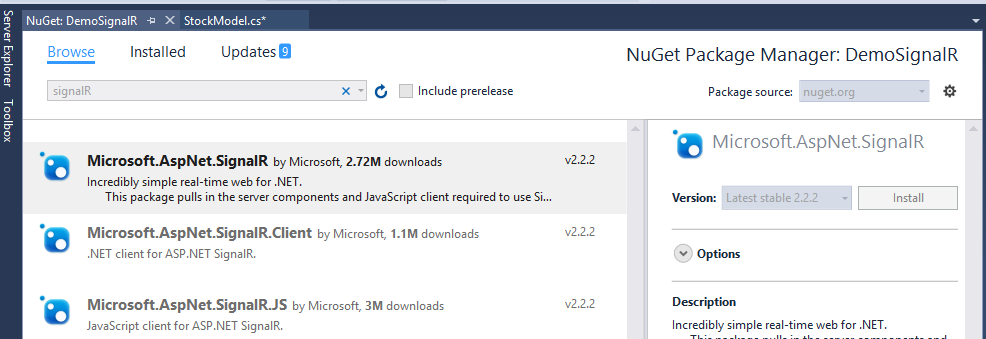
# Bước 2: Tạo một thư mục **StockTicker** trong project **DemoSignalR**

Tạo một class **StockModel.cs** trong thư mục **StockTicker** và mã code trong class như sau:

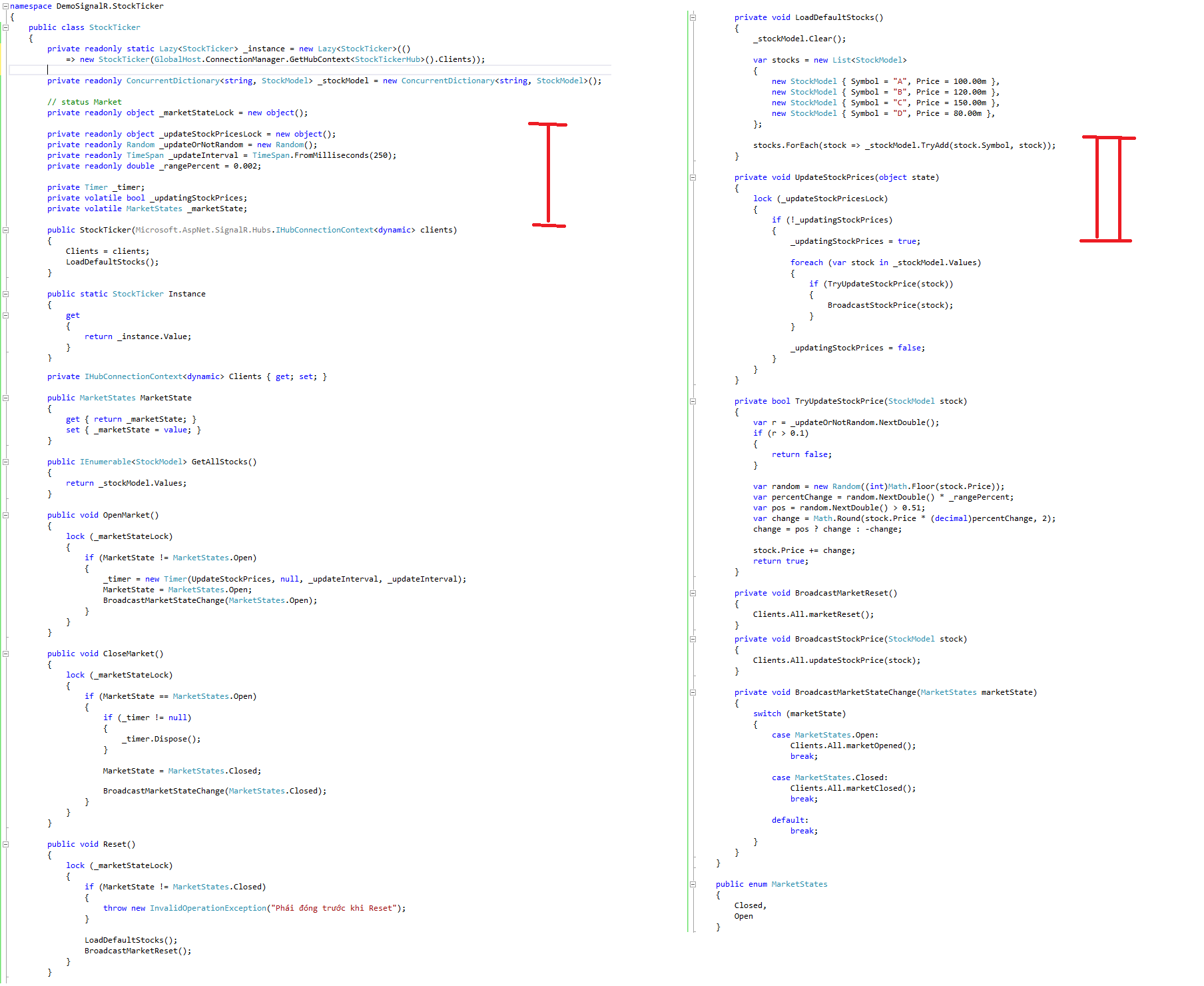


# Bước 3: Tạo **StockSticker** và **StockStickHub** class

Click vào **References** -> **Manage Nuget Packages**… để cài đặt **Microsoft.AspNet.SignalR** packge để tải thư viện **SignalR**



Tạo một StockTicker.cs class trong thư mục StockStick và mã code như sau:



public class StockTicker

{

private readonly static Lazy<StockTicker> \_instance = new Lazy<StockTicker>(()

=> new StockTicker(GlobalHost.ConnectionManager.GetHubContext<StockTickerHub>().Clients));

private readonly ConcurrentDictionary<string, StockModel> \_stockModel = new ConcurrentDictionary<string, StockModel>();

// status Market

private readonly object \_marketStateLock = new object();

private readonly object \_updateStockPricesLock = new object();

private readonly Random \_updateOrNotRandom = new Random();

private readonly TimeSpan \_updateInterval = TimeSpan.FromMilliseconds(250);

private readonly double \_rangePercent = 0.002;

private Timer \_timer;

private volatile bool \_updatingStockPrices;

private volatile MarketStates \_marketState;

public StockTicker(Microsoft.AspNet.SignalR.Hubs.IHubConnectionContext<dynamic> clients)

{

Clients = clients;

LoadDefaultStocks();

}

public static StockTicker Instance

{

get

{

return \_instance.Value;

}

}

private IHubConnectionContext<dynamic> Clients { get; set; }

public MarketStates MarketState

{

get { return \_marketState; }

set { \_marketState = value; }

}

public IEnumerable<StockModel> GetAllStocks()

{

return \_stockModel.Values;

}

public void OpenMarket()

{

lock (\_marketStateLock)

{

if (MarketState != MarketStates.Open)

{

\_timer = new Timer(UpdateStockPrices, null, \_updateInterval, \_updateInterval);

MarketState = MarketStates.Open;

BroadcastMarketStateChange(MarketStates.Open);

}

}

}

public void CloseMarket()

{

lock (\_marketStateLock)

{

if (MarketState == MarketStates.Open)

{

if (\_timer != null)

{

\_timer.Dispose();

}

MarketState = MarketStates.Closed;

BroadcastMarketStateChange(MarketStates.Closed);

}

}

}

public void Reset()

{

lock (\_marketStateLock)

{

if (MarketState != MarketStates.Closed)

{

throw new InvalidOperationException("Phái đóng trước khi Reset");

}

LoadDefaultStocks();

BroadcastMarketReset();

}

}

private void LoadDefaultStocks()

{

\_stockModel.Clear();

var stocks = new List<StockModel>

{

new StockModel { Symbol = "A", Price = 100.00m },

new StockModel { Symbol = "B", Price = 120.00m },

new StockModel { Symbol = "C", Price = 150.00m },

new StockModel { Symbol = "D", Price = 80.00m },

};

stocks.ForEach(stock => \_stockModel.TryAdd(stock.Symbol, stock));

}

private void UpdateStockPrices(object state)

{

lock (\_updateStockPricesLock)

{

if (!\_updatingStockPrices)

{

\_updatingStockPrices = true;

foreach (var stock in \_stockModel.Values)

{

if (TryUpdateStockPrice(stock))

{

BroadcastStockPrice(stock);

}

}

\_updatingStockPrices = false;

}

}

}

private bool TryUpdateStockPrice(StockModel stock)

{

var r = \_updateOrNotRandom.NextDouble();

if (r > 0.1)

{

return false;

}

var random = new Random((int)Math.Floor(stock.Price));

var percentChange = random.NextDouble() \* \_rangePercent;

var pos = random.NextDouble() > 0.51;

var change = Math.Round(stock.Price \* (decimal)percentChange, 2);

change = pos ? change : -change;

stock.Price += change;

return true;

}

private void BroadcastMarketReset()

{

Clients.All.marketReset();

}

private void BroadcastStockPrice(StockModel stock)

{

Clients.All.updateStockPrice(stock);

}

private void BroadcastMarketStateChange(MarketStates marketState)

{

switch (marketState)

{

case MarketStates.Open:

Clients.All.marketOpened();

break;

case MarketStates.Closed:

Clients.All.marketClosed();

break;

default:

break;

}

}

}

public enum MarketStates

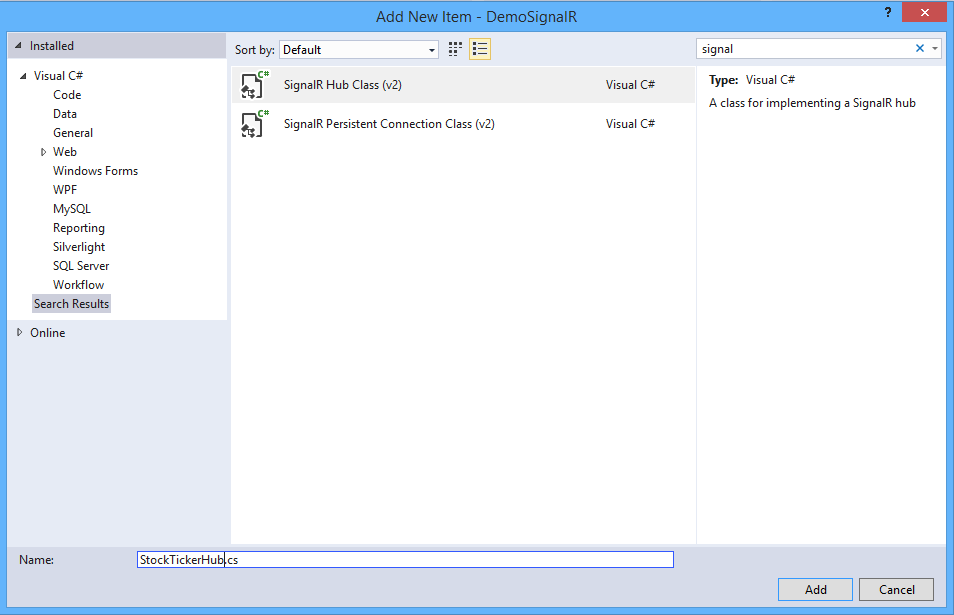
{

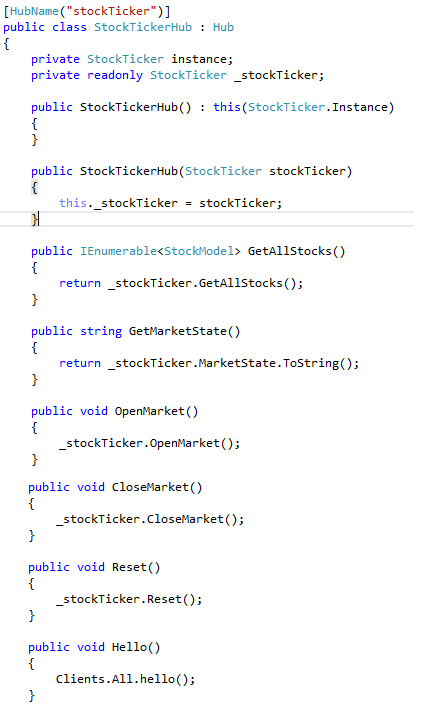
Closed,

Open

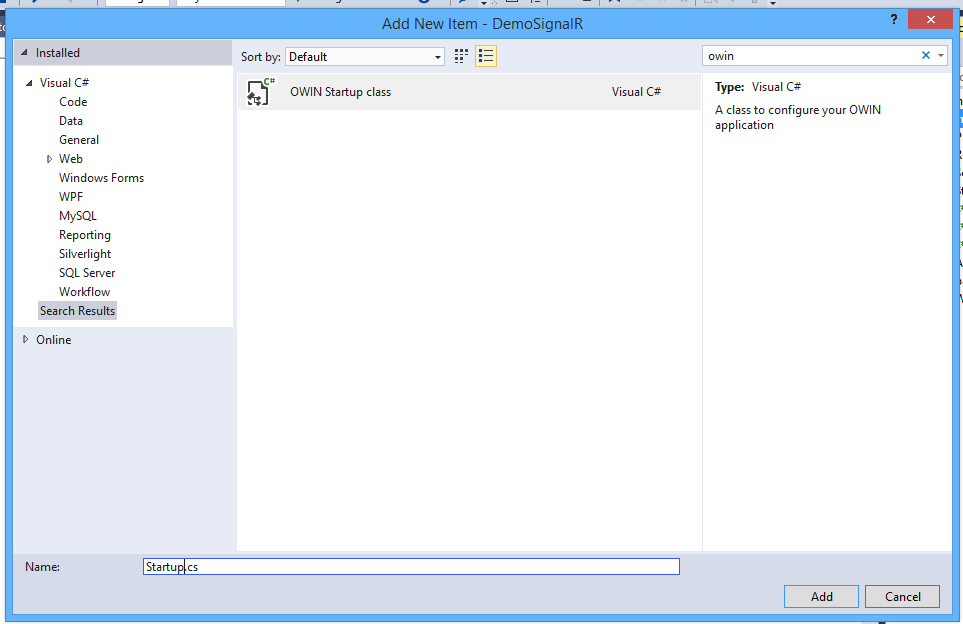
}

Tạo một class SignalR Hub có tên là **StockStickHub.cs** và mã code của hub này như sau:

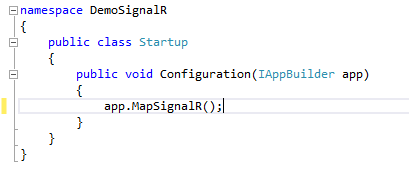




# Bước 4: Tạo một **OWIN Startup Class**

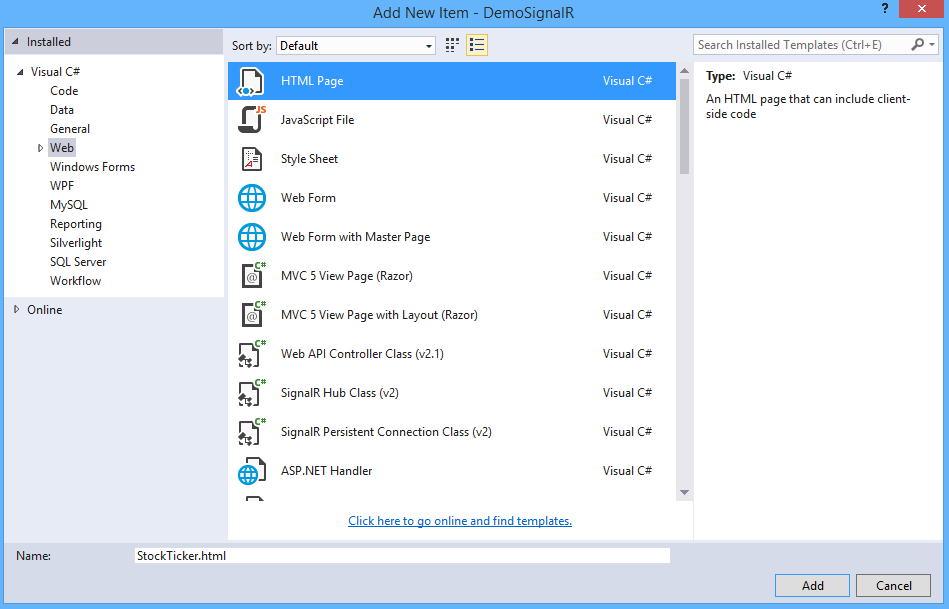


Map SignalR vào **Startup.cs** class vừa tạo

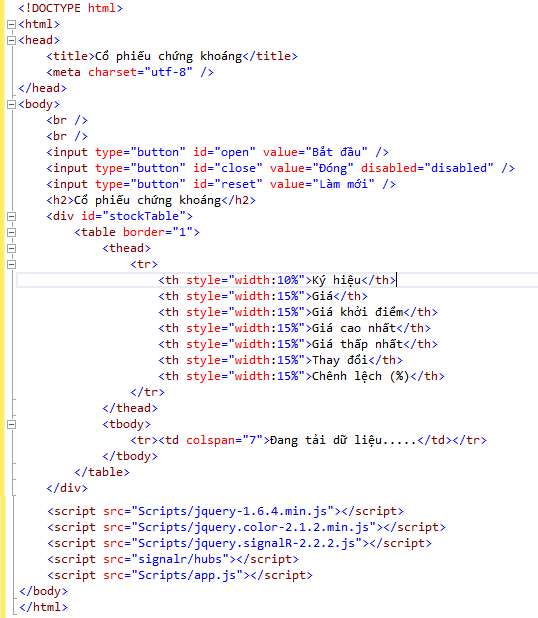


# Bước 5 : Tạo view StockTicker bằng HTML đặt tên là **StockTicker.html**

Click DemoSignalR -> Add -> New Item -> Đặt tên cho view và click OK



Mã code view như sau:



Lưu ý:

* jquery.color-2.1.2.min.js có thể tải tại địa chỉ sau: <https://github.com/jquery/jquery-color> sau đó add vào thư mục Scripts của project
* app.js sẽ được thực hiện ở Bước 6

# Bước 6: Tạo một JavaScript file đặt tên **app.js** trong thư mục **Scripts** tương tự như tạo view HTML và mã code như sau:

if (!String.prototype.supplant) {

String.prototype.supplant = function (o) {

return this.replace(/{([^{}]\*)}/g,

function (a, b) {

var r = o[b];

return typeof r === 'string' || typeof r === 'number' ? r : a;

}

);

};

}

// hiệu ứng màu

jQuery.fn.flash = function (color, duration) {

var current = this.css('backgroundColor');

this.animate({ backgroundColor: 'rgb(' + color + ')' }, duration / 2)

.animate({ backgroundColor: current }, duration / 2);

};

$(function () {

var ticker = $.connection.stockTicker,

up = '▲',

down = '▼',

$stockTable = $('#stockTable'),

$stockTableBody = $stockTable.find('tbody'),

rowTemplate = '<tr data-symbol="{Symbol}"><td>{Symbol}</td><td>{Price}</td><td>{DayOpen}</td><td>{DayHigh}</td><td>{DayLow}</td><td><span class="dir {DirectionClass}">{Direction}</span> {Change}</td><td>{PercentChange}</td></tr>',

$stockTicker = $('#stockTicker'),

$stockTickerUl = $stockTicker.find('ul'),

liTemplate = '<li data-symbol="{Symbol}"><span class="symbol">{Symbol}</span> <span class="price">{Price}</span> <span class="change"><span class="dir {DirectionClass}">{Direction}</span> {Change} ({PercentChange})</span></li>';

function formatStock(stock) {

return $.extend(stock, {

Price: stock.Price.toFixed(2),

PercentChange: (stock.PercentChange \* 100).toFixed(2) + '%',

Direction: stock.Change === 0 ? '' : stock.Change >= 0 ? up : down,

DirectionClass: stock.Change === 0 ? 'even' : stock.Change >= 0 ? 'up' : 'down'

});

}

function stopTicker() {

$stockTickerUl.stop();

}

// khởi tạo

function init() {

return ticker.server.getAllStocks().done(function (stocks) {

$stockTableBody.empty();

$.each(stocks, function () {

var stock = formatStock(this);

$stockTableBody.append(rowTemplate.supplant(stock));

});

});

}

// bên client thêm phương thức hub khi server gọi

$.extend(ticker.client, {

updateStockPrice: function (stock) {

var displayStock = formatStock(stock),

$row = $(rowTemplate.supplant(displayStock)),

bg = stock.LastChange < 0

? '255,148,148' // red

: '154,240,117'; // green

$stockTableBody.find('tr[data-symbol=' + stock.Symbol + ']')

.replaceWith($row);

$row.flash(bg, 1000); // call funtion tạo hiệu ứng màu

},

marketOpened: function () {

$("#open").prop("disabled", true);

$("#close").prop("disabled", false);

$("#reset").prop("disabled", true);

},

marketClosed: function () {

$("#open").prop("disabled", false);

$("#close").prop("disabled", true);

$("#reset").prop("disabled", false);

stopTicker();

},

// reset

marketReset: function () {

return init();

}

});

// Bắt đầu connection

$.connection.hub.start()

.then(init)

.then(function () {

})

.done(function (state) {

if (state === 'Open') {

ticker.client.marketOpened();

} else {

ticker.client.marketClosed();

}

// open

$("#open").click(function () {

ticker.server.openMarket();

});

// clode

$("#close").click(function () {

ticker.server.closeMarket();

});

// reset

$("#reset").click(function () {

ticker.server.reset();

});

});

});

# Bước 7: Run project

