# Calling Other Systems

# Project Set Up

## Setup Steps for Visual Studio

From within VS (using point and click):

* Create new solution called ConnectingToOtherSystems
* Add console project called CallingOtherSystemsDemo.csproj to the solution
* Use NuGet to add a reference to the following package:
  + Newtonsoft.json

## Setup Steps for Visual Studio Code

From Command Prompt:

* Prepare solution by typing:

dotnet new solution --name connecting\_to\_other\_systems

* Create console project and add to solution by typing:

dotnet new console --output calling\_other\_systems-demo

dotnet sln add calling\_other\_systems-demo

* Install the following packages by typing:

dotnet add calling\_other\_systems-demo package newtonsoft.json

## Now do the following regardless of project type:

* Replace the Program.cs file’s content with the following:

using System.Net;

using System.Text;

using Newtonsoft.Json;

namespace CallingOtherSystemsDemo

{

internal class Program

{

static void Main(string[] args)

{

// Outdoor cinema weather check

// Create an HTTP client (should only have one of these

// for whole application)

var http = new HttpClient();

// Send a GET request and process the response as JSON,

// using C# dynamic objects

string city = "Leeds";

string url = $"https://weather-api.qaalabs.com/api/weather/{city}";

string json = http.GetStringAsync(url).Result;

dynamic obj = JsonConvert.DeserializeObject(json);

string temp = obj["TemperatureInCelsius"] >= 15 ? "warm" : "cold";

Console.WriteLine($"The weather for the outdoor cinema event " +

$"in {obj["City"]} is {obj["WeatherDescription"]} and it will be {temp}.");

// Send a POST request with movie blog data to an API,

// and collect the response

var data = new

{

title = "my movie blog post",

body = "Apollo 10.5 is a great movie. I'd rate it at 10 out of 10",

userId = 101

};

string dataJson = JsonConvert.SerializeObject(data);

url = "https://jsonplaceholder.typicode.com/posts";

HttpResponseMessage response = http.PostAsync(url,

new StringContent(dataJson, Encoding.UTF8, "application/json")).Result;

string responseJson = response.Content.ReadAsStringAsync().Result;

dynamic responseData = JsonConvert.DeserializeObject(responseJson);

Console.WriteLine($"New post has ID {responseData["id"]}");

Console.WriteLine($"New post has title text of {responseData["title"]}");

Console.WriteLine($"New post has body text of {responseData["body"]}");

// Sending headers with request, and extracting headers from response

var data = new

{

title = "My movie blog post - Up",

body = "Up is an uplifting film",

userId = 101

};

dataJson = JsonConvert.SerializeObject(data);

url = "https://jsonplaceholder.typicode.com/posts";

HttpContent content = new StringContent(

dataJson, Encoding.UTF8, "application/json");

content.Headers.Add("movie\_title", "Up");

content.Headers.Add("rating", "10");

response = http.PostAsync(url, content).Result;

responseJson = response.Content.ReadAsStringAsync().Result;

responseData = JsonConvert.DeserializeObject(responseJson);

Console.WriteLine($"New post has ID {responseData["id"]}");

Console.WriteLine($"New post has title text of {responseData["title"]}");

Console.WriteLine($"New post has body text of {responseData["body"]}");

// NOTE: the typicode site does not do anything

// with the additional header content and so

// it does not get returned in the repsonse

foreach (var header in response.Headers)

{

Console.WriteLine($"{header.Key} = {header.Value.First()}");

}

}

}

}

* Review the code and try to work out what it’s doing.
* Build and run the code and confirm it behaves as you expected.