CROSS DOMAIN AJAX

Browser security prevents cross domain AJAX requests.

AJAX requests can only be made within the same domain. Browser security prevents a web page from making AJAX requests to another domain.

This is referred to as the ‘same origin policy’. URL’s must be in the same domain for AJAX requests to go through.

Here I have set up a client at a different port than my Web API.

When I send an AJAX request in my test system I get the following message :

Failed to load http://localhost:56356/api/Employees_: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://localhost:54898' is therefore not allowed access.

Because the AJAX request is coming from a different port on localhost, I receive the message No ‘Access-Control-Allow-Origin’ and the request doesn’t go through.

There is a way to send AJAX requests across domains This involves using JSONP which stands for JSON with Padding. JSONP wraps the data in a JavaScript function allowing it to cross domains because JavaScript is allowed to function across domains. We are simply disguising it inside a JavaScript function.

For example, I have a record in JSON format

```
{“EmpID”:”100”,”EmpName”:”Bonnie”,” JobTitle”:”Programmer”}
```

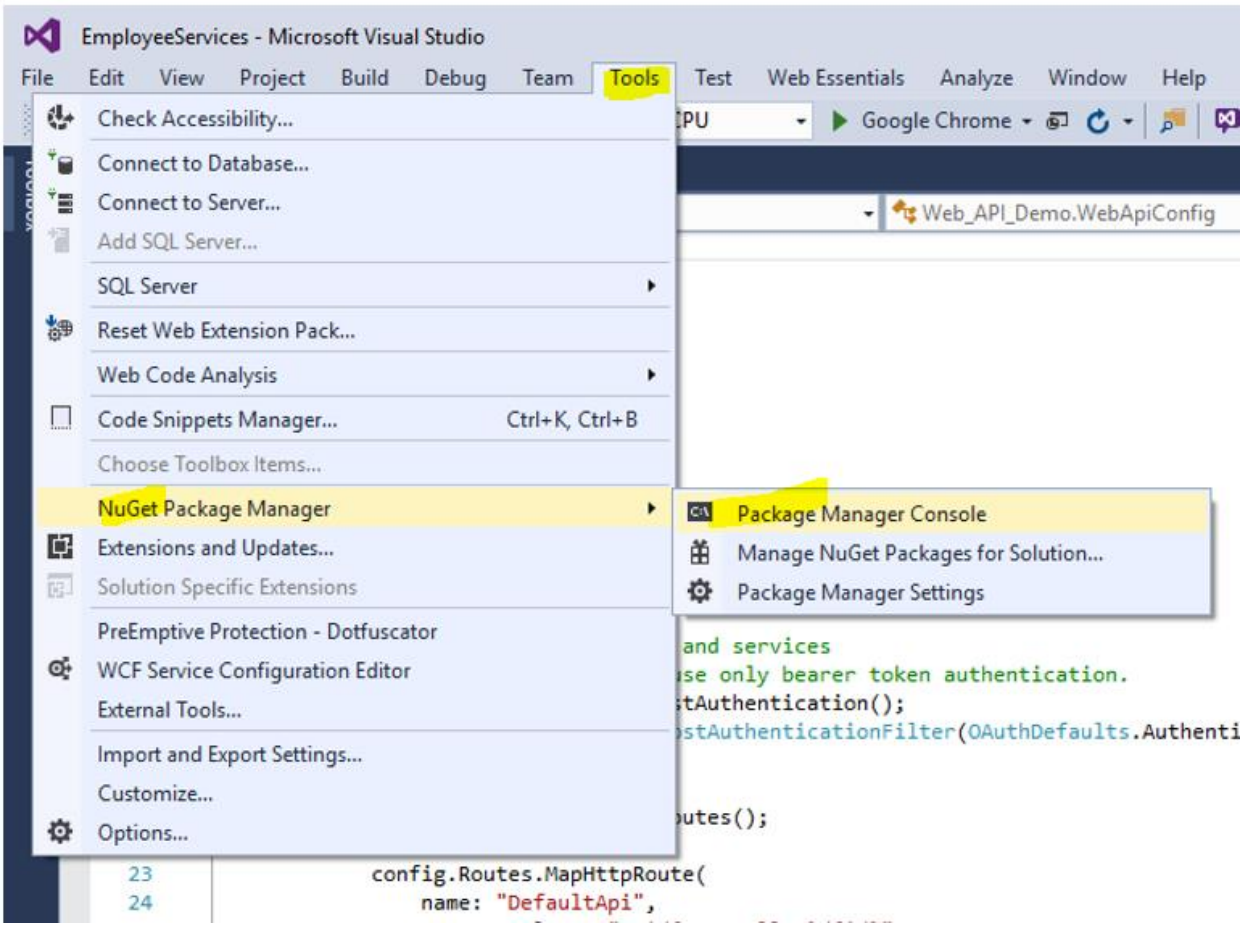
I will need to wrap the data in a JavaScript function:

```
CallbackFunction({“EmpID”:”100”,”EmpName”:”Bonnie”,” JobTitle”:”Programmer”})
```

Web browsers allow JavaScript to be consumed from a different domain, so this data can be consumed by a web page that is in a cross domain.

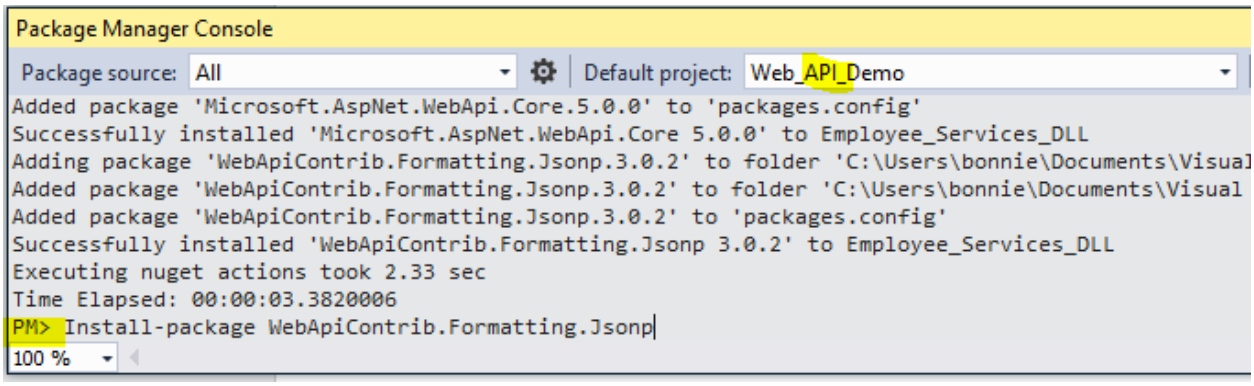
The support for JSONP format is in a separate package which needs to be installed **in the Web API** (containing controllers). Using Visual Studio 2015 I used the NUGET package manager.

Go to ‘Tools’ -> ‘NuGet Package Manager’ -> ‘Package Manager Console’

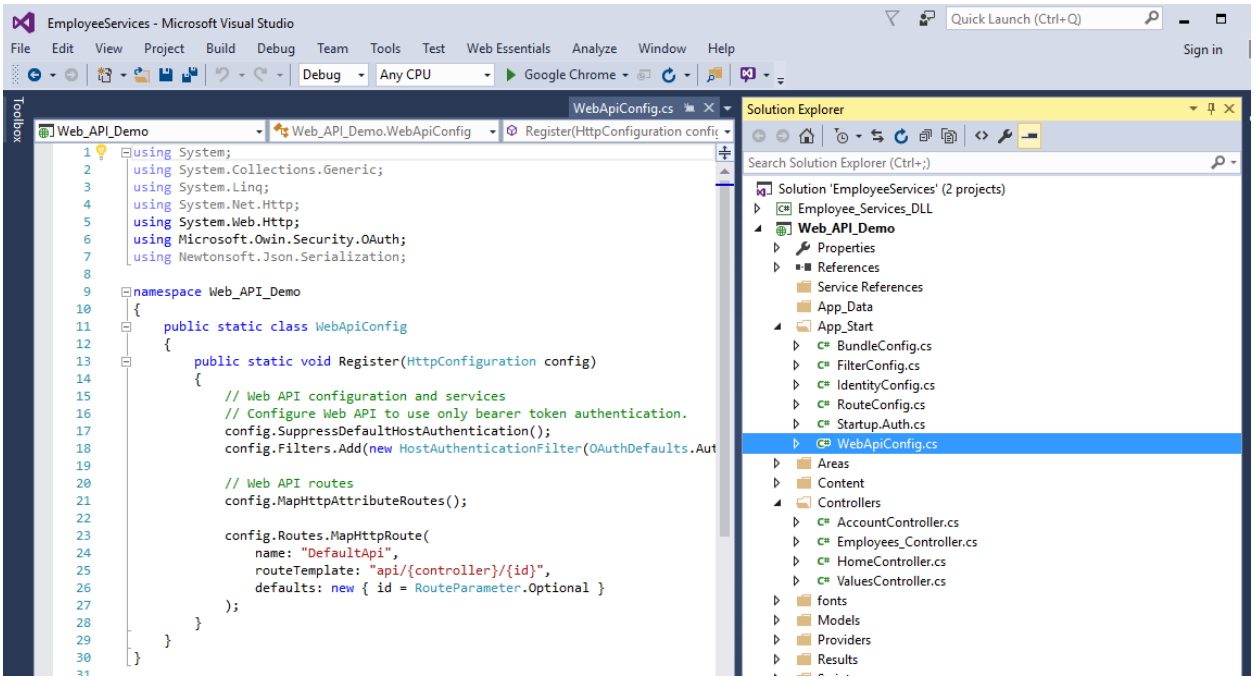


For the Web API, when the 'Package Manager' Window comes up enter :

> `Install-package WebApiContrib.Formatting.Jsonp`



Now go to WebAPIConfig.cs file in the App_Start folder:



In the Register create a new instance of `JsonMediaTypeFormatter` which will require the use of namespace `WebApiContrib.Formatting.Jsonp` as for example see below.

```
using WebApiContrib.Formatting.Jsonp;
```

In the Register method, to the constructor of JsonMediaTypeFormatter I will pass a JSON formatter from the config object that is being passed to the class i.e. `public static void Register(HttpConfiguration config)`

```
var jsonpFormatter = new JsonMediaTypeFormatter(config.Formatters.JsonFormatter);
```

Inject the jsonpFormatter into the Formatters collection of the config object.

Insert this formatter as the first formatter into the config object. Insert at position 0 to make it the first formatter that is used.

```
config.Formatters.Insert(0, jsonpFormatter);
```

In the HTML page that is using the JSON call across domains set the dateType to jsonp and not json.

```
var cFJF = config.Formatters.JsonFormatter;
var jsonpFormatter = new JsonpMediaTypeFormatter(cFJF);
config.Formatters.Insert(0, jsonpFormatter);
```

To be added – invoking this in AJAX on web page as jsonp

Cors or Cross Origin Resource Method

Using the NuGet PM add the following: `PM> Install-Package Microsoft.AspNet.WebApi.Cors`

And install the Cors package.

Use the namespace `using System.Web.Http.Cors;`

IN PROGRESS….