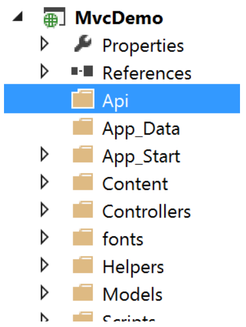
ASP.Net Web API: Lab 1 – Getting Started With WebApi

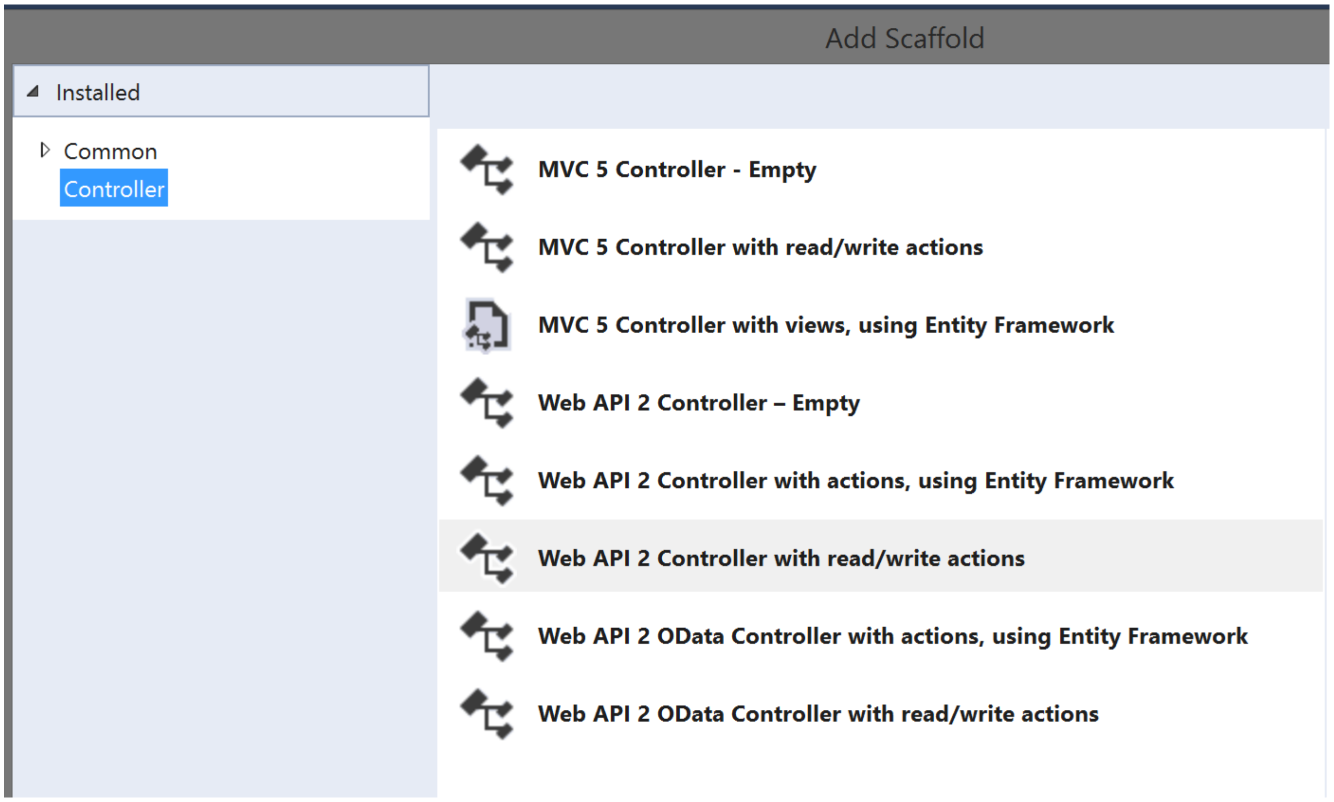
Lab Setup: Open the MVCDemo solution file in the Lab-4-Exercise-1-Begin

In this lab we're going to create a simple ActionFilter to demonstrate how they work and how to implement them.

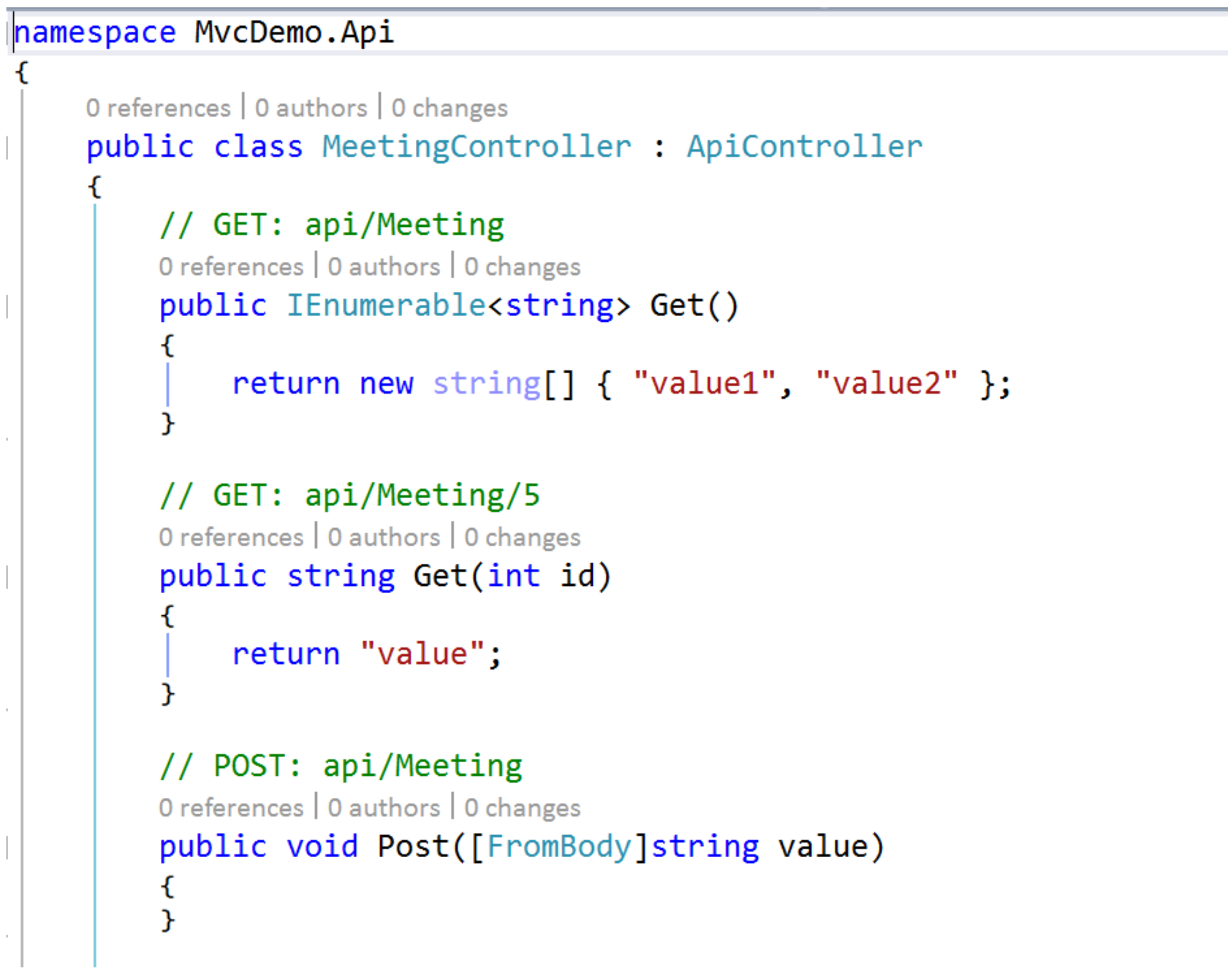
1. Create a new folder and name it Api



2. Next we want to add a Web Api 2 Controller. To do this right-click on Api and select Add-> Controller and select "Web API 2 Controller with read/write actions" and name it MeetingController


Add Scaffold
Installed
Common
Controller
MVC 5 Controller - Empty
MVC 5 Controller with read/write actions
MVC 5 Controller with views, using Entity Framework
Web API 2 Controller – Empty
Web API 2 Controller with actions, using Entity Framework
Web API 2 Controller with read/write actions
Web API 2 OData Controller with actions, using Entity Framework
Web API 2 OData Controller with read/write actions

This will create a new controller with a number of methods.



```
namespace MvcDemo.Api
{
    0 references | 0 authors | 0 changes
    public class MeetingController : ApiController
    {
        // GET: api/Meeting
        0 references | 0 authors | 0 changes
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        // GET: api/Meeting/5
        0 references | 0 authors | 0 changes
        public string Get(int id)
        {
            return "value";
        }

        // POST: api/Meeting
        0 references | 0 authors | 0 changes
        public void Post([FromBody]string value)
        {
        }
```

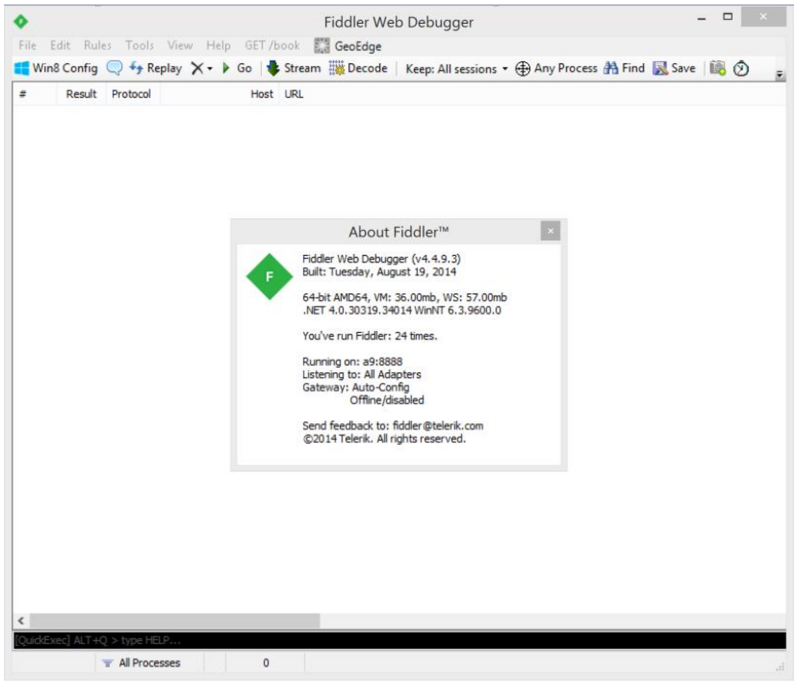
You will notice that similar to when we created our other MVC controllers a number of methods have been created for us. In this case however their names reflect http verbs. (Get, Post etc.)

3. Next we will need to update Global.asax to register the Api routing

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

namespace MvcDemo
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            GlobalConfiguration.Configure(WebApiConfig.Register);
            AreaRegistration.RegisterAllAreas();
            RouteConfig.RegisterRoutes(RouteTable.Routes);
        }
    }
}
```

4. Let’s take a look at how the API responds using a http request tool such as Fiddler. (Other options include PostMan for Chrome)


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Win8 Config Replay Go Stream Decode Keep: All sessions Any Process Find Save
# Result Protocol Host URL
[QuickExec] ALT+Q > type HELP...
All Processes 0

About Fiddler™

Fiddler Web Debugger (v4.4.9.3)
Built: Tuesday, August 19, 2014

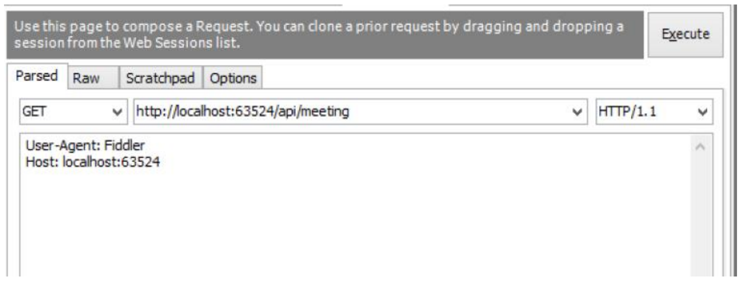
64-bit AMD64, VM: 36.00mb, WS: 57.00mb
.NET 4.0.30319.34014 WinNT 6.3.9600.0

You've run Fiddler: 24 times.

Running on: a9:8888
Listening to: All Adapters
Gateway: Auto-Config
Offline/disabled

Send feedback to: fiddler@telerik.com
©2014 Telerik. All rights reserved.

Using the composer we can make http requests against our service


Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

GET http://localhost:63524/api/meeting HTTP/1.1

```
User-Agent: Fiddler
Host: localhost:63524
```

And see the results that are returned:

5. To make things more interesting, let’s update our controller to return some meeting data:

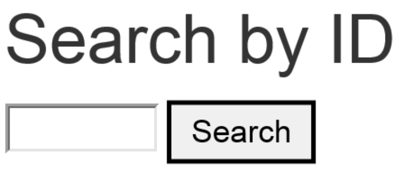
```
// GET: api/Meeting
public IEnumerable<Meeting> Get()
{
    var meetingRepository = new MockMeetingRepository();
    List<Meeting> meetings = meetingRepository.GetAll();
    return meetings;
}

// GET: api/Meeting/5
public Meeting Get(int id)
{
    var meetingRepository = new MockMeetingRepository();
    Meeting meeting = meetingRepository.Get(id);
    return meeting;
}
```

6. The next step is to setup some client side code (Javascript in the browser) to access this data.First add some the following HTML to our Meetings/Index view above the first <h2> tag:

```
<div>
    <h2>Search by ID</h2>
    <input type="text" id="meetingId" size="5" />
    <input type="button" value="Search" onclick="find();" />
    <p id="meeting" />
</div>
<div>
    <h2>All Meetings</h2>
    <ul id="meetings" />
</div>
```

This will give us a text box that we can use to filter data from the user.



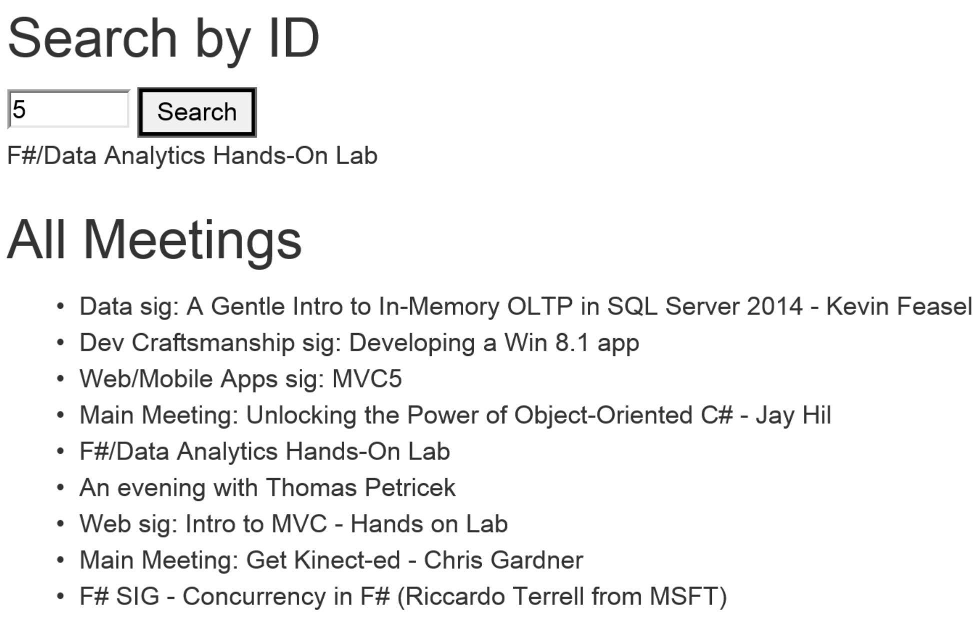
7. Next add the following JavaScript code to the very bottom of the view.

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-2.0.3.min.js"></script>
<script>
    var uri = 'api/meeting';

    $(document).ready(function () {
        // Send an AJAX request
        $.getJSON(uri)
            .done(function (data) {
                // On success, 'data' contains a list of meetings.
                $.each(data, function (key, item) {
                    // Add a list item for the meeting.
                    $('<li>', { text: item.Title }).appendTo($('#meetings'));
                });
            });
    });


    function find() {
        var id = $('#meetingId').val();
        $.getJSON(uri + '/' + id)
            .done(function (data) {
                $('#meeting').text(data.Title);
            })
            .fail(function (jqXHR, textStatus, err) {
                $('#product').text('Error: ' + err);
            });
        }
</script>
```

The result displays a list from the API of all of the meetings and give you the ability to enter the id of a single meeting and see the results.



Search by ID

F#/Data Analytics Hands-On Lab

All Meetings

- Data sig: A Gentle Intro to In-Memory OLTP in SQL Server 2014 - Kevin Feasel
- Dev Craftsmanship sig: Developing a Win 8.1 app
- Web/Mobile Apps sig: MVC5
- Main Meeting: Unlocking the Power of Object-Oriented C# - Jay Hil
- F#/Data Analytics Hands-On Lab
- An evening with Thomas Petricek
- Web sig: Intro to MVC - Hands on Lab
- Main Meeting: Get Kinect-ed - Chris Gardner
- F# SIG - Concurrency in F# (Riccardo Terrell from MSFT)