

ASP.NET MVC - Web API

Advertisements

[⬅ Previous Page](#)[Next Page ➡](#)

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building RESTful applications on the .NET Framework.

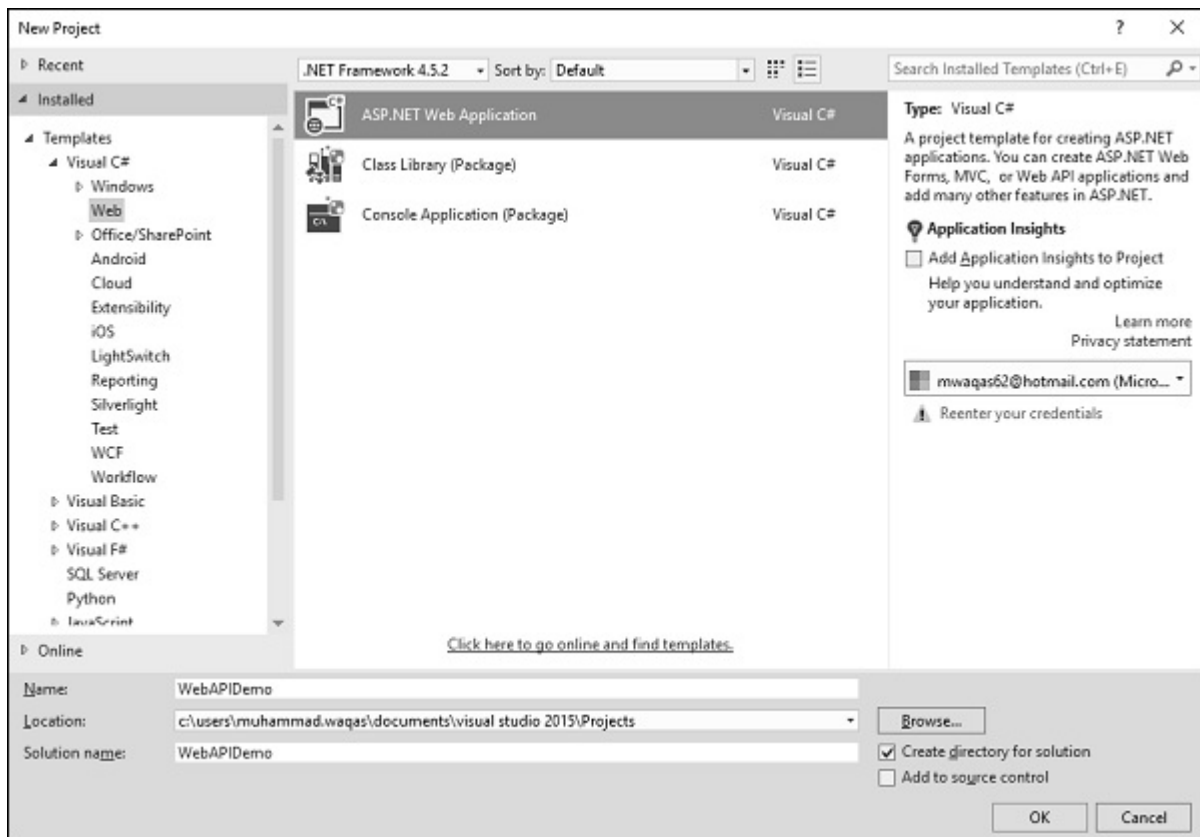
When you're building APIs on the Web, there are several ways you can build APIs on the Web. These include HTTP/RPC, and what this means is using HTTP in Remote Procedure Call to call into things, like Methods, across the Web.

The verbs themselves are included in the APIs, like Get Customers, Insert Invoice, Delete Customer, and that each of these endpoints end up being a separate URI.

Let's take a look at a simple example of Web API by creating a new ASP.NET Web Application.

Step 1 – Open the Visual Studio and click File → New → Project menu option.

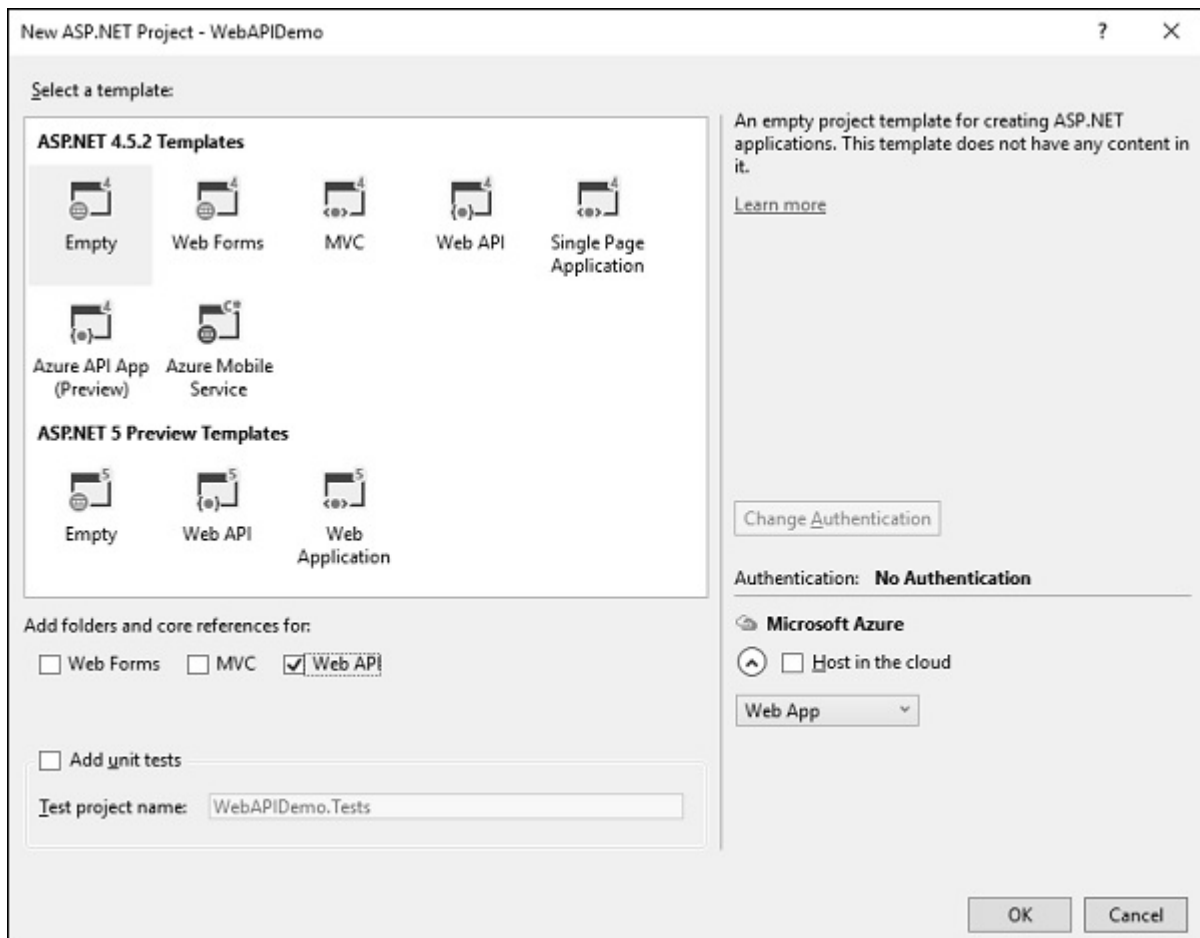
A new Project dialog opens.



Step 2 – From the left pane, select Templates → Visual C# → Web.

Step 3 – In the middle pane, select ASP.NET Web Application

Enter project name WebAPIDemo in the Name field and click Ok to continue. You will see the following dialog, which asks you to set the initial content for the ASP.NET project.

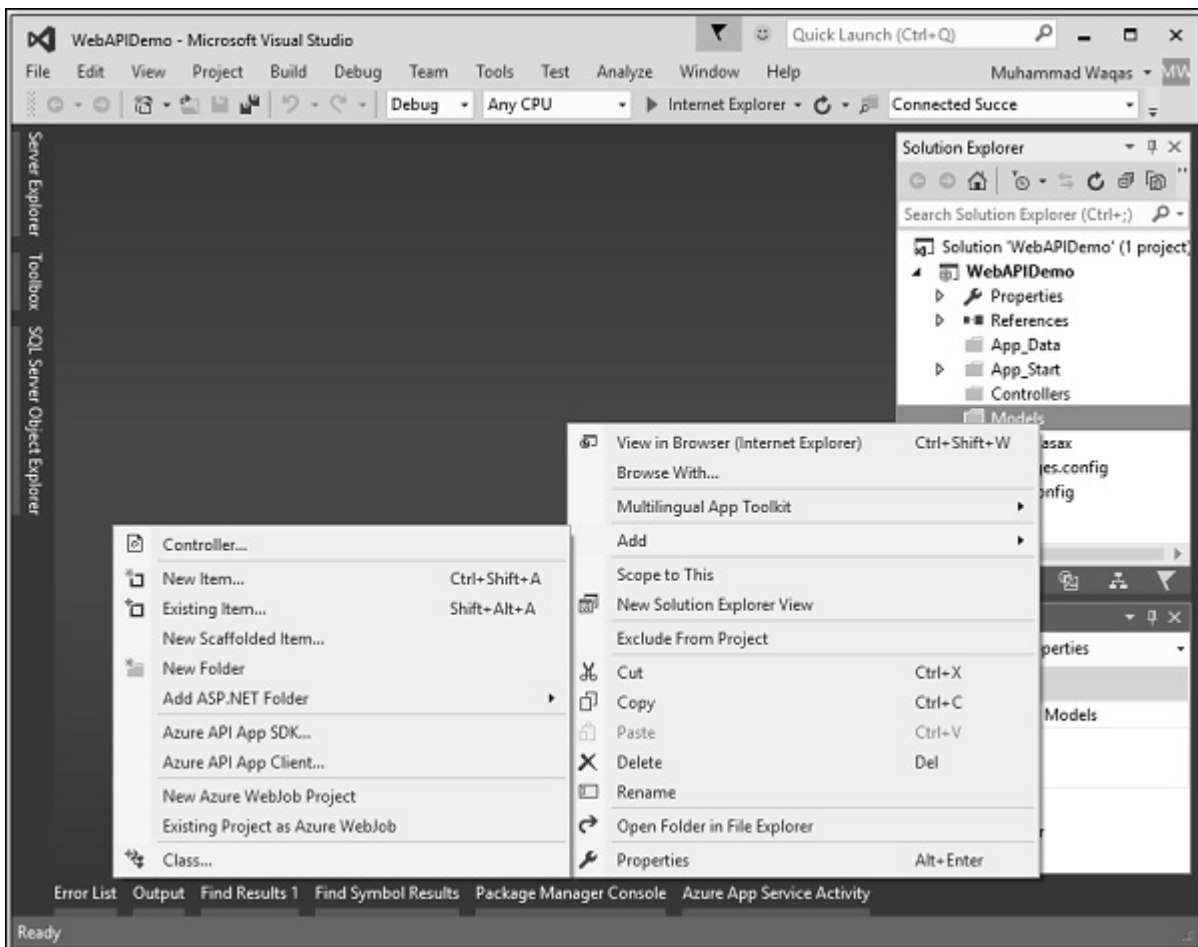


Step 4 – To keep things simple, select the Empty option and check the Web API checkbox in the 'Add folders and core references for' section and click Ok.

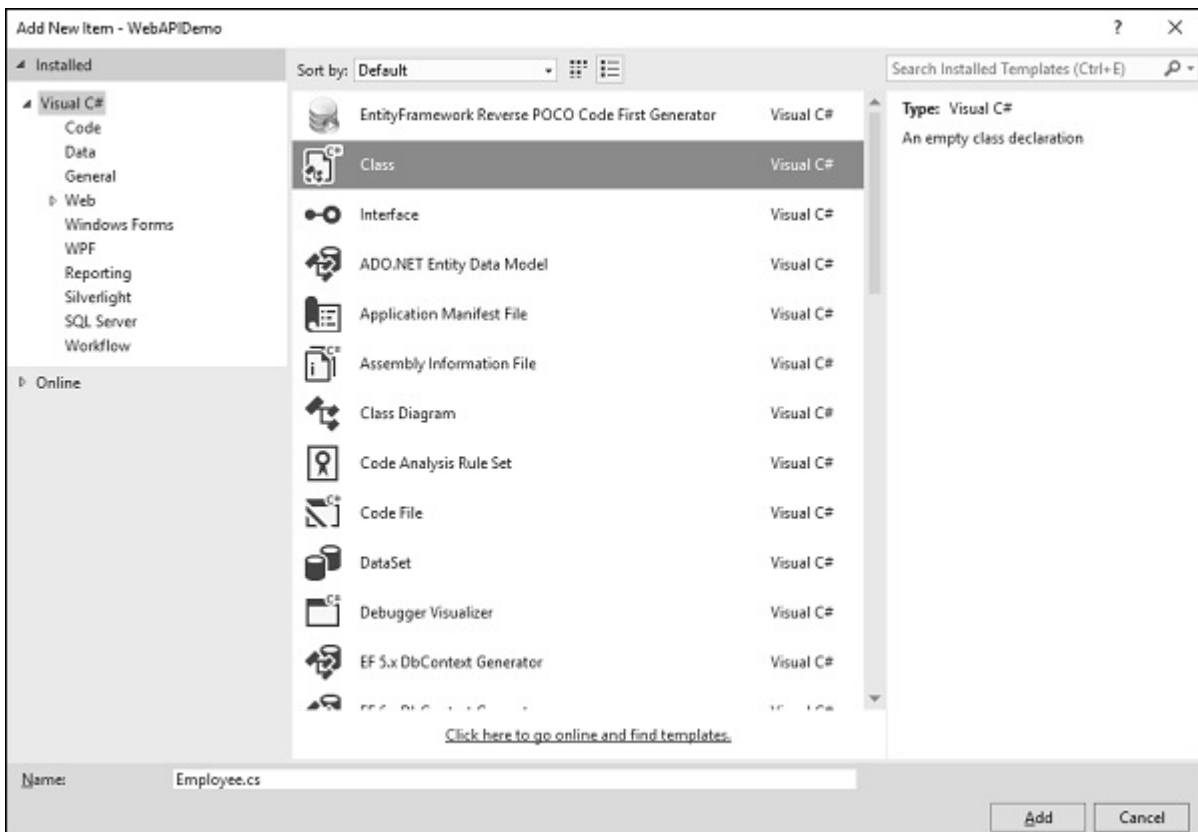
Step 5 – It will create a basic MVC project with minimal predefined content.

Once the project is created by Visual Studio, you will see a number of files and folders displayed in the Solution Explorer window.

Step 6 – Now we need to add a model. Right-click on the Models folder in the solution explorer and select Add → Class.



You will now see the Add New Item dialog.



Step 7 – Select Class in the middle pan and enter Employee.cs in the name field.

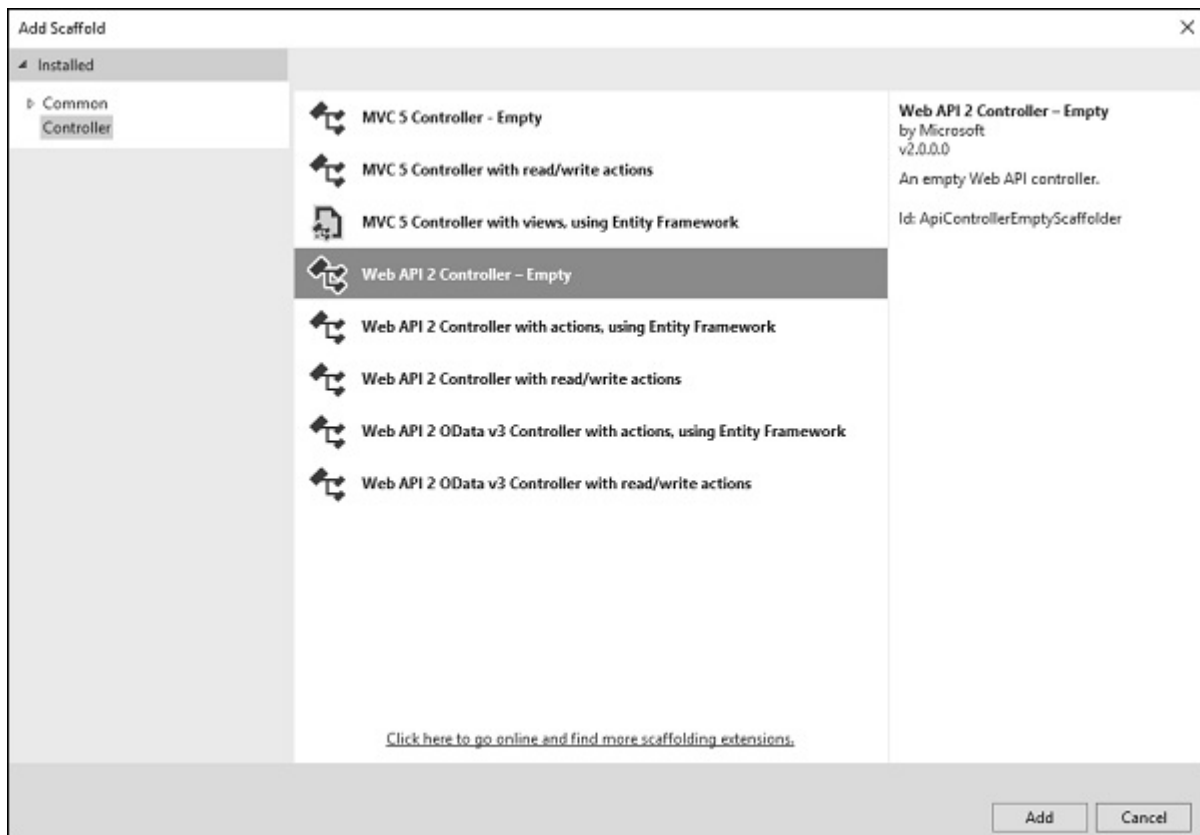
Step 8 – Add some properties to Employee class using the following code.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebAPIDemo.Models {
    public class Employee{
        public int ID { get; set; }
        public string Name { get; set; }
        public DateTime JoiningDate { get; set; }
        public int Age { get; set; }
    }
}
```

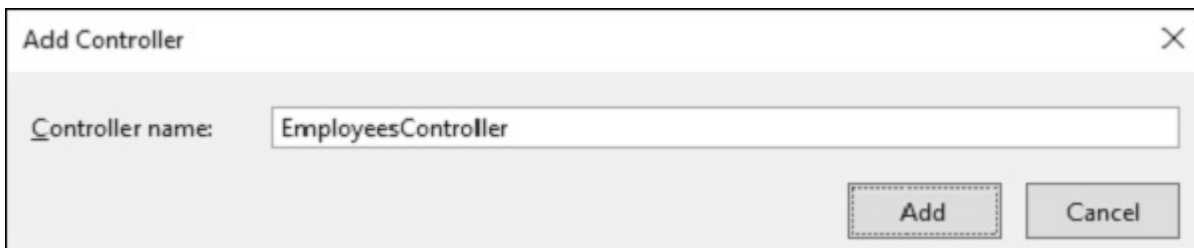
Step 9 – Let's add the controller. Right-click on the controller folder in the solution explorer and select Add → Controller.

It will display the Add Scaffold dialog.



Step 10 – Select the Web API 2 Controller - Empty option. This template will create an Index method with default action for controller.

Step 11 – Click 'Add' button and the Add Controller dialog will appear.

A screenshot of the 'Add Controller' dialog box in Visual Studio. The dialog has a title bar with a close button. Inside, there is a label 'Controller name:' followed by a text box containing 'EmployeesController'. At the bottom right, there are two buttons: 'Add' and 'Cancel'.

Step 12 – Set the name to EmployeesController and click 'Add' button.

You will see a new C# file 'EmployeeController.cs' in the Controllers folder, which is open for editing in Visual Studio with some default actions.

```
using System;
using System.Collections.Generic;
using System.Linq;

using System.Web.Http;
using WebAPIDemo.Models;

namespace WebAPIDemo.Controllers{
    public class EmployeesController : ApiController{
        Employee[] employees = new Employee[]{
            new Employee { ID = 1, Name = "Mark", JoiningDate =
                DateTime.Parse(DateTime.Today.ToString()), Age = 30 },
            new Employee { ID = 2, Name = "Allan", JoiningDate =
                DateTime.Parse(DateTime.Today.ToString()), Age = 35 },
            new Employee { ID = 3, Name = "Johny", JoiningDate =
                DateTime.Parse(DateTime.Today.ToString()), Age = 21 }
        };

        public IEnumerable<Employee> GetAllEmployees(){
            return employees;
        }

        public IHttpActionResult GetEmployee(int id){
            var employee = employees.FirstOrDefault((p) => p.ID == id);
            if (employee == null){
                return NotFound();
            }
            return Ok(employee);
        }
    }
}
```

Step 13 – Run this application and specify /api/employees/ at the end of the URL and press 'Enter'. You will see the following output.



Step 14 – Let us specify the following URL **http://localhost:63457/api/employees/1** and you will see the following output.



⏪ Previous Page

Next Page ⏩

Advertisements



[Privacy Policy](#) [Cookies Policy](#) [Contact](#)

© Copyright 2019. All Rights Reserved.