

## WCF or ASP.NET Web API

Below points can help you to choose WCF or ASP.NET Web API

- If you want to create resource based services and need to take full advantage of HTTP like cache control for browsers, transfer different kind of content types like documents, images, HTML pages **ASP.NET Web API** should be selected.
- If you have heterogeneous clients for same service and needs to call service using different protocols like netTCPBinding, netNamedPipeBinding, wsHttpBinding **WCF** is the obvious choice.
- If you have special requirements like CallbackContract, One way communication or queuing then **WCF** should be selected.

We will read the data from [Customers.xml](#) file and show it on web page using jQuery.

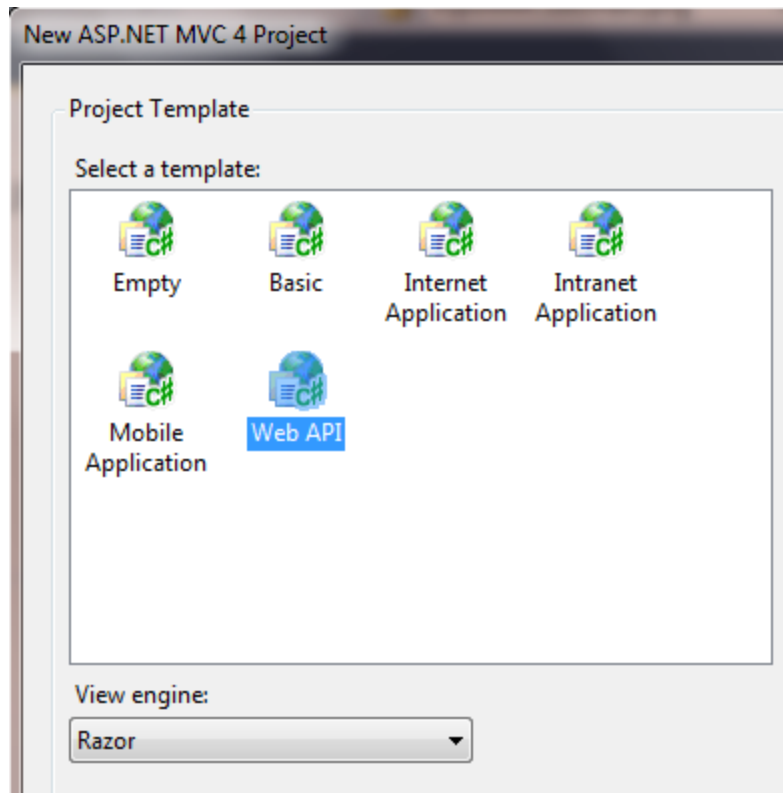
## Implementation of ASP.NET Web API in MVC

### Create new ASP.NET Web API

Open visual studio and click on **File -> New Project -> Select ASP.NET MVC4 Web Application**

Name the application as **NorthwindWebApi**

From next window select **Web API** template and **Razor** as ViewEngine.



## Customer Model

In ASP.NET MVC, Model manages properties and data of application and it contains all application logic like business logic, validation logic or data access logic. Model binders helps controller to keep its code cleanly separated from actual request and its environment. See more on [ASP.NET MVC Application Request life cycle](#)

ASP.NET web api will produce serialized data of model to JSON, XML or some other format. It also writes HTTP Response messages with serialized data. Client can use **GET** verb to get data in specific format like JSON or XML by setting **Accept** header HTTP Request message.

Create a new class in **Model** folder. Name it as **Customer**

Customer Model will represent Customer entity from datastore. Add below properties to Model.

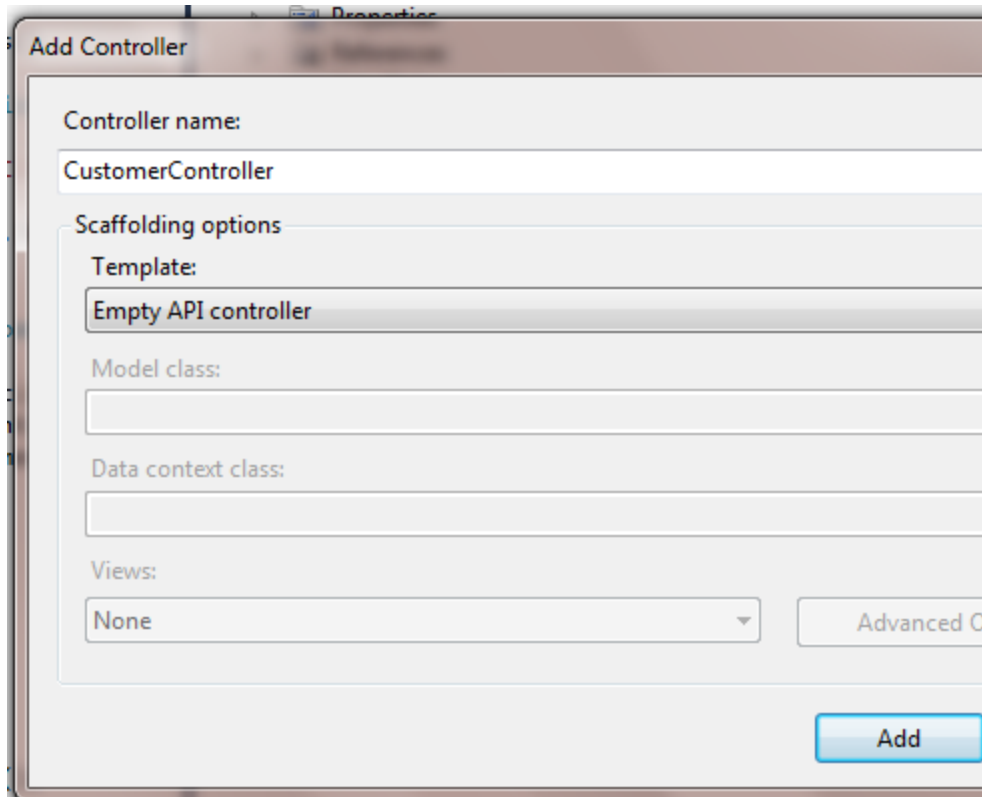
```
namespace NorthwindWebApi.Models
{
    public class Customer
    {
        public string CustomerID { get; set; }
        public string CompanyName { get; set; }
        public string City { get; set; }
    }
}
```

## Customer ApiController

**The controllers in Web API inherits from ApiController class instead of Controller. The major difference between them is, actions from ApiController returns data and not Views. [See difference between ApiController and Controller.](#)**

Add Customer ApiController by right click on **Controllers** folder from Solution Explorer. Select **Add -> Controller**. Name it as **CustomerController**

From next window select **Empty Api Controller**.



Add below code to **CustomerController**. It has two methods **GetAllCustomers** which returns all the customers from Customer.xml and **GetCustomer** which takes **id** as input parameter and return its customer details.

```
using NorthwindWebApi.Models;
using System.Xml.Linq;
using System.Collections;

namespace NorthwindWebApi.Controllers
{
    public class CustomerController : ApiController
    {
        public List<customer> GetAllCustomers()
        {
            List<Customer> customers = new List<Customer>();
```

```

XDocument doc = XDocument.Load("C:\\Customers.xml");

foreach (XElement element in doc.Descendants("DocumentElement")
    .Descendants("Customer"))
{
    Customer customer = new Customer();

    customer.CustomerID = element.Element("CustomerID").Value;
    customer.CompanyName = element.Element("CompanyName").Value;
    customer.City = element.Element("City").Value;

    customers.Add(customer);
}

return customers;
}

public Customer GetCustomer(int id)
{
    Customer customer = new Customer();

    XDocument doc = XDocument.Load("C:\\Customers.xml");

    XElement element = doc.Element("DocumentElement")
        .Elements("Customer").Elements("CustomerID").
        SingleOrDefault(x => x.Value == id.ToString());

    if (element != null)
    {
        XElement parent = element.Parent;

        customer.CustomerID =
            parent.Element("CustomerID").Value;
    }
}

```

```

        customer.CompanyName =
            parent.Element("CompanyName").Value;
        customer.City =
            parent.Element("City").Value;

        return customer;
    }
    else
    {
        throw new HttpResponseException
            (new HttpResponseMessage(HttpStatusCode.NotFound));
    }
}
}
}

```

## Customer View

As Web API does not return View or ActionResult, for this tutorial you can use default **Index.cshtml** under **Views -> Home** folder.

Replace html from **Index.cshtml** with below code which shows all customers by retrieving JSON through jQuery call to Customer Web API.

```

<div id="body">
<section class="content-wrapper main-content clear-fix">
    <div>
        <h2>All Customers</h2>
    </div>
    <div>
        <table id="customers" style="border: 1px solid black;" ></table>
    </div>
</section>
</div>

```

```

    </div>

</section>
</div>

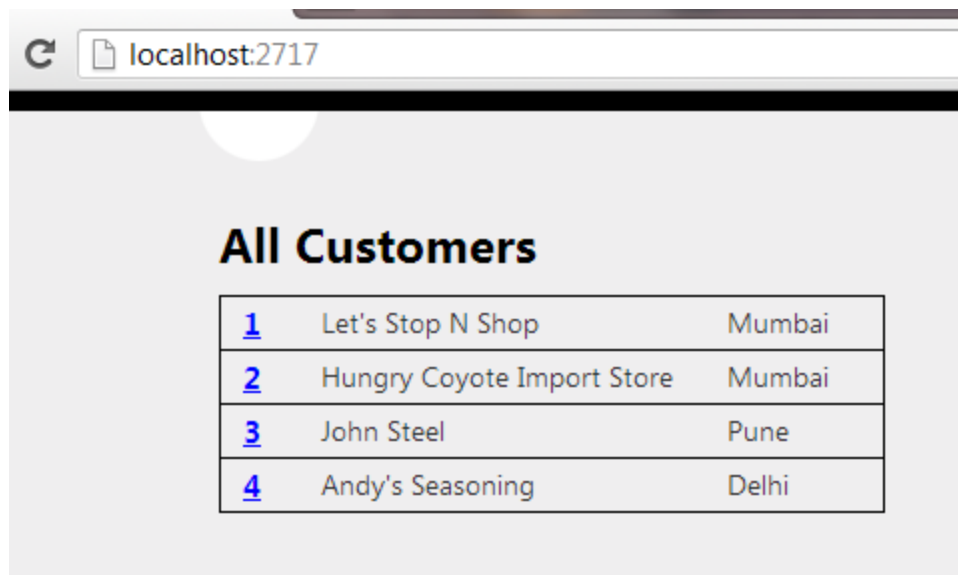
@section scripts {
<script>

    var apiCustomers = 'api/customer';
    $(document).ready(function () {
        // Send an AJAX request
        var rows = '';
        $('#customers').empty();

        $.getJSON(apiCustomers,
            function (data) {
                $.each(data, function (key, val) {
                    var customerURL = apiCustomers + '/' + val.CustomerID;
                    rows = rows + '<tr style="border: 1px solid black;"><td> ' +
                        '<a style="color: Blue; font-weight:bold; font-size:15px"' +
                            'href="' + customerURL + '">' + val.CustomerID + '</a>' +
                            '</td><td>' + val.CompanyName +
                            '</td><td>' + val.City + '</td></tr>';
                });
                $('#customers').html(rows);
            });
    });
</script>
}

```

It shows all the customers and link to each customer.



A screenshot of a web browser window with the address bar showing 'localhost:2717'. The page has a light gray background and a dark header bar. Below the header, the title 'All Customers' is displayed in bold black text. Underneath the title is a table with 4 rows and 3 columns. The first column contains blue underlined links numbered 1 to 4. The second column contains the company names, and the third column contains the cities.

<a href="#">1</a>	Let's Stop N Shop	Mumbai
<a href="#">2</a>	Hungry Coyote Import Store	Mumbai
<a href="#">3</a>	John Steel	Pune
<a href="#">4</a>	Andy's Seasoning	Delhi

Now click on CustomerID link to see Customer details.



A screenshot of a web browser window with the address bar showing 'localhost:2717/api/customer/1'. The page content shows a message: 'This XML file does not appear to have any style information associated with it. The'. Below this message is a collapsed XML view icon (a small triangle) followed by the XML code for CustomerID 1.

```

▼<Customer xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
  <City>Mumbai</City>
  <CompanyName>Let's Stop N Shop</CompanyName>
  <CustomerID>1</CustomerID>
</Customer>

```

As we haven't implemented any HTML view for customer details it shows customer details in browser. Depending on your browser it shows the customer details Chrome and Firefox will show xml whereas Internet Explorer will show it in JSON. It is because the request attribute **Accept-headers** send by browsers are different.