



Become a member

Login

Post ▼

Ask Question



Sandeep Singh Shekhawat



Updated date, May 13, 2019



490.3k



7



4

[Download Free .NET & JAVA Files API](#)[Try Free File Format APIs for Word/Excel/PDF](#)

Introduction

This article explains how to render a partial view and JSON data using AJAX. I have divided this article into three sections to understand both concepts, the first section describes the basic code and structure that is common in both concepts, the second section describes how to render a partial view using AJAX and the last section describes how to render JSON data on a web using AJAX.

I will explain these concepts with a simple example. The example is that books are showing in a web depending on publisher. I choose a publisher from a dropdown list then the books information is shown in the web page depending on publisher. So let's see this example in detail.

Getting Started



The ADO.NET Entity Model is an Object-Relational Mapping (ORM) that creates a higher-level abstraction of the database. This ADO.NET Entity Model is mapped with "Development" database so that it inherits the DbContext class.

Post ▼

Ask Question

This "Development" database has two tables, one is the Publisher table and the other is the BOOK table. Both tables have 1-to-many relationships, in other words one publisher can publish multiple books but each book is associated with one publisher. If you want to learn more about this application database design then please check my previous article "[An MVC Application with LINQ to SQL](#)".

The ADO.NET Entity Model is mapped to both tables. The connection string for this has the same name as the context class name and this connection string is created in the web.config file. You can change the name of the connection string. The context class name and connection string name is just a convention, not a configuration, so you can change it with a meaningful name. The following Figure 1.1 shows the ADO.NET Entity Model mapping with both tables.

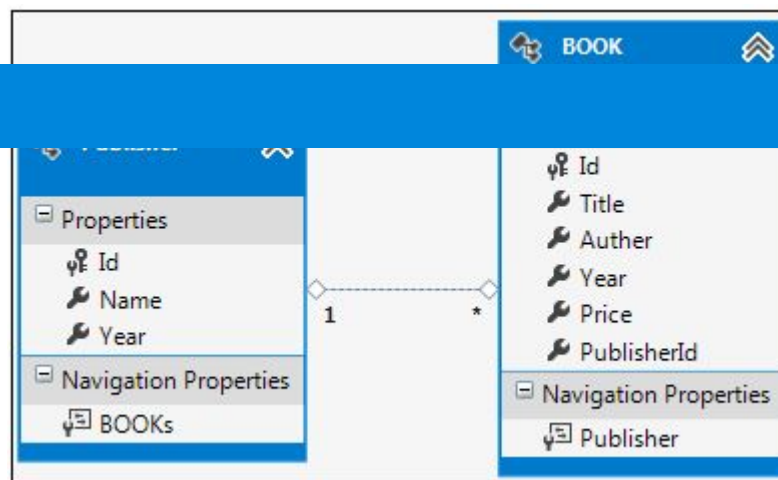


Figure 1.1 The ADO.NET Entity Model mapping with Publisher and Book tables.

Now the ADO.NET Entity Model is ready for the application and it's time to move on the next step of the application that is the model design so let's see the application's model.

Model Design

[become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

as the same therefore I need to create two models, one for the publisher and the other for the book.

I create a publisher model (Publisher.cs) under the Model folder that has a publisher id and publisher list as in the following code snippet.

```
01. using System.Collections.Generic;
02. using System.ComponentModel.DataAnnotations;
03. using System.Web.Mvc;
04.
05. namespace JsonRenderingMvcApplication.Models
06. {
07.     public class PublisherModel
08.     {
09.         public PublisherModel()
10.         {
11.             PublisherList = new List<SelectListItem>();
12.         }
13.
14.
15.
16.         public IEnumerable<SelectListItem> PublisherList { get; set; }
17.     }
18. }
```

After the publisher model, I create a book model (Book.cs) in the same folder. That has basic properties related to a book. The following is a code snippet for the book model.

```
01. namespace JsonRenderingMvcApplication.Models
02. {
03.     public class BookModel
04.     {
05.         public string Title { get; set; }
06.         public string Author { get; set; }
07.         public string Year { get; set; }
08.         public decimal Price { get; set; }
09.     }
```



[Become a member](#)[Login](#)

Now the models are ready for use so now to move on to the controller.

[Post ▼](#)[Ask Question](#)

Controller Design

I create two controllers, one for publisher that shows a publisher's list in a dropdown list and another is a book controller that shows book details depending on publisher. The publisher controller defines a single action method that is the same in both concepts; rendering a partial view and JSON data while the book controller defines two action methods, one for partial view rendering and another for JSON data rendering so you will look it later in this article.

Now create a publisher controller (PublisherController.cs) under the Controllers folder as per the MVC convention. This control has a single action method to show the publisher list on the view. The following is a code snippet for the publisher controller.

```
01. using System.Collections.Generic;
02. using System.Linq;
03. using System.Web.Mvc;
04. using JsonRenderingMvcApplication.Models;

07. {
08.     public class PublisherController : Controller
09.     {
10.         public ActionResult Index()
11.         {
12.             PublisherModel model = new PublisherModel();
13.             using (DAL.DevelopmentEntities context = new DAL.DevelopmentEntities())
14.             {
15.                 List<DAL.Publisher> PublisherList = context.Publishers.ToList();
16.                 model.PublisherList = PublisherList.Select(x =>
17.                     new SelectListItem()
18.                     {
19.                         Text = x.Name,
20.                         Value = x.Id.ToString()
21.                     });
22.             }
23.             return View(model);
```



Now create book a controller (BookController.cs) under the Controllers folder of the application and leave it empty without any action method, you will define two action methods in this controller, one for partial view rendering and another for JSON data rendering. Now the article's first section is completed as defined in the introduction and now to move both approaches one by one.

Rendering a partial view

When making AJAX requests, it is very simple to return HTML content as the result. Simply return an ActionResult using the PartialView method that will return rendered HTML to the calling JavaScript.

Now define an action method in the book controller that returns an ActionResult using the PartialView. This action method retrieves a list of books depending on publisher id that passes as a parameter in this action method. The following is a code snippet for this action method.

```
02. {
03.     IEnumerable<BookModel> modellist = new List<BookModel>();
04.     using (DAL.DevelopmentEntities context = new DAL.DevelopmentEntities())
05.     {
06.         var books = context.BOOKs.Where(x => x.PublisherId == id).ToList();
07.         modellist = books.Select(x =>
08.             new BookModel()
09.             {
10.                 Title = x.Title,
11.                 Author = x.Auther,
12.                 Year = x.Year,
13.                 Price = x.Price
14.             });
15.     }
16.     return PartialView(modellist);
17. }
```



```
01. routes.MapRoute("BookByPublisher",
02.     "book/bookbypublisher/",
03.     new { controller = "Book", action = "BookByPublisher" },
04.     new[] { "JsonRenderingMvcApplication.Controllers" });
```


Post ▼


Ask Question

Now create a view for the publisher that has a dropdown list for the publisher and shows the book details depending on the value selected in the publisher dropdown list using AJAX. This view is an index (Views/Publisher/Index.cshtml). The following is a code snippet for the Index view.

```
01. @model JsonRenderingMvcApplication.Models.PublisherModel
02. <script src="~/Scripts/jquery-1.7.1.min.js"></script>
03. <script type="text/javascript">
04.
05.     $(document).ready(function ()
06.     {
07.         $("#Id").change(function ()
08.         {
09.             var id = $("#Id").val();
10.
11.             $.ajax({
12.                 cache: false,
13.                 type: "GET",
14.                 url: "@(Url.RouteUrl("BookByPublisher"))",
15.                 data: { "id": id },
16.                 success: function (data)
17.                 {
18.                     booksDiv.html('');
19.                     booksDiv.html(data);
20.                 },
21.                 error: function (xhr, ajaxOptions, thrownError)
22.                 {
23.                     alert('Failed to retrieve books.');
```



 C# Corner

 Become a member

Login

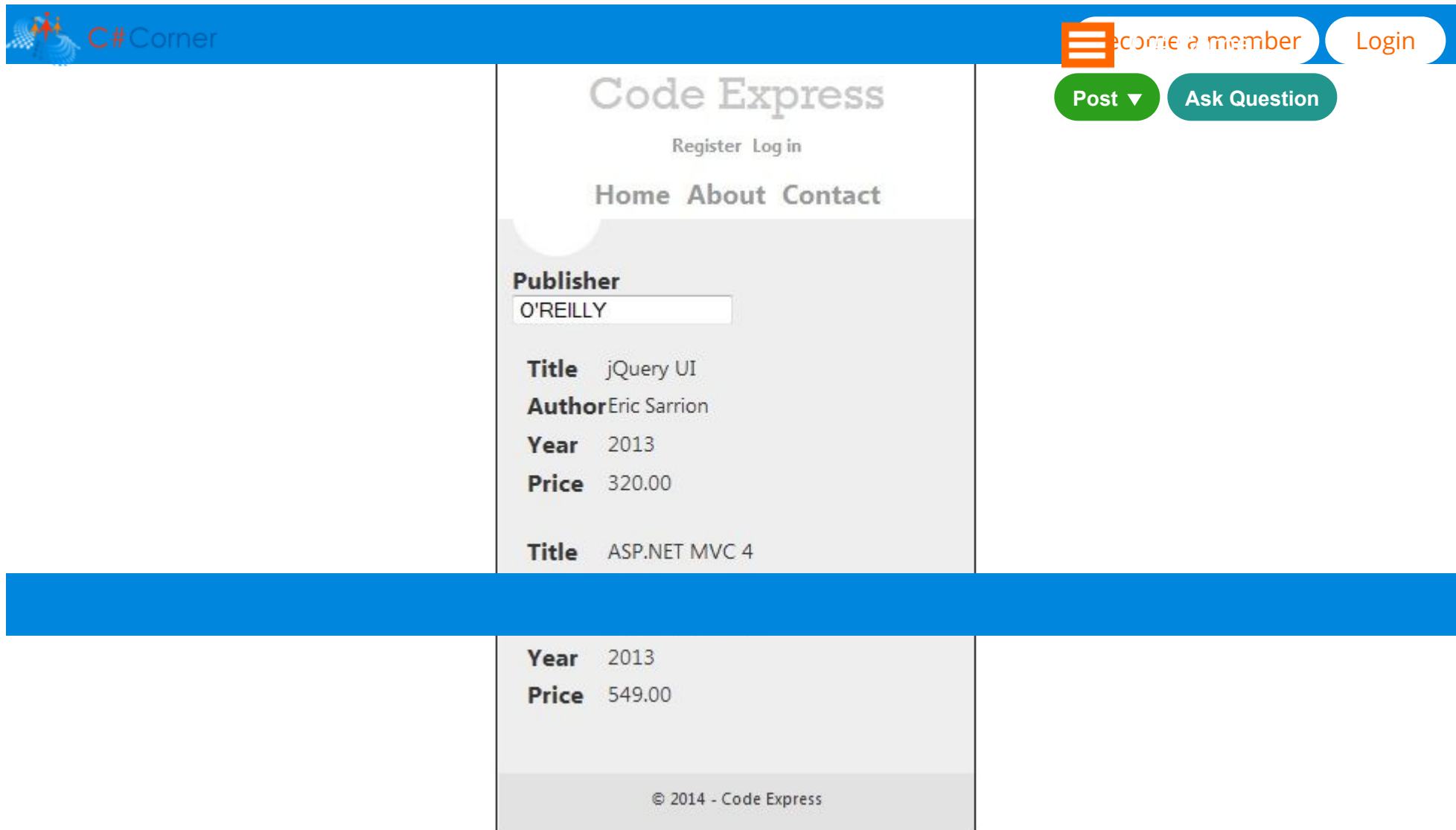
Post ▼

Ask Question

```
31.         @Html.DropDownListFor(model => model.Id, Model.PublisherList)
32.     </div>
33.     <div id="booksDiv">
34. </div>
```

Run the application and choose an item from the publisher dropdown list. You will then get the result as in Figure 1.2.





The screenshot shows the Code Express website. The header includes the C#Corner logo, a navigation menu with 'Home', 'About', and 'Contact', and user options: 'Register', 'Log in', 'Become a member', and 'Login'. Below the header, there are buttons for 'Post' and 'Ask Question'. The main content area displays a book listing for 'jQuery UI' by Eric Sarrion, published by O'Reilly in 2013 for \$320.00. A blue horizontal bar separates this from a second listing for 'ASP.NET MVC 4' by Eric Sarrion, published in 2013 for \$549.00. The footer shows the copyright notice '© 2014 - Code Express'.

Code Express

Register Log in

Home About Contact

Publisher
O'REILLY

Title jQuery UI
Author Eric Sarrion
Year 2013
Price 320.00

Title ASP.NET MVC 4

Year 2013
Price 549.00

© 2014 - Code Express

Figure 1.2: output result using HTML rendering

Rendering JSON Data

In the previous section you have learned that you can render an HTML on AJAX request but in this section you will learn about rendering only serialized data, not entire HTML. ASP.NET MVC offers native JSON support in the form of the JsonResult action result, which accepts a model object that it serialized into the JSON format. In order to add AJAX support to your controller actions via JSON, simply use the Controller.Json() method to create a new JsonResult containing the object to be serialized.

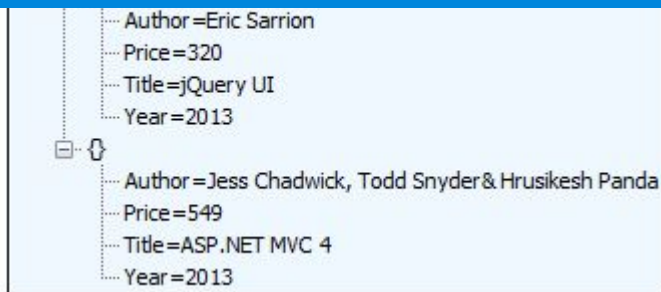


Figure 1.3 JSON Data from action method

I define a route for this action in the RegisterRoute() method under the RouteConfig class (App_Start/RouteConfig.cs).

```

01. routes.MapRoute("BooksByPublisherId",
02.     "book/booksbypublisherid/",
03.     new { controller = "Book", action = "BooksByPublisherId" },
04.     new[] { "JsonRenderingMvcApplication.Controllers" });

```

Now modify the previously created Index view for the publisher that has a dropdown list for the publisher and that shows the

Index view.

```

01. @model JsonRenderingMvcApplication.Models.PublisherModel
02.
03. <script src="~/Scripts/jquery-1.7.1.min.js"></script>
04. <script type="text/javascript">
05.     $(document).ready(function () {
06.         $("#Id").change(function () {
07.             var id = $("#Id").val();
08.             var booksDiv = $("#booksDiv");
09.             $.ajax({
10.                 cache: false,
11.                 type: "GET",
12.                 url: "@(Url.RouteUrl("BooksByPublisherId"))",
13.                 data: { "id": id },
14.                 success: function (data) {
15.                     var result = "";

```

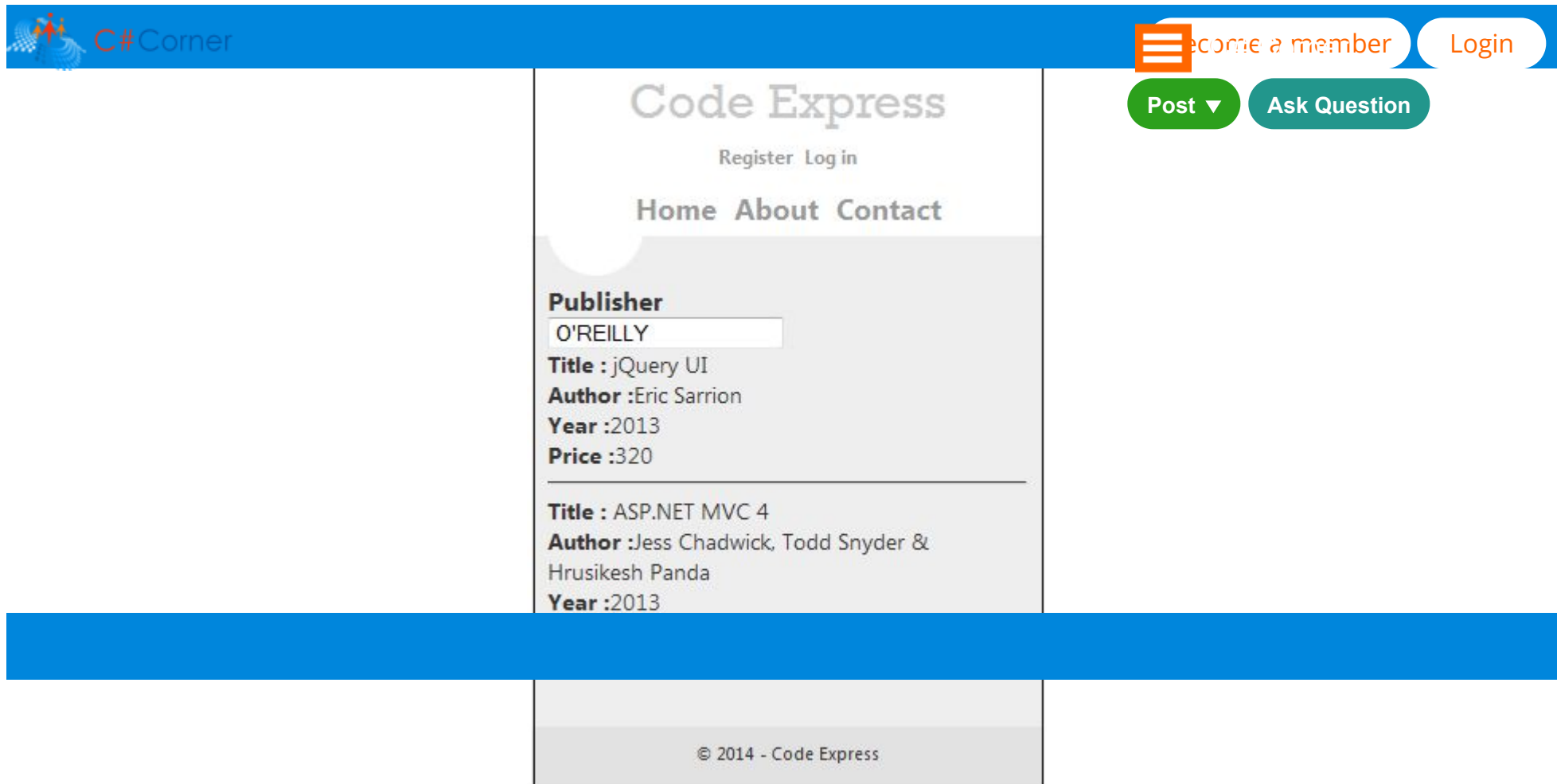




```
18.         result += '<b>Title : </b>' + book.Title + '<br/>' +  
19.             '<b> Author :</b>' + book.Author + '  
20.             '<b> Year :</b>' + book.Year + '<br/>' +  
21.             '<b> Price :</b>' + book.Price + '<hr/>';  
22.         });  
23.         booksDiv.html(result);  
24.     },  
25.     error: function (xhr, ajaxOptions, thrownError) {  
26.         alert('Failed to retrieve books.');27.     }  
28. });  
29. });  
30. });  
31. </script>  
32. <div>  
33.     @Html.LabelFor(model=>model.Id)  
34.     @Html.DropDownListFor(model => model.Id, Model.PublisherList)  
35. </div>  
36. <div id="booksDiv">
```

Run the application and choose an item from the publisher dropdown list; you will then get the result as in Figure 1.3.





The screenshot shows the Code Express website. The header includes the C# Corner logo, a navigation menu with 'Home', 'About', and 'Contact', and user options: 'Register', 'Log in', 'Become a member', and 'Login'. Below the header, there are two buttons: 'Post' and 'Ask Question'. The main content area displays a book listing for 'jQuery UI' by Eric Sarrion, published in 2013 for \$320. Below this, there is a section for 'ASP.NET MVC 4' by Jess Chadwick, Todd Snyder & Hrusikesh Panda, published in 2013. The footer shows the copyright notice '© 2014 - Code Express'.

Figure 1.4: output result using JSON data rendering

Conclusion

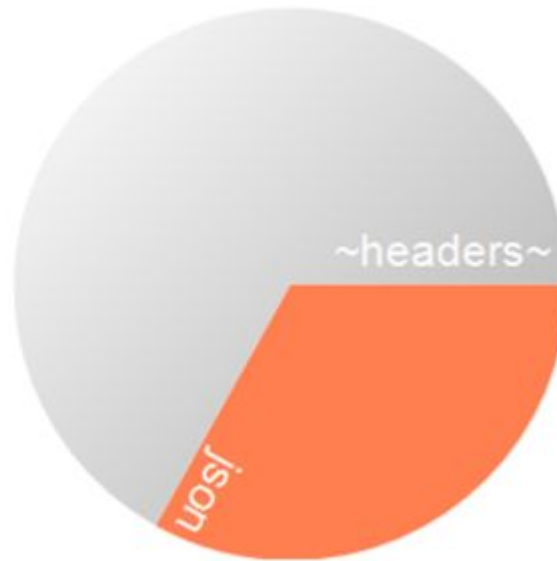
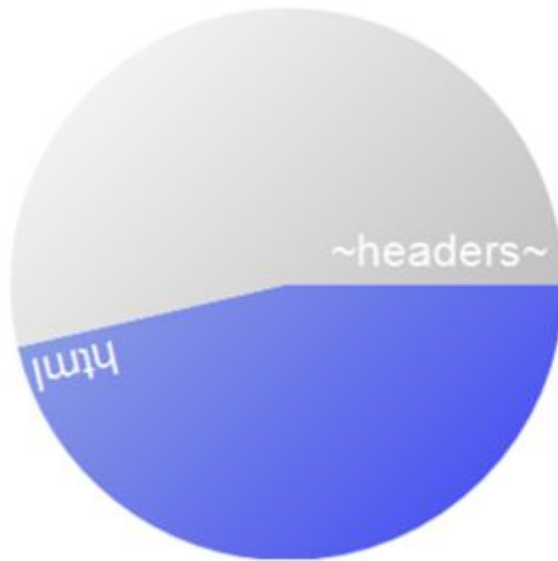
Which one is fast? You can say that JSON data rendering is faster compared to partial view rendering. I make the same request using both approaches and get response data in bytes as shown in the following table.

Content Type	Header	Body	Total (Byte)
text/html	434	375	809
application/ json	398	197	595

It is big difference when working with a small amount server data. When I represent these statistics in a pie chart then these look like:

Post ▼

Ask Question



Brought to you by: [Embed Analytics and Dashboards](#) into your product with a JavaScript SDK. Free trial.

Next Recommended Article

[Collect Form Data From Both Main and Partial View in MVC](#)

AJAX

JSON

MVC

MVC Rendering

Sandeep Singh Shekhawat *TOP 100*



[Become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

96



18.8m



5



4



7



Type your comment here and press Enter Key (Minimum 10 characters)



Good Explanation. Thanks.

[Imran Hassan](#)

1831 96 0

Apr 18, 2020



0



0

Reply



How can we modify this accordingly to suit the need



Sandeep, our requirement is to populate the intel xdk mobile app HTML combo box incorporating json/jquery as a web service and c# web method

[Rejith Harisen](#)

1916 11 0

Oct 11, 2017



1



0

Reply



good one...

[Sonu Chaudhary](#)

803 1.7k 186.7k

Jun 06, 2016



1



0

Reply



good work

[Bhushan Gupta](#)

1733 194 2.6k

Sep 10, 2015



1



0

Reply

Thanks Vishnujeet..

[Sandeep Singh Shekhawat](#)

Jan 23



[Become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

Nice one !!

Vishnujeet Kumar

466 4.1k 1.1m

Jan 23, 2014

2

0

Reply



[Become a member](#)[Login](#)[Post ▼](#)[Ask Question](#)

FEATURED ARTICLES

[CI/CD Implementation For A Simple Web Application Using Azure DevOps](#)

[Async Await Reference Implementation](#)



[Become a member](#)[Login](#)

Registration And Login Functionality In ASP.NET Core 3.0 MVC Web Application Using Identity

[Post ▼](#)[Ask Question](#)

C# 9 Preview

[View All ➔](#)

TRENDING UP




- 01 Implement Azure File Storage Using ASP.NET Core Console Application
- 02 Implement Azure Queue Using ASP.NET Core Console Application
- 03 How To Fetch Data Using React Hooks
- 04 What Is NSX Logical Switching And NSX Logical Routing?
- 05 What Is Network-Attached Storage?
- 06 What are Virtual Switches and Standard Switches?
- 07 What Is Local Disk Storage?
- 08 What Is Load Balancing In Networking?




[become a member](#)[Login](#)

10 Covid-19 Tracker Website With React, Material.UI And Chart.js

[Post ▼](#)[Ask Question](#)[View All ➔](#)

C# Corner

Become a member

Login

About Us Contact Us Privacy Policy Terms Media Kit Sitemap Report a Bug

Post ▾ Ask Question

C# Tutorials Common Interview Questions Stories Consultants Ideas Certifications

©2020 C# Corner. All contents are copyright of their authors.

