
**CODE
PROJECT**
For those who code

Build apps that matter
[GET THE EBOOK](#)

**VMware
Pivotal Labs**

[articles](#) [quick answers](#) [discussions](#) [features](#) [community](#) [help](#)

Search for articles, questions, tips



Articles » Web Development » ASP.NET » General



Rendering a Partial View and JSON Data Using AJAX in ASP.NET MVC

**Sandeep Singh Shekhawat**18 Sep 2014 [CPOL](#)Rate this:  4.82 (21 votes)

This article explains how to render a partial view and JSON data using AJAX.

[Download JSON Rendering MVC application - 3.3 MB](#)

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 on the 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

I add the ADO.NET Entity Model to the application to do the database operations mapped from the "**Development**" database. The ADO.NET Entity Model is an Object Relational Mapping (ORM) that creates a higher abstract object model over ADO.NET components. This ADO.NET Entity Model is mapped the with "**Development**" database so context class is "**DevelopmentEntities**" that inherits the **DbContext** class.

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.

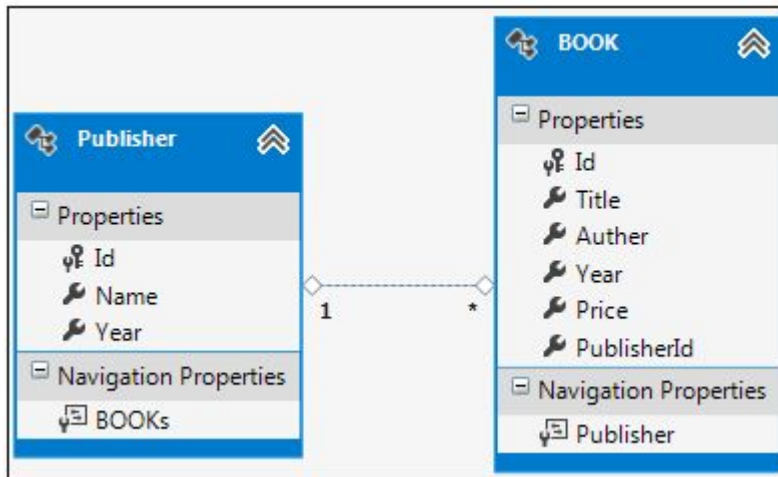


Figure 1.1 The ADO.NET Entity Model mapping with **Publisher** and **Book** tables.

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

As you already know, the purpose of the application is to create two entities (**Publisher** and **BOOK**) that I have already created as the same therefore I need to create two models, one for the publisher and the other for the books.

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.

[Hide](#) [Copy Code](#)

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;
```

```
namespace JsonRenderingMvcApplication.Models
{
    public class PublisherModel
    {
        public PublisherModel()
        {
            PublisherList = new List<SelectListItem>();
        }

        [Display(Name="Publisher")]
        public int Id { get; set; }
        public IEnumerable<SelectListItem> PublisherList { get; set; }
    }
}
```

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.

[Hide](#) [Copy Code](#)

```
namespace JsonRenderingMvcApplication.Models
{
    public class BookModel
    {
        public string Title { get; set; }
        public string Author { get; set; }
        public string Year { get; set; }
        public decimal Price { get; set; }
    }
}
```

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

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 at 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.

[Hide](#) [Copy Code](#)

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using JsonRenderingMvcApplication.Models;
```

```

namespace JsonRenderingMvcApplication.Controllers
{
    public class PublisherController : Controller
    {
        public ActionResult Index()
        {
            PublisherModel model = new PublisherModel();
            using (DAL.DevelopmentEntities context = new DAL.DevelopmentEntities())
            {
                List<DAL.Publisher> PublisherList = context.Publishers.ToList();
                model.PublisherList = PublisherList.Select(x =>
                    new SelectListItem()
                    {
                        Text = x.Name,
                        Value = x.Id.ToString()
                    });
            }
            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.

[Hide](#) [Copy Code](#)

```

public ActionResult BookByPublisher(int id)
{
    IEnumerable<BookModel> modelList = new List<BookModel>();
    using (DAL.DevelopmentEntities context = new DAL.DevelopmentEntities())
    {
        var books = context.BOOKs.Where(x => x.PublisherId == id).ToList();
        modelList = books.Select(x =>
            new BookModel()
            {

```

```

                Title = x.Title,
                Author = x.Auther,
                Year = x.Year,
                Price = x.Price
            });
        }
        return PartialView(modellist);
    }

```

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

[Hide](#) [Copy Code](#)

```

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

```

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.

[Hide](#) [Shrink ▲](#) [Copy Code](#)

```

@model JsonRenderingMvcApplication.Models.PublisherModel
<script src="~/Scripts/jquery-1.7.1.min.js"></script>
<script type="text/javascript">

    $(document).ready(function ()
    {
        $("#Id").change(function ()
        {
            var id = $("#Id").val();
            var booksDiv = $("#booksDiv");
            $.ajax({
                cache: false,
                type: "GET",
                url: "@(Url.RouteUrl("BookByPublisher"))",
                data: { "id": id },
                success: function (data)
                {
                    booksDiv.html('');
                    booksDiv.html(data);
                },
                error: function (xhr, ajaxOptions, thrownError)
                {
                    alert('Failed to retrieve books.');
```

```
});  
</script>  
<div>  
    @Html.LabelFor(model=>model.Id)  
    @Html.DropDownListFor(model => model.Id, Model.PublisherList)  
</div>  
<div id="booksDiv">  
</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 a web application titled "Code Express" with a header containing "Register" and "Log in" links, and navigation links "Home", "About", and "Contact". The main content area displays a search result for a book. The "Publisher" dropdown menu is set to "O'REILLY". The book details are as follows:

Field	Value
Title	jQuery UI
Author	Eric Sarrion
Year	2013
Price	320.00

Below this, another book entry is partially visible:

Field	Value
Title	ASP.NET MVC 4
Author	Jess Chadwick, Todd Snyder & Hrusikesh Panda
Year	2013
Price	549.00

The footer of the application shows "© 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.

Now, create an action method **BooksByPublisherId()** in the book controller that returns **JsonResult**. 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.

[Hide](#) [Copy Code](#)

```
public JsonResult BooksByPublisherId(int id)
{
    IEnumerable<BookModel> modellist = new List<BookModel>();
    using (DAL.DevelopmentEntities context = new DAL.DevelopmentEntities())
    {
        var books = context.BOOKs.Where(x => x.PublisherId == id).ToList();
        modellist = books.Select(x =>
            new BookModel()
            {
                Title = x.Title,
                Author = x.Auther,
                Year = x.Year,
                Price = x.Price
            });
    }
    return Json(modellist, JsonRequestBehavior.AllowGet);
}
```

The action method was created to return book details. Here the **Controller.Json()** method has two parameters, the first one is for the data source that will be serialized and the second parameter is **JsonRequestBehavior.AllowGet**, which explicitly informs the ASP.NET MVC Framework that it's acceptable to return JSON data in a response to an HTTP **GET** request.

The **JsonRequestBehavior.AllowGet** parameter is necessary in this case because, by default, ASP.NET MVC disallows returning JSON in response to an HTTP **GET** request in order to avoid potentially dangerous security vulnerability known as JSON hijacking.

This action method returns JSON data as shown in Figure 1.3.

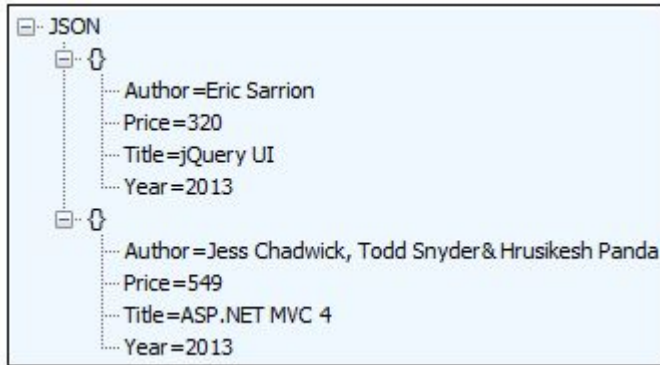


Figure 1.3 JSON Data from action method

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

Hide Copy Code

```

routes.MapRoute("BooksByPublisherId",
    "book/booksbypublisherid/",
    new { controller = "Book", action = "BooksByPublisherId" },
    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 book details depending on selecting a value in the publisher dropdown list using AJAX. The following is a code snippet for the Index view:

Hide Shrink ▲ Copy Code

```

@model JsonRenderingMvcApplication.Models.PublisherModel

<script src="~/Scripts/jquery-1.7.1.min.js"></script>
<script type="text/javascript">
    $(document).ready(function () {
        $("#Id").change(function () {
            var id = $("#Id").val();
            var booksDiv = $("#booksDiv");
            $.ajax({
                cache: false,
                type: "GET",
                url: "@(Url.RouteUrl("BooksByPublisherId"))",
                data: { "id": id },
                success: function (data) {
                    var result = "";
                    booksDiv.html('');
                    $.each(data, function (id, book) {
                        result += '<b>Title : </b>' + book.Title + '<br/>' +
                            '<b> Author :</b>' + book.Author + '<br/>' +
                            '<b> Year :</b>' + book.Year + '<br/>' +

```



```
                '<b> Price :</b>' + book.Price + '<hr/>';
            });
            booksDiv.html(result);
        },
        error: function (xhr, ajaxOptions, thrownError) {
            alert('Failed to retrieve books.');
```

Run the application and choose an item from the publisher dropdown list; you will then get the result as in Figure 1.4.



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

Table 1.1: Summary of response bytes

It is a big difference when working with a small amount of server data. When I represent these statistics using a circle graph, then these look like:



License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOl\)](#)

Share



About the Author

Sandeep Singh Shekhawat

  Software Developer
India 

Azure + C3.ai

10x less code

25x faster development

1/10 time to deploy

Get the report





He is awarded for Microsoft TechNet Guru, CodeProject MVP and C# Corner MVP. <http://i-knowtech.com/>

Comments and Discussions

You must **Sign In** to use this message board.



Spacing **Relaxed** ▾ Layout **Normal** ▾ Per page **25** ▾ **Update**

First Prev Next

Ajax code Problem	Member 11956823	25-Apr-16 1:13
WHAT IS "DAL"	Diego Flores	21-Aug-15 7:46
Re: WHAT IS "DAL"	Mario Smolčić	17-Oct-15 7:47
My vote of 5	Humayun Kabir Mamun	23-Oct-14 21:58
what is the need to use two parameters for the function in \$.each() function(id,book)	Srinivas Ra	21-Sep-14 20:36
doubt in view part	NithyaGopu	30-Mar-14 20:21

Last Visit: 13-Jun-20 12:12 Last Update: 13-Jun-20 13:43

[Refresh](#)

1

[General](#) [News](#) [Suggestion](#) [Question](#) [Bug](#) [Answer](#) [Joke](#) [Praise](#) [Rant](#) [Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#)[Advertise](#)[Privacy](#)[Cookies](#)[Terms of Use](#)Layout: [fixed](#) | [fluid](#)Article Copyright 2014 by Sandeep Singh Shekhawat
Everything else Copyright © [CodeProject](#), 1999-2020

Web01 2.8.200606.1