پشتیبانی ASP.NET Web API 2.1 از BSON

BSON چیست؟

BSON یک قالب serial کردن به صورت باینری می باشد. BSON مخفف Binary JSON می باشد اما نوع serial کردن BSON و JSON بسیار با هم تفاوت دارد. BSON یک JSON-like می باشد زیرا objectها به عنوان نماینده ای از جفت مقدار-نام (name-value pairs) شبیه به JSON می باشند. بر خلاف JSON، مقادیر عددی به صورت رشته ای و بایت ذخیره نمی شوند.

BSON برای آن طراحی شده است تا باعث سرعت بخشیدن به کدگذاری و کد گشایی، اسکن راحت و کمتر کردن حجم شود.

- BSON را می توان در اندازه فایل نیز با JSON مقایسه کرد. با توجه به داده، ظرفیت انتقال در BSON ممکن است کوچکتر یا بزرگتر از JSON شود. برای serial کردن داده باینری مثل عکس، BSON حجم کمتری از JSON دارد زیرا داده باینری بر اساس کدگذاری ۶۴ بیتی نمی باشد.
- مستندات BSON آسان اسکن می شود زیرا عنصرها پیشوندی از رشته ها دارند . در نتیجه ، تفسیرگر(parser) می تواند بدون کدگشایی آن ها از عنصر ها عبور کند.
- کدگذاری و کدگشایی بهینه تر می شود زیرا مقادیر عددی به صورت عددی و نه بصورت رشته ای ذخیره می شوند.

کاربران محلی مثل برنامه های .NET. کاربر می توانند از مزایای استفاده از BSON در یک مکان متنی مثل JSON یا XML استفاده کنند.

خوشبختانه Web API از محتوای مذاکره ای (Content negotiation)استفاده می کند و API شما از هر دو قالب پشتیبانی می کند و به کاربر اجازه انتخاب می دهد.

فعالسازی BSON در سمت سرور

در تنظیمات Web API خود، BsonMediaTypeFormatter را به مجموعه قالب ها اضافه کنید.

```
public static class WebApiConfig
{
   public static void Register(HttpConfiguration config)
   {
       config.Formatters.Add(new BsonMediaTypeFormatter());
       // Other Web API configuration not shown...
   }
}
```

حالا اگر کاربر application/bson را درخواست کند، Web API از قالب BSON استفاده می کند.

برای وابسته کردن BSON به انواع دیگر داده ها، آن ها را به مجموعه SupportedMediaTypes اضافه کنید. کد زیر application/vnd.contoso را به انواع داده های پشتیبانی شده اضافه می کند:

```
var bson = new BsonMediaTypeFormatter();
bson.SupportedMediaTypes.Add(new MediaTypeHeaderValue("application/vnd.contoso"));

config.Formatters.Add(bson);
```

مثالی از HTTP Session

برای این مثال، ما از کلاس زیر به علاوه ی یک Web API Controller ساده استفاده می کنیم:

```
public class Book
{
   public int Id { get;set;}
   public string Title { get;set;}
```

```
    public string Author { get;set;}
    public decimal Price { get;set;}
    public DateTime PublicationDate { get;set;}
}
public class BooksController : ApiController
{
    public IHttpActionResult GetBook(int id)
    {
        var book = new Book()
        {
            Id = id,
            Author = "Charles Dickens",
            Title = "Great Expectations",
            Price = 9.95M,
            PublicationDate = new DateTime(2014, 1, 20)
        };
        return Ok(book);
    }
}
```

کاربر یک درخواست HTTP مشابه درخواست زیر می فرستد:

```
GET http://localhost:15192/api/books/1 HTTP/1.1
User-Agent: Fiddler
Host: localhost:15192
Accept: application/bson
```

و در اینجا پاسخ آن را می بینید:

```
Content-Type: application/bson;charset=utf-8
Date: Fri, 17 Jan 2014 01:05:40 GMT
Content-Length: 111
.....Id......Title.....Great Expectations..Author.....Charles
Dickens..Price.........PublicationDate.........
```

در اینجا کاراکتر های باینری را با "." جا به جا می کنیم. تصویر زیر یک نمایه از مقادیر خام در Fiddler نشان می دهد.

```
48 54 54 50 2F 31 2E 31 20 32 30 30 20 4F 4B 0D 0A 43 61 63 68   HTTP/1.1 200 OK..Cach
65 2D 43 6F 6E 74 72 6F 6C 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A   e-Control: no-cache..
50 72 61 67 6D 61 3A 20 6E 6F 2D 63 61 63 68 65 0D 0A 43 6F 6E   Pragma: no-cache..Con
74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C 69 63 61 74 69 6F   tent-Type: applicatio
6E 2F 62 73 6F 6E 3B 20 63 68 61 72 73 65 74 3D 75 74 66 2D 38   n/bson; charset=utf-8
0D 0A 45 78 70 69 72 65 73 3A 20 2D 31 0D 0A 53 65 72 76 65 72   ..Expires: -1..Server
3A 20 4D 69 63 72 6F 73 6F 66 74 2D 49 49 53 2F 38 2E 30 0D 0A   : Microsoft-IIS/8.0..
58 2D 41 73 70 4E 65 74 2D 56 65 72 73 69 6F 6E 3A 20 34 2E 30   X-AspNet-Version: 4.0
2E 33 30 33 31 39 0D 0A 58 2D 53 6F 75 72 63 65 46 69 6C 65 73   .30319..X-SourceFiles
3A 20 3D 3F 55 54 46 2D 38 3F 42 3F 51 7A 70 63 56 58 4E 6C 63   : =?UTF-8?B?QzpcVXNlc
6E 4E 63 62 58 64 68 63 33 4E 76 62 6C 78 45 62 32 4E 31 62 57   nNcbXdhc3NvblxEb2N1bW
56 75 64 48 4E 63 56 6D 6C 7A 64 57 46 73 49 46 4E 30 64 57 52   VudHNcVmlzdWFsIFN0dWR
70 62 79 41 79 4D 44 45 7A 58 46 42 79 62 32 70 6C 59 33 52 7A   pbyAyMDEzXFByb2plY3Rz
58 45 4A 7A 62 32 35 46 65 47 46 74 63 47 78 6C 58 45 4A 7A 62   XEJzb25FeGFtcGxlXEJzb
32 35 46 65 47 46 74 63 47 78 6C 58 47 46 77 61 56 78 69 62 32   25FeGFtcGxlXGFwaVxib2
39 72 63 31 77 78 3F 3D 0D 0A 58 2D 50 6F 77 65 72 65 64 2D 42   9rc1wx?=..X-Powered-B
79 3A 20 41 53 50 2E 4E 45 54 0D 0A 44 61 74 65 3A 20 46 72 69   y: ASP.NET..Date: Fri
2C 20 31 37 20 4A 61 6E 20 32 30 31 34 20 30 31 3A 30 35 3A 34   , 17 Jan 2014 01:05:4
30 20 47 4D 54 0D 0A 43 6F 6E 74 65 6E 74 2D 4C 65 6E 67 74 68   0 GMT..Content-Length
3A 20 31 31 31 0D 0A 0D 0A 6F 00 00 00 10 49 64 00 01 00 00 00   : 111....o....Id.....
02 54 69 74 6C 65 00 13 00 00 00 47 72 65 61 74 20 45 78 70 65   .Title.....Great Expe
63 74 61 74 69 6F 6E 73 00 02 41 75 74 68 6F 72 00 10 00 00 00   ctations..Author.....
43 68 61 72 6C 65 73 20 44 69 63 6B 65 6E 73 00 01 50 72 69 63   Charles Dickens..Pric
65 00 66 66 66 66 66 E6 23 40 09 50 75 62 6C 69 63 61 74 69 6F   e.fffffæ#@.Publicatio
6E 44 61 74 65 00 00 CC A9 AE 43 01 00 00 00                     nDate..Ì©®C....
```

استفاده کردن از BSON با HttpClient

کاربران می توانند از برنامه های .NET. در قالب BSON با HttpClient استفاده کنند. کد زیر یک درخواست GET که از BSON پشتیبانی می کند می فرستد و سپس ظرفیت انتقال BSON، پاسخ را deserial می کند.

```
HTTP/1.1 200 OK
static async Task RunAsync()
{
   using (HttpClient client = new HttpClient())
   {
       client.BaseAddress = new Uri("http://localhost");
       // Set the Accept header for BSON.
       client.DefaultRequestHeaders.Accept.Clear();
       client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/bson"));
       // Send GET request.
       result = await client.GetAsync("api/books/1");
       result.EnsureSuccessStatusCode();
       // Use BSON formatter to deserialize the result.
       MediaTypeFormatter[] formatters = new MediaTypeFormatter[] {
           new BsonMediaTypeFormatter()
       };
       var book = await result.Content.ReadAsAsync(formatters);
   }
}
```

برای درخواست BSON از سرور، مقدار سربرگ پذیرفته شده(Accept header) را application/bson قرار دهید.

```
client.DefaultRequestHeaders.Accept.Clear();
client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/bson"));
```

برای deserial کردن پاسخ از BsonMediaTypeFormatter استفاده کنید. این قالب در قالب های پیش فرض موجود نیست بنابراین شما وقتی پاسخ را می خوانید باید آن را مشخص کنید.

```
MediaTypeFormatter[] formatters = new MediaTypeFormatter[] {
   new BsonMediaTypeFormatter()
};
var book = await result.Content.ReadAsAsync(formatters);
```

مثال بعدی نشان می دهد چطور یک درخواست POST که شامل BSON می شود بفرستید.

```
static async Task RunAsync()
{
   using (HttpClient client = new HttpClient())
   {
       client.BaseAddress = new Uri("http://localhost:15192");
       // Set the Accept header for BSON.
       client.DefaultRequestHeaders.Accept.Clear();
       client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/bson"));
       var book = new Book()
       {
           Author = "Jane Austen",
           Title = "Emma",
           Price = 9.95M,
           PublicationDate = new DateTime(1815, 1, 1)
       };
       // POST using the BSON formatter.
       MediaTypeFormatter bsonFormatter = new BsonMediaTypeFormatter();
       var result = await client.PostAsync("api/books", book, bsonFormatter);
```

```
        result.EnsureSuccessStatusCode();
    }
}
```

بسیاری از قسمت های این کد مثل قبل می باشد اما متد PostAsync مقدار BsonMediaTypeFormatter را به عنوان یک قالب مشخص می کند:

```
MediaTypeFormatter bsonFormatter = new BsonMediaTypeFormatter();
var result = await client.PostAsync("api/books", book, bsonFormatter);
```

serial کردن داده های اولیه سطح بالا

همه ی اسناد BSON یک لیست از جفت کلید مقدار ها(key/value pairs) می باشند. BSON یک ترکیب برای serial کردن یک مقدار مفرد خام برای نوع عددی و رشته ای تعریف نمی کند.

برای کار کردن در این سطح، BsonMediaTypeFormatter مربوط به داده های اولیه به صورت مقدار ویژه رفتار می کند. قبل از serial کردن، این مقدار را به key-value pair با کلید Value تبدیل می کند . برای مثال فرض کنید API Controller یک مقدار عددی را برگرداند:

```
public class ValuesController : ApiController
{
    public IHttpActionResult Get()
    {
        return Ok(42);
    }
}
```

قبل از serial کردن، قالب BSON این را به key-value pair تبدیل می کند.

```
{ "Value": 42 }
```

وقتی شما عمل deserial انجام می دهید، قالب ، اطلاعات برگشتی را به مقدار اصلی تبدیل می کند. با این حال، اگر Web API شما مقادیر خام را برگرداند، کاربران از یک تفسیرگر BSON متفاوت که بتواند این مورد را مدیریت کند استفاده می کنند.