بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

قالب بندی داده (media formatter) در ASP.NET Web API 2

مدرس : مهندس افشین رفوآ

دوره آموزشی Web API

Media Type های اینترنت

media type که به آن MIME نیز گفته می شود، فرمت یک قطعه از داده را شناسایی می کند. در HTTP، نوع داده، فرمت پیام را توصیف می کند. یک داده شامل دو رشته می شود، یکی نوع آن و دیگری زیر گروه آن. برای مثال:

- text/html
- image/png
- application/json

زمانی که یک پیام HTTP شامل یک entity-body می شود، سر برگ محتوا (Contetnt-type header)، نوع پیام را مشخص می سازد. این به گیرنده نشان می دهد چگونه محتوای پیام را تفسیر کند.

```
HTTP/1.1 200 OK
Content-Length: 95267
Content-Type: image/png
```

وقتی کاربر یک درخواست ارسال می کند، این پیام می تواند شامل سر برگ پذیرنده (Accept header) باشد. Accept header به سرور اعلام می کند کاربر کدام نوع داده را از سرور درخواست می کند. برای مثال:

```
Accept: text/html,application/xhtml+xml,application/xml
```

header بالا به سرور اعلام می کند که کاربر هم HTML و هم XHTML و نیز XML را می خواهد. همچنین به سرور اعلام می کند Web API چگونه پیام را serial و deserial کند. Web API از XML، JSON، BSON پشتیبانی می کند و شما می توانید با نوشتن یک media formatter ، از انواع دیگر داده ها پشتیبانی کنید.

برای ساخت یک media formatter، کلاس شما باید از یکی از کلاس های زیر مشتق شود:

- MediaTypeFormatter که به صورت غیر همسان متد ها را می خواند و می نویسد.
- BufferedMediaTypeFormatter که از MediaTypeFormatter مشتق می شود و به صورت همسان متد ها را می خواند و می نویسد.

مشتق کردن از BufferedMediaTypeFormatter آسان تر است زیرا کد غیر همسانی وجود ندارد و به این معنی است که فراخوانی Thread می تواند در زمان I/O مسدود شود.

مثال: ساختن یک media formatter از نوع CSV

این مثال نشان می دهد که media formatter می تواند یک Product به عنوان object را به مقادیر جدا از هم به وسیله کاما(CSV) serial کند.در زیر، Product را تعریف می کنیم:

```
namespace ProductStore.Models
    {
        public class Product
        {
            public int Id { get; set; }
            public string Name { get; set; }
            public string Category { get; set; }
            public decimal Price { get; set; }
        }
     }

   }
```

برای پیاده سازی یک CSV media formatter، یک کلاس که از BufferedMediaTypeFormater مشتق شود تعریف می کنیم:

```
namespace ProductStore.Formatters
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using ProductStore.Models;
namespace ProductStore.Formatters
```

```
{
    public class ProductCsvFormatter : BufferedMediaTypeFormatter
    {
    }

}
```

برای آنکه نشان دهیم کدام نوع از formatter می تواند serial کند، متد CanWriteType را override می کنیم:

```
public override bool CanWriteType(System.Type type)
{
    if (type == typeof(Product))
    {
        return true;
    }
    else
    {
        Type enumerableType = typeof(IEnumerable<Product>);
        return enumerableType.IsAssignableFrom(type);
    }

}
```

به صورت مشابه، برای آنکه نشان دهیم کدام formatter می تواند deserial کند، متد CanReadType را override می کنیم. در این مثال، formatter از deserial کردن پشتیبانی نمی کند و برای همین مقدار false را بر می گرداند.

```
public override bool CanReadType(Type type)
{
    return false;

}
```

در آخر نیز متد WriteToStream را override می کنیم. این متد با نوشتن داده ها در یک Stream، یک نوع داده را serial می کند. اگر formatter شما از deserialization پشتیبانی کند نیز متد ReadFromStream را override می کنیم:

```
public override void WriteToStream(Type type, object value, Stream writeStream, HttpContent
content)
{
    using (var writer = new StreamWriter(writeStream))
    {
        var products = value as IEnumerable<Product>;
        if (products != null)
        {
            foreach (var product in products)
            {
                WriteItem(product, writer);
            }
        }
        else
        {
```

```
            var singleProduct = value as Product;
            if (singleProduct == null)
            {
                throw new InvalidOperationException("Cannot serialize type");
            }
            WriteItem(singleProduct, writer);
        }
    }
}
// Helper methods for serializing Products to CSV format.
private void WriteItem(Product product, StreamWriter writer)
{
    writer.WriteLine("{0},{1},{2},{3}", Escape(product.Id),
        Escape(product.Name), Escape(product.Category), Escape(product.Price));
}
static char[] _specialChars = new char[] { ',', '\n', '\r', '"' };
private string Escape(object o)
{
    if (o == null)
    {
        return "";
    }
    string field = o.ToString();
    if (field.IndexOfAny(_specialChars) != -1)
    {
        // Delimit the entire field with quotes and replace embedded quotes with "".
        return String.Format("\"{0}\"", field.Replace("\"", "\"\""));
    }
    else return field;
}
```

اضافه کردن یک media formatter به خط لوله Web API (Pipeline)

برای اضافه کردن یک نوع media formatter به Web API pipeline از خاصیت Formatters در HttpConfiguration استفاده می کنیم:

```
public static void ConfigureApis(HttpConfiguration config)
{
    config.Formatters.Add(new ProductCsvFormatter());

}
```

رمزگذاری کاراکتر

در صورت تمایل، media formatter می تواند از چند نوع رمزگذاری مثل UTF-8 و ISO 8859-1 پشتیبانی کند. در constructor یک یا چند نوع System.Text.Encoding به مجموعه SupportedEncodings اضافه کنید. حالت رمزگذاری را در ابتدا به صورت پیش فرض قرار دهید:

```
public ProductCsvFormatter()
{
    SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/csv"));
    // New code:
    SupportedEncodings.Add(new UTF8Encoding(encoderShouldEmitUTF8Identifier: false));
    SupportedEncodings.Add(Encoding.GetEncoding("iso-8859-1"));

  }
```

در متد های WriteToStream و ReadFromStream برای رمزگذاری بهتر، متد MediaTypeFormatter.SelectCharacterEncoding را فراخوانی می کنیم. این متد ، Accept Header را به لیستی از رمزگذاری های پشتیبانی شده بسط می دهد. برای خواندن و نوشتن Stream از Encoding استفاده کنید:

```
public override void WriteToStream(Type type, object value, Stream writeStream, HttpContent
content)
{
    Encoding effectiveEncoding = SelectCharacterEncoding(content.Headers);
    using (var writer = new StreamWriter(writeStream, effectiveEncoding))
    {
        // Write the object (code not shown)
    }

  }
```