عنوان: **روشی برای DeSerialize کردن QueryString به یک کلاس**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۷/۱۴ ۱۹:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Reflection

چند روز پیش در حال استفاده از افزونه‌ی jQuery Bootgrid بودم که داده‌های خود را در قالب زیر به صورت کوئری استرینگ ارسال می‌کند.

```
current=1&rowCount=10&sort[sender]=asc&searchPhrase=&id=b0df282a-0d67-40e5-8558-c9e93b7befed
```

قبلا هم با کوئری استرینگ‌ها کار کرده‌ایم و نحوه دریافت آن را یاد گرفته‌ایم و میدانیم که اگر کلاس شما شامل پراپرتی‌های همنام با کلیدهای کوئری استرینگ باشد مستقیما در کلاس شما جا می‌گیرند؛ ولی من دوست داشتم که پراپرتی‌های کلاسم نام دلخواه من را داشته باشد و اجباری به استفاده از نام‌های کوئری استرینگ نداشته باشد. اصلا ممکن است افراد Back End یک سری کد نوشته‌اند و کلاسشان را هم ساخته‌اند و اصلا کاری با من ندارند که چطوری داده‌ها را و با چه اسامی از آن‌ها دریافت می‌کنم و فقط انتظار دارند که کلاس آن‌ها را با اطلاعات دریافتی پر کنم و ارسال کنم. اگر بخواهیم به طور دستی هر یک از کلیدها را چک کنیم، هم کدنویسی طولانی می‌شود و هم کد قابلیت استفاده مجدد ندارد. پس بهترین کار این است که یک کد با قابلیت استفاده مجدد بنویسیم.

دوست دارم چیزی شبیه به DeSerialize کردن فرمت json توسط کتابخانه Json.net داشته باشم؛ پس در اولین قدم یک attribute با مشخصات زیر می‌سازیم:

```
[AttributeUsage(AttributeTargets.Property,Inherited = true)]
    public class RequestBodyField:Attribute
    {
        public string Field;
        public RequestBodyField(string field)
        {
            this.Field = field;
        }
    }
```

سپس در کلاس اصلی، ما این خصوصیت‌ها را در بالای propertyها تعریف کرده و با کلیدهای موجود در کوئری استرینگ برابر می‌کنیم:

```
public class EmployeesRequestBody
    {
        [RequestBodyField("current")]
        public  int CurrentPage { get; set; }

        [RequestBodyField("rowcount")]
        public int RowCount { get; set; }

        [RequestBodyField("searchPhrase")]
        public string SearchPhrase { get; set; }

        [RequestBodyField("sort")]
        public NameValueCollection SortDictionary { get; set; }
    }
```

سپس کلاس زیر را می‌نویسیم که وظیفه دارد کلاس‌های از جنس بالا را با query stringها رسیده در درخواست مطابقت دهد:

```
public T GetFromQueryString<T>() where T : new()
        {
            var obj = new T();

            var queryString = HttpContext.Current.Request.QueryString;
            var queries = HttpUtility.ParseQueryString(queryString.ToString());
```

```
            var properties = typeof(T).GetProperties();
            foreach (var property in properties)
            {
                foreach (Attribute attribute in property.GetCustomAttributes(true))
                {
                    var requestBodyField = attribute as RequestBodyField;
                    if (requestBodyField == null) continue;

                    //get value of query string
                    var valueAsString = queries[requestBodyField.Field];

                    var converter = TypeDescriptor.GetConverter(property.PropertyType);
                    var value = converter.ConvertFrom(valueAsString);

                    if (value == null)
                        continue;

                    property.SetValue(obj, value, null);
                }
            }
            return obj;
        }
```

این متد یک تعریف کلاس را دریافت می‌کند. سپس رشته‌ی کوئری استرینگ موجود در بدنه درخواست را دریافت کرده و با استفاده از کد زیر اشیا را به صورت nameValueCollection دریافت می‌کنیم.

```
HttpContext.Current.Request.QueryString
```

نکته: همچنین متد httputility.parseQueryString یک رشته کوئری استرینگ دریافت می‌کند و کوئری استرینگ را به زوج نام و مقدار nameValueCollection تبدیل میکند.

سپس در مرحله‌ی بعدی با استفاده از Reflection پراپرتی‌هایی را که دارای attribute تعریف شده هستند، پیدا می‌کنیم.

مقدار داده شده به attribute را در nameValueCollection بررسی می‌کنیم و در صورت موجود بودن، مقدار آن را می‌گیریم. از آنجا که این مقدار از نوع رشته است و ممکن است مقدار داخل آن عددی یا هر نوع دیگری باشد، باید آن را به نوع صحیح تبدیل کنیم که خطوط زیر کار تبدیل را انجام می‌دهند:

```
var converter = TypeDescriptor.GetConverter(property.PropertyType);
   var value = converter.ConvertFrom(valueAsString);
```

در خط اول بر اساس نوع property کلاس، یک converter دریافت می‌کنیم و سپس مقدار ارسال شده را به آن می‌دهیم تا مقدار جدید را با نوع صحیح خود، دریافت کنیم.
سپس در صورتی که مقدار صحیح دریافت شود و برابر null نباشد، مقدار را در پراپرتی مربوطه جا می‌دهیم.

نکته‌ای که در اینجا نیاز به تلاش بیشتر دارد، کلید sort در کوئری استرینگ است. با نگاهی دقیق‌تر متوجه می‌شوید که خود کلید دو مقدار دارد که یکی از مقادیرش با کلید ترکیب شده است. این حالت روش ارسال آرایه‌ها با نام کلیدی متفاوت در کوئری استرینگ است. این حالت ارسال باعث می‌شود که گرید بتواند حالت multi sort را نیز پیاده سازی کند.
پس برای دریافت این نوع مقادیر کمی کد به آن اضافه می‌کنیم. برای دریافت مقادیر آرایه‌ای کد زیر را به سیستم اضافه می‌کنیم:

```
if (valueAsString == null)
                    {
                        var keys = from key in queries.AllKeys where
key.StartsWith(requestBodyField.Field) select key;

                        var collection = new NameValueCollection();

                        foreach (var key in keys)
                        {
```

```
                            var openBraketIndex = key.IndexOf("[", StringComparison.Ordinal);
                            var closeBraketIndex = key.IndexOf("]", StringComparison.Ordinal);

                            if (openBraketIndex < 0 || closeBraketIndex < 0)
                                throw new Exception("query string is corrupted.");

                            openBraketIndex++;
                            //get key in [...]
                            var fieldName = key.Substring(openBraketIndex, closeBraketIndex -
openBraketIndex);
                            collection.Add(fieldName, queries[key] );
                        }
                        property.SetValue(obj, collection, null);
                        continue;
                    }
```

در صورتیکه شما کلید sort را درخواست کنید و از آنجا که کلید اصلی با نام [sender]sort است، مقدار null بازگشت می‌دهد. پس ما می‌توانیم به این مقدار شک کنیم که شاید این کلید حاوی مقدار مورد نظر ماست؛ پس این حالت را بررسی میکنیم. برای بررسی، با استفاده از linq بررسی می‌کنیم که اگر کلیدهای namValueCollection با این کلید (در اینجا sort) آغاز می‌شوند، پس به احتمال زیاد همان حالت مورد نظر ما رخ داه است. پس اندیس‌های [و] را می‌گیریم و اگر اندیس هر دو بزرگتر از صفر بود مقدار ما بین آن را به عنوان کلید بیرون می‌کشیم و در یک namValueCollection جدید قرار می‌دهیم و در نهایت به پراپرتی پاس می‌دهیم. کد نهایی این متد به شکل زیر است:

```
public T GetFromQueryString<T>() where T : new()
        {
            var obj = new T();
            var properties = typeof(T).GetProperties();

            var queryString = HttpContext.Current.Request.QueryString;
            var queries = HttpUtility.ParseQueryString(queryString.ToString());

            foreach (var property in properties)
            {
                foreach (Attribute attribute in property.GetCustomAttributes(true))
                {
                    var requestBodyField = attribute as RequestBodyField;
                    if (requestBodyField == null) continue;

                    //get value of query string
                    var valueAsString = queries[requestBodyField.Field];

                    if (valueAsString == null)
                    {
                        var keys = from key in queries.AllKeys where
key.StartsWith(requestBodyField.Field) select key;

                        var collection = new NameValueCollection();

                        foreach (var key in keys)
                        {
                            var openBraketIndex = key.IndexOf("[", StringComparison.Ordinal);
                            var closeBraketIndex = key.IndexOf("]", StringComparison.Ordinal);

                            if (openBraketIndex < 0 || closeBraketIndex < 0)
                                throw new Exception("query string is corrupted.");

                            openBraketIndex++;
                            //get key in [...]
                            var fieldName = key.Substring(openBraketIndex, closeBraketIndex -
openBraketIndex);
                            collection.Add(fieldName, queries[key]);
                        }
                        property.SetValue(obj, collection, null);
                        continue;
                    }

                    var converter = TypeDescriptor.GetConverter(property.PropertyType);
                    var value = converter.ConvertFrom(valueAsString);

                    if (value == null)
                        continue;

                    property.SetValue(obj, value, null);
                }
```

```
            }
            return obj;
        }
```

حال به صورت زیر این متد را صدا می‌زنیم:

```
public virtual ActionResult GetEmployees()
{
     var request = new Requests().GetFromQueryString<EmployeesRequestBody>();
}
```

نظرات خوانندگان

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۷/۱۷ ۱۴:۰

جهت رفع باگ کد زیر را

```
var keys = from key in queries.AllKeys where key.StartsWith(requestBodyField.Field+"[") select key;

if(!keys.Any())
    continue;
```

بعد از خط

```
if (valueAsString == null)
{
```

اضافه کنید.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۷/۱۷ ۱۴:۲۶

جهت افزودن پشتیبانی از حالت POST کد متد را به شکل زیر تغییر دهید:

```
public T GetFromQueryString<T>(RequestType type=RequestType.GET) where T : new()
        {
            var obj = new T();
            var properties = typeof(T).GetProperties();

            var queries =new NameValueCollection();
            if(type==RequestType.GET)
                queries= HttpContext.Current.Request.QueryString;
            else
            {
                queries = HttpContext.Current.Request.Form;
            }

            foreach (var property in properties)
            {
                foreach (Attribute attribute in property.GetCustomAttributes(true))
                {
                    var requestBodyField = attribute as RequestBodyField;
                    if (requestBodyField == null) continue;

                    //get value of query string
                    var valueAsString = queries[requestBodyField.Field];

                    if (valueAsString == null)
                    {
                        var keys = from key in queries.AllKeys where
key.StartsWith(requestBodyField.Field) select key;

                        if(!keys.Any())
                            continue;

                        var collection = new NameValueCollection();

                        foreach (var key in keys)
                        {
                            var openBraketIndex = key.IndexOf("[", StringComparison.Ordinal);
                            var closeBraketIndex = key.IndexOf("]", StringComparison.Ordinal);

                            if (openBraketIndex < 0 || closeBraketIndex < 0)
                                throw new Exception("query string is crupted.");

                            openBraketIndex++;
                            //get key in [...]
```

```
                            var fieldName = key.Substring(openBraketIndex, closeBraketIndex -
openBraketIndex);
                            collection.Add(fieldName, queries[key]);
                        }
                        property.SetValue(obj, collection, null);
                        continue;
                    }

                    var converter = TypeDescriptor.GetConverter(property.PropertyType);
                    var value = converter.ConvertFrom(valueAsString);

                    if (value == null)
                        continue;

                    property.SetValue(obj, value, null);
                }
            }
            return obj;
        }
```