بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

انقیاد (Binding) پارامتر در ASP.NET Web API

مدرس : مهندس افشین رفوآ

دوره آموزشی Web API

وقتی Web API یک متد از یک Controller فراخوانی می کند باید مقادیر را برای پارامترها تنظیم کند، به این رویه Binding می گوییم.

به صورت پیش فرض، Web API از قوانین زیر برای bind کردن پارامتر ها استفاده می کند:

- اگر پارامتر از نوع ساده باشد، Web API مقدار را از URI دریافت می کند. انواع داده های .NET. شامل (int،bool،double و غیره) به علاوه TimeSpan، DateTime، Guid، decimal و string می شود.
- برای داده های پیچیده، Web API با استفاده از قالب media-type مقادیر را می خواند.

برای مثال، در زیر یک نوع متد Web API Controller نشان داده شده است:

```
HttpResponseMessage Put(int id, Product item) { ... }
```

پارامتر id یک نوع داده ساده می باشد که Web API مقدار آن را با درخواست URI دریافت می کند.

پارامتر item یک نوع داده پیچیده محسوب می شود که Web API از قالب media-type مقدار را از درخواست دریافت می کند.

Web API با استفاده از اطلاعات مسیرها (Route) مقادیر را از URI دریافت می کند. مکانیزم مسیریابی، URI را دریافت می کند و مسیر را تفسیر می کند, آنگاه مقادیر را از مسیریابی استخراج می کند.

در ادامه این مقاله، به شما نشان می دهیم چگونه می توان عملیات model binding را سفارشی سازی کنیم. با این حال برای داده های پیچیده، به نظر می رسد هر گاه امکان داشته باشد می توان

از media-type formatter استفاده کرد. یک اصل کلیدی در HTTP این است که با استفاده از Content negotiation منابع، در پیام ارسال شده است .

استفاده کردن از [FromUri]

برای اینکه Web API را وادار کنیم یک نوع داده پیچیده را از URI دریافت کند ، [FromUri] را به یک پارامتر اضافه می کنیم. مثال زیر یک نوع GeoPoint همراه با یک متد Controller که GeoPoint را از URI می گیرد، تعریف می کند.

```
public class GeoPoint
{
    public double Latitude { get; set; }
    public double Longitude { get; set; }
}
public ValuesController : ApiController
{
    public HttpResponseMessage Get([FromUri] GeoPoint location) { ... }

   }
```

کاربر می تواند طول و عرض جغرافیایی مقادیر را در آدرس قرار دهد تا Web API از آن ها برای ساخت GeoPoint استفاده کند. برای مثال:

```
http://localhost/api/values/?Latitude=47.678558&Longitude=-122.130989
```

استفاده کردن از [FromBody]

برای اینکه Web API را وادار کنیم یک نوع داده ساده از درخواست دریافت کند، [FromBody] attribute را به پارامتر اضافه می کنیم:

```
public HttpResponseMessage Post([FromBody] string name) { ... }
```

در این مثال Web API از media-type formatter برای خواندن مقدار name در درخواست استفاده می کند.

در اینجا یک مثال از درخواست کاربر مشاهده می کنیم:

```
POST http://localhost:5076/api/values HTTP/1.1
User-Agent: Fiddler
Host: localhost:5076
Content-Type: application/json
```

```
Content-Length: 7
"Alice"
```

زمانی که یک پارامتر، attribute به نام [FromBody] دارد، Web API از سربرگ نوع محتوا (Content-Type header)برای انتخاب کردن قالب استفاده می کند. در این مثال، نوع محتوا application/json و درخواست از نوع رشته JSON خام و نه یک object از نوع JSON می باشد. حداکثر یک پارامتر اجازه ی دریافت پیام را دارد. بنابراین این کد جواب نمی دهد:

```
// Caution: Will not work!
public HttpResponseMessage Post([FromBody] int id, [FromBody] string name) { ... }
```

دلیل این اتفاق این است که درخواست تنها یک بار خوانده شده و در جایی ذخیره نمی شود.

تبدیل کننده نوع داده(Type Converters)

شما می توانید Web API را به صورت یک کلاس با داده ساده بسازید تا Web API بتواند با ساختن یک TypeConverter و تبدیل رشته ای از URI ، آن را bind کند.

کد زیر یک کلاس GeoPoint که یک نقطه جغرافیایی را مشخص می کند نشان می دهد به علاوه ی یک TypeConverter که مورد GeoPoint را به رشته ها تبدیل می کند. برای مشخص کردن نوع Converter ، کلاس GeoPoint از [TypeConverter] استفاده می کند.

```
[TypeConverter(typeof(GeoPointConverter))]
public class GeoPoint
{
    public double Latitude { get; set; }
    public double Longitude { get; set; }
    public static bool TryParse(string s, out GeoPoint result)
    {
        result = null;
        var parts = s.Split(',');
        if (parts.Length != 2)
        {
            return false;
        }
        double latitude, longitude;
        if (double.TryParse(parts[0], out latitude) &&
            double.TryParse(parts[1], out longitude))
        {
            result = new GeoPoint() { Longitude = longitude, Latitude = latitude };
            return true;
        }
```

```
        return false;
    }
}
class GeoPointConverter : TypeConverter
{
    public override bool CanConvertFrom(ITypeDescriptorContext context, Type sourceType)
    {
        if (sourceType == typeof(string))
        {
            return true;
        }
        return base.CanConvertFrom(context, sourceType);
    }
    public override object ConvertFrom(ITypeDescriptorContext context,
        CultureInfo culture, object value)
    {
        if (value is string)
        {
            GeoPoint point;
            if (GeoPoint.TryParse((string)value, out point))
            {
                return point;
            }
        }
        return base.ConvertFrom(context, culture, value);
    }
}
```

حالا Web API به صورت نوع ساده ی GeoPoint نشان داده می شود یعنی پارامترهای GeoPoint را از URI دریافت می کند. لازم نیست شما [FromUri] را به پارامتر اضافه کنید.

```
public HttpResponseMessage Get(GeoPoint location) { ... }
```

کاربر می تواند متد را با یک URI مثل متد زیر فراخوانی کند:

```
http://localhost/api/values/?location=47.678558,-122.130989
```

Model Binder ها

گزینه انعطاف پذیرتر از type converter این است که یک model binder سفارشی بسازید. شما می توانید با یک model binder به مقادیری مثل درخواست HTTP، توضیحات action و مقادیر خام از داده مسیر دسترسی داشته باشید.

برای ساختن یک model binder، رابط IModelBinder را پیاده سازی کنید. این رابط یک متد منفرد تعریف می کند، BindModel:

```
bool BindModel(HttpActionContext actionContext, ModelBindingContext bindingContext);
```

در اینجا یک model binder برای GeoPoint می بینید:

```
public class GeoPointModelBinder : IModelBinder
{
    // List of known locations.
    private static ConcurrentDictionary<string, GeoPoint> _locations
        = new ConcurrentDictionary<string, GeoPoint>(StringComparer.OrdinalIgnoreCase);
    static GeoPointModelBinder()
    {
        _locations["redmond"] = new GeoPoint() { Latitude = 47.67856, Longitude = -122.131 };
        _locations["paris"] = new GeoPoint() { Latitude = 48.856930, Longitude = 2.3412 };
        _locations["tokyo"] = new GeoPoint() { Latitude = 35.683208, Longitude = 139.80894 };
    }
    public bool BindModel(HttpActionContext actionContext, ModelBindingContext bindingContext)
    {
        if (bindingContext.ModelType != typeof(GeoPoint))
        {
            return false;
        }
        ValueProviderResult val = bindingContext.ValueProvider.GetValue(
            bindingContext.ModelName);
        if (val == null)
        {
            return false;
        }
        string key = val.RawValue as string;
        if (key == null)
        {
            bindingContext.ModelState.AddModelError(
                bindingContext.ModelName, "Wrong value type");
            return false;
        }
        GeoPoint result;
        if (_locations.TryGetValue(key, out result) || GeoPoint.TryParse(key, out result))
        {
            bindingContext.Model = result;
            return true;
        }
        bindingContext.ModelState.AddModelError(
            bindingContext.ModelName, "Cannot convert value to Location");
        return false;
    }
}
```

یک model binder مقادیر خام ورودی از یک Value provider دریافت می کند. این طراحی 2 تابع متمایز هم را جدا می کند:

- Value provider درخواست HTTP را می گیرد و یک dictionary از جفت کلید مقدار ها(key-value pairs) آماده می کند.

- Model binder از این dictionary برای پر کردن model استفاده می کند.

مقدار پیش فرض در Web API مقادیر را از داده های مسیر و آدرس دریافت می کند. برای مثال اگر URI به صورت زیر باشد:

```
http://localhost/api/values/1?location=48,-122
```

value provider شکل key-value pair را به صورت زیر طراحی می کند:

- id = "1"
- location = "48,122"

در خاصیت ModelBindingContext.ModelName، نام پارامتر برای bind کردن در آن ذخیره می شود. Model binder دیکشنری را برای پیدا کردن کلیدی که با این مقدار همخوانی داشته باشد جستجو می کند. اگر مقدار موجود باشد و بتواند به یک GeoPoint تبدیل شود، model binder یک مقدار حدی به خاصیت ModelBindingContext.Model اختصاص می دهد.

تنظیم کردن model binder

چندین راه برای تنظیم کردن model binderوجود دارد. یکی از آن ها این است که شما می توانید [ModelBinder] را به پارامتر خود اضافه کنید.

```
public HttpResponseMessage Get([ModelBinder(typeof(GeoPointModelBinder))] GeoPoint
location)
```

شما همچنین می توانید [ModelBinder] را به type آن اضافه کنید و با این کار،Web API از model binder مشخص برای تمامی پارامتر های آن type استفاده می شود.

```
[ModelBinder(typeof(GeoPointModelBinder))]
public class GeoPoint
{
    // ....
}
```

سرانجام شما می توانید model binder را به HttpConfiguration اضافه کنید. شما می توانید یک provider با اشتقاق از کلاس ModelBinderProvider بسازید. با این حال، اگر شما از یک نوع ساده استفاده می کنید،برای شما آسان تر خواهد بود از SimpleModelBinderProvider در حال اجرا که برای این منظور طراحی شده است استفاده کنید. کد زیر چگونگی عملکرد آن را توضیح می دهد:

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        var provider = new SimpleModelBinderProvider(
            typeof(GeoPoint), new GeoPointModelBinder());
        config.Services.Insert(typeof(ModelBinderProvider), 0, provider);
        // ...
    }
}
```

برای آنکه Web API بتواند از model binder استفاده کند، شما هنوز نیاز دارید اتربیوت [ModelBinder] را به پارامتر اضافه کنید.

```
public HttpResponseMessage Get([ModelBinder] GeoPoint location) { ... }
```

value providerها

باید این را هم اضافه کرد که model binder مقادیر را از یک value provider دریافت می کند. برای طراحی یک مقدار سفارشی، رابط IValueProvider را پیاده سازی کنید. در زیر مثالی را می بینید که مقادیر را از کوکی می گیرد:

```
public class CookieValueProvider : IValueProvider
{
    private Dictionary<string, string> _values;
    public CookieValueProvider(HttpActionContext actionContext)
    {
        if (actionContext == null)
        {
            throw new ArgumentNullException("actionContext");
        }
        _values = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase);
        foreach (var cookie in actionContext.Request.Headers.GetCookies())
        {
            foreach (CookieState state in cookie.Cookies)
            {
                _values[state.Name] = state.Value;
            }
        }
    }
    public bool ContainsPrefix(string prefix)
    {
        return _values.Keys.Contains(prefix);
```

```
    }
    public ValueProviderResult GetValue(string key)
    {
        string value;
        if (_values.TryGetValue(key, out value))
        {
            return new ValueProviderResult(value, value, CultureInfo.InvariantCulture);
        }
        return null;
    }
}
```

شما همچنین نیاز دارید تا یک value provider با اشتقاق از کلاس ValueProviderFactory بسازید:

```
public class CookieValueProviderFactory : ValueProviderFactory
{
    public override IValueProvider GetValueProvider(HttpActionContext actionContext)
    {
        return new CookieValueProvider(actionContext);
    }
}
```

به صورت زیر، value provider را به HttpConfiguration اضافه کنید:

```
public static void Register(HttpConfiguration config)
{
    config.Services.Add(typeof(ValueProviderFactory), new CookieValueProviderFactory());
    // ...
}
```

Web API همه ی value provider ها را می سازد و درنتیجه هرگاه model binder متد ValueProvider.GetValue را فراخوانی کند model binder مقدار را از اولین value provider که بتواند بسازد دریافت می کند.

در عوض شما می توانید value provider را در سطح پارامتر با استفاده از ValueProvider attribute به صورت زیر تنظیم کنید:

```
public HttpResponseMessage Get(
[ValueProvider(typeof(CookieValueProviderFactory))] GeoPoint location)
```

HttpParameterBinding

Model binder ها موارد خاص از یک مکانیزم کلی تر می باشند. اگر به [ModelBinder] دقت کنید متوجه خواهید شد که از کلاس ParameterBindingAttribute مشتق شده است. این کلاس یه متد منفرد تعریف می کند مانند GetBinding که یک HttpParameterBinding بر می گرداند:

```
public abstract class ParameterBindingAttribute : Attribute
```

```
{
    public abstract HttpParameterBinding GetBinding(HttpParameterDescriptor parameter);
}
```

یک HttpParameterBinding مسئول bind کردن یک پارامتر به یک مقدار می باشد. در شرایطی که [ModelBinder]، HttpParameterBinding را بر می گرداند، از یک IModelBinder برای bind کردن واقعی استفاده می شود. شما همچنین می توانید HttpParameterBinding خود را پیاده سازی کنید.

برای مثال، فرض کنید می خواهید از headerهای درون درخواست if-match و if-none-match ، Etag ها را دریافت کنید. برای شروع، یک کلاس که Etag ها را نشان دهد تعریف می کنیم:

```
public class ETag
{
    public string Tag { get; set; }
}
```

همچنین برای اینکه header مربوط به if-match یا if-none-match را دریافت کنیم می توانیم یک enum تعریف کنیم.

```
public enum ETagMatch
{
    IfMatch,
    IfNoneMatch
}
```

در کد زیر یک HttpParameterBinding که Etag را از header دلخواه دریافت و آن را به یک پارمتر از نوع Etag تبدیل می کند می بینید:

```
public class ETagParameterBinding : HttpParameterBinding
{
    ETagMatch _match;
    public ETagParameterBinding(HttpParameterDescriptor parameter, ETagMatch match)
        : base(parameter)
    {
        _match = match;
    }
    public override Task ExecuteBindingAsync(ModelMetadataProvider metadataProvider,
        HttpActionContext actionContext, CancellationToken cancellationToken)
    {
        EntityTagHeaderValue etagHeader = null;
        switch (_match)
        {
            case ETagMatch.IfNoneMatch:
                etagHeader = actionContext.Request.Headers.IfNoneMatch.FirstOrDefault();
                break;
```

```
            case ETagMatch.IfMatch:
                etagHeader = actionContext.Request.Headers.IfMatch.FirstOrDefault();
                break;
        }
        ETag etag = null;
        if (etagHeader != null)
        {
            etag = new ETag { Tag = etagHeader.Tag };
        }
        actionContext.ActionArguments[Descriptor.ParameterName] = etag;
        var tsc = new TaskCompletionSource<object>();
        tsc.SetResult(null);
        return tsc.Task;
    }
}
```

متد ExecuteBindingAsync عمل bind کردن را انجام می دهد. در این متد، مقدار پارامتر کران را به ActionArgument در HttpActionContext اضافه می کنیم.

اگر متد ExecuteBindingAsync شما درخواستی دریافت کند، برای برگرداندن true خاصیت WillReadBody را override می کند. درخواست تنها یک بار خوانده می شود و قابلیت ذخیره ندارد بنابراین Web API یک قانون که حداکثر یک پیام بتواند خوانده شود را اعمال می کند.

برای پیاده سازی یک HttpParameterBinding سفارشی، شما می توانید یک attribute که از ParameterBindingAttribute مشتق شده را تعریف کنید. برای ETagParameterBinding ما دو attribute برای if-match و دیگری برای if-none-match تعریف می کنیم که هر دو از یک کلاس مشتق می شوند.

```
public abstract class ETagMatchAttribute : ParameterBindingAttribute
{
    private ETagMatch _match;
    public ETagMatchAttribute(ETagMatch match)
    {
        _match = match;
    }
    public override HttpParameterBinding GetBinding(HttpParameterDescriptor parameter)
    {
        if (parameter.ParameterType == typeof(ETag))
        {
            return new ETagParameterBinding(parameter, _match);
        }
        return parameter.BindAsError("Wrong parameter type");
    }
}
public class IfMatchAttribute : ETagMatchAttribute
{
    public IfMatchAttribute()
        : base(ETagMatch.IfMatch)
    {
```

```
    }
}
public class IfNoneMatchAttribute : ETagMatchAttribute
{
    public IfNoneMatchAttribute()
        : base(ETagMatch.IfNoneMatch)
    {
    }
}
```

در زیر نیز یک متد Controller که از [IfNoneMatch] استفاده می کند نشان داده است:

```
public HttpResponseMessage Get([IfNoneMatch] ETag etag) { ... }
```

علاوه بر ParameterBindingAttribute یک روش زیرکانه برای اضافه کردن HttpParameterBinding سفارشی وجود دارد. در HttpConfiguration، خاصیت ParameterBindingRules یک مجموعه از توابع ناشناس(anomymous) از نوع (HttpParameterDescriptor -> HttpParameterBinding) می باشد. برای مثال، شما می توانید این قانون که هر پارامتر Etag در متد Get از ETagParameterBinding با if-none-match استفاده کند تعریف می کنیم.

```
config.ParameterBindingRules.Add(p =>
{
    if (p.ParameterType == typeof(ETag) &&
        p.ActionDescriptor.SupportedHttpMethods.Contains(HttpMethod.Get))
    {
        return new ETagParameterBinding(p, ETagMatch.IfNoneMatch);
    }
    else
    {
        return null;
    }
})
```

تابع باید در جایی که bind کردن انجام نمی گیرد مقدار null را برگرداند.

IActionValueBinder

تمام عملیات bind کردن پارامتر با یک سرویس به نام IActionValueBinder مدیریت می شود. پیاده سازی پیش فرض IActionValueBinder شامل مراحل زیر می باشد:

1- ParameterBindingAttribute را جستجو کنید که شامل [FromBody] و [FromUri] و [ModelBinder] یا attribute های سفارشی را شامل می شود.

2– همچنین، HttpConfiguration.ParameterBindingRules برای یک تابع که مقدار غیر خالی HttpParameterBinding بر می گرداند.

3–و نیز، از قوانینی که در قسمت های قبل توضیح دادم استفاده کنید:

- اگر نوع پارامتر ساده و یا یک type converter داشته باشد از URI بسط داده می شود. این عبارت معادل آن است که [FromUri] را روی پارامتر قرار دهید.
- همچنین، خواندن پارامتر از متن پیام. این معادل آن است که [FromBody] را روی پارامتر قرار دهیم.

در صورت نیاز شما می توانید کل سرویس IActionValueBinder را با یک پیاده سازی سفارشی، جا به جا کنید.