

Forum:[Hack Forums](#)

Thread: [MVC Custom ValueProvider for nested object?](#)

severos - July 12, 2017, 6:39 am

I'm writing a custom ValueProvider that will find encrypted values in the request and decrypt them so they can be used during Model Binding phase.

I seem to have most of the process done, I can decrypt the values and registered the value provider with the framework.

The decrypted string is actually a json string, a simple parsing to Dictionary<string,string> will allow me to provide values for the first level.

Now, that's the hardest thing I've found online, but what if I have a nested object?? how would I go for creating a value provider for it??

P.S: still need to have the ContainsPrefix function to work properly.

ShadowCloudGH - July 12, 2017, 10:17 pm

(Jul 12, 2017 06:39 AM)severos Wrote: [\[CITE\]](#) I'm writing a custom ValueProvider that will find encrypted values in the request and decrypt them so they can be used during Model Binding phase.

I seem to have most of the process done, I can decrypt the values and registered the value provider with the framework.

The decrypted string is actually a json string, a simple parsing to Dictionary<string,string> will allow me to provide values for the first level.

Now, that's the hardest thing I've found online, but what if I have a nested object?? how would I go for creating a value provider for it??

P.S: still need to have the ContainsPrefix function to work properly.

I might be oversimplifying the issue here, but can't you just serialize the object to JSON? That's what an AJAX request would most likely do anyway?

Have a look at this for some background:

<http://haacked.com/archive/2011/06/30/whatsquos-the-difference-between-a-value-provider-and-model-binder.aspx/>

EDIT:

I just read some of your other posts and with the context for this question, might I suggest a slightly different approach? Create a custom actionFilter to do the decryption for you :

PHP Code

```
<?php
public class CustomFilterAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        // This method is executed before calling the action
        // and here you have access to the route data:
```

```

var foo = filterContext.RouteData.Values["foo"];
// TODO: use the routeData to decrypt the values before it's passed to the controller
base.OnActionExecuting(filterContext);
}
}

```

And if you don't want to decorate all your controller actions with this action filter register it as a global in your global.asax's Application_Start event like this;

```

GlobalFilters.Filters.Add(new CustomFilterAttribute ()); // ADDED
RegisterGlobalFilters(GlobalFilters.Filters);

```

This was mostly taken from this as a source >>

<https://stackoverflow.com/questions/4104107/asp-net-mvc-event-that-happens-just-before-action-is-called>

severos - July 12, 2017, 10:48 pm

(Jul 12, 2017 10:17 PM)ShadowCloudGH Wrote: [\[CITE\]](#) I might be oversimplifying the issue here, but can't you just serialize the object to JSON? That's what an AJAX request would most likely do anyway?

Have a look at this for some background:

<http://haacked.com/archive/2011/06/30/whatsquos-the-difference-between-a-value-provider-and-model-binder.aspx/>

Actually what you mean is deserialize, I'm reading the request not writing it.

Problem is that I'm writing middleware, so I have no idea what's the output type required, I could deserialize JSON to dictionary as I said earlier, but then what about nested objects? consider the following:

Code {"field1":["value1","value2"], "field2":{"inner1":1, "inner2":2}}how would I write a value provider for that???? I have a strong feeling that I'm missing some key stuff related to ContainsPrefix function and how model binder deals with nested objects.

ShadowCloudGH - July 12, 2017, 10:52 pm

(Jul 12, 2017 10:48 PM)severos Wrote: [\[CITE\]](#) Actually what you mean is deserialize, I'm reading the request not writing it.

Problem is that I'm writing middleware, so I have no idea what's the output type required, I could deserialize JSON to dictionary as I said earlier, but then what about nested objects? consider the following:

Code {"field1":["value1","value2"], "field2":{"inner1":1, "inner2":2}}how would I write a value provider for that???? I have a strong feeling that I'm missing some key stuff related to ContainsPrefix function and how model binder deals with nested objects.

Have you seen the updated answer?

This will be the decrypted result from the actionFilter
"field2":{"inner1":1, "inner2":2}

So just deserialize these into JSON? I presume that once you have field2 you'll know what you

actually expected in terms of incoming values?

severos - July 12, 2017, 10:52 pm

Update:

Nope, using action filters won't work, you're assuming I have the value of Foo and want to decrypt it, in my case I have an encrypted object in the body, so the context doesn't even know there's something called Foo, and it'll be too late for ValueProvider or ModelBinder to kick in during ActionFilter.

(Jul 12, 2017 10:52 PM)ShadowCloudGH Wrote: [\[CITE\]](#) Have you seen the updated answer?

This will be the decrypted result from the actionFilter

```
"field2":{"inner1":1, "inner2":2}
```

So just deserialize these into JSON? I presume that once you have field2 you'll know what you actually expected in terms of incoming values?

Just added an update.

And no, just because I know there's something called field2 doesn't mean I know what's inside, I'm writing middleware to be consumed by others, so I have no idea what's the types they're going to use, and I don't want to limit them.

ShadowCloudGH - July 12, 2017, 11:07 pm

(Jul 12, 2017 10:52 PM)severos Wrote: [\[CITE\]](#) Update:

Nope, using action filters won't work, you're assuming I have the value of Foo and want to decrypt it, in my case I have an encrypted object in the body, so the context doesn't even know there's something called Foo, and it'll be too late for ValueProvider or ModelBinder to kick in during ActionFilter.

Just added an update.

And no, just because I know there's something called field2 doesn't mean I know what's inside, I'm writing middleware to be consumed by others, so I have no idea what's the types they're going to use, and I don't want to limit them.

Uhm... Foo was really just the layman's example, there's no reason you can't loop through all the values in the RouteValueDictionary?

The only requirement from the consuming code would be to expect a JSON string rather than an object? Apart from that you don't need to limit the objects or the levels of nesting?

severos - July 12, 2017, 11:13 pm

(Jul 12, 2017 11:07 PM)ShadowCloudGH Wrote: [\[CITE\]](#) Uhm... Foo was really just the layman's example, there's no reason you can't loop through all the values in the RouteValueDictionary?

The only requirement from the consuming code would be to expect a JSON string rather than an object? Apart from that you don't need to limit the objects or the levels of nesting?

Not really, you can loop through RouteValueDictionary only if it had the data, but in my case, it has no idea what the hell is that string in body is because it's encrypted. And the one doing decryption is the ValueProvider.

I've spent quite few days looking for other work arounds, the only thing that I can see it working is if I can alter the request body before it gets read by the MVC framework, but I believe rewriting the stream will break the whole request, and I'm not really sure it'll allow me to do it.

ShadowCloudGH - July 13, 2017, 12:52 am

(Jul 12, 2017 11:13 PM)severos Wrote: [\[CITE\]](#) Not really, you can loop through RouteValueDictionary only if it had the data, but in my case, it has no idea what the hell is that string in body is because it's encrypted. And the one doing decryption is the ValueProvider.

I've spent quite few days looking for other work arounds, the only thing that I can see it working is if I can alter the request body before it gets read by the MVC framework, but I believe rewriting the stream will break the whole request, and I'm not really sure it'll allow me to do it.

I'm not following...

1. (Encrypted)Request received by Managed Pipeline (IIS stuff before your code)
2. The encrypted request is received by the ActionFilter
3. In the actionfilter loop through the RouteValueDictionary and decrypt all values.
4. Serialize all decrypted values to JSON (To deal with nested objects)
5. Controllers now have decrypted JSON
6. YAY

If this still doesn't work for you, then creating a managed handler and injecting into your managed pipeline might be what you are looking for. This will allow you to decrypt earlier in the lifecycle, but won't solve your nested object issue.

severos - July 13, 2017, 2:34 am

1- Pipeline isn't IIS, I'm using .Net Core, but that's not really relevant much here.

2- The encrypted part is the body of request not the route values, also it's fully encrypted object - the whole thing is encrypted not just the values of fields - and it's encrypted as a whole not every part alone.

3- The whole process should be invisible to controllers, developers will still want to write it as they used to:

PHP Code

```
<?php
```

```
class MyController : Controller
```

```
{
    public string SomeAction(SomeClassHere s)
    {
```

```
//s was actually encrypted in the body, but value provider decrypted it and passed the value to mode
l binder.
```

```
    }
}
```

Ninja - July 13, 2017, 7:41 am

(Jul 12, 2017 10:52 PM)severos Wrote: [\[CITE\]](#) so the context doesn't even know there's something called Foo

Let it know then.

Buffer the JSON string with a type name or something to let it know what it is. Or, if you use public/private key encryption you can just decrypt it using the private key and get the clean JSON.

EDIT: If you are inheriting from like a default value provider can't you just decrypt the JSON and then call the base method?

severos - July 13, 2017, 11:57 am

(Jul 13, 2017 07:41 AM)Ninja Wrote: [\[CITE\]](#) Let it know then.

Buffer the JSON string with a type name or something to let it know what it is. Or, if you use public/private key encryption you can just decrypt it using the private key and get the clean JSON.

EDIT: If you are inheriting from like a default value provider can't you just decrypt the JSON and then call the base method?

Wow... that didn't come to my mind, I'll see if there's a default value provider and inherit it.

Ninja - July 13, 2017, 8:13 pm

(Jul 13, 2017 11:57 AM)severos Wrote: [\[CITE\]](#) Wow... that didn't come to my mind, I'll see if there's a default value provider and inherit it.

Long problem solving frustrations can make you blind to many obvious solutions. So did you manage to get it to work? I am pretty interested in the solution.

severos - July 13, 2017, 10:09 pm

Well, there doesn't seem to be a default value provider, however there are couple providers (Form, JQuery form, Query string, Route) but no idea which one should I use, also I'm not sure they'll work as intended since they have their own way of accessing the data through actionContext. I'll be taking a look at their sources to see what they have, might inspire me for the solution.

Snoweyes - July 14, 2017, 8:48 am

(Jul 13, 2017 10:09 PM)severos Wrote: [\[CITE\]](#) Well, there doesn't seem to be a default value provider, however there are couple providers (Form, JQuery form, Query string, Route) but no idea which one should I use, also I'm not sure they'll work as intended since they have their own way of accessing the data through actionContext. I'll be taking a look at their sources to see what they have, might inspire me for the solution.

Can't you just create a modelbinder which reads the data from your source (for example the body of the request).

In this example i read the body and convert it to a string, but you can ofcourse do anything you want.

Example model binder:

PHP Code

```
<?php
public class MyCustomModelBinder : IModelBinder
{
    public Task BindModelAsync(ModelBindingContext bindingContext)
    {
        var request = bindingContext.HttpContext.Request;
        var data = new StreamReader(request.Body).ReadToEnd();

        bindingContext.Result = ModelBindingResult.Success(data);

        return Task.CompletedTask;
    }
}
```

Usage:

PHP Code

```
<?php
[HttpPost]
public IEnumerable<string> ReadData([ModelBinder(BinderType = typeof(MyCustomModelBinder
))] string data)
```

```
{
    return Enumerable.Empty<string>();
}
```

Keep in mind the above code example is made with .NET Core.

Ninja - July 14, 2017, 8:53 am

(Jul 14, 2017 08:48 AM)Snoweyes Wrote: [\[CITE\]](#) Can't you just create a modelbinder which reads the data from your source (for example the body of the request).

In this example i read the body and convert it to a string, but you can ofcourse do anything you want.

Example model binder:

PHP Code

```
<?php
public class MyCustomModelBinder : IModelBinder
{
    public Task BindModelAsync(ModelBindingContext bindingContext)
    {
        var request = bindingContext.HttpContext.Request;
        var data = new StreamReader(request.Body).ReadToEnd();

        bindingContext.Result = ModelBindingResult.Success(data);

        return Task.CompletedTask;
    }
}
```

Usage:

PHP Code

```
<?php
[HttpPost]
public IEnumerable<string> ReadData([ModelBinder(BinderType = typeof(MyCustomModelBinder))] string data)
{
    return Enumerable.Empty<string>();
}
```

Keep in mind the above code example is made with .NET Core.

He wants everything done under the hood for him, so the developer still use the controllers as normal, while under the hood the encryption and stuff is handled.

Snoweyes - July 14, 2017, 10:54 am

(Jul 14, 2017 08:53 AM)Ninja Wrote: [\[CITE\]](#) He wants everything done under the hood for him, so the developer still use the controllers as normal, while under the hood the encryption and stuff is handled.

Ahh. I didn't understood. Still the model binder could be provided by a model binder provider and automatically converted. Maybe you can make this generic enough, so developers have minimal work and only have to implement the middleware. See example:

<http://www.dotnet-programming.com/post/2017/03/17/Custom-Model-Binding-in-Aspnet-Core-2-Model-Binding-Interfaces.aspx>

severos - *July 14, 2017, 11:02 am*

I can create a model binder, that was my initial thinking. However, this isn't really how the framework does things inside, I want to do it the 'right' way, ValueProvider is responsible for parsing values from various places in request (route, query, body, headers, etc...) but it's not responsible for whatever model. The ModelBinder on the other hand is responsible to create the model and bind values they get from ValueProviders.