FromBody string parameter is giving null

24

2

This is probably something very basic, but I am having trouble figuring out where I am going wrong.

I am trying to grab a string from the body of a POST, but "jsonString" only shows as null. I also want to avoid using a model, but maybe this isn't possible. The piece of code that I am hitting with PostMan is this chunk:

```
[Route("Edit/Test")]
[HttpPost]
public void Test(int id, [FromBody] string jsonString)
{
    ...
}
```

Maybe it is something I am doing incorrectly with postman, but I have been trying to use "=test" (as seen in other questions asked about this topic) in the value section of the body - x-www-form-urlencoded section with the key as jsonString and nothing. I have also tried using raw - text and raw - text/plain. I get the id so I know the url is correct. Any help with this would be greatly appreciated.

PostMan is set up like this currently:

```
POST http://localhost:8000/Edit/Test?id=111
key = id  value = 111
Body - x-www-form-urlencoded
key = jsonString  value = "=test"
```

c# asp.net-web-api asp.net-web-api2 postman asp.net-web-api-routing

edited Dec 6 '18 at 10:43
NH Nkosi
129k 21 158 225

asked Nov 28 '16 at 20:48
Robert Prine
648 2 9 15

Related question for Asp.Net Core stackoverflow.com/questions/31952002/… – Michael Freidgeim Nov 18 '17 at 1:18

I have been batling with this for two days and after reading every article I could find about it, it turned out to be as simple as formatting the JSON string correctly in the WebRequest: The data must start and end with double quotes (I.E. Add double quotes inside your string of data around the json data) and if you then use single quotes throughout your json data it all plays nice. – Gineer Jun 7 '18 at 13:49

8 Answers

35

By declaring the jsonString parameter with `[FromBody]` you tell ASP.NET Core to use the input formatter to bind the provided JSON (or XML) to a model. So your test should work, if you provide a simple model class

```
public class MyModel
{
    public string Key {get; set;}
}

[Route("Edit/Test")]
[HttpPost]
public void Test(int id, [FromBody] MyModel model)
{
    ... model.Key....
}
```

and a sent JSON like

```
{
    key: "value"
}
```

Of course you can skip the model binding and retrieve the provided data directly by accessing `HttpContext.Request` in the controller. The `HttpContext.Request.Body` property gives you the content stream or you can access the form data via `HttpContext.Request.Forms` .

I personally prefer the model binding because of the type safety.

2 In question was mentioned: *I also want to avoid using a model...* – Fabio Nov 28 '16 at 21:08

2 @Fabio - then [FromBody] will not work, because it tells the framework you want to bind the data to a model class. In order to avoid the binding, skip this parameter and access the sent data directly as hinted in the last paragraph. Hope that will help. – Ralf Bönning Nov 28 '16 at 21:10

This worked for me - you just have to use RAW instead of FORM in the POST options for POSTMan. – Codeman Jun 19 '18 at 23:48

36

Referencing Parameter Binding in ASP.NET Web API

> ### Using [FromBody]
>
> To force Web API to read a simple type from the request body, add the **[FromBody]** attribute to the parameter:
>
> ```
> [Route("Edit/Test")]
> [HttpPost]
> public IHttpActionResult Test(int id, [FromBody] string jsonString) { ... }
> ```
>
> In this example, Web API will use a media-type formatter to read the value of *jsonString* from the request body. Here is an example client request.
>
> ```
> POST http://localhost:8000/Edit/Test?id=111 HTTP/1.1
> User-Agent: Fiddler
> Host: localhost:8000
> Content-Type: application/json
> Content-Length: 6
>
> "test"
> ```
>
> When a parameter has [FromBody], Web API uses the Content-Type header to select a formatter. In this example, the content type is "application/json" and the request body is a raw JSON string (not a JSON object).

In the above example no model is needed if the data is provided in the correct format in the body.

For URL encoded a request would look like this

```
Content-Length: 5

=test
```

answered Nov 29 '16 at 2:37

NH Nkosi
129k 21 158 225

2 Exactly! That little = solves the problem. I remember it took me many days when I worked on my first project, because it was not mentioned in any web api book I had. :(– Al Kepp Jan 31 '18 at 22:52

13

When having [FromBody]attribute, the string sent should not be a raw string, but rather a JSON string as it includes the wrapping quotes:

```
"test"
```

Based on https://weblog.west-wind.com/posts/2017/Sep/14/Accepting-Raw-Request-Body-Content-in-ASPNET-Core-API-Controllers

Similar answer string value is Empty when using FromBody in asp.net web api

edited Jan 10 '18 at 9:27

answered Jan 10 '18 at 9:20

Michael Freidgeim
14.4k 6 94 117

6

You are on the right track.

On your header set

See also POST string to ASP.NET Web Api application - returns null

edited May 23 '17 at 12:34
Community ♦
1 1

answered Nov 28 '16 at 21:26
Igor
44.4k 4 55 114

3

In my case I forgot to use JSON.stringify(bodyStuff).

answered Jun 1 '18 at 8:36
Eugene
351 3 9

you saved my life buddy. – Khizar Iqbal May 2 at 12:00

this is what solved my issues, thanks! – ragnarswanson May 22 at 15:34

2

I know this answer is kinda old and there are some very good answers who already solve the problem.

In order to expand the issue I'd like to mention one more thing that has driven me crazy for the last 4 or 5 hours.

It is **VERY VERY VERY** important that your properties in your model class have the **set** attribute enabled.

This **WILL NOT** work (parameter still null):

```
/* Action code */
[HttpPost]
public Weird NOURLAuthenticate([FromBody] Weird form) {
    public string UserPwd {get;}
}
```

This **WILL** work:

```
/* Action code */
[HttpPost]
public Weird NOURLAuthenticate([FromBody] Weird form) {
    return form;
}
/* Model class code */
public class Weird {
    public string UserId {get; set;}
    public string UserPwd {get; set;}
}
```

answered May 1 '18 at 17:57

Some random IT boy

534 1 4 18

0

Finally got it working after 1 hour struggle.

This will remove null issue, also gets the JSON *key1*'s value of *value1*, in a generic way (no model binding), .

For a new WebApi 2 application example:

> Postman (looks exactly, like below):

```
POST    http://localhost:61402/api/values   [Send]

              Body
                   (*) raw             JSON (application/json) v

"{  \"key1\": \"value1\" }"
```

```
using Newtonsoft.Json;
// ..

// POST api/values
[HttpPost]
public object Post([FromBody]string jsonString)
{
    // add reference to Newtonsoft.Json
    //  using Newtonsoft.Json;

    // jsonString to myJsonObj
    var myJsonObj = JsonConvert.DeserializeObject<Dictionary<string, dynamic>>
(jsonString);

    // value1 is myJsonObj[key1]
    var valueOfkey1 = myJsonObj["key1"];

    return myJsonObj;
}
```

All good for now, not sure if model binding to a class is required if I have sub keys, or, may be DeserializeObject on sub key will work.

answered Aug 24 '18 at 11:12
Manohar Reddy Poreddy
6,946 57 59

0

After a long nightmare of fiddling with Google and trying out the wrong code in Stack Overflow I discovered changing ([FromBody] string model) to ([FromBody] object model) does wonders please not i am using .NET 4.0 yes yes i know it s old but ...

answered Jun 24 at 9:16
Kudakwashe Mafutah
1 2