**Created, CreatedAtAction, CreatedAtRoute Methods In ASP.NET Core Explained With Examples**

In this post we will discuss **Created**, **CreatedAtAction** and **CreatedAtRoute** methods available in ASP.NET Core controllers and, in particular, answer such questions as what it is and why it is needed, how do these methods work and how to use them, how to pass parameters and create link for an action in a different controller, examples how to use different overloaded versions, common errors we might encounter and how to resolve them.

**Contents:**

* What and Why
  + Why Do I Need Location Header?
* Difference Between Created, CreatedAtAction and CreatedAtRoute
* Created Explained
  + Created Examples
    - Created (string uri, object value)
    - Created (Uri uri, object value)
* CreatedAtAction Explained
  + CreatedAtAction Examples
    - CreatedAtAction (string actionName, object routeValues, object value)
    - CreatedAtAction (string actionName, string controllerName, object routeValues, object value)
    - CreatedAtAction (string actionName, object value)
* CreatedAtRoute Explained
  + CreatedAtRoute Examples
    - CreatedAtRoute (string routeName, object routeValues, object value)
    - CreatedAtRoute (string routeName, object value)
    - CreatedAtRoute (object routeValues, object value)
* About **object routeValues** parameter
* CreatedAtRoute vs CreatedAtAction
* Common Errors and Troubleshooting

**What and Why**

So, first of all, let us understand what these methods are and what they do.

**Created, CreatedAtAction, CreatedAtRoute and their overloads are methods of ControllerBase class, they provide convenient ways to return 201 Created response from Web API that signifies a successful request completion. Response includes a Location header with an URI that can be used to retrieve newly created resource.**

So, the ***key points*** about these methods are:

* In contrast to other methods (Ok, NotFound, BadRequest), they create a response with status code 201 and a **Location** header.
* They accept an object as the last parameter, this object will be returned as a response body. Usually, it is just the value of newly created resource.

**Why Do I Need Location Header?**

It is a ***convention*** to include Location header in 201 Created HTTP response. It is a good practice to follow this convention, however, it is not strictly required because this is only a “SHOULD” in RFC 2616.

Another question is do we need to return 201 Created instead of simple 200 OK response. Short answer is - yes, we should return 201 response when a resource is created in case we follow REST architectural style. But this is another topic which is not discussed in this post.

**Difference Between Created, CreatedAtAction and CreatedAtRoute**

As a result, these methods and their overloads produce similar HTTP responses. However, t are some differences to consider:

* ***The main difference*** is how you specify the URI of the created resource which will be included in the Location header.
* **Created** gives you ***more control*** over URI creation, w as **CreatedAtAction** and **CreatedAtRoute** gives ***more safety***.
* If your server is behind ***load balancer*** or ***reverse proxy***, you might want to use **Created** method to craft URI based on **X-Forwarded-...** headers from the request.

Which method to use depends on the use case and your preference. Next, we will discuss each of them in detail.

**Created Explained**

This method is very straightforward and has only two overloaded versions. Detailed signatures of both methods can be found  .

Since **Created** method is very simple, it needs only these two pieces of information:

* **uri** - simply the URI that should be returned in the **Location** header
* **value** - content to return in a response body

Basically, the main difference between them is how you pass the value of the location header - as **string** or **Uri**.

***Important points*** about passing location header using these methods:

* We are responsible for ensuring URI correctness and existence on the server at all times, for example, when we update paths or arguments.
* Using **string** means that value will be returned as is, without any validation.
* **Uri** class helps ensure that the URI format is correct. However, it doesn’t check that this path exists on the server.
* Value can be either absolute or relative. Examples below use absolute URI.

**Created Examples**

The following code snippets demonstrate sample usage of **Created** method overloads.

**Created (string uri, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet("{id}")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**string uri = $"https://www.example.com/api/values/{createdResource.Id}?version={createdResource.Version}";**

**return Created(uri, createdResource);**

**// Location: https://www.example.com/api/values/1?version=1.0**

**}**

**}**

**Created (Uri uri, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet("{id}")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**Uri uri = new Uri($"https://www.example.com/api/values/{createdResource.Id}?version={createdResource.Version}");**

**return Created(uri, createdResource);**

**// Location: https://www.example.com/api/values/1?version=1.0**

**}**

**}**

**CreatedAtAction Explained**

This method provides more support in generating URI for the Location header.

As the name suggests, this method allows us to set Location URI of the newly created resource by ***specifying the name of an action*** w we can retrieve our resource.

To achieve that, ASP.NET Core framework might need the following information depending on how and w you define your action:

* **actionName** - by default it is controller method name but can also be assigned using **[ActionName("...")]** attribute
* **controllerName** - name of the controller w our action resides
* **routeValues** - info necessary to generate a correct URL, for example, path or query parameters
* **value** - content to return in a response body

Overloaded version are only different in the way they handle **controllerName** and **routeValues** parameters because:

* **controllerName** - can be omitted if action is in the same controller
* **routeValues** - not needed in case your action doesn’t have any route parameters

Next, let us look at examples of how to use each of the overloaded methods.

**CreatedAtAction Examples**

Below are examples how we can use each of the **CreatedAtAction** method overloads.

**Note:** response **Location** header will contain your base URI instead of **...**, for example, **https://localhost:5001/api/Values/1?version=1.0** in case of local development.

**CreatedAtAction (string actionName, object routeValues, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet("{id}")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var actionName = nameof(GetValue);**

**var routeValues = new { id = createdResource.Id, version = createdResource.Version };**

**return CreatedAtAction(actionName, routeValues, createdResource);**

**// Location: .../api/Values/1?version=1.0**

**}**

**}**

**CreatedAtAction (string actionName, string controllerName, object routeValues, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesV2Controller : ControllerBase**

**{**

**[HttpGet("{id}")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**}**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet]**

**public IActionResult GetValue()**

**{**

**var value = "Value";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var actionName = nameof(ValuesV2Controller.GetValue);**

**var controllerName = "ValuesV2";**

**var routeValues = new { id = createdResource.Id, version = createdResource.Version };**

**return CreatedAtAction(actionName, controllerName, routeValues, createdResource);**

**// Location: .../api/ValuesV2/1?version=1.0**

**}**

**}**

**CreatedAtAction (string actionName, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet]**

**[ActionName("RetrieveValue")]**

**public IActionResult GetValue()**

**{**

**var value = "Value";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var actionName = "RetrieveValue";**

**return CreatedAtAction(actionName, createdResource);**

**// Location: .../api/Values**

**}**

**}**

**CreatedAtRoute Explained**

In my opinion, this method is more interesting and a bit harder to understand than others, but at the end it’s just an another way to specify location header in a 201 Created response.

In this case framework might need the following:

* **routeName** - name of the route, it could be assigned for a particular action or declared in the Startup class
* **routeValues** - info necessary to generate a correct URL, for example, path or query parameters
* **value** - same as in other methods, simply content of a response body

Usually, the main source of confusion is **routeName** parameter since we have to set it by ourselves. In the examples we’ll see how to specify a name for a route.

**Note:** t is an interesting and often less discussed overload **CreatedAtRoute (object routeValues, object value)** which does not accept **routeName**, it is covered  .

**CreatedAtRoute Examples**

**Note:** response **Location** header will contain your base URI instead of **...**, for example, **https://localhost:5001/api/Values/1?version=1.0** in case of local development.

**CreatedAtRoute (string routeName, object routeValues, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet("{id}", Name = "NameForGetValueEndpoint")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var routeValues = new { id = createdResource.Id, version = createdResource.Version };**

**return CreatedAtRoute("NameForGetValueEndpoint", routeValues, createdResource);**

**// Location: .../api/Values/1?version=1.0**

**}**

**}**

**CreatedAtRoute (string routeName, object value)**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet(Name = "NameForGetValueEndpoint")]**

**public IActionResult GetValue()**

**{**

**var value = "Value";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**return CreatedAtRoute("NameForGetValueEndpoint", createdResource);**

**// Location: .../api/Values**

**}**

**}**

**CreatedAtRoute (object routeValues, object value)**

I think this overload is less commonly used (I can be wrong) and that’s why deserves some comments.

**Example 1**

By default, if no information about target route is passed, it will take ***the path of the current method*** and use it for creating **Location** header.

In the following example it takes the route of POST method and ***attaches all route values as query parameters***. More about **routeValues** read in the next section.

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpGet]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var routeValues = new { id = createdResource.Id, version = createdResource.Version };**

**return CreatedAtRoute(routeValues, createdResource);**

**// Location: .../api/Values?id=1&version=1.0**

**}**

**}**

**Example 2**

You can ***specify action and controller inside of routeValues*** and, as a result, get behavior similar to the one of **CreatedAtAction** method. As we discussed before, we can omit **controller** if both methods are inside the same controller.

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesV2Controller : ControllerBase**

**{**

**[HttpGet("{id}")]**

**public IActionResult GetValue(int id, string version)**

**{**

**var value = $"Value {id} of version {version}";**

**return Ok(value);**

**}**

**}**

**[Route("api/[controller]")]**

**[ApiController]**

**public class ValuesController : ControllerBase**

**{**

**[HttpPost]**

**public IActionResult Create()**

**{**

**var createdResource = new { Id = 1, Version = "1.0" };**

**var routeValues = new**

**{**

**action = nameof(ValuesV2Controller.GetValue),**

**controller = "ValuesV2",**

**id = createdResource.Id,**

**version = createdResource.Version**

**};**

**return CreatedAtRoute(routeValues, createdResource);**

**// Location: .../api/ValuesV2/1?version=1.0**

**}**

**}**

**About object routeValues parameter**

In many overloaded versions of **CreatedAtRoute** and **CreatedAtAction** we can pass **routeValues** parameter. Let’s understand what it is and how to use it.

Route values are used by ASP.NET Core framework to generate the correct URI. are the values that can be passed:

* **action** - name of a controller method
* **controller** - name of a controller ***without*** **Controller** suffix
* **area** - optional level of organization for web apps, not used for web apis: documentation
* path parameters - e.g. **id** parameter in the path from examples above
* query parameters - all other parameters that are not recognized as those described above will be attached as part of a query string

**Type of routeValues parameter**

Most often, an anonymous type is used, it is clean and easy to use.

However, since the accepting type is **object**, we can pass any type. If the value is **IEnumerable** of **KeyValuePair**, for example, **Dictionary**, then its entries will be copied. Otherwise the object is interpreted as a set of key-value pairs w (property name, property value) is a key-value pair.

**CreatedAtRoute vs CreatedAtAction**

Now we know a lot about these two methods and, t fore, can discuss them together. are some ***key points*** to consider:

* Both of them provide different ways of achieving the same thing - returning 201 Created response with a **Location** header
* Both of them check correctness and existence of the target action, thus, ensuring validness of the returned URI
* **CreatedAtAction** finds target action by method and controller names **CreatedAtRoute** finds target action by a route name
* **CreatedAtAction** requires action name, default is controller method name but can be assigned with **ActionName** attribute **CreatedAtRoute** requires a name of the target route, it can be assigned to particular route by us or declared in Startup
* **CreatedAtRoute** covers functionality of **CreatedAtAction** when using overload that doesn’t require **routeName** parameter

As always, which one to use depends on the use case. In general, **CreatedAtRoute** gives more options and includes functionality of **CreatedAtAction**. However, in some cases **CreatedAtAction** is more convenient, for example, when handling actions inside of the same controller.

**Common Errors and Troubleshooting**

If you encounter errors, for example, **No route matches the supplied values** or **Cannot resolve action**, you might want to check the following things:

* Correct overloaded method is used - it could happen that another version is used because of the argument types, not the one you intended to use
* All necessary route parameters for the target action are passed as part of **routeValues** argument
* **CreatedAtAction** - if target action is in another controller, then **controllerName** is provided
* **CreatedAtAction** - **controllerName** does ***not*** contain **Controller** suffix
* **CreatedAtAction** - t was an issue with **actionName** containing **Async** suffix in some versions of ASP.NET Core
* **CreatedAtRoute** - target route has a unique name assigned to it

← PrevNext →↑ Top ↑

© 2023 Oleksandr Chzhen