Request Validation

Request validation is used to ensure that the incoming request message is properly formatted and contains the proper attributes. You can set up request validators in an API's OpenAPI definition file and then import the OpenAPI definitions into API Gateway. You can also set them up in the API Gateway console or by calling the API Gateway REST API, AWS CLI, or one of the AWS SDKs.

The x-amazon-apigateway-request-validators object allows you to define custom validators for your API, for this there are three extensions that can be used:

- x-amazon-apigateway-request-validators - Defines the supported request validators for the containing API as a map between a validator name and the associated request validation rules. This extension applies to a REST API.
- x-amazon-apigateway-request-validators.requestValidator - Specifies the validation rules of a request validator as part of the x-amazon-apigateway-request-validators object map definition.
- x-amazon-apigateway-request-validator - Specifies a request validator, by referencing a request_validator_name of the x-amazon-apigateway-request-validators object map, to enable request validation on the containing API or a method. The value of this extension is a JSON string.

> The x-amazon-apigateway-request-validator extension can be specified at the API level or at the method level. The API-level validator applies to all of the methods unless it is overridden by the method-level validator.

Create and configure request validators for your first API

In this section, we will configure three request validators, being:

- **All** - This validator is for both body and parameters validation.
- **Body** - This is the body-only validator.
- **Params** - This is the parameters-only validator.

We also will configure the extension x-amazon-apigateway-request-validator to use the **Body** validator.

1. Go back to the AWS Cloud9 console.
2. Add a new **x-amazon-apigateway-request-validators** and **x-amazon-apigateway-request-validator** to your file `openapi.yaml`.

   Review the code and then **copy/paste** it into the `openapi.yaml` file

```
1    openapi: "3.0.1"
2    info:
3      title: "module-2-my-first-api"
4      description: "First API with IaC example"
5      version: "1.0"
6
7    x-amazon-apigateway-request-validators:
8        All:
9            validateRequestParameters: true
10           validateRequestBody: true
11       Body:
12           validateRequestParameters: false
13           validateRequestBody: true
14       Params:
15           validateRequestParameters: true
16           validateRequestBody: false
17
18   x-amazon-apigateway-request-validator: Body
19
20   paths:
21
22       /pricepermeter:
23           post:
24               consumes:
25               - "application/json"
26               produces:
27               - "application/json"
28               responses:
29                   "200":
30                       description: "200 response"
31               x-amazon-apigateway-integration:
32                   httpMethod: "POST"
33                   credentials:
34                       Fn::GetAtt: [LambdaExecutionRole, Arn]
35                   uri:
36                     Fn::Sub: "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${CostCalculator.Arn}/invocations"
37                   responses:
38                       default:
39                           statusCode: "200"
40                           responseTemplates:
41                               application/json: |
42                                   #set($allParams = $input.params())
43                                   {
44                                   "body-json" : $input.json('$'),
45                                   "params" : {
46                                   #foreach($type in $allParams.keySet())
47                                       #set($params = $allParams.get($type))
48                                   "$type" : {
49                                       #foreach($paramName in $params.keySet())
50                                       "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
51                                           #if($foreach.hasNext),#end
52                                       #end
53                                   }
54                                       #if($foreach.hasNext),#end
55                                   #end
56                                   },
57                                   "stage-variables" : {
58                                   #foreach($key in $stageVariables.keySet())
59                                   "$key" : "$util.escapeJavaScript($stageVariables.get($key))"
60                                       #if($foreach.hasNext),#end
61                                   #end
62                                   },
63                                   "context" : {
64                                       "account-id" : "$context.identity.accountId",
65                                       "api-id" : "$context.apiId",
66                                       "api-key" : "$context.identity.apiKey",
67                                       "authorizer-principal-id" : "$context.authorizer.principalId",
68                                       "caller" : "$context.identity.caller",
69                                       "cognito-authentication-provider" : "$context.identity.cognitoAuthenticationProvider",
70                                       "cognito-authentication-type" : "$context.identity.cognitoAuthenticationType",
71                                       "cognito-identity-id" : "$context.identity.cognitoIdentityId",
72                                       "cognito-identity-pool-id" : "$context.identity.cognitoIdentityPoolId",
73                                       "http-method" : "$context.httpMethod",
74                                       "stage" : "$context.stage",
75                                       "source-ip" : "$context.identity.sourceIp",
76                                       "user" : "$context.identity.user",
77                                       "user-agent" : "$context.identity.userAgent",
78                                       "user-arn" : "$context.identity.userArn",
79                                       "request-id" : "$context.requestId",
80                                       "resource-id" : "$context.resourceId",
81                                       "resource-path" : "$context.resourcePath"
82                                       }
83                                   }
84                   requestTemplates:
85                       application/json: |
86                           #set($inputRoot = $input.path("$"))
87                           {
88                               "price": "$inputRoot.price",
89                               "size": "$inputRoot.size",
90                               "unit": "$inputRoot.unit",
91                               "downPaymentAmount": $inputRoot.downPayment
92                           }
93                   passthroughBehavior: "when_no_match"
94                   type: "aws"
95
96   components:
97       schemas:
98           costCalculatorRequest:
99               type: object
100              properties:
101                  price:
102                      type: number
103                  size:
104                      type: number
105                  unit:
106                      type: string
107                  downPayment:
108                      type: number
```

On **line 7 to 16** we are defining the **All**, **Body** and **Params** validators.

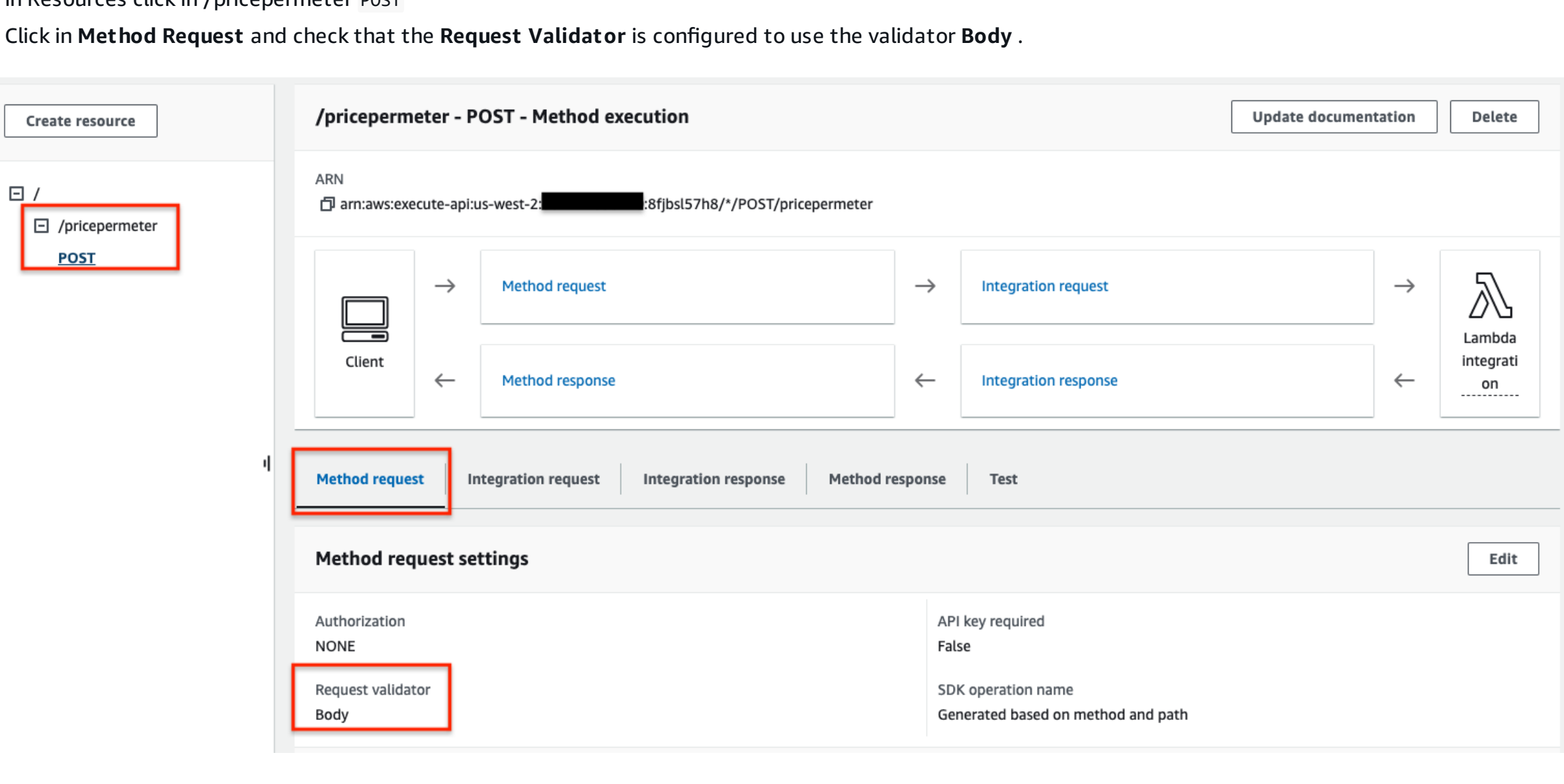
On **line 18** we are defining that the entire API paths will use the validator **Body**.

> **Important**
> Remember that you can override the API validator method by specifying the x-amazon-apigateway-request-validator inside the method.

3. Deploy the project again using SAM to verify that the **Request Validator** was added to your API Method Request.

```
sam build && sam deploy
```

4. Check if the stack was successfully updated and open the Amazon API Gateway console.
5. Choose your REST API named, `module-2-my-first-api`.
6. In Resources click in /pricepermeter `POST`
7. Click in **Method Request** and check that the **Request Validator** is configured to use the validator **Body** .



The next step is to map the **costCalculatorRequest** model to the **Request Body**. After configuring the **Request Body** you should be able to validate the request body against the model that you defined previously.

Configure the Request Body for request validation

1. Go back to the AWS Cloud9 console.
2. Add a new requestBody object to your file `openapi.yaml`.

   Review the code and then **copy/paste** it into the `openapi.yaml` file

```
1    openapi: "3.0.1"
2    info:
3      title: "module-2-my-first-api"
4      description: "First API with IaC example"
5      version: "1.0"
6
7    x-amazon-apigateway-request-validators:
8        All:
9            validateRequestParameters: true
10           validateRequestBody: true
11       Body:
12           validateRequestParameters: false
13           validateRequestBody: true
14       Params:
15           validateRequestParameters: true
16           validateRequestBody: false
17
18   x-amazon-apigateway-request-validator: Body
19
20   paths:
21
22       /pricepermeter:
23           post:
24               consumes:
25               - "application/json"
26               produces:
27               - "application/json"
28               responses:
29                   "200":
30                       description: "200 response"
31               requestBody:
32                   required: true
33                   content:
34                       application/json:
35                           schema:
36                               $ref: '#/components/schemas/costCalculatorRequest'
37               x-amazon-apigateway-integration:
38                   httpMethod: "POST"
39                   credentials:
40                       Fn::GetAtt: [LambdaExecutionRole, Arn]
41                   uri:
42                     Fn::Sub: "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${CostCalculator.Arn}/invocations"
43                   responses:
44                       default:
45                           statusCode: "200"
46                           responseTemplates:
47                               application/json: |
48                                   #set($allParams = $input.params())
49                                   {
50                                   "body-json" : $input.json('$'),
51                                   "params" : {
52                                   #foreach($type in $allParams.keySet())
53                                       #set($params = $allParams.get($type))
54                                   "$type" : {
55                                       #foreach($paramName in $params.keySet())
56                                       "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
57                                           #if($foreach.hasNext),#end
58                                       #end
59                                   }
60                                       #if($foreach.hasNext),#end
61                                   #end
62                                   },
63                                   "stage-variables" : {
64                                   #foreach($key in $stageVariables.keySet())
65                                   "$key" : "$util.escapeJavaScript($stageVariables.get($key))"
66                                       #if($foreach.hasNext),#end
67                                   #end
68                                   },
69                                   "context" : {
70                                       "account-id" : "$context.identity.accountId",
71                                       "api-id" : "$context.apiId",
72                                       "api-key" : "$context.identity.apiKey",
73                                       "authorizer-principal-id" : "$context.authorizer.principalId",
74                                       "caller" : "$context.identity.caller",
75                                       "cognito-authentication-provider" : "$context.identity.cognitoAuthenticationProvider",
76                                       "cognito-authentication-type" : "$context.identity.cognitoAuthenticationType",
77                                       "cognito-identity-id" : "$context.identity.cognitoIdentityId",
78                                       "cognito-identity-pool-id" : "$context.identity.cognitoIdentityPoolId",
79                                       "http-method" : "$context.httpMethod",
80                                       "stage" : "$context.stage",
81                                       "source-ip" : "$context.identity.sourceIp",
82                                       "user" : "$context.identity.user",
83                                       "user-agent" : "$context.identity.userAgent",
84                                       "user-arn" : "$context.identity.userArn",
85                                       "request-id" : "$context.requestId",
86                                       "resource-id" : "$context.resourceId",
87                                       "resource-path" : "$context.resourcePath"
88                                       }
89                                   }
90                   requestTemplates:
91                       application/json: |
92                           #set($inputRoot = $input.path("$"))
93                           {
94                               "price": "$inputRoot.price",
95                               "size": "$inputRoot.size",
96                               "unit": "$inputRoot.unit",
97                               "downPaymentAmount": $inputRoot.downPayment
98                           }
99                   passthroughBehavior: "when_no_match"
100                  type: "aws"
101
102  components:
103      schemas:
104          costCalculatorRequest:
105              type: object
106              properties:
107                  price:
108                      type: number
109                  size:
110                      type: number
111                  unit:
112                      type: string
113                  downPayment:
114                      type: number
```

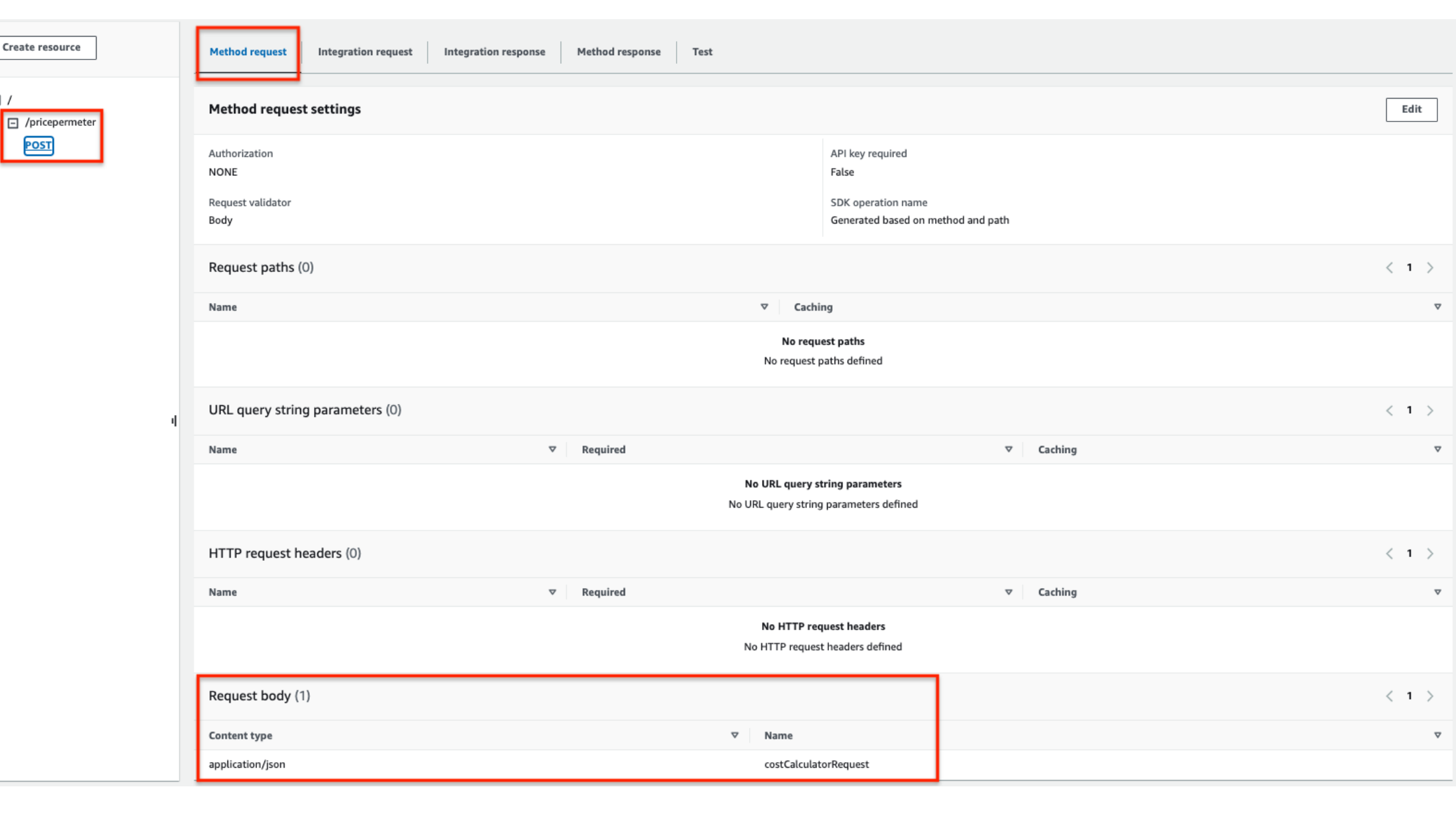
On **line 31** we are defining the **requestBody** object to the resource /pricemeter, method `POST`.

On **line 36** we are referencing the schema the contains the definition for the `costCalculatorRequest` that was previously defined in this module.

3. Deploy the project again using SAM to verify that the **Request Body** was mapped to your API Method Request.

```
sam build && sam deploy
```

4. Check if the stack was successfully updated and open the Amazon API Gateway console.
5. Choose your REST API named, `module-2-my-first-api`.
6. In Resources click in /pricepermeter `POST`
7. Click in **Method Request**, scroll down to the **Request Body** section and see that the model **costCalculatorRequest** is mapped to your **Request Body**.

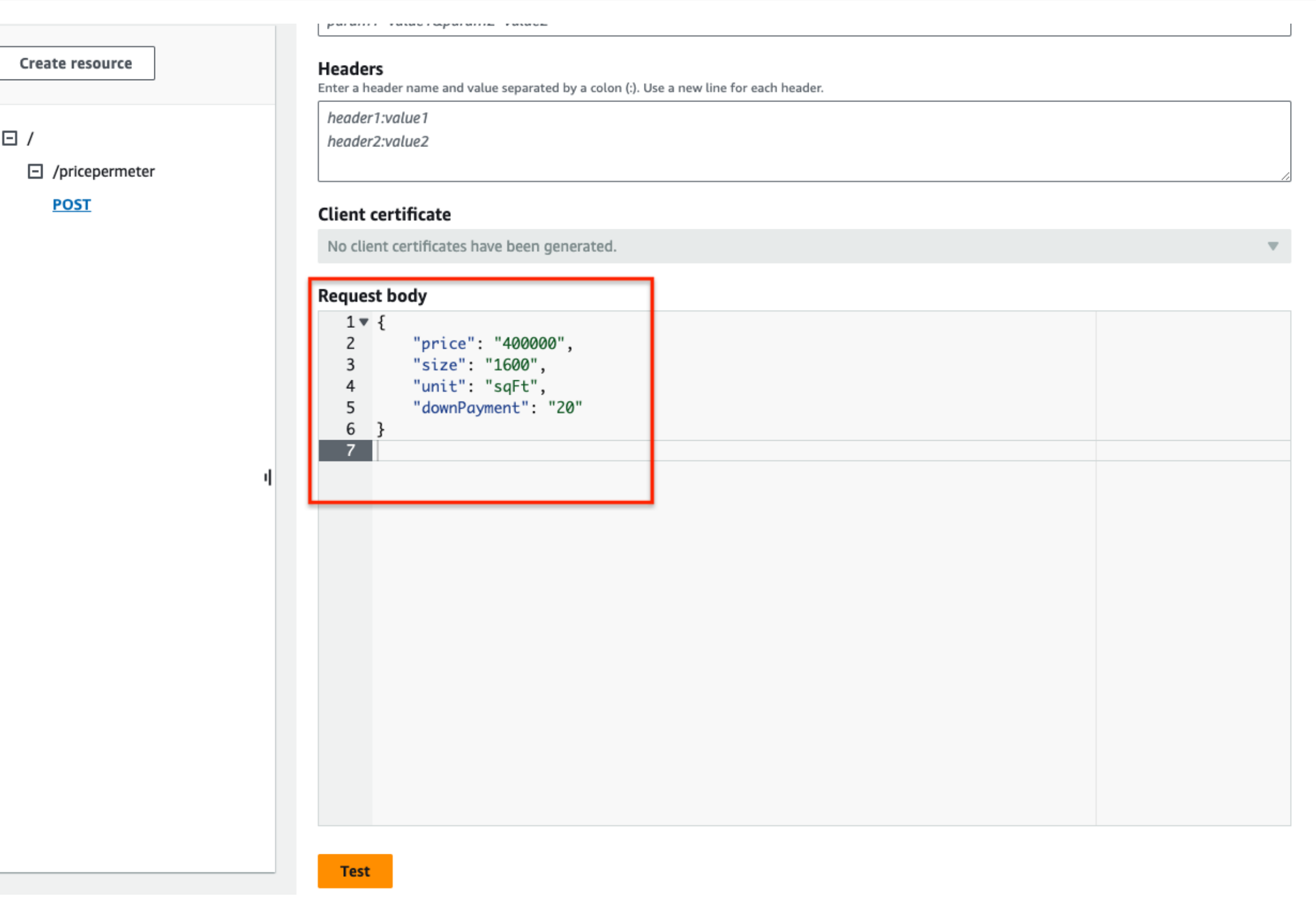


Test the API

Go back to the API Gateway console and select the POST integration method to test out the API.

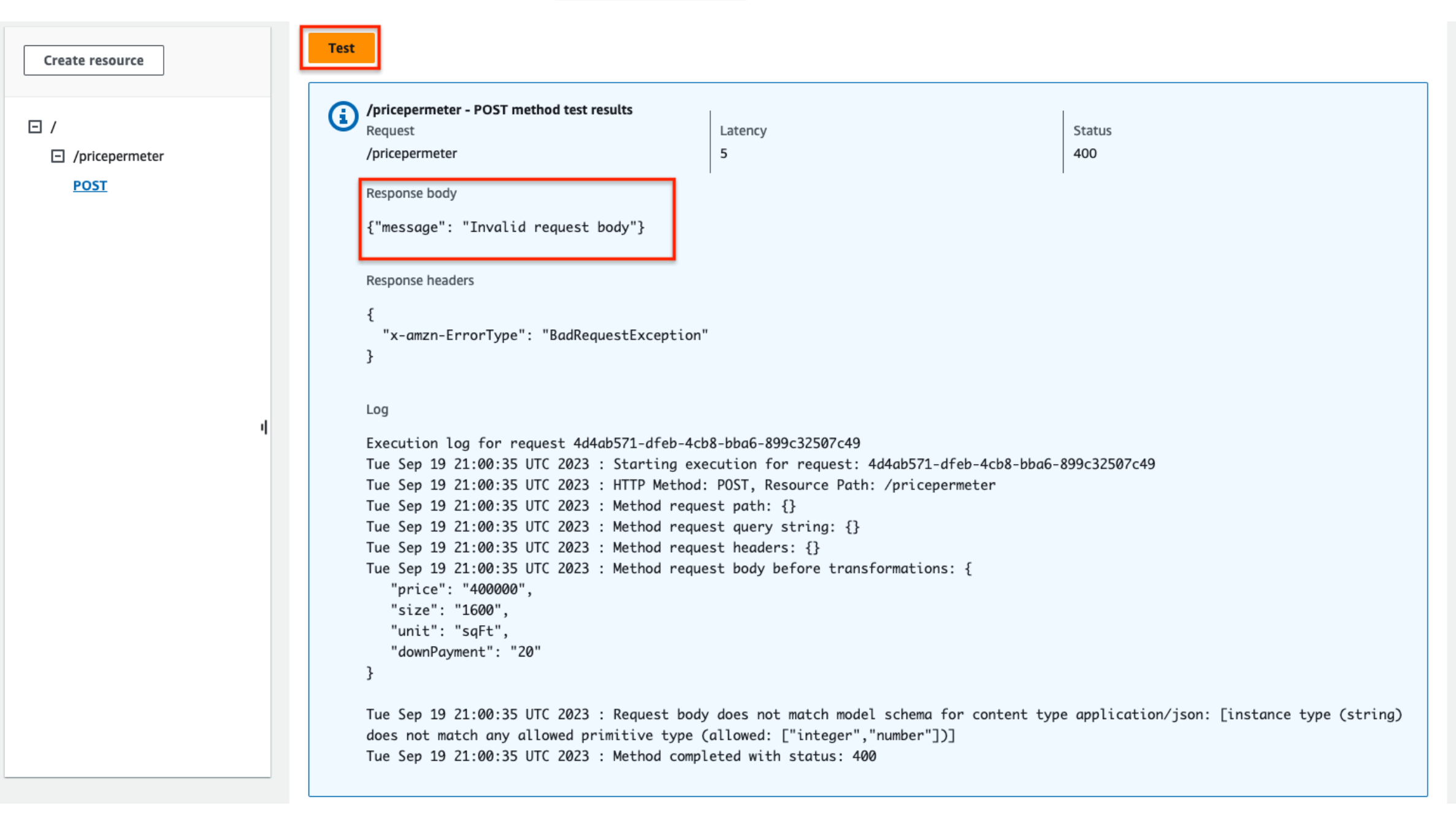
1. Click on the **Test** option to provide a sample request message.
2. Copy and Paste the following JSON Sample in the Request *Body* section

```
{
  "price": "400000",
  "size": "1600",
  "unit": "sqFt",
  "downPayment": "20"
}
```



3. Click on **Test**

   Notice that this time, the message failed with a message of `Invalid request body`.



   The request failed validation because price, size and downPayment are defined as **number** in our model/schema. Remove the string **quotes (" ")** from price, size, and downPayment and try your request again.

4. Click on the **Test** icon to provide a sample request message.
5. Copy and Paste the following JSON Sample in the *Request Body* section

```
{
  "price": 400000,
  "size": 1600,
  "unit": "sqFt",
  "downPayment": 20
}
```

6. Click on **Test**

You should see that the normal response is returned by the service.