# TOOL DEVELOPMENT

## CONSUME WEB API'S (ASYNCHRONOUSLY)

A DIGITAL ARTS AND ENTERTAINMENT PRESENTATION

## HTTP REST API

CONCEPT

## GOAL OF AN HTTP REST API

- There is a server that contains data
  - All student data on the Howest server
  - Personal data (name, picture, …)
  - Enrollment data (modules, ..)
  - Results data
  - …
  - RESOURCES

- (Part of) this data needs to be accessible through a web client
  - The user can receive a list of all modules that are available in Howest
  - The user can authorize himself as a student to see its results
  - …
  - REST Services define what resources are available, and how

## RESTful PARTS

1. Resources

2. Request verbs
3. Request headers
4. Request body

5. Response body
6. Respone status code

http rest api

## RESTful PARTS - RESOURCES

- Any RESOURCE that:
  - Is on the server (data, image, ….),
  - Is made available by the web service
    - might have extra authentication, …
    - is available through an endpoint
- an API endpoint:
  - makes a recource available through the cloud
  - has a base API url (eg.: https://deckofcardsapi.com/api/)
  - has a sub path per resource (eg.: https://deckofcardsapi.com/api/deck/new/)
  - might allow extra key-value pairs in the querystring
    - ✓ ?key1=value1&key2=value2&...
    - ✓ eg.:

https://deckofcardsapi.com/api/deck/new/?deck_count=2&cards=AS,2S,KS,AD

http rest api

## RESTful PARTS

1. Resources
2. Request verbs
3. Request headers
4. Request body
5. Response body
6. Respone status code

## HTTP REQUESTS

USING HTTP REQUEST WITH A REST API

http rest api

## RESTful PARTS - REQUESTS

- ➢ Consumer sends a REQUEST, eg.:
  - ✓ get me a list of all available modules
  - ✓ update my personal data
  - ✓ create a new enrolment for me
  - ➢ only predefined actions can be performed!

- ➢ How is this REQUEST made?
  - ✓ By entering the url (endpoint) in a browser
  - ✓ By sending an HTTP Request in C# code
  - ✓ By making a request in POSTMAN
  - ✓ ...

https://www.postman.com/downloads/

http rest api

## RESTful PARTS

1. Resources

2. **Request verbs**
3. Request headers
4. Request body

5. Response body
6. Respone status codes

- What do you want to do?
  - ✓ get some resource(s)
  - ✓ send new data to be saved
  - ✓ change / delete existing data
  - ✓ …

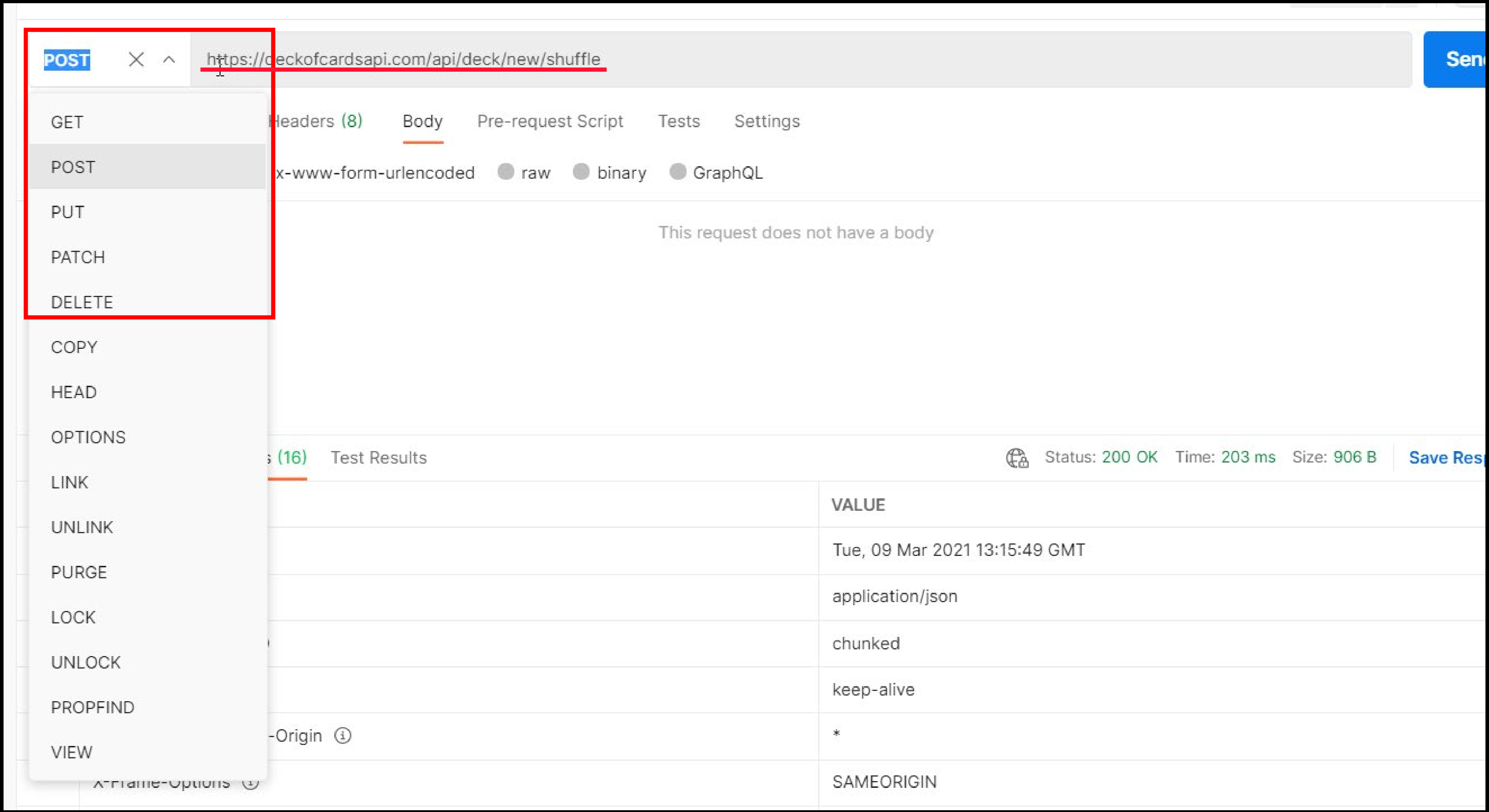
- REST http methods, usually:

| | | |
|---|---|---|
| ✓ | **GET**: | get a list of data, details, file,… |
| ✓ | **POST**: | add/create new data |
| ✓ | **PUT**: | change (update) existing data |
| ✓ | **DELETE**: | delete exisiting data |

  - *This is a only a recommendation.*

http rest api

# POSTMAN EXAMPLE: REQUEST VERBS


POST

https://deckofcardsapi.com/api/deck/new/shuffle

Sen

GET
POST
PUT
PATCH
DELETE
COPY
HEAD
OPTIONS
LINK
UNLINK
PURGE
LOCK
UNLOCK
PROPFIND
VIEW

Headers (8) Body Pre-request Script Tests Settings

x-www-form-urlencoded ● raw ● binary ● GraphQL

This request does not have a body

s (16) Test Results

Status: 200 OK Time: 203 ms Size: 906 B Save Res

| | VALUE |
|---|---|
| | Tue, 09 Mar 2021 13:15:49 GMT |
| | application/json |
| | chunked |
| | keep-alive |
| -Origin | * |
| X-Frame-Options | SAMEORIGIN |

http rest api

## RESTful PARTS

1. Resources

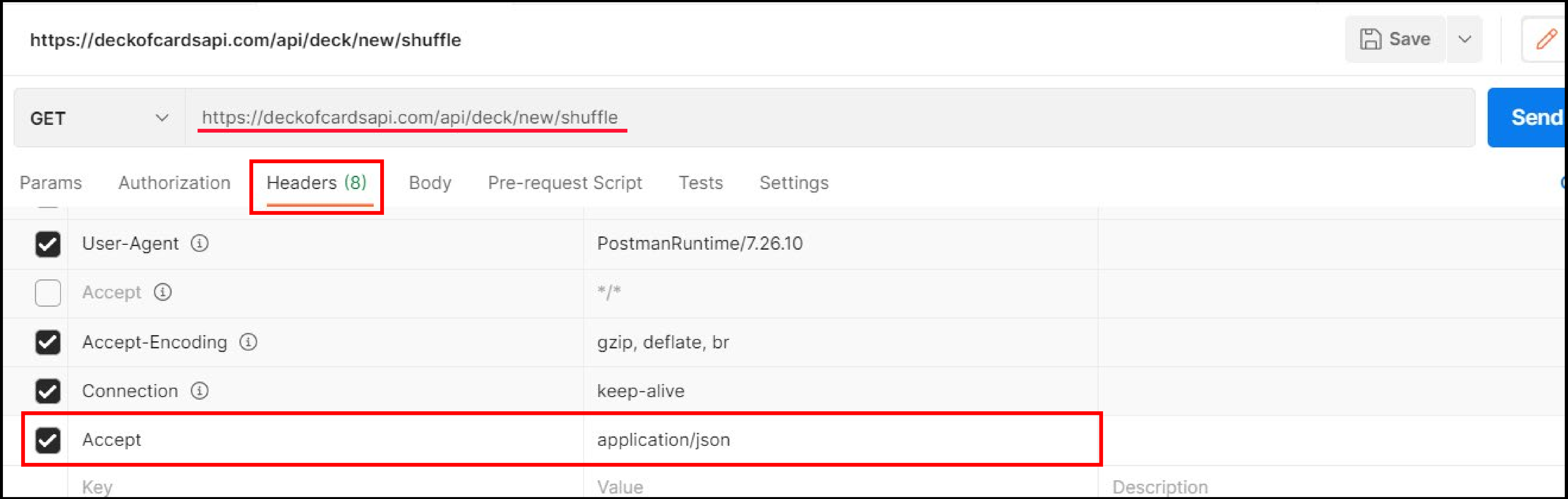
2. Request verbs
3. **Request headers**
4. Request body

5. Response body
6. Respone status code

- Extra information on how to make the request, eg.:
  - A key to authorize if necessary
  - The expected format of the result (xml, JSON,…) if possible by API
  - ….
  - not the same as querystring!

http rest api

## POSTMAN EXAMPLE: REQUEST HEADERS



https://deckofcardsapi.com/api/deck/new/shuffle

GET https://deckofcardsapi.com/api/deck/new/shuffle

| | Key | Value | Description |
|---|---|---|---|
| ✓ | User-Agent | PostmanRuntime/7.26.10 | |
| ☐ | Accept | */* | |
| ✓ | Accept-Encoding | gzip, deflate, br | |
| ✓ | Connection | keep-alive | |
| ✓ | Accept | application/json | |

http rest api

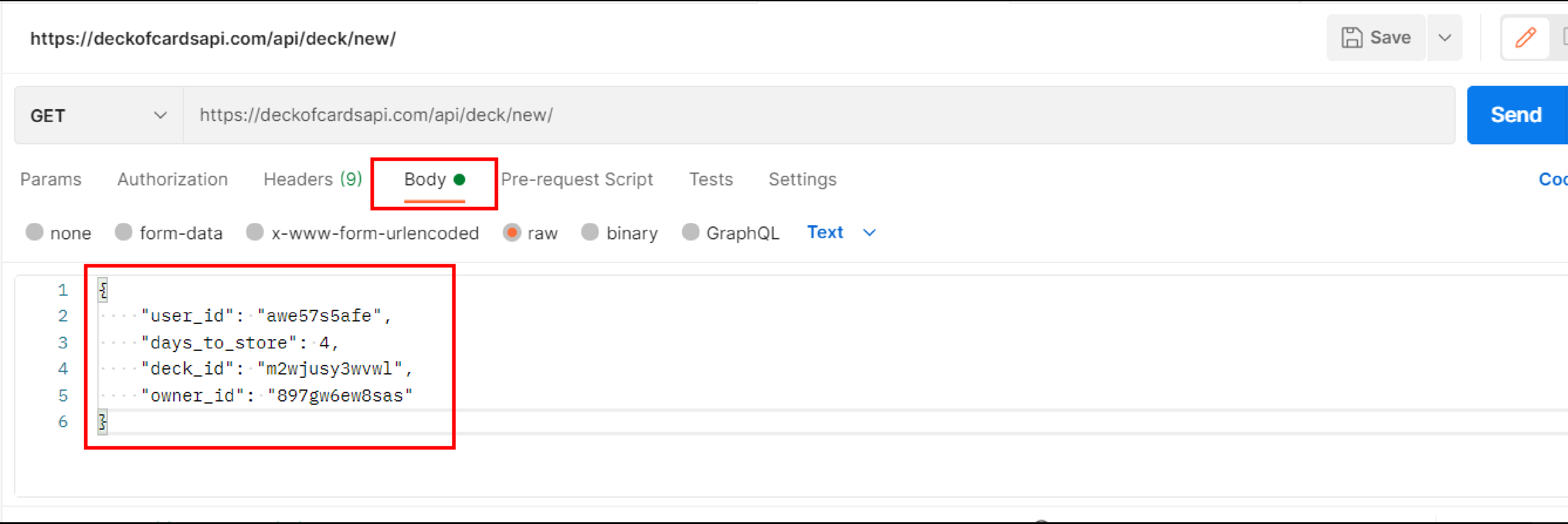
# RESTful PARTS

1. Resources
2. Request verbs
3. Request headers
4. **<u>Request body</u>**
5. Response body
6. Respone status code

- Information / data to send along with the request, eg.:
  - ✓ a new Enrollment object to be added
  - ✓ updated data of an existing student
  - ✓ ...
  - serialized data, eg. JSON format

http rest api

## POSTMAN EXAMPLE: REQUEST BODY



```
{
    "user_id": "awe57s5afe",
    "days_to_store": 4,
    "deck_id": "m2wjusy3wvwl",
    "owner_id": "897gw6ew8sas"
}
```

## HTTP RESPONSE

HANDLING RESPONSES FROM THE HTML REST API

RESTful webservice

## RESTful PARTS

1. Resources

2. Request verbs
3. Request headers
4. Request body

5. Response body
6. Respone status code

## RESTful PARTS - RESPONSE

- Consumer receives a RESPONSE, eg.:
  - ✓ the list of all available modules to enrol in `(after GET request)`
  - ✓ an error message: not authorized
  - ✓ a success code for the enrolment registration `(after POST request)`
  - ✓ ....

- This is the result of a request and:
  - ✓ gives a CODE to indicate the result
    - was it a success? If not, why?
  - ✓ might contain one or more resources
    - following the request `(GET/POST/PUT/..)`

# RESTful PARTS

1. Resources

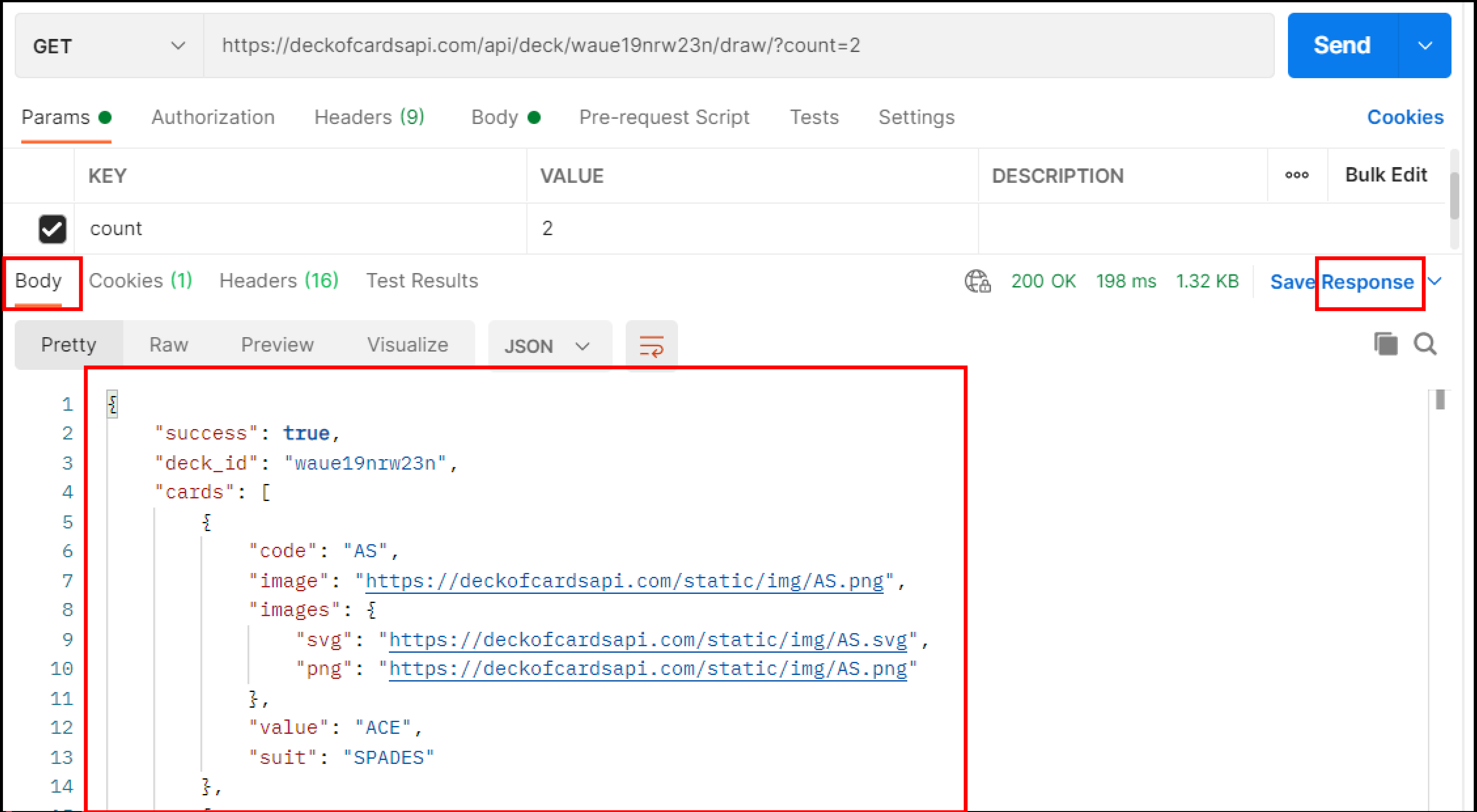
2. Request verbs
3. Request headers
4. Request body

5. **Response body**
6. Respone status code

- ➢ RESOURCE received based on request
  - ✓ a list of Enrollments (after GET)
  - ✓ the picture of a student
  - ✓ the Enrollment that was just saved in the database (after POST)
  - ✓ an error message if something went wrong
  - ✓ ...
  - ➢ serialized data, eg. JSON format

http rest api

## POSTMAN EXAMPLE: RESPONSE BODY (OK)



https://deckofcardsapi.com/api/deck/waue19nrw23n/draw/?count=2

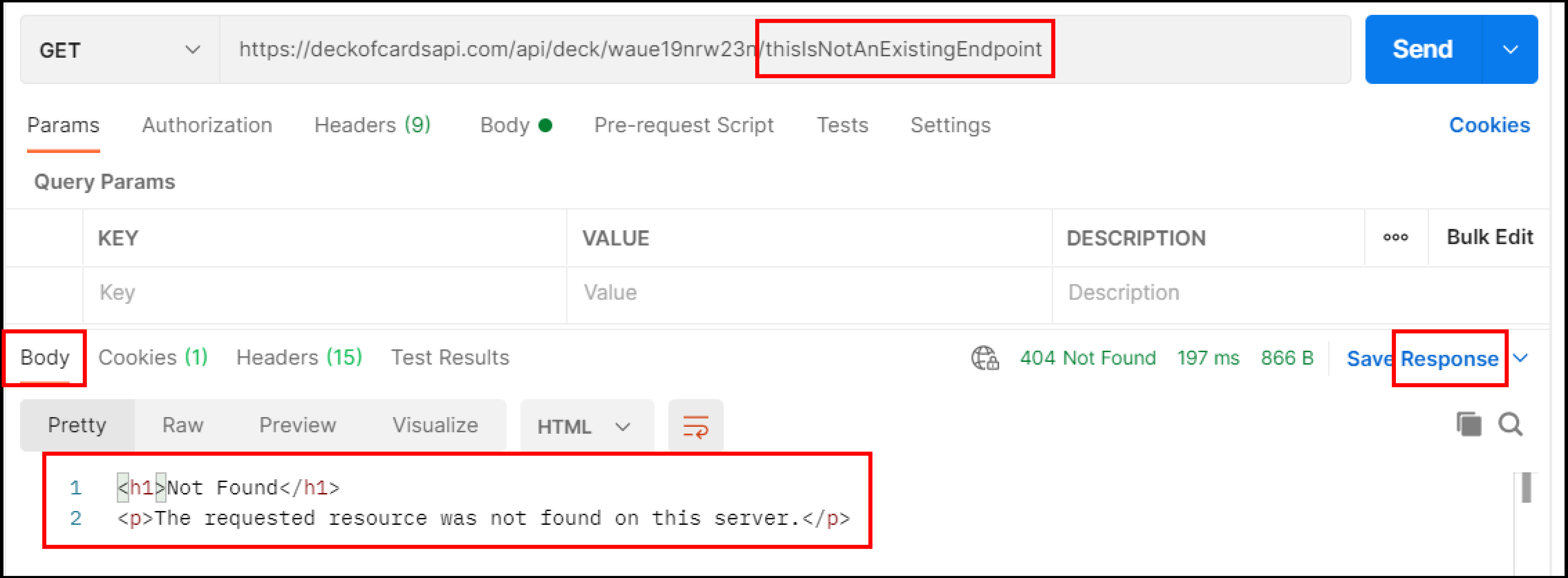
| KEY | VALUE | DESCRIPTION |
|---|---|---|
| count | 2 | |

200 OK 198 ms 1.32 KB

```
{
    "success": true,
    "deck_id": "waue19nrw23n",
    "cards": [
        {
            "code": "AS",
            "image": "https://deckofcardsapi.com/static/img/AS.png",
            "images": {
                "svg": "https://deckofcardsapi.com/static/img/AS.svg",
                "png": "https://deckofcardsapi.com/static/img/AS.png"
            },
            "value": "ACE",
            "suit": "SPADES"
        },
```

http rest api

# POSTMAN EXAMPLE: RESPONSE BODY (ERROR)



GET https://deckofcardsapi.com/api/deck/waue19nrw23n/thisIsNotAnExistingEndpoint

| KEY | VALUE | DESCRIPTION |
|---|---|---|
| Key | Value | Description |

404 Not Found 197 ms 866 B

```
<h1>Not Found</h1>
<p>The requested resource was not found on this server.</p>
```

RESTful webservice

# RESTful PARTS

1. Resources

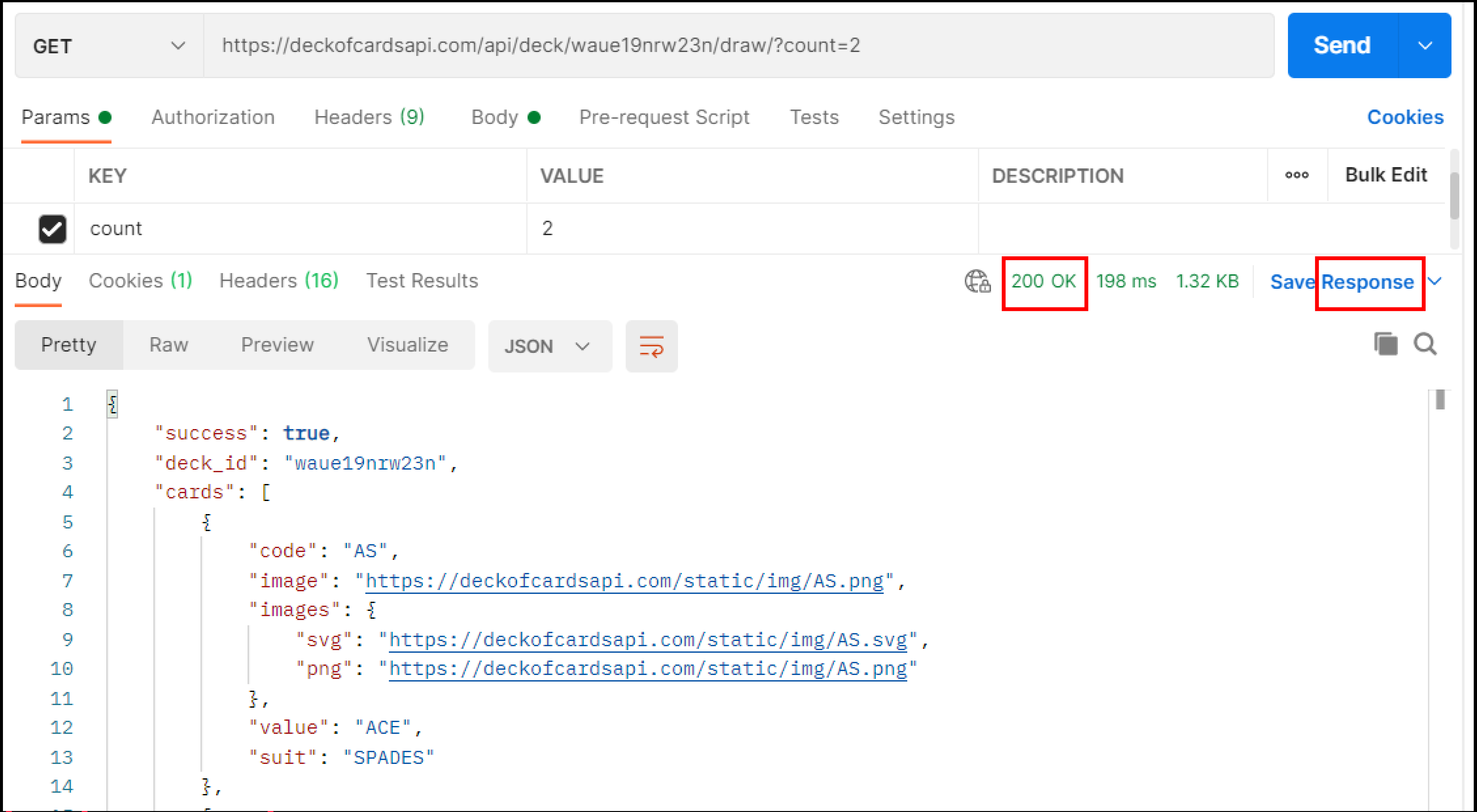
2. Request verbs
3. Request headers
4. Request body

5. Response body
6. **Respone status code**

- ➢ CODE to indicate response STATUS
  - ✓ a code for success (eg. 200 after GET)
  - ✓ successfully updated (eg. 201 after PUT)
  - ✓ endpoint not found (Not Found 404)
  - ✓ unauthorized (eg. 401)
  - ✓ …
  - ➢ these are not fixed; webservice decides
  - ➢ however, there are recommendations:

https://developer.mozilla.org/nl/docs/Web/HTTP/Status

# POSTMAN EXAMPLE: STATUS CODE (OK)



GET https://deckofcardsapi.com/api/deck/waue19nrw23n/draw/?count=2

Params ● | Authorization | Headers (9) | Body ● | Pre-request Script | Tests | Settings | Cookies

| KEY | VALUE | DESCRIPTION |
|---|---|---|
| ✓ count | 2 | |

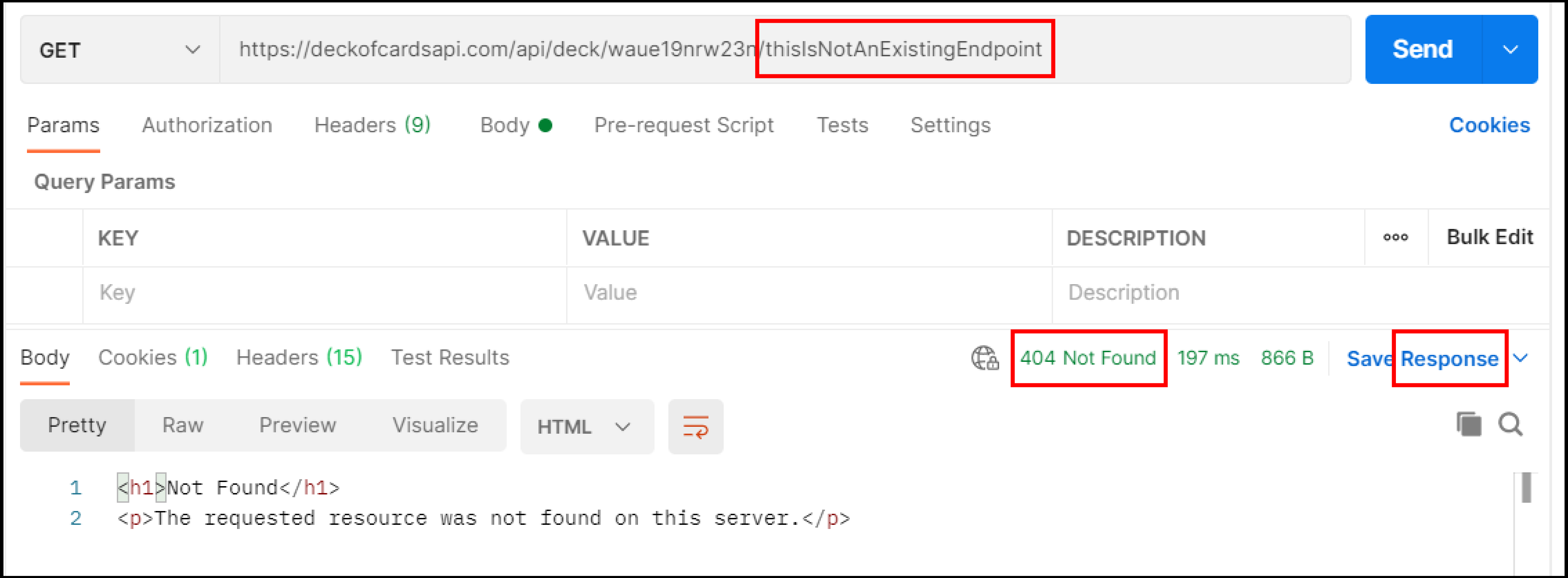
Body | Cookies (1) | Headers (16) | Test Results — 200 OK 198 ms 1.32 KB Save Response

Pretty | Raw | Preview | Visualize | JSON

```
{
    "success": true,
    "deck_id": "waue19nrw23n",
    "cards": [
        {
            "code": "AS",
            "image": "https://deckofcardsapi.com/static/img/AS.png",
            "images": {
                "svg": "https://deckofcardsapi.com/static/img/AS.svg",
                "png": "https://deckofcardsapi.com/static/img/AS.png"
            },
            "value": "ACE",
            "suit": "SPADES"
        },
```

http rest api

## POSTMAN EXAMPLE: RESPONSE BODY (ERROR)



https://deckofcardsapi.com/api/deck/waue19nrw23n/thisIsNotAnExistingEndpoint

404 Not Found 197 ms 866 B

```
<h1>Not Found</h1>
<p>The requested resource was not found on this server.</p>
```

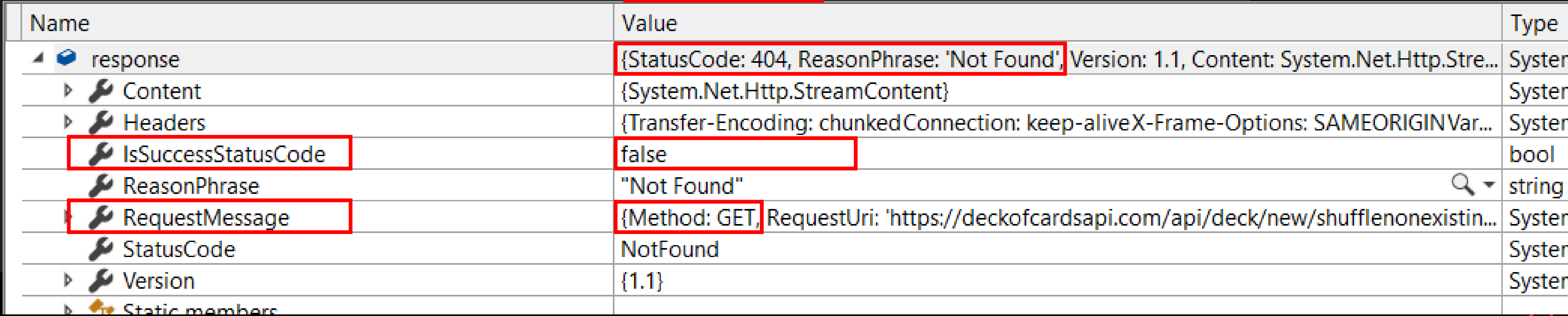
## HTTP REQUESTS IN C#

CONSUMING AN HTTP REST API

http requests in C#

## USING THE HTTPCLIENT OBJECT – GET

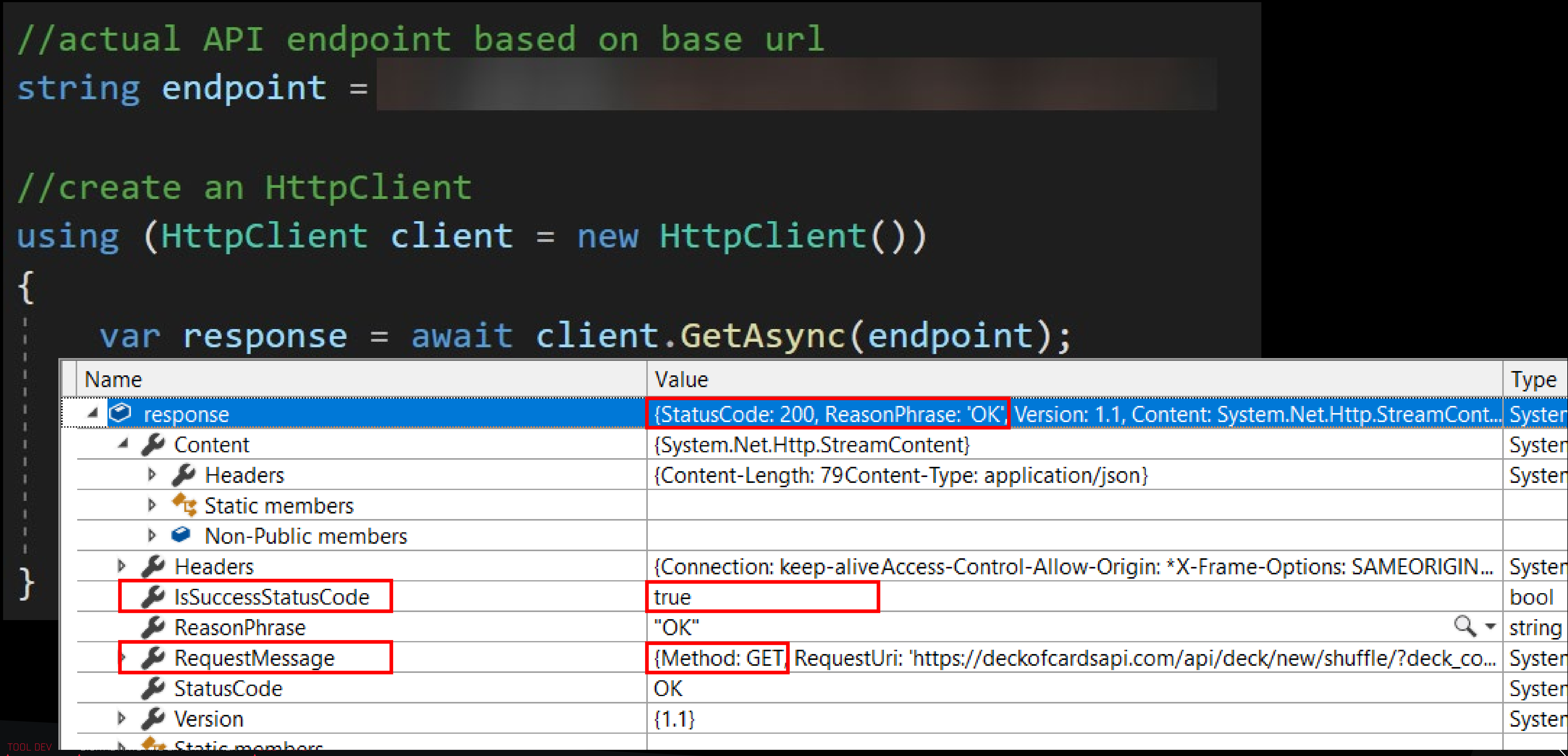
```
//actual API endpoint based on base url
string endpoint = [redacted]

//create an HttpClient
using (HttpClient client = new HttpClient())
{
    var response = await client.GetAsync(endpoint);
}
```



| Name | Value | Type |
|---|---|---|
| response | {StatusCode: 404, ReasonPhrase: 'Not Found', Version: 1.1, Content: System.Net.Http.Stre... | Syster |
| Content | {System.Net.Http.StreamContent} | Syster |
| Headers | {Transfer-Encoding: chunked Connection: keep-alive X-Frame-Options: SAMEORIGIN Var... | Syster |
| IsSuccessStatusCode | false | bool |
| ReasonPhrase | "Not Found" | string |
| RequestMessage | {Method: GET, RequestUri: 'https://deckofcardsapi.com/api/deck/new/shufflenonexistin... | Syster |
| StatusCode | NotFound | Syster |
| Version | {1.1} | Syster |
| Static members | | |

http requests in C#

## USING THE HTTPCLIENT OBJECT – GET



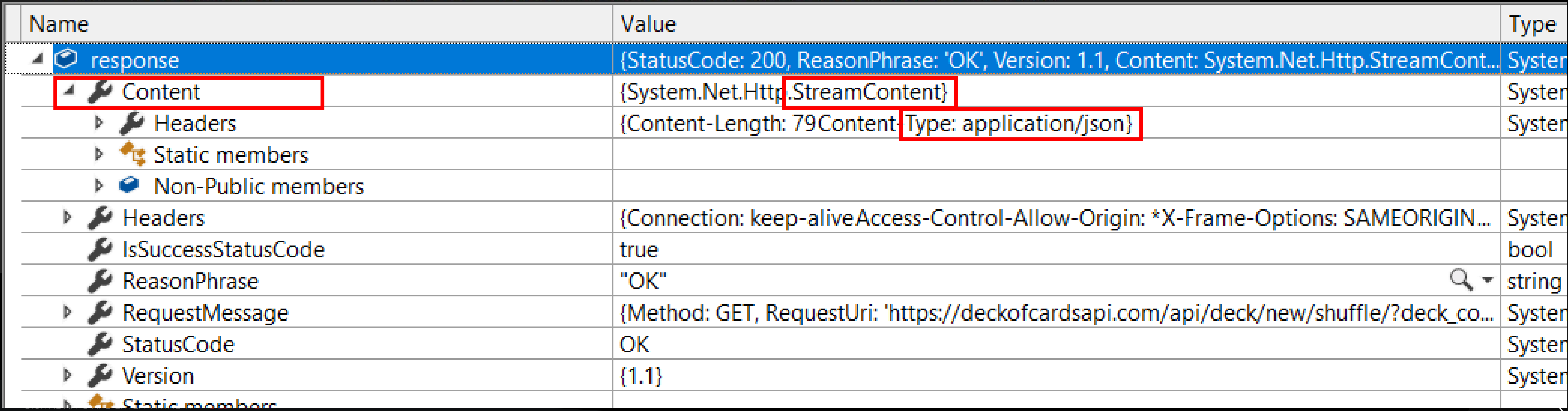
```
//actual API endpoint based on base url
string endpoint = [redacted]

//create an HttpClient
using (HttpClient client = new HttpClient())
{
    var response = await client.GetAsync(endpoint);
}
```

| Name | Value | Type |
|---|---|---|
| response | {StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamCont... | Syster |
| Content | {System.Net.Http.StreamContent} | Syster |
| Headers | {Content-Length: 79Content-Type: application/json} | Syster |
| Static members | | |
| Non-Public members | | |
| Headers | {Connection: keep-aliveAccess-Control-Allow-Origin: *X-Frame-Options: SAMEORIGIN... | Syster |
| IsSuccessStatusCode | true | bool |
| ReasonPhrase | "OK" | string |
| RequestMessage | {Method: GET, RequestUri: 'https://deckofcardsapi.com/api/deck/new/shuffle/?deck_co... | Syster |
| StatusCode | OK | Syster |
| Version | {1.1} | Syster |
| Static members | | |

http requests in C#

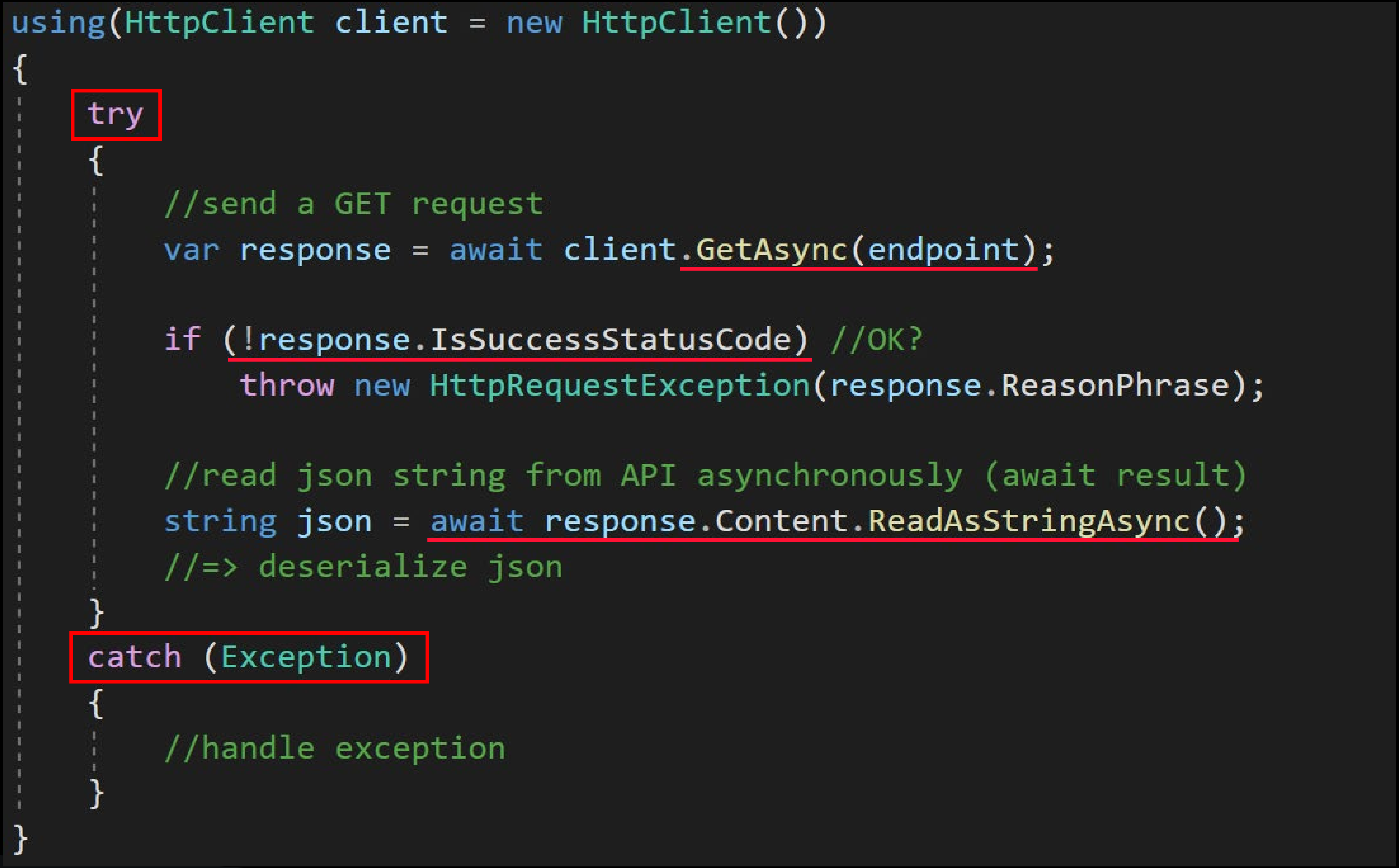
## USING THE HTTPCLIENT OBJECT – GET

```
//actual API endpoint based on base url
string endpoint = [redacted]

//create an HttpClient
using (HttpClient client = new HttpClient())
{
    var response = await client.GetAsync(endpoint);
}
```



| Name | Value | Type |
|---|---|---|
| response | {StatusCode: 200, ReasonPhrase: 'OK', Version: 1.1, Content: System.Net.Http.StreamCont... | Syster |
| Content | {System.Net.Http.StreamContent} | Syster |
| Headers | {Content-Length: 79Content-Type: application/json} | Syster |
| Static members | | |
| Non-Public members | | |
| Headers | {Connection: keep-alive Access-Control-Allow-Origin: *X-Frame-Options: SAMEORIGIN... | Syster |
| IsSuccessStatusCode | true | bool |
| ReasonPhrase | "OK" | string |
| RequestMessage | {Method: GET, RequestUri: 'https://deckofcardsapi.com/api/deck/new/shuffle/?deck_co... | Syster |
| StatusCode | OK | Syster |
| Version | {1.1} | Syster |
| Static members | | |

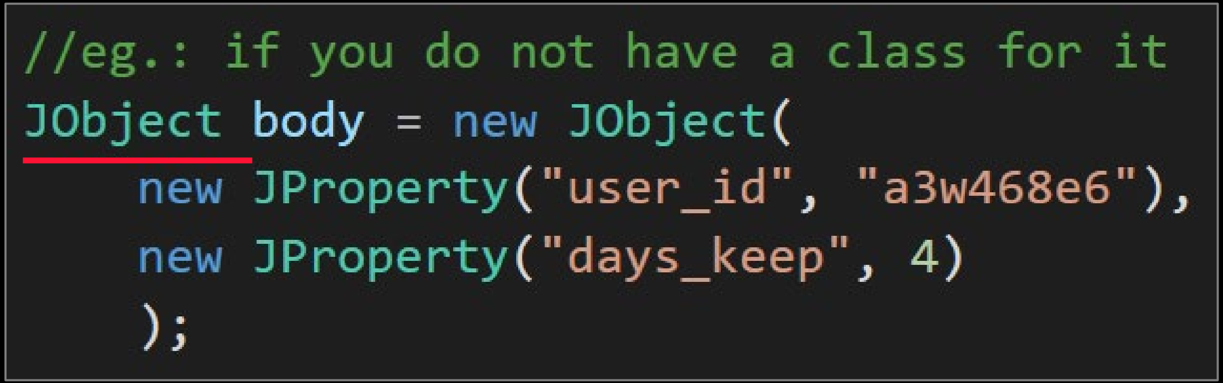
## GET REQUEST + RESPONSE



```
using(HttpClient client = new HttpClient())
{
    try
    {
        //send a GET request
        var response = await client.GetAsync(endpoint);

        if (!response.IsSuccessStatusCode) //OK?
            throw new HttpRequestException(response.ReasonPhrase);

        //read json string from API asynchronously (await result)
        string json = await response.Content.ReadAsStringAsync();
        //=> deserialize json
    }
    catch (Exception)
    {
        //handle exception
    }
}
```

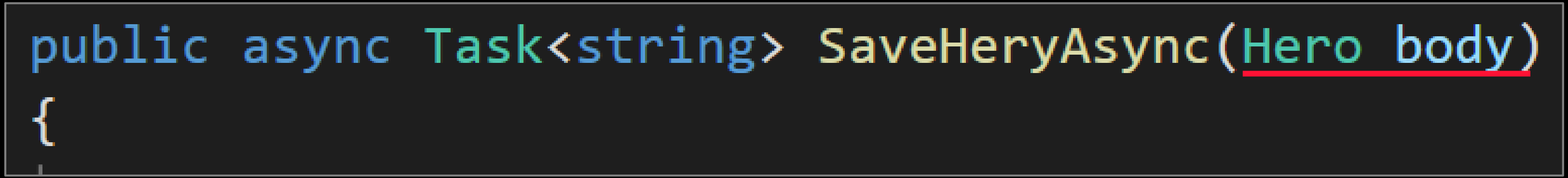
## POST REQUEST - BODY

- Prepare body to send:
  - Eg.: extra info needed, no class



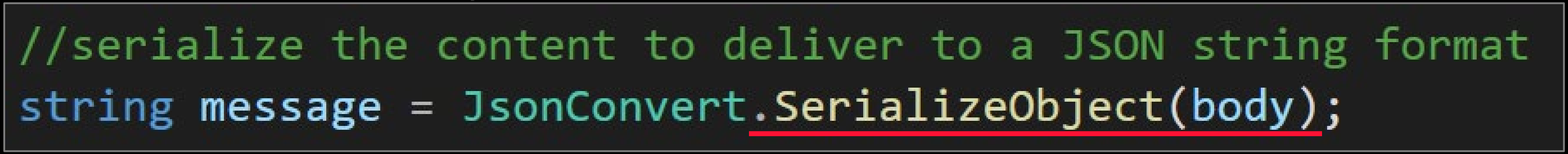
```
//eg.: if you do not have a class for it
JObject body = new JObject(
    new JProperty("user_id", "a3w468e6"),
    new JProperty("days_keep", 4)
    );
```

  - Eg.: new Hero object to be saved in the cloud:



```
public async Task<string> SaveHeryAsync(Hero body)
{
```

- Serialize the body (to format that web service expects):



```
//serialize the content to deliver to a JSON string format
string message = JsonConvert.SerializeObject(body);
```

## POST REQUEST + RESPONSE

```
using(HttpClient client = new HttpClient())
{
    try
    {
        //send a POST to the API + catch the result
        var response = await client.PostAsync(endpoint,
            new StringContent(message));

        if(!response.IsSuccessStatusCode) //OK?
            throw new HttpRequestException(response.ReasonPhrase);

        //read the result json string asynchronously
        string json = await response.Content.ReadAsStringAsync();

        //deserialize json object (in this case we only need 1 property)
        return JsonConvert.DeserializeObject<JObject>(json)
            .SelectToken("deck_id").Value<string>();
    }
    catch (Exception)
```