

ASP Net RESTFul Web API Core

03 Web API Fundamentals

Topics

- **01 Fundamentals**
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

What is Web API?


- Application Programming Interface – API
- Web API – שירותי HTTP
- Representational State Transfer – RESTFul Service. ארכיטקטורת שירותי אינטרנט שמורכבת ממספר מאפיינים:
 - שרת – לקוח
 - Stateless – אין שמירת מצב בין קריאות HTTP
 - Cacheable – ניתן לשמור את תגובת השרת בלקוח למועד מאוחר יותר
 - Uniform interface – ממשק אחיד שמכיל מספיק אינפורמציה בכדי לעבד אותו. כמו למשל מידע אודות המשאב והפעולה הרצויה עליו.

Resource	Verb	כוונה	CRUD
/Students	GET	מחזיר רשימת סטודנטים	read
/Students/1	GET	מחזיר סטודנט בעל ID=1	read
/Students	POST	מייצר סטודנט חדש	create
/Students/1	PUT	מעדכן סטודנט בעל ID=1	update
/Students/1	DELETE	מוחק סטודנט בעל ID=1	delete

Open a project

web api x Clear all

All languages All platforms All project types

 ASP.NET Core **Web API**
A project template for creating an ASP.NET Core application with an example template can also be used for ASP.NET Core MVC Views and Controllers.

C# Linux macOS Windows Cloud Service **Web** **WebAPI**

Framework ⓘ
.NET 6.0 (Long Term Support)

Authentication type ⓘ
None
☒ **Configure for HTTPS** ⓘ
☐ Enable Docker ⓘ

Docker OS ⓘ
Linux

☒ Use controllers (uncheck to use minimal APIs) ⓘ
☒ Enable OpenAPI support ⓘ
☐ Do not use top-level statements ⓘ

Open a project

- יש לשים לב במקרה שנבחר פרוייקט ללא HTTPS אז נוכל לבדוק אותו מול POSTMAN וגם מול THUNDER CLIENT לעומת זאת בפרוייקט של HTTPS ניתן לבדוק מול THUNDER CLIENT אבל מול POSTMAN מצריך לחיצה על

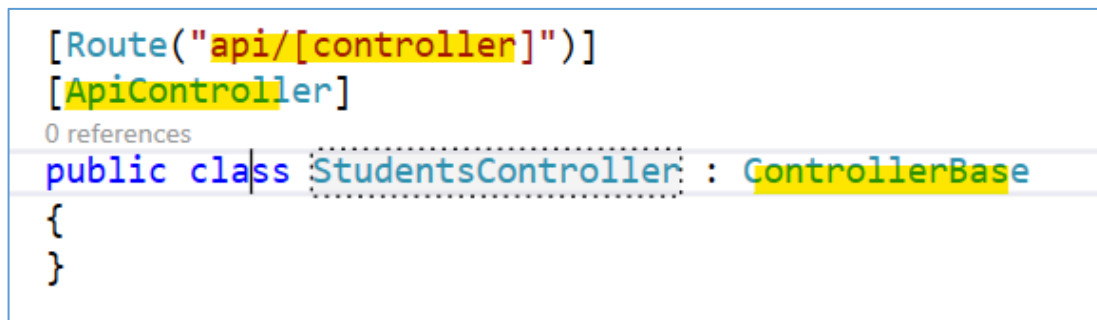
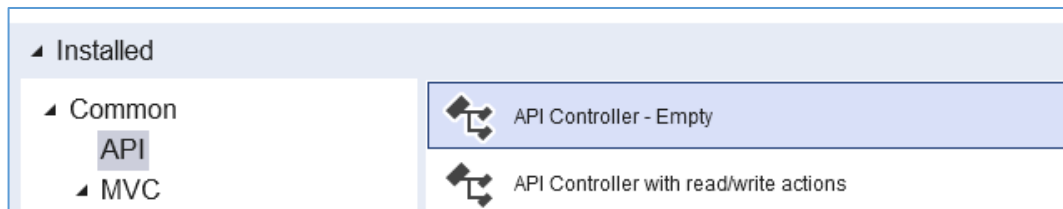
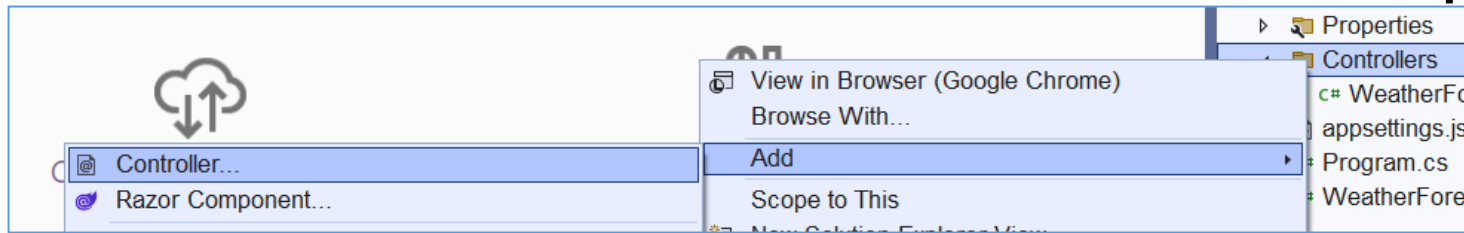


- כמובן שניתן לבדוק מול אפליקציית ריאקט בכל מקרה גם HTTPS אבל רק בתנאי שמימשנו את CORS

Configuration and basic Routing

Controller and basic Routing

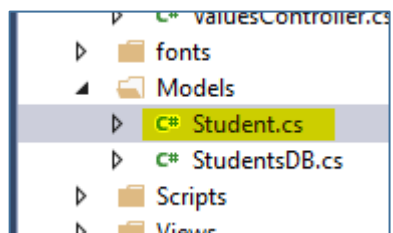
- נוסף קונטרולר עבור resource של סטודנט.



- פה נמצא הנתיב הדיפולטיבי.
- שימו לב שהוא מתחיל ב- api/...

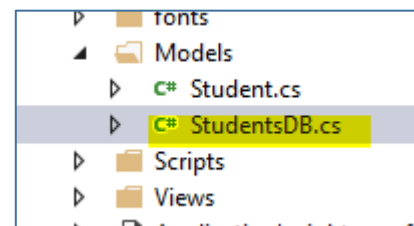
Model and Controller

- בד"כ הקונטרולר משויך למשאב מסויים כמו למשל סטודנט במקרה שלנו. מטרת הקונטרולר היא לנהל את המשאב ע"י שליפה, הכנסה, מחיקה ועידכון של המשאב.
- לעיתים קרובות כדאי לייצר מודל (מחלקה) מסוג המשאב כך שניתן יהיה לעבוד איתה בקלות רבה.
- נעשה זאת תחת הספרייה של models. כך ע"י using נוכל להשתמש במחלקה בקונטרולר.



```
public class Student
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int Grade { get; set; }

    public override string ToString()
    {
        return $"{ID},{ Name},{ Grade}";
    }
}
```



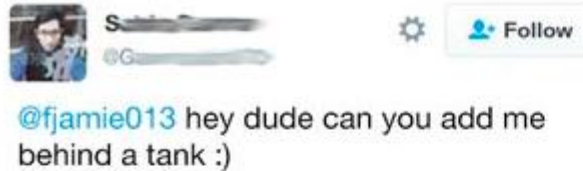
```
public class StudentsDB
{
    static public List<Student> students = new List<Student>()
    {
        new Student(){ID=1, Name="avi", Grade=100 },
        new Student(){ID=2, Name="charlie", Grade=90 },
        new Student(){ID=3, Name="benny", Grade=95 },
        new Student(){ID=4, Name="dora", Grade=97 },
    };
}
```


Topics

- 01 Fundamentals
- **02 Methods: Get, Post, Put, Delete – the basic (wrong) way**
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

http request - response

request



response



בחמשת השקפים הבאים איך להשתמש בVERBS עבור CRUD. נא לשים לב שנתתי דוגמאות שונות עבור הערך המוחזר. הדוגמאות הללו לאו דווקא הנכונות ביותר, בהמשך נלמד מה באמת כדאי להחזיר בכל VERB!

```

public class StudentsController : ControllerBase
{
    [HttpGet]
    0 references
    public IEnumerable<Student> Get() {
        Student[] sa = StudentsMockDB.studnets.ToArray();
        return sa;
    }
}

```

The function's name does not matter

http Get – get all - Read

request

סוג הבקשה – GET
המשאב המתאים – שם הקונטרולר

סוג המידע המוחזר – פה JSON
...JPG,XML יכול להיות גם

```

1 GET /api/students HTTP/1.1
2 Host: localhost:58563
3 Accept: application/json
4 cache-control: no-cache
5 Postman-Token: e36aea79-c2ad-43e6-ba57-041245e0009b
6 accept: application/xml-----WebKitFormBoundary7MA4YWxkTrZu0gW--|

```

response

סטטוס מוחזר – 2XX success

סוג המידע המוחזר – פה JSON

```

HTTP/1.1 200
status: 200
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?RDpcV29ya1xydXBwaw4gZW5nXExlY3R1cmVzXHhBhcnQgSU1JIC0=
X-Powered-By: ASP.NET
Date: Tue, 18 Jun 2019 07:50:54 GMT
Content-Length: 141

[{"ID":1,"Name":"avi","Grade":100},{ "ID":2,"Name":
{"ID":4,"Name":"dora","Grade":97}]

```

המידע המוחזר

```
[HttpGet("{id}")]
0 references
public Student Get(int id)
{
    return StudentsMockDB.studnets.SingleOrDefault(stu=> stu.ID == id);
}
```

http Get – get one - Read

request

סוג הבקשה – GET
המשאב המתאים – שם הקונטרולר וגם הפרמטר - פה id=2

```
1 GET /api/students/2 HTTP/1.1
2 Host: localhost:58563
3 Accept: application/json
4 cache-control: no-cache
5 Postman-Token: 25b33761-7558-4e2e-a024-493564697156
6 accept: application/xml-----WebKitFormBoundary7MA4YWxkTrZu0gW--|
```

סוג המידע המוחזר – פה JSON
...JPG,XML יכול להיות גם

response

```
HTTP/1.1 200
status: 200
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?
RDpcV29ya1xydXBwaw4gZW5nXEXlY3R1cmVzXHhBhcnQgSU1J
=
X-Powered-By: ASP.NET
Date: Tue, 18 Jun 2019 08:15:20 GMT
Content-Length: 36
```

סטטוס מוחזר – 2XX success

סוג המידע המוחזר – פה JSON

המידע המוחזר

```
{"ID":2,"Name":"charlie","Grade":90}
```

```
[HttpPost]
0 references
public int Post([FromBody] Student value)
{
    StudentsMockDB.studnets.Add(value);
    return value.ID;
}
```

http Post – insert - Create

request

POST – סוג הבקשה
המשאב המתאים – שם הקונטרולר

סוג המידע המוחזר – פה JSON
...JPG,XML יכול להיות גם

סוג המידע הנשלח

```
1 POST /api/students HTTP/1.1
2 Host: localhost:58563
3 Accept: application/json
4 Content-Type: application/json
5 cache-control: no-cache
6 Postman-Token: c79c04a5-da10-4641-9f03-e41282a49ae6
7
8 "ID": 5,
9 "Name": "sivan",
10 "Grade": 105
11 }-----WebKitFormBoundary7MA4YWxkTrZu0gW--
```

המידע הנשלח

response

סטטוס מוחזר – success 2XX

סוג המידע המוחזר – פה JSON

```
HTTP/1.1 200
status: 200
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?RDpcV29ya1xydXBwaw4gZW5nXExlY3R1cmVzXHBhcnQgSU1=
X-Powered-By: ASP.NET
Date: Tue, 18 Jun 2019 08:28:15 GMT
Content-Length: 1
```

המידע המוחזר

```
[HttpPut("{id}")]
```

0 references

```
public string Put(int id, [FromBody] Student value)
```

```
{  
    Student stu = StudentsMockDB.studnets.SingleOrDefault(stu => stu.ID == id);  
    stu.Name = value.Name;  
    stu.Grade = value.Grade;  
    return "done:");  
}
```

http Put – Update

סוג הבקשה – PUT

המשאב המתאים – שם הקונטרולר והפרמטר לעדכון – פה id=2

request

סוג המידע המוחזר – פה JSON

יכול להיות גם XML, JPG, ...

סוג המידע הנשלח

```
PUT /api/students/2
```

```
Accept: application/json
```

```
Content-Type: application/json
```

```
cache-control: no-cache
```

```
Postman-Token: 295af416-25f0-498f-afe0-d68e0fad1eec
```

```
User-Agent: PostmanRuntime/7.6.0
```

```
Host: localhost:58563
```

```
accept-encoding: gzip, deflate
```

```
content-length: 56
```

```
{ "ID": 2, "Name": "charlie", "Grade": 102 }
```

המידע הנשלח

response

סטטוס מוחזר – success 2XX

סוג המידע המוחזר – פה JSON

```
HTTP/1.1 200
```

```
status: 200
```

```
Cache-Control: no-cache
```

```
Pragma: no-cache
```

```
Content-Type: application/json; charset=
```

```
Expires: -1
```

```
Server: Microsoft-IIS/10.0
```

```
X-AspNet-Version: 4.0.30319
```

```
X-SourceFiles: =?UTF-8?B?RDpcV29ya1xydXBwaw4gZW5r
```

```
X-Powered-By: ASP.NET
```

```
Date: Tue, 18 Jun 2019 18:35:43 GMT
```

```
Content-Length: 8
```

```
"done:)"
```

המידע המוחזר

[HttpDelete]

0 references

```
public IActionResult Delete(int id)
{
    Student s = StudentsMockDB.studnets.SingleOrDefault(stu => stu.ID == id);
    StudentsMockDB.studnets.Remove(s);
    var v = new { Result = "Deleted Successfully" };
    var j = new JsonResult(v);
    return j;
}
```

request

DELETE – סוג הבקשה
המשאב המתאים – שם הקונטרולר וגם הפרמטר - פה id=2

```
DELETE /api/students/2
Accept: application/json
cache-control: no-cache
Postman-Token: 76bb6284-f907-4882-9434-bf6b
User-Agent: PostmanRuntime/7.6.0
Host: localhost:58563
accept-encoding: gzip, deflate
content-length:
```

סוג המידע המוחזר – פה JSON
יכול להיות גם XML, JPG, ...

response

```
HTTP/1.1 200
status: 200
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 34
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?RDpCV29ya1xydXBwaW4gZl
X-Powered-By: ASP.NET
Date: Tue, 18 Jun 2019 19:02:09 GMT

{"Result": "deleted successfully!"}
```

סטטוס מוחזר – 2XX success

סוג המידע המוחזר – פה JSON

המידע המוחזר

http Delete – delete one

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- **03 Content Negotiation**
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Content Negotiation

- ניתן לבקש סוגים שונים של מידע בחזרה מהשרת.
- שדה ה Accept אשר נמצא ב- Header אחראי על סוג הערך המוחזר.
- יכול להיות למשל:
 - Accept: application/json
 - Accept: application/xml
 - Accept: application/json,application/xml
- אם לא קיים בכלל אז Json יהיה ברירת במחדל.
- כאשר שולחים מידע לשרת ניתן להשתמש בשדה ה- Content-Type למשל:
 - Content-Type: application/json

POST	http://localhost:58563/api/students		
Params	Auth	Headers (2)	Body
	KEY	VALUE	DESCRIPT
<input checked="" type="checkbox"/>	Accept	application/json	
<input checked="" type="checkbox"/>	Content-Type	application/json	
	Key 17	Value	Descript

POST	http://localhost:58563/api/students		
Params	Auth	Headers (2)	Body
<input type="radio"/> none <input type="radio"/> form-data <input type="radio"/> x-www-form-urlencoded			
JSON (application/json)			
1	{		
2	"ID": 5,		
3	"Name": "sivan",		
4	"Grade": 105		
5	}		

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- **04 Status Codes and IHttpActionResult**
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Status Codes and IActionResult

- כדי להקל על הלקוח בהבנת המידע המוחזר מהשרת צריך להשתמש בקוד של סטטוס אשר מסביר האם הבקשה עברה בהצלחה או לא.
- יש מגוון סטטוסים מוחזרים בקטגוריות שונות ע"פ ספרת המאות כגון:
 - **2xx Success**
 - 200 OK – למשל כאשר ה-GET חוזר בהצלחה.
 - 201 Created – למשל כאשר POST חוזר בהצלחה.
 - 204 No Content – למשל כאשר המתודה מחזירה VOID.
 - **4xx Client Error**
 - 400 Bad Request
 - 401 Unauthorized
 - 404 Not Found
 - **5xx Server Error**
 - 500 Internal Server Error – למשל כאשר יש חריגה בצד שרת.
- בכדי שהתקשורת בין השרת ללקוח תהיה אופטימלית כדאי להחזיר מהשרת IActionResult. כך נוכל לקבוע מהו הסטטוס הרצוי בצירוף המידע המבוקש.
- בשקפים הבאים נראה איך להשתמש נכון בסטטוסים ובהחזר מידע נכון מהשרת ע"י IActionResult.

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- **05 Methods: Get, Post, Put, Delete – the right way**
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

StudentsRW

http Get – get all - Read

מחזיר סטטוס 200 וגם את
הרשימה

```
public class StudentsRWController : ControllerBase
{
    [HttpGet]
    [ProducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]
    0 references
    public ActionResult<Student[]> Get() {
        try
        {
            Student[] sa = StudentsMockDB.studnets.ToArray();
            return Ok(sa);
        }
        catch (Exception e)
        {
            //return BadRequest(e.Message);
            return StatusCode(StatusCodes.Status400BadRequest, e.Message);
        }
    }
}
```

swagger	
Code	Description
200	Success
Media type	
text/plain	
Controls Accept header.	
Example Value Schema	
[{ "id": 0, "name": "string", "grade": 0 }]	
400	Bad Request
Media type	
text/plain	
Example Value Schema	
{	

מחזיר סטטוס 400 וגם את ההודעה
של החריגה.

Status: 400 Bad Request Time: 228 ms Size: 469 B		
Body	Cookies	Headers (10) Test Results
Pretty Raw Preview JSON		
1	{	
2	"Message": "nirs message - invalid op!"	
3	}	

http Get – get one - Read

```
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status200OK, Type = typeof(Student))]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
0 references
public IActionResult Get(int id)
{
    try
    {
        Student value = StudentsMockDB.studnets.FirstOrDefault(stu=> stu.ID == id);
        if (value == null)
        {
            return NotFound($"studnt with id={id} was not found in GET by id!");
        }
        return Ok(value);
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
```

מחזיר סטטוס 200 וגם
את הסטודנט המבוקש

מחזיר 404 NOT FOUND

Responses	
Code	Description
200	Success
Media type	
text/plain	
Controls Accept header.	
Example Value Schema	
{ "id": 0, "name": "string", "grade": 0 }	

Status: 404 Not Found Time: 3121 ms Size: 462 B

Body Cookies Headers (10) Test Results

Pretty

Raw

Preview

JSON



1 "student with id=22 was not found!"

http Post – insert - Create

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status201Created, Type = typeof(Student))]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
0 references
public IActionResult Post([FromBody] Student value)
{
    try
    {
        if (value == null)
            return BadRequest(value);
        else if (value.Id != 0)
            return StatusCode(StatusCodes.Status500InternalServerError);

        value.Id = StudentsDbMOCK.students.Max(stu => stu.Id) + 1;
        StudentsDbMOCK.students.Add(value);

        return CreatedAtAction(nameof(Get), new { id = value.Id }, value);
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
```

```
[HttpGet("{id}")]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
1 reference
public IActionResult Get(int id)
```

מחזיר סטטוס 201 וגם את הסטודנט שנוצר
בשרת – בד"כ כולל את המספר רץ של ה-ID.
וגם קישור לGET עם ה-ID המתאים.

http Put – Update

```
[HttpPut("{id}")]
[ProducesResponseType(StatusCodes.Status204NoContent)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
0 references
public IActionResult Put(int id, [FromBody] Student value)
{
    try
    {
        if (value == null || value.ID != id)
        {
            return BadRequest();
        }
        Student stu = StudentsMockDB.students.FirstOrDefault(stu => stu.ID == id);
        if (stu == null)
        {
            return NotFound($"student with id={id} was not found to update!");
        }
        stu.Name = value.Name;
        stu.Grade = value.Grade;
        return NoContent();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
```

מחזיר 204 ואת הסטודנט
המעודכן

Status: 404 Not Found	Time: 54 ms	Size: 472 B
Body	Cookies	Headers (10)
Test Results		
Pretty Raw Preview JSON		
1	"Student with id=22 was not found to update!"	

מחזיר 404 NOT FOUND

http Delete – delete one

```
[HttpDelete("{id}")]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status204NoContent)]
0 references
public IActionResult Delete(int id)
{
    try
    {
        if (id==0)
        {
            return BadRequest();
        }
        Student value = StudentsMockDB.students.FirstOrDefault(stu => stu.ID == id);
        if (value == null)
        {
            return NotFound($"student with id={id} was not found to delete!");
        }
        StudentsMockDB.students.Remove(value);
        return NoContent();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
```

מחזיר 404 NOT FOUND

Status: 404 Not Found Time: 129 ms Size: 471 B

Body Cookies Headers (10) Test Results

Pretty Raw Preview JSON

1 "Student with id=2 was not found to delete!"

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- **06 Custom Method Names**
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Custom Method Names

- אין חשיבות לשם הפונקציה אלא רק לATTRIBUTE

```
[HttpGet]  
public IActionResult LoadStudents()
```

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- **07 Attribute Routing**
- 08 Query String Parameters
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Attribute Routing


- נוסיף routing attribute וכך נפתור את הבעיה

Id פרמטר

```
[Route("api/StudentsCustom/{id}/grades")]  
public IActionResult GetStudentGrades(int id)
```

- ניתן עדין להגדיר את כל המסלול באופן אבסולוטי ע"י שימוש ב ~ למשל:

```
[Route("{id}/grades")]  
[Route("~/sg/{id}")]  
public IActionResult GetStudentGrades(int id)
```

GET  https://localhost:7092/stam

Attribute Routing continue

- ניתן להגדיר מספר מגבלות על הפרמטרים שנשלחים למתודה בכדי להבדיל ביניהם למשל אם נרצה מתודת get שמקבלת גם בולאני – האם להחזיר את אבי או את כל השאר:
- פה נקבל שגיאה

```
[HttpGet("{isAvi}")]
[ProducesResponseType(StatusCodes.Stat
[ProducesResponseType(StatusCodes.Stat
0 references
public IActionResult Get(bool isAvi)
{
```

Error: response status is 500

Response body

Microsoft.AspNetCore.Routing.Matching.AmbiguousMatchException: The request matched multiple endpoints. Matches:

- בכדי לפתור את זה ניתן להוסיף את הסוג של הפרמטר (במקרה של שניים אפשר לשים רק על הבוליאני וזה יספיק) למשל:

```
[Route("{isAvi:bool}")]
public IActionResult Get(bool isAvi) {...}

[Route("{id:int}")]
public IActionResult Get(int id) {...}
```

Attribute Routing continue

- ניתן גם להגדיר מגבלות נוספות כמו מספר מינימום למשל:

```
[Route("{id:int:min(1)}")]  
public IActionResult Get(int id)
```

- אם נשלח למשל 0 נקבל שגיאה:

Error: response status is 404

Response headers

```
content-length: 0  
date: Wed, 20 Sep 2023 08:37:42 GMT  
server: Kestrel
```

- בעמוד הבא נמצאת רשימת המגבלות...

Attribute Routing continue

Constraint	Description	Example
alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	{x:alpha}
bool	Matches a Boolean value.	{x:bool}
datetime	Matches a Date T ime value.	{x:datetime}
decimal	Matches a decimal value.	{x:decimal}
double	Matches a 64-bit floating-point value.	{x:double}
float	Matches a 32-bit floating-point value.	{x:float}
guid	Matches a GUID value.	{x:guid}
int	Matches a 32-bit integer value.	{x:int}
length	Matches a string with the specified length or within a specified range of lengths.	{x:length(6)} {x:length(1,20)}
long	Matches a 64-bit integer value.	{x:long}
max	Matches an integer with a maximum value.	{x:max(10)}
maxlength	Matches a string with a maximum length.	{x:maxlength(10)}
min	Matches an integer with a minimum value.	{x:min(10)}
minlength	Matches a string with a minimum length.	{x:minlength(10)}
range	Matches an integer within a range of values.	{x:range(10,50)}
regex ³²	Matches a regular expression.	{x:regex(^\d{3}-\d{3}- \d{4}\$)}

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- **08 Query String Parameters**
- 09 FromUri and FromBody
- 10 Cross-Origin Resource Sharing (CORS)

Query String Parameters

- ניתן לשרשר ל-URL פרמטרים אשר ישלחו לפונקציה ע"י שימוש באופרטור ?

`http://localhost:58563/api/studentsCustom?gradeStatus=pass`

```
public IActionResult Get(string gradeStatus="all")
{
    try
    {
        Student[] sa = null;
        switch (gradeStatus)
        {
            case "all":
                sa = StudentsDB.students.ToArray();
                break;
            case "pass":
                sa = StudentsDB.students.Where(s => s.Grade >= 60).ToArray();
                break;
            case "fail":
                sa = StudentsDB.students.Where(s => s.Grade < 60).ToArray();
                break;
            default:
                return BadRequest($"gradeStatus must be: all\\pass\\fail");
        }
        return Ok(sa);
    }
    catch (Exception ex)
    {
        return Content(HttpStatusCode.BadRequest, ex);
    }
}
```

ברירת מחדל

Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- **09 FromUri and FromBody**
- 10 Cross-Origin Resource Sharing (CORS)

FromUri and FromBody

- כבירת מחדל, ערכים פרימיטיביים בשפה: bool, string, int... נלקחים מהו URI לעומת ערכים מורכבים כמו מחלקה שנקראים מה-BODY. לדוגמה:

http://localhost:58563/api/studentsCustom/true →

```
public IActionResult Put( bool id)
{
    return Ok(id);
}
```

→

Status: 200 OK Time: 81 ms Size: 431 B

Body Cookies Headers (10) Test Results

Pretty Raw Preview JSON ▼

1 true

http://localhost:58563/api/studentsCustom/nir →

```
public IActionResult Put( string id)
{
    return Ok(id);
}
```

→

Status: 200 OK Time: 3215 ms Size: 432 B

Body Cookies Headers (10) Test Results

Pretty Raw Preview JSON

1 "nir"

http://localhost:58563/api/studentsCustom/ →

```
public IActionResult Put( Student id)
{
    return Ok(id);
}
```

→

Status: 200 OK Time: 3265 ms Size: 460 B

1 {

2 "ID": 2,

3 "Name": "charlie",

4 "Grade": 50

5 }

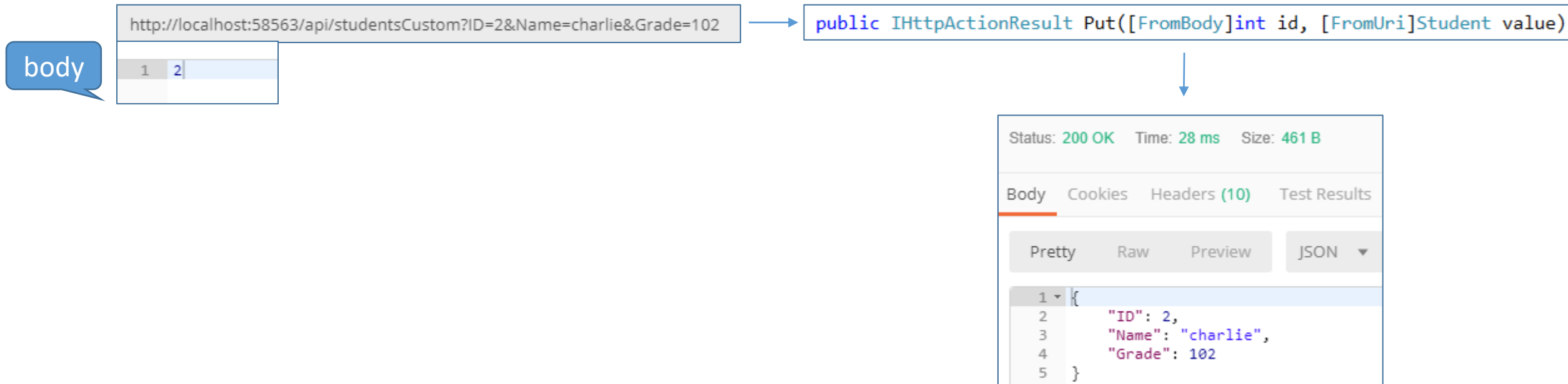
body

36

ניר חן

FromUri and FromBody continue

- שימו לב שה attribute פה מיותר: `public IHttpActionResult Put(int id, [FromBody]Student value)`
- ניתן לתת הוראה מפורשת מאיפה לקחת את המידע למשל נוכל להחליף בדוגמה הזו את המקור למידע למשל:



Topics

- 01 Fundamentals
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody
- **10 Cross-Origin Resource Sharing (CORS)**

Cross-Origin Resource Sharing (CORS)

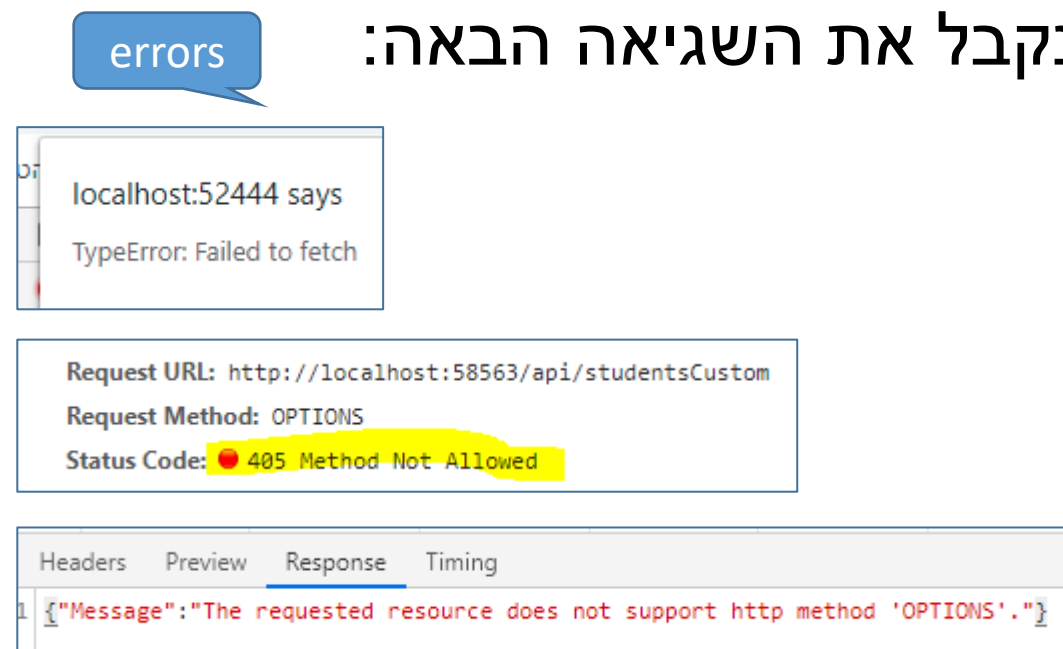
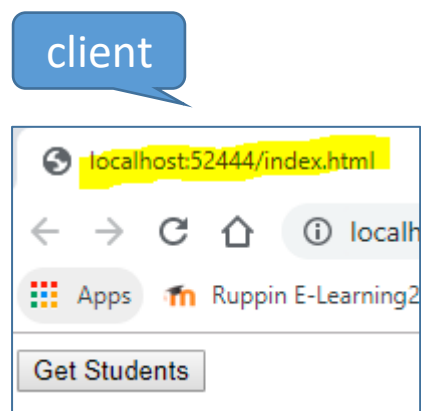
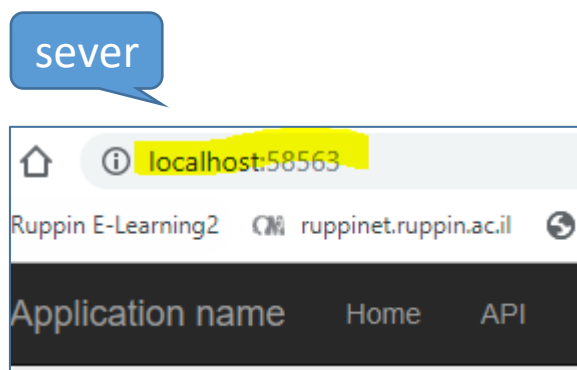
- כאשר נרצה לקרוא ע"י FETCH ל- WEB API מדומיין\אתר (פרויקט - SOLUTION) אחר למשל. נצטרך לאפשר זאת ע"י הוספת יכולת CORS, אחרת נקבל שגיאה.
- נניח שהלקוח שלנו הוא אתר אחר עם הקוד JS הבא:

```
<script>
function btnGetStudents() {
  alert(1);

  var url = 'http://localhost:58563/api/studentsCustom';
  fetch(url,
    {
      method: 'GET', // 'GET', 'PUT', 'DELETE', etc.
      headers: new Headers({
        'Content-Type': 'application/json'
      }),
    }) // Call the fetch function passing the url of the
    .then((resp) => resp.json()) // Transform the data
    .then(function (data) {
      alert(data);
    })
    .catch(function (err) {
      alert(err);
    });
}
</script>
</head>
<body>
  <button onclick="btnGetStudents()">Get Students</button>
</body>
```

Cross-Origin Resource Sharing (CORS) continue

- שימו לב שהלקוח והשרת רצים תחת דומיין (פה ה-PORT) שונה.
- נקבל את השגיאה הבאה:



Cross-Origin Resource Sharing (CORS) continue

- בכדי לאפשר CORS יש צורך להוסיף את הקוד הבא:
- ניתן להגביל את המאפיינים לפתיחת ה-CORS

- origins
- headers
- methods

```
builder.Services.AddCors(p => p.AddPolicy("corspolicy", build => build.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader()));  
//builder.Services.AddCors(p => p.AddPolicy("corspolicy",  
//    build => build.SetIsOriginAllowed(origin => new Uri(origin).Host == "localhost").AllowAnyMethod().AllowAnyHeader()));  
//builder.Services.AddCors(p => p.AddPolicy("corspolicy",  
//    build => build.WithOrigins("http://example.com").AllowAnyMethod().AllowAnyHeader()));  
  
var app = builder.Build();  
  
// Configure the HTTP request pipeline.  
if (app.Environment.IsDevelopment())  
{  
    app.UseSwagger();  
    app.UseSwaggerUI();  
}  
  
app.UseHttpsRedirection();  
  
app.UseCors("corspolicy");
```

- ניתן לאפשר את ה-CORS בכמה רמות :

- ברמת הפרויקט כולו

- או

```
.WithHeaders(HeaderNames.ContentType, "x-custom-header")  
.WithMethods("PUT", "DELETE", "GET")
```

Cross-Origin Resource Sharing (CORS) continue

- ניתן לאפשר את הCORS בכמה רמות :

- ברמת הפרויקט כולו

- או

- ברמת הקונטרולר

- ברמת המתודה הבודדת.

- ניתן גם לאסור עבור רמה מסויימת ע"י
[DisableCors]

```
[EnableCors("MyPolicy")]  
[Route("api/[controller]")]  
[ApiController]  
public class ValuesController :  
{
```

```
// GET api/values  
[EnableCors("AnotherPolicy")]  
[HttpGet]  
public ActionResult<IEnumerable<string>> Get()  
{
```

```
// GET: api/values/GetValues2  
[DisableCors]  
[HttpGet("{action}")]  
public IActionResult GetValues2() =>  
    ControllerContext.MyDisplayRouteInfo
```

Cross-Origin Resource Sharing (CORS) continue

- ניתן לראות שה- HEADER מסוג CORS
נוסף ל- RESPONSE

