

# ASP Net RESTFul Web API2

## 01 Web API Fundamentals

# Topics

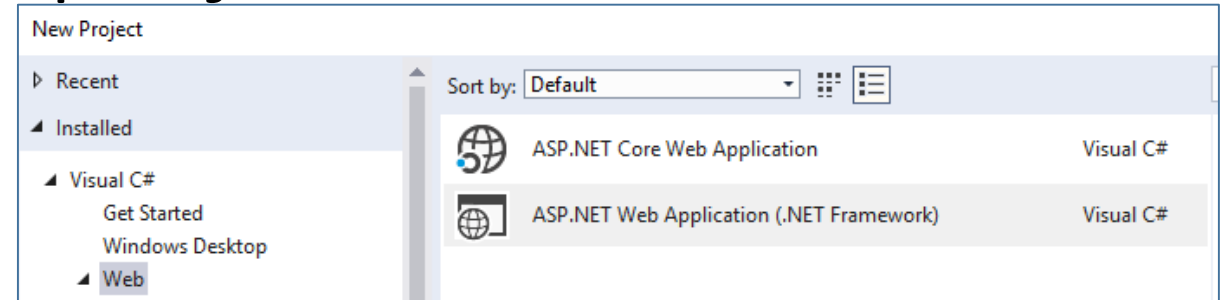
- **01 Fundamentals**
- 02 Methods: Get, Post, Put, Delete – the basic (wrong) way
- 03 Content Negotiation
- 04 Status Codes and IHttpActionResult
- 05 Methods: Get, Post, Put, Delete – the right way
- 06 Custom Method Names
- 07 Attribute Routing
- 08 Query String Parameters
- 09 FromUri and FromBody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



 **Empty**  
An empty project template for creating ASP.NET applications. This template does not have any content in it.

 **Web Forms**  
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

 **MVC**  
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

 **Web API**  
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

 **Single Page Application**

## Authentication

No Authentication

[Change](#)

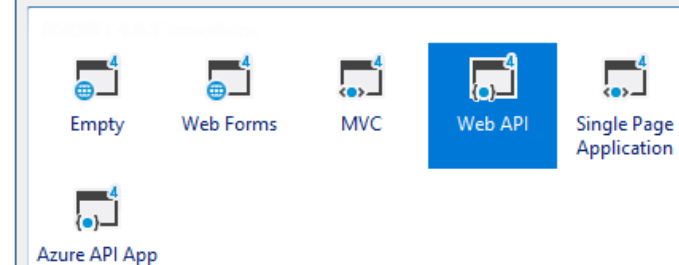
## Add folders & core refe

- ☐ Web Forms
- ☒ MVC
- ☒ Web API

## Advanced

- ☐ Configure for HTTPS
- ☐ Docker support  
(Requires [Docker Desktop](#))

## New ASP.NET Web Application - MyWebAPIProj



A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

[Learn more](#)

## Add folders and core references for:

- ☐ Web Forms ☒ MVC ☒ Web API
- ☐ Enable Docker Compose support (Requires [Docker for Windows](#))
- ☐ Add unit tests

Test project name:

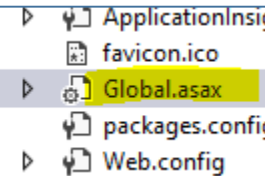
Authentication: **No Authentication**

[Change Authentication](#)

OK

# Configuration and basic Routing

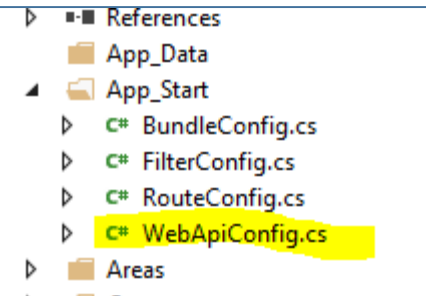
```
AreaRegistration.RegisterAllAreas();  
GlobalConfiguration.Configure(WebApiConfig.Register);  
FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);  
RouteConfig.RegisterRoutes(RouteTable.Routes);
```



- פה נמצא הנתיב הדיפולטיבי.

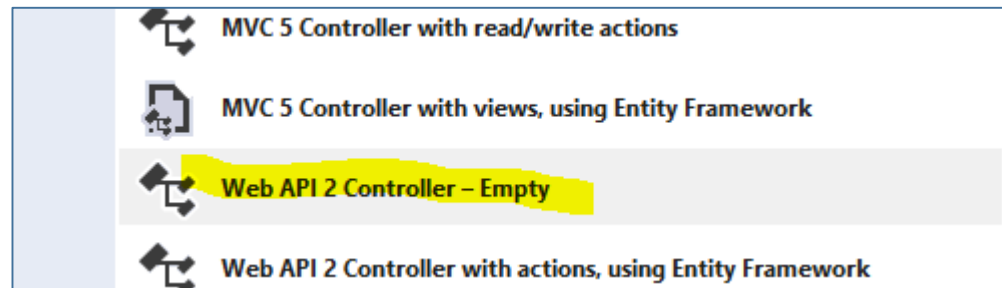
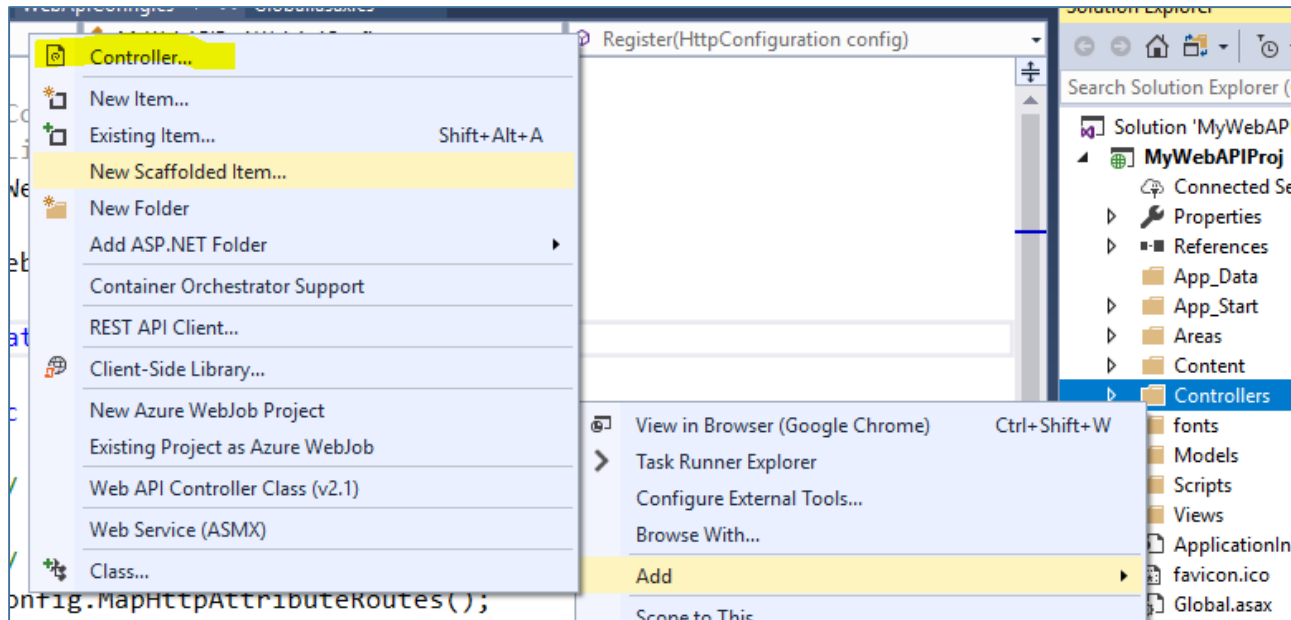
- שימו לב שהוא מתחיל ב- api/...

```
public static class WebApiConfig  
{  
    public static void Register(HttpConfiguration config)  
    {  
        // Web API configuration and services  
  
        // Web API routes  
        config.MapHttpAttributeRoutes();  
  
        config.Routes.MapHttpRoute(  
            name: "DefaultApi",  
            routeTemplate: "api/{controller}/{id}",  
            defaults: new { id = RouteParameter.Optional }  
        );  
    }  
}
```



# Controller

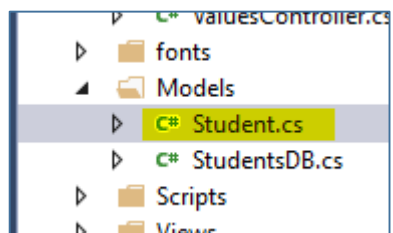
- נוסף קונטרולר עבור resource של סטודנט.



```
public class StudentsController : ApiController
{
```

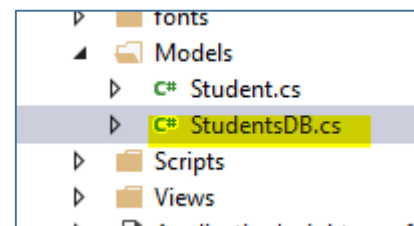
# Model and Controller

- 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 },
    };
}
```

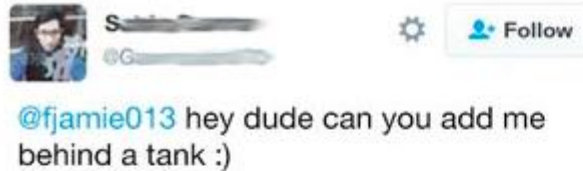
# Topics

- 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 : ApiController
{
    public IEnumerable<Student> Get() {
        Student[] sa = StudentsDB.students.ToArray();
        return sa;
    }
}
```

# http Get – get all - Read

request

סוג הבקשה – GET  
המשאב המתאים – שם הקונטרולר

```
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--|
```

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=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}]
```

סטטוס מוחזר – 2XX success

סוג המידע המוחזר – פה JSON

המידע המוחזר

```
public Student Get(int id)
{
    return StudentsDB.students.SingleOrDefault(x=>x.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?
RDpcV29ya1xydXBwaw4gZW5nXEXlY3R1cmVzXHBhcnQgSU1J
=
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}
```

```
public int Post([FromBody]Student value)
{
    StudentsDB.students.Add(value);
    return value.ID;
}
```

# http Post – insert - Create

request

POST – סוג הבקשה  
המשאב המתאים – שם הקונטרולר

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

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?RDpcV29ya1xydXBwaw4gZW5nXExlY3R1cmVzXHhcnQgsU1=
X-Powered-By: ASP.NET
Date: Tue, 18 Jun 2019 08:28:15 GMT
Content-Length: 1
```

המידע המוחזר

5

# http Put – Update

```
public string Put(int id, [FromBody]Student value)
{
    Student s = StudentsDB.students.SingleOrDefault(x => x.ID == id);
    s.Name = value.Name;
    s.Grade = value.Grade;
    return "done:");
}
```

request

סוג הבקשה – PUT  
המשאב המתאים – שם הקונטרולר והפרמטר לעדכון – פה id=2

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

```
public IActionResult Delete(int id)
{
    Student s = StudentsDB.students.SingleOrDefault(x => x.ID == id);
    StudentsDB.students.Remove(s);
    var v = new { Result = "deleted successfully!" };
    var j = Json(v);
    return j;
}
```

# http Delete – delete one

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

# 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 16	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 IHttpActionResult

- 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 IHttpActionResult. כך נוכל לקבוע מהו הסטטוס הרצוי בצירוף המידע המבוקש.
- בשקפים הבאים נראה איך להשתמש נכון בסטטוסים ובהחזר מידע נכון מהשרת ע"י IHttpActionResult.

# 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 Get – get all - Read

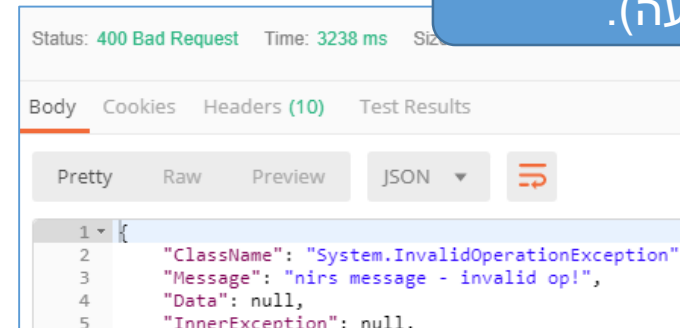
StudentsRW

```
public class StudentsRWController : ApiController
{
    public IHttpActionResult Get()
    {
        try
        {
            Student[] sa = StudentsDB.students.ToArray();
            return Ok(sa);
        }
        catch (Exception ex)
        {
            //return BadRequest(ex.Message); //opt 1
            return Content(HttpStatusCode.BadRequest, ex); //opt 2
        }
    }
}
```

מחזיר סטטוס 200 וגם את  
הרשימה

מחזיר סטטוס 400 וגם את ההודעה  
של החריגה.

מחזיר סטטוס 400 וגם את החריגה  
עצמה (כולל ההודעה).



# http Get – get one - Read

```
public IActionResult Get(int id)
{
    try
    {
        Student value = StudentsDB.students.FirstOrDefault(x => x.ID == id);
        if (value == null)
        {
            //return NotFound(); //opt1
            return Content(HttpStatusCode.NotFound, "student with id=" + id + " was not found!"); //opt2
        }
        return Ok(value);
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

מחזיר 404 NOT FOUND

מחזיר סטטוס 200 וגם את  
הסטודנט המבוקש

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

```
// POST api/students
public IHttpActionResult Post([FromBody]Student value)
{
    try
    {
        StudentsDB.students.Add(value);
        return Created(new Uri(Request.RequestUri.AbsoluteUri + value.ID), value);
        //normally returns back the object with the auto-generated id (autonumber)
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

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

בשקף [הזה](#) יש את השיטה הכי נכונה למימוש  
POST כי היא כוללת route name לGET  
שמקבלים חזרה

Status: 201 Created	Time: 3224 ms	Size: 511 B
Body	Cookies	Headers (11)
Cache-Control → no-cache		
Pragma → no-cache		
Content-Type → application/json; charset=utf-8		
Expires → -1		
Location → http://localhost:58563/api/studentsRW/5		

# http Put – Update

```
public IActionResult Put(int id, [FromBody]Student value)
{
    try
    {
        Student s = StudentsDB.students.SingleOrDefault(x => x.ID == id);
        if (s != null)
        {
            s.Name = value.Name;
            s.Grade = value.Grade;
            return Content(HttpStatusCode.OK, s);
        }
        else
        {
            return Content(HttpStatusCode.NotFound, $"Student with id={id} was not found to update!");
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Status: 200 OK Time: 108 ms Size: 457 B

Body Cookies Headers (10) Test Results

Pretty

Raw

Preview

JSON ▼



```
1 {
2   "ID": 2,
3   "Name": "charlie",
4   "Grade": 102
5 }
```

מחזיר 200 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

```
public IActionResult Delete(int id)
{
    try
    {
        Student s = StudentsDB.students.SingleOrDefault(x => x.ID == id);
        if (s != null)
        {
            StudentsDB.students.Remove(s);
            return Ok();
        }
        else
        {
            return Content(HttpStatusCode.NotFound, $"Student with id={id} was not found to delete!");
        }
    }
    catch (Exception ex)
    {
        return BadRequest(ex.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)

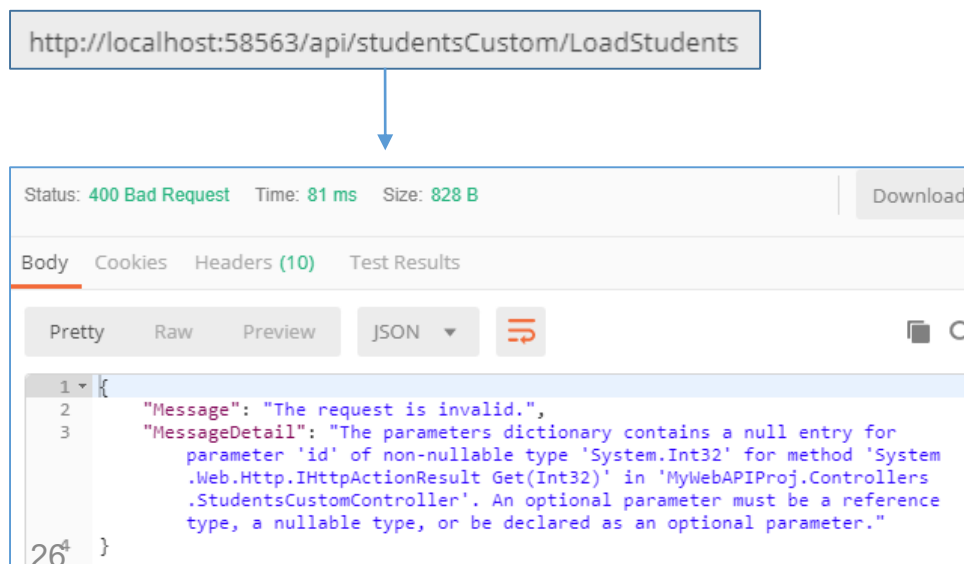
# Custom Method Names

- כל 3 החתימות הבאות ירוצו באותו אופן:

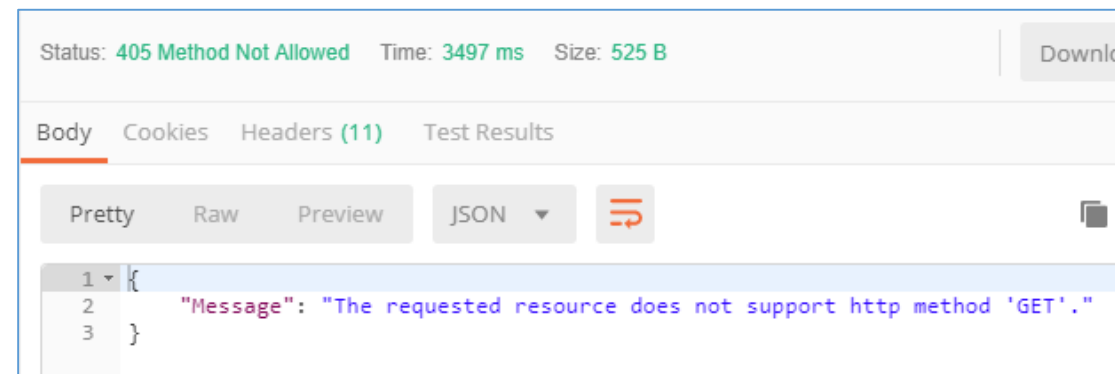
```
public IHttpActionResult Get()  
  
public IHttpActionResult GetStudents()  
  
[HttpGet]  
public IHttpActionResult LoadStudents()
```

- שימו לב לשגיאות הבאות:

במקרה ולא קיימת בכלל פונקציית GET



ניר חן



# 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

- עד עכשיו למדנו את שיטת Convention Based Routing כפי שמומש ב:

```
public static void Register(HttpConfiguration config)
{
    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );
}
```

- עכשיו נלמד גם Attribute Routing - ע"י attribute ניתן לבחור route שיוביל לקונטרולר או מתודה שניבחר.

- דורש את השורה הבאה:

```
public static void Register(HttpConfiguration config)
{
    config.MapHttpAttributeRoutes();
}
```

# Attribute Routing continue

- נניח שנרצה מספר פונקציות שונות של Get אשר מקבלות id. למשל נוסיף לקוד שלנו עוד פונקציה שמחזירה ציונים עבור סטודנט עם id מסויים.

```
public IHttpActionResult GetStudentGrades(int id)
```

```
http://localhost:58563/api/studentsCustom/2
```

- במקרה שננסה לקרוא למתודה כך:

נקבל

```
{
  "Message": "An error has occurred.",
  "ExceptionMessage": "Multiple actions were found that match the request:
    \r\nGetStudentGrades on type MyWebAPIProj.Controllers.StudentsCustomController\r\nGet on
    type MyWebAPIProj.Controllers.StudentsCustomController",
  "ExceptionType": "System.InvalidOperationException",
}
```

```
http://localhost:58563/api/studentsCustom/2/grades
```

- אנחנו נרצה לקרוא למתודה הזו בצורה הבאה:

ואז נקבל את השגיאה הבאה:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.
  /xhtml1-strict.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <title>IIS 10.0 Detailed Error - 404.0 - Not Found</title>
5 <style type="text/css">
```

פתרון בעמוד  
הבא...

# Attribute Routing continue

Id פרמטר

- נוסף routing attribute וכך נפתור את הבעיה

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

- ניתן להוסיף routeprefix לקונטרולר בכדי למנוע חזרה של תחילית מסויימת בכל המתודות למשל:

```
[RoutePrefix("api/StudentsCustom")]  
public class StudentsCustomController : ApiController
```

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

- ואז ניתן לקצר את ה-route

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

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

# Attribute Routing continue

- ניתן להגדיר מספר routes על אותה מתודה.
- ניתן עדין להגדיר את כל המסלול באופן אבסולוטי ע"י שימוש ב ~ למשל:

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

http://localhost:58563/api/studentsCustom/2/grades

http://localhost:58563/sg/2

ואז שתי הקריאות הבאות יעשו אותו דבר

# 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:

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

Status: 500 Internal Server Error

```
{  
  "Message": "An error has occurred.",  
  "ExceptionMessage": "Multiple actions were found that match the request: \r\nGet on type  
MyWebAPIProj.Controllers.StudentsCustomController\r\nGet on type MyWebAPIProj.Controllers  
.StudentsCustomController",  
}
```

- פה נקבל שגיאה

- בכדי לפתור את זה ניתן להוסיף את הסוג של הפרמטר 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 IHttpActionResult Get(int id)
```

- אם נשלח למשל 0 נקבל שגיאה:

Status: 405 Method Not Allowed

```
{  
  "Message": "The requested resource does not support http method 'GET'."  
}
```

- בעמוד הבא נמצאת רשימת המגבלות...

# Attribute Routing continue

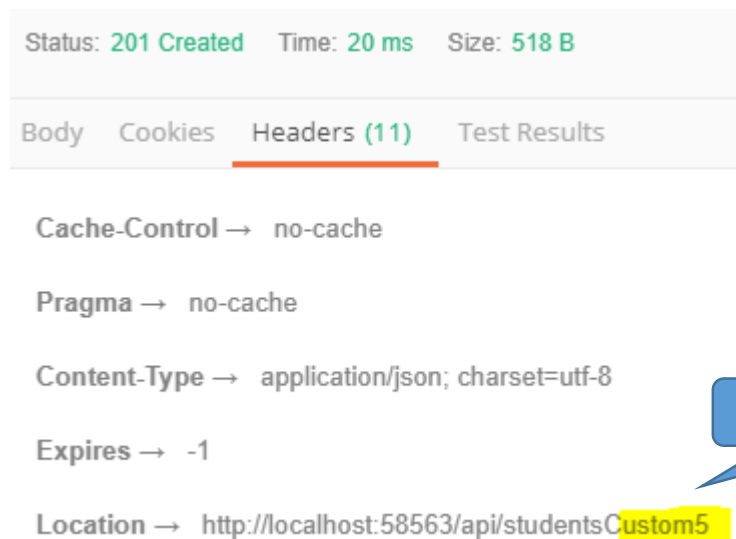
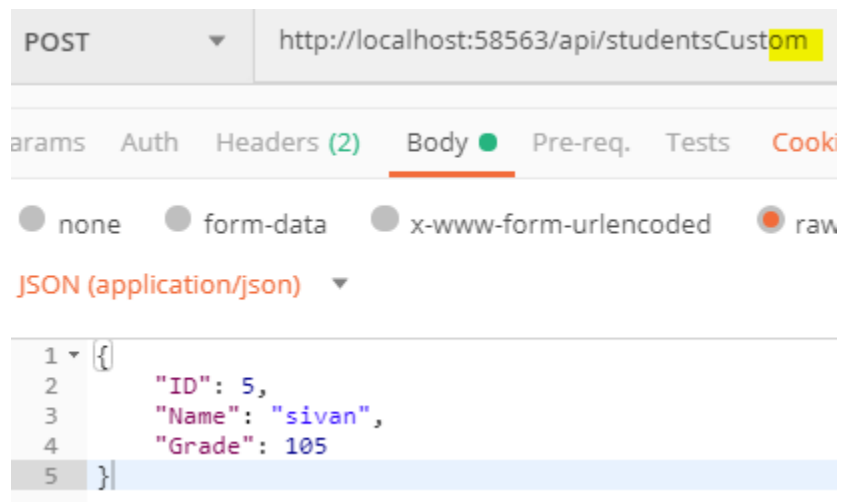
Constraint	Description	Example
alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	{x:alpha}
bool	Matches a Boolean value.	{x:bool}
datetime	Matches a <code>DateTime</code> value.	{x:datetime}
decimal	Matches a decimal value.	{x:decimal}
double	Matches a 64-bit floating-point value.	{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 34	Matches a regular expression.	{x:regex(^\d{3}-\d{3}- \d{4}\$)}

# Attribute Routing continue

- עבור מתודת POST אנו רוצים להחזיר קריאת GET לאובייקט שנוצר. ישנה בעיה מסויימת בשורה הבאה:

```
return Created(new Uri(Request.RequestUri.AbsoluteUri + value.ID), value);
```

- כאשר נקרה ל-POST עם '/' בסוף הכל ירוץ טוב אבל אם נקרה לזה בלי, נקבל חזרה



חסר הסימן /

פתרון בעמוד  
הבא...

# Attribute Routing continue

- נפתור את הבעיה ע"י route name באופן הבא:

```
[Route("{id:int:min(1)}", Name = "GetStudentById")]
public IHttpActionResult Get(int id) {...}

// POST api/students
public IHttpActionResult Post([FromBody]Student value)
{
    try
    {
        StudentsDB.students.Add(value);
        return Created(new Uri(Url.Link("GetStudentById", new { id=value.ID})), value);
        //normally returns back the object with the auto-generated id (autonumber)
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

- זו השיטה הכי נכונה לממש את POST

# 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

- כברירת 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/ →

body

```
{
  "ID": 2,
  "Name": "charlie",
  "Grade": 50
}
```

40

→

```
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 }

ניר חן



# FromUri and FromBody continue

- שימו לב שה attribute פה מיותר: `public IActionResult 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)

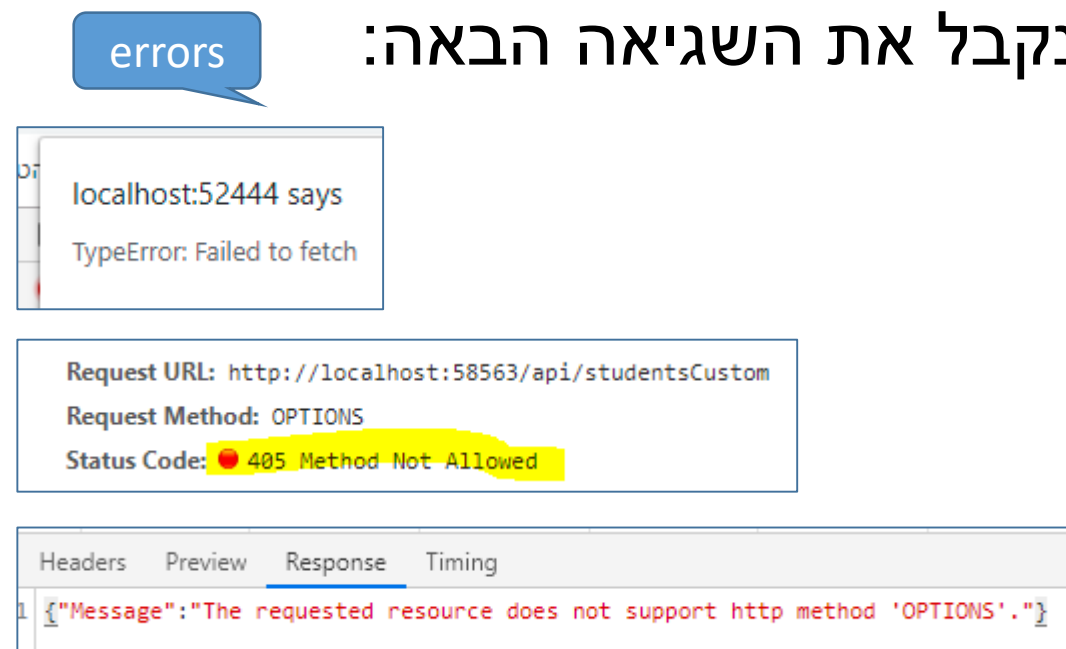
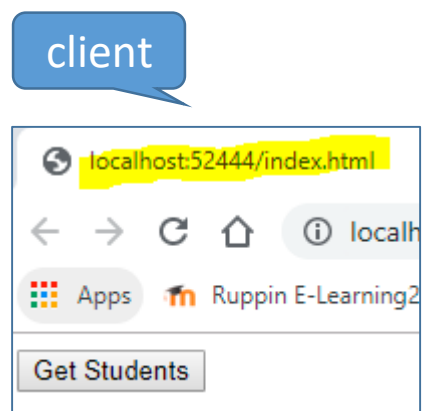
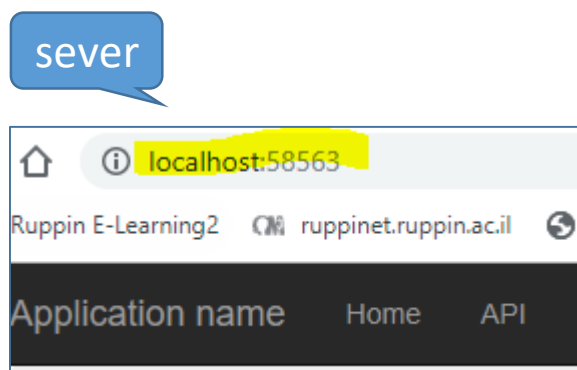
- כאשר נרצה לקרוא ע"י FETCH ל- WEB API מדומיין\אתר (פרויקט - SOLUTION) אחר למשל. נצטרך לאפשר זאת ע"י הוספת יכולת CORS, אחרת נקבל שגיאה.
- נניח שהלקוח שלנו הוא אתר אחר עם הקוד JS הבא:

```
<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>
```

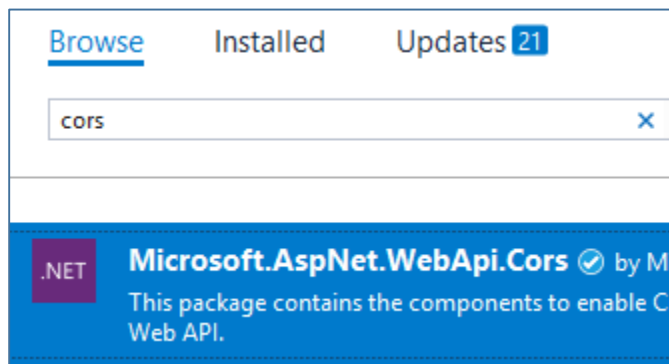
# Cross-Origin Resource Sharing (CORS) continue

- 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 להתקין את הPACKAGE הבא:

- ניתן לאפשר את הCORS בכמה רמות :

- ברמת הפרויקט כולו

- או

- ברמת הקונטרולר

- ברמת המתודה הבודדת.

```
public static void Register(HttpConfiguration config)
{
    EnableCorsAttribute cors = new EnableCorsAttribute("*", "*", "*");
    config.EnableCors(cors);
}
```

```
//FOR ATTRIBUTE ON ACTION OR CONTROLLER ONLY!
config.EnableCors();
```

```
[EnableCors("*", "*", "*")]
public class StudentsCustomController : ApiController
{
}
```

```
[EnableCors("*", "*", "*")]
public IHttpActionResult Get(string gradeStatus = "all")
{
}
```

# Cross-Origin Resource Sharing (CORS) continue

• ניתן להגביל את המאפיינים לפתיחת ה-CORS

```
EnableCorsAttribute cors =  
    new EnableCorsAttribute("http://localhost:52444", "accept,Content-Type", "GET");
```

- origins
- headers
- methods

```
[EnableCors("*", "*", "*")]  
public IHttpActionResult Get(string gradeSt
```

• "\*" מאפשר את כל האופציות

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

```
[DisableCors]  
public IHttpActionResult Get(string gradeStatus = "all")  
{
```

# Cross-Origin Resource Sharing (CORS) continue

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

