



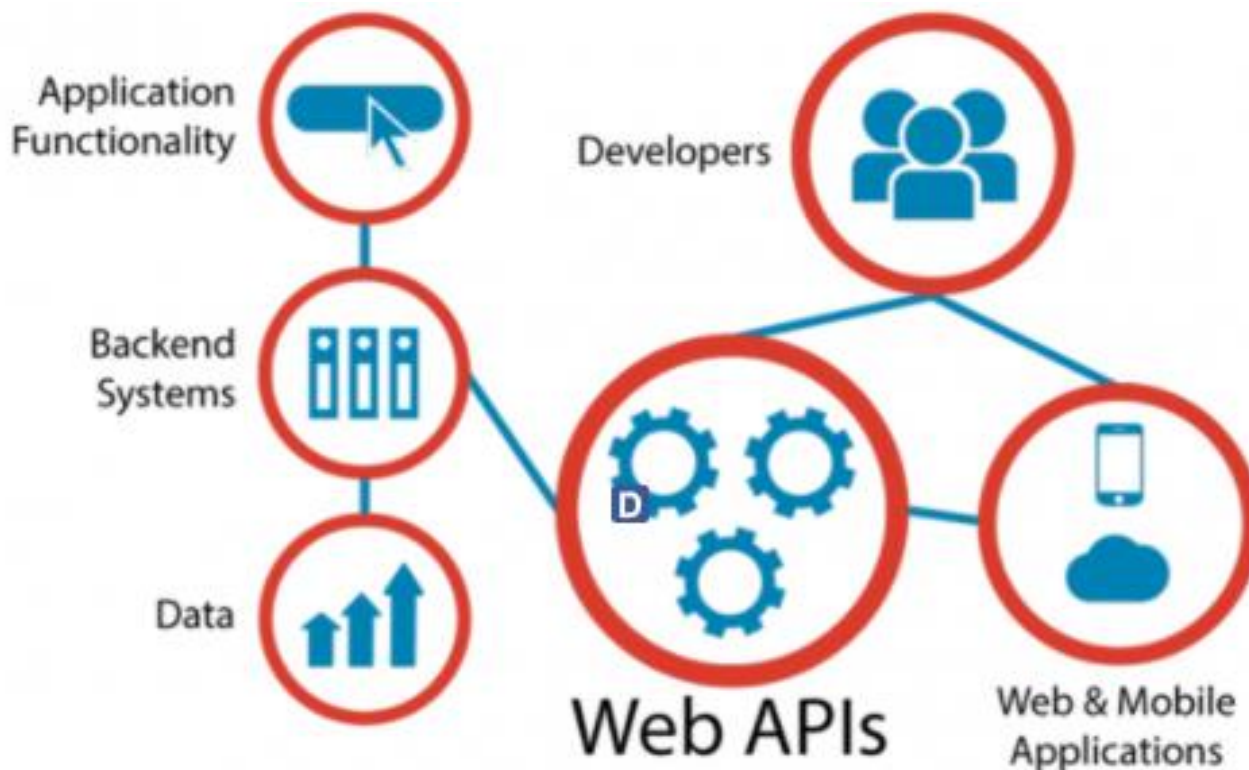
LẬP TRÌNH C# 5

BÀI 1: WEB APIs - ASP.NET CORE

- ◎ Web Api
- ◎ Testing Api - Postman
- ◎ Web APIs in ASP.NET Core



- ❑ Kiến trúc ứng dụng hiện đại ngày nay ngày càng phân tán, không phụ thuộc ngôn ngữ đã thúc đẩy việc ứng dụng API
- ❑ API là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác(Application Programming Interface)

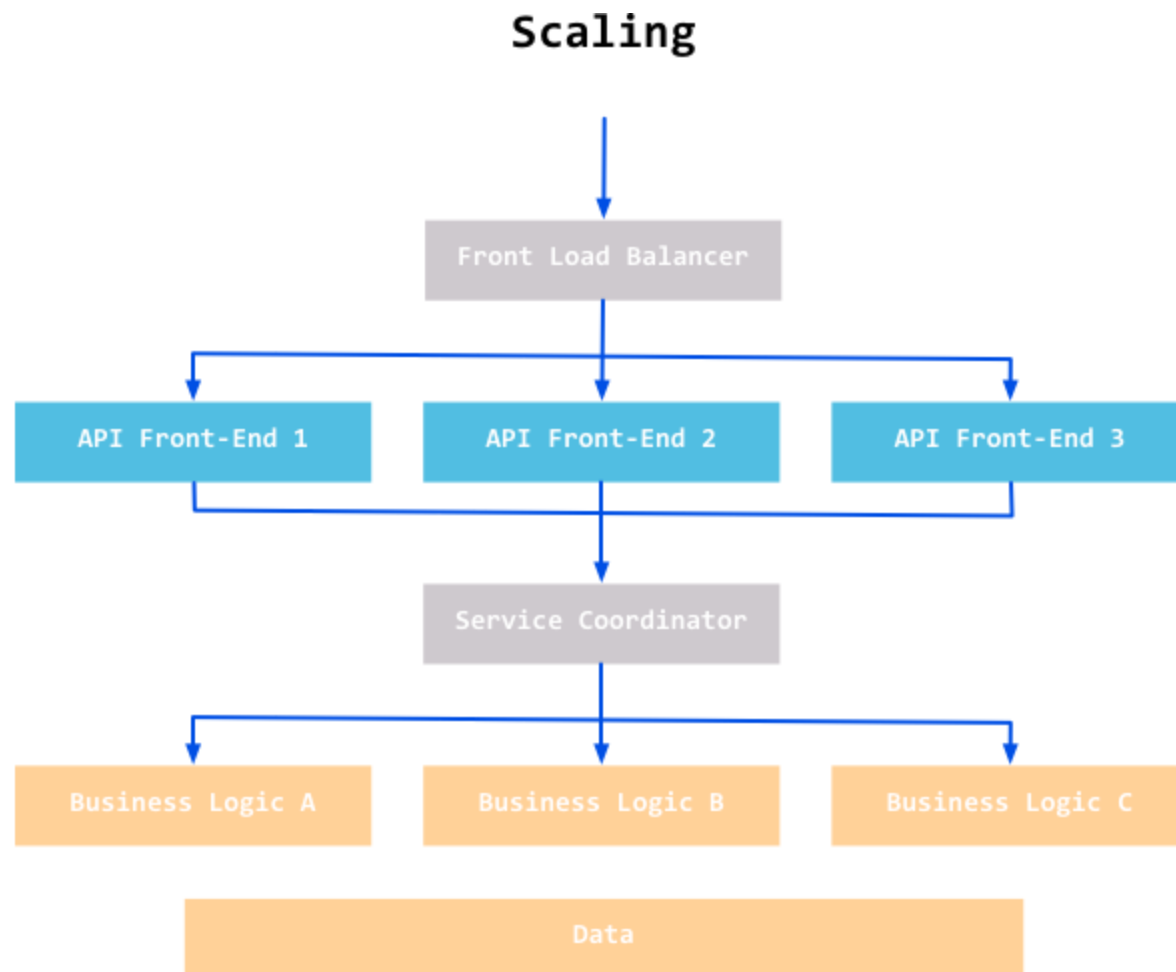


- ❑ Web API là một phương thức dùng để cho phép các ứng dụng khác nhau có thể giao tiếp, trao đổi dữ liệu qua lại. Dữ liệu được Web API trả lại thường ở dạng JSON hoặc XML thông qua giao thức HTTP hoặc HTTPS
- ❑ Web API hỗ trợ restful đầy đủ các phương thức: Get, Post, Put Delete dữ liệu
- ❑ Hỗ trợ đầy đủ các thành phần HTTP: URI, request/response headers, caching, versioning, content format.
- ❑ Hầu hết các website đều ứng dụng đến Web API cho phép kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu (login thông qua Google, Facebook, Twitter)

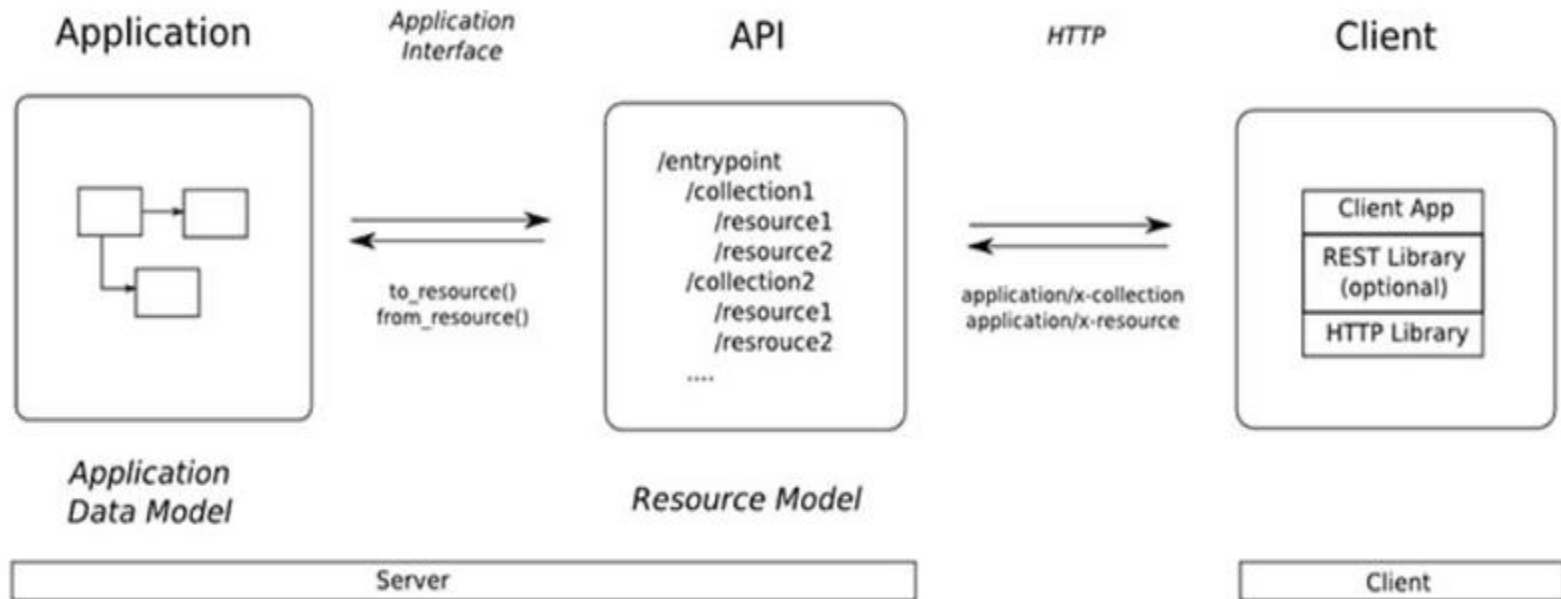
- ❑ REST(REpresentational State Transfer) là một hệ thống các ràng buộc (constraints)
- ❑ Là một kiểu kiến trúc được sử dụng trong việc giao tiếp giữa các máy tính (máy tính cá nhân và máy chủ của trang web) trong việc quản lý các tài nguyên trên internet.
- ❑ RESTful API là một tiêu chuẩn tuân thủ Rest dùng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) để tiện cho việc quản lý các resource, chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

❑ REST constraints

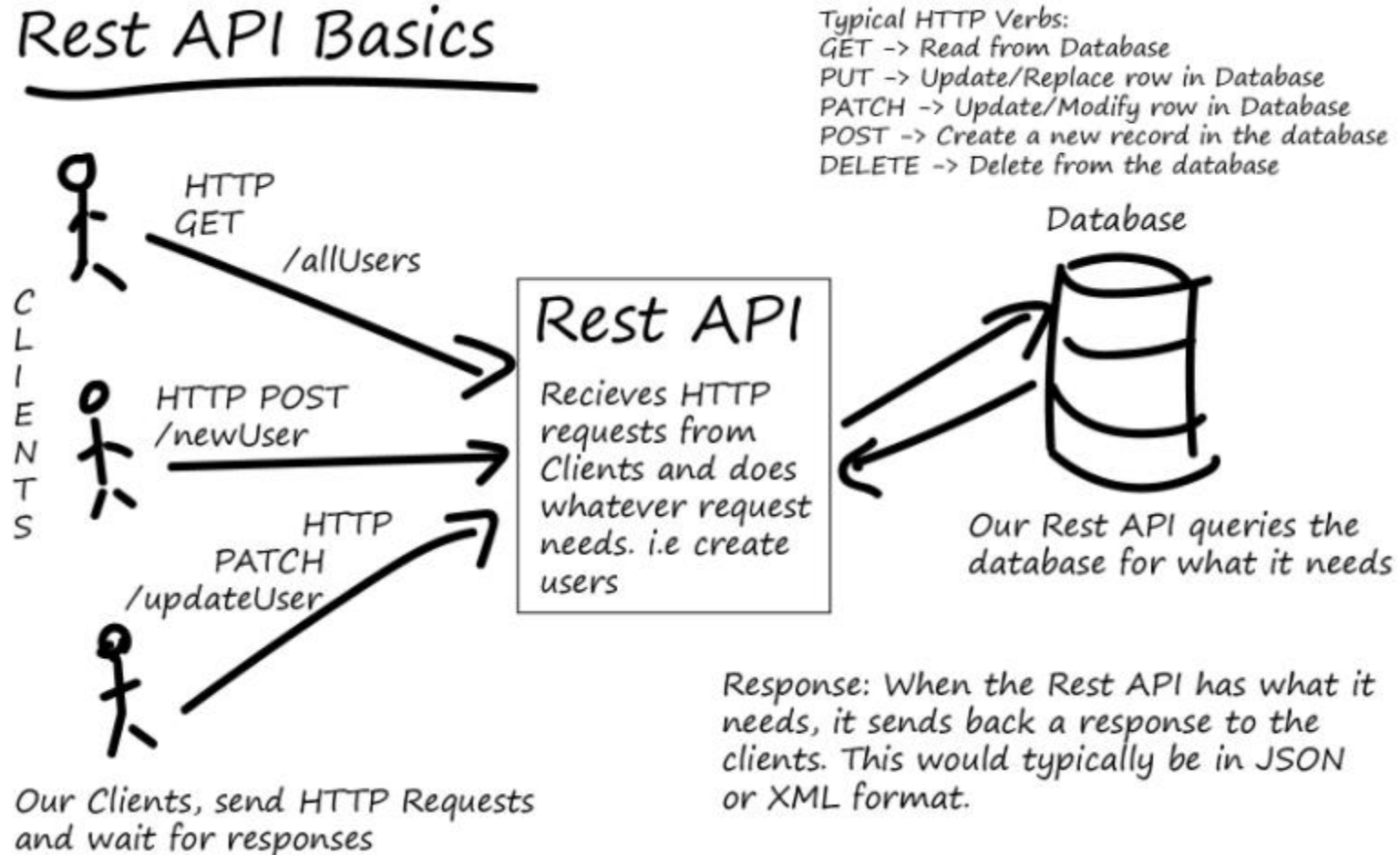
- ❖ Kiến trúc client-server
- ❖ Stateless
- ❖ Cacheability
- ❖ Layered system
- ❖ Uniform interface



□ Nguyên lý hoạt động RESTful API





Rest API Basics



- GET(SELECT): Trả về một Resource hoặc một danh sách Resource.
- POST (CREATE): Tạo mới một Resource.
- PUT (UPDATE): Cập nhật thông tin cho Resource.
- DELETE (DELETE): Xóa một Resource

- ❑ Không nên dùng các động từ, chỉ sử dụng danh từ số nhiều

GET /getAllCars		GET /cars	
POST /createNewCar		POST /cars	
PUT /updateAllRedCars		PUT /cars/123	
DELETE /deleteAllRedCars		DELETE /cars/123	

An orange arrow points from the left column to the right column.

- ❑ Sử dụng '/' để chỉ ra mối quan hệ thứ bậc

```
http://api.example.com/cars
http://api.example.com/cars/{id}
```

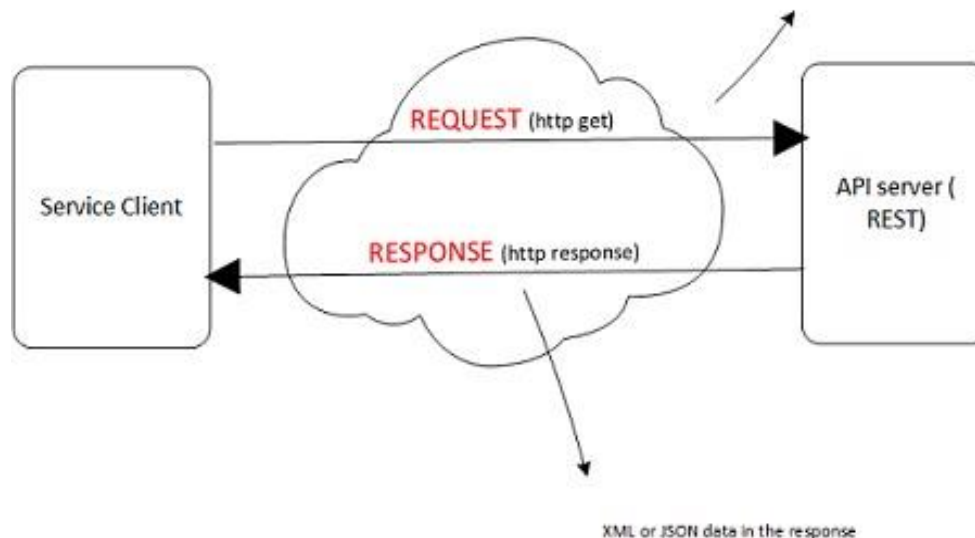
- ❑ Sử dụng chữ thường trong URIs

```
http://api.example.com/GET-DEVICES // Bad
http://api.example.com/Get-Devices // Bad
http://api.example.com/get-devices // Good
```

- ❑ Sử dụng SSL/TLS

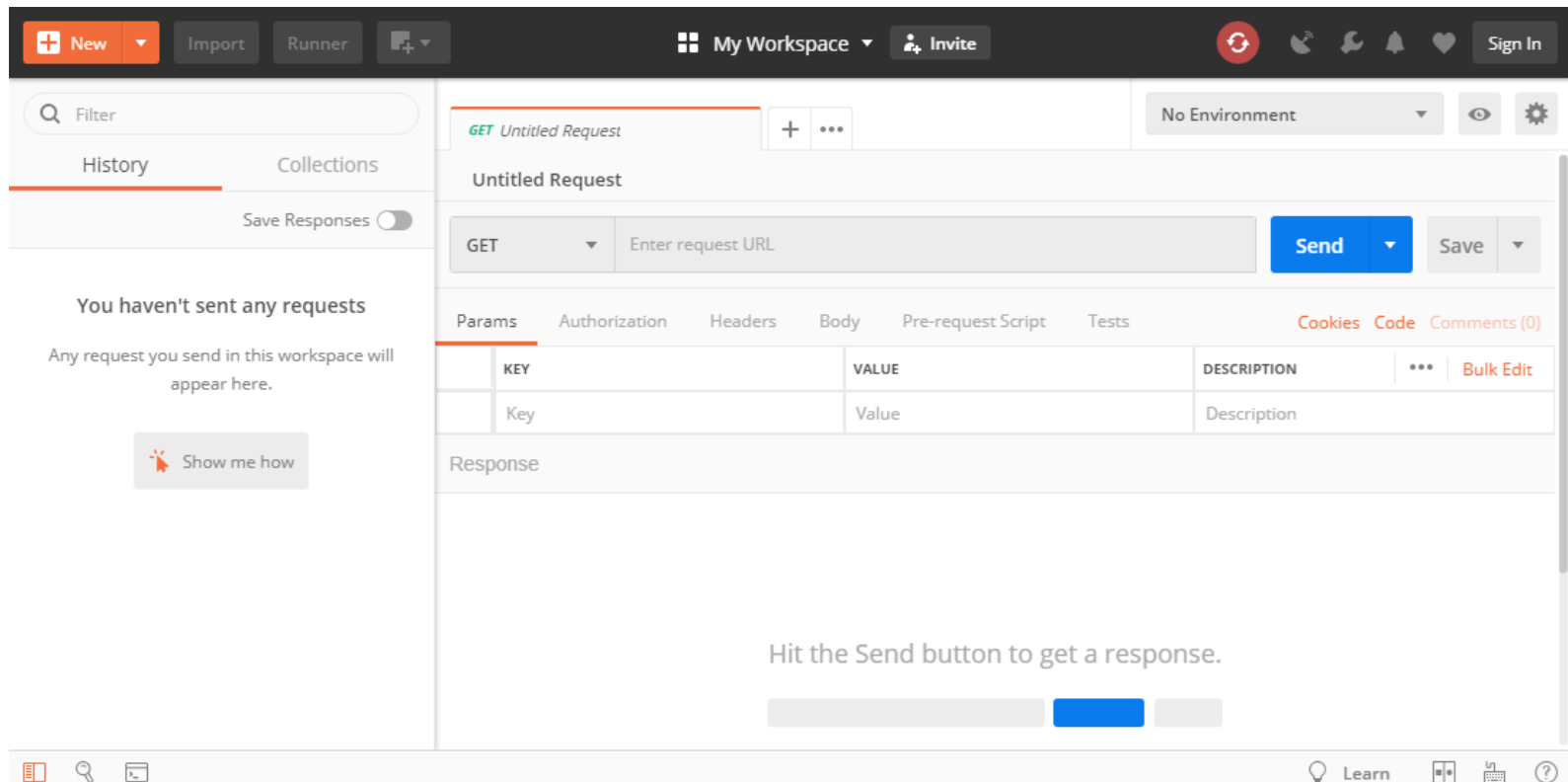
```
http://api.example.com/cars // Bad
https://api.example.com/cars // Good
```

- ❑ Postman là một App Extensions, cho phép làm việc với các API, nhất là REST, giúp ích rất nhiều cho việc testing. Hỗ trợ tất cả các phương thức HTTP (GET, POST, PUT, DELETE, OPTIONS, HEAD ...) Postman cho phép lưu lại các lần sử dụng.
- ❑ Có thể thiết kế, mô phỏng, gỡ lỗi, kiểm tra, tài liệu, theo dõi và xuất bản tất cả các API



❑ Cách sử dụng Postman

- Cho phép gửi HTTP Request với các method GET, POST, PUT, DELETE.
- Cho phép post dữ liệu dưới dạng form (key-value), text, [json](#).
- Hiện kết quả trả về dạng text, hình ảnh, XML, JSON.
- Hỗ trợ authorization (OAuth1, 2).
- Cho phép thay đổi header của các request.

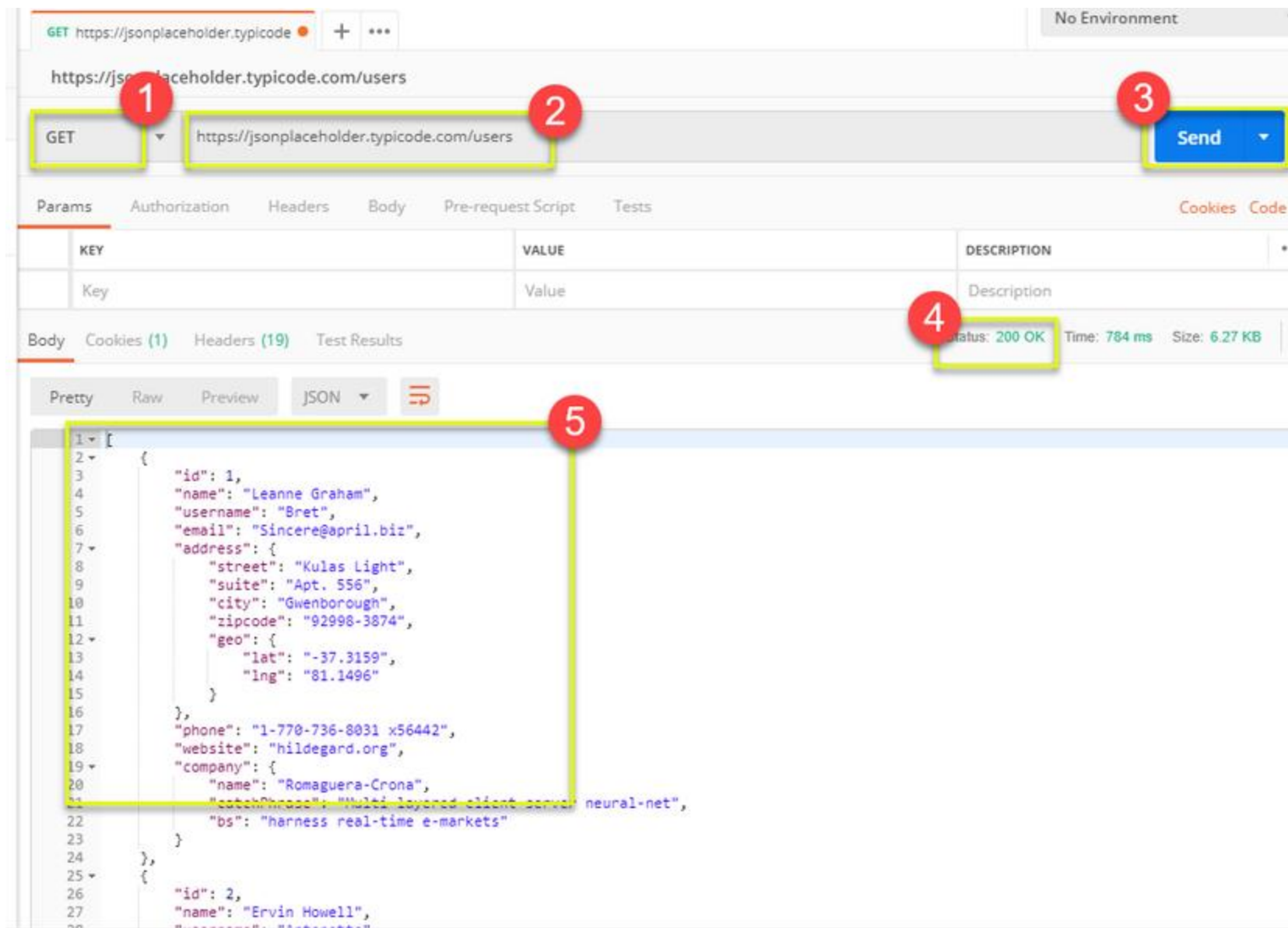


- ❑ Json là một kiểu định dạng dữ liệu trong đó sử dụng văn bản thuần túy, định dạng JSON sử dụng các cặp key - value để dữ liệu sử dụng
- ❑ Được xây dựng dựa trên một tiêu chuẩn của ngôn ngữ JavaScript và là viết tắt của cụm từ JavaScript Object Notation
- ❑ JSON là một tiêu chuẩn mở để trao đổi dữ liệu trên web

```
{  
    "name" : "Asp-Core-API",  
    "version" : "3.1",  
    "license" : "commercial"  
}
```

- ❑ Có thể thấy cú pháp của JSON có 2 phần đó là key và value:
 - ❖ Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}
 - ❖ Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép {}, nếu bạn đặt nó trong dấu nháy đơn thì đây không phải là một chuỗi JSON đúng chuẩn. Nếu trường hợp trong value của bạn có chứa dấu nháy kép " thì hãy dùng dấu \ để đặt trước nó, ví dụ "json là gì\".
 - ❖ Nếu có nhiều dữ liệu thì dùng dấu phẩy , để ngăn cách.
 - ❖ Các key của JSON bạn nên đặt chữ cái không dấu hoặc số, dấu _ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số.
- ❑ C# hỗ trợ nhiều cách tương tác với Json thông qua các Namespaces (System.Text.Json, Newtonsoft.Json...)

- ❑ Truy cập vào trang chủ: <https://www.getpostman.com/> chọn nền tảng muốn cài đặt như cho Mac, Windows hoặc Linux
- ❑ Làm việc với request get (sử dụng URL làm ví dụ <https://jsonplaceholder.typicode.com/users>)
 1. Thiết lập request HTTP của bạn là GET
 2. Trong trường URL yêu cầu, nhập vào link
 3. Kích nút Send
 4. Bạn sẽ nhìn thấy message là 200 ok
 5. Sẽ hiển thị kết quả 10 người dùng trong phần Body



GET https://jsonplaceholder.typicode.com/users

GET https://jsonplaceholder.typicode.com/users Send

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION
Key	Value	Description

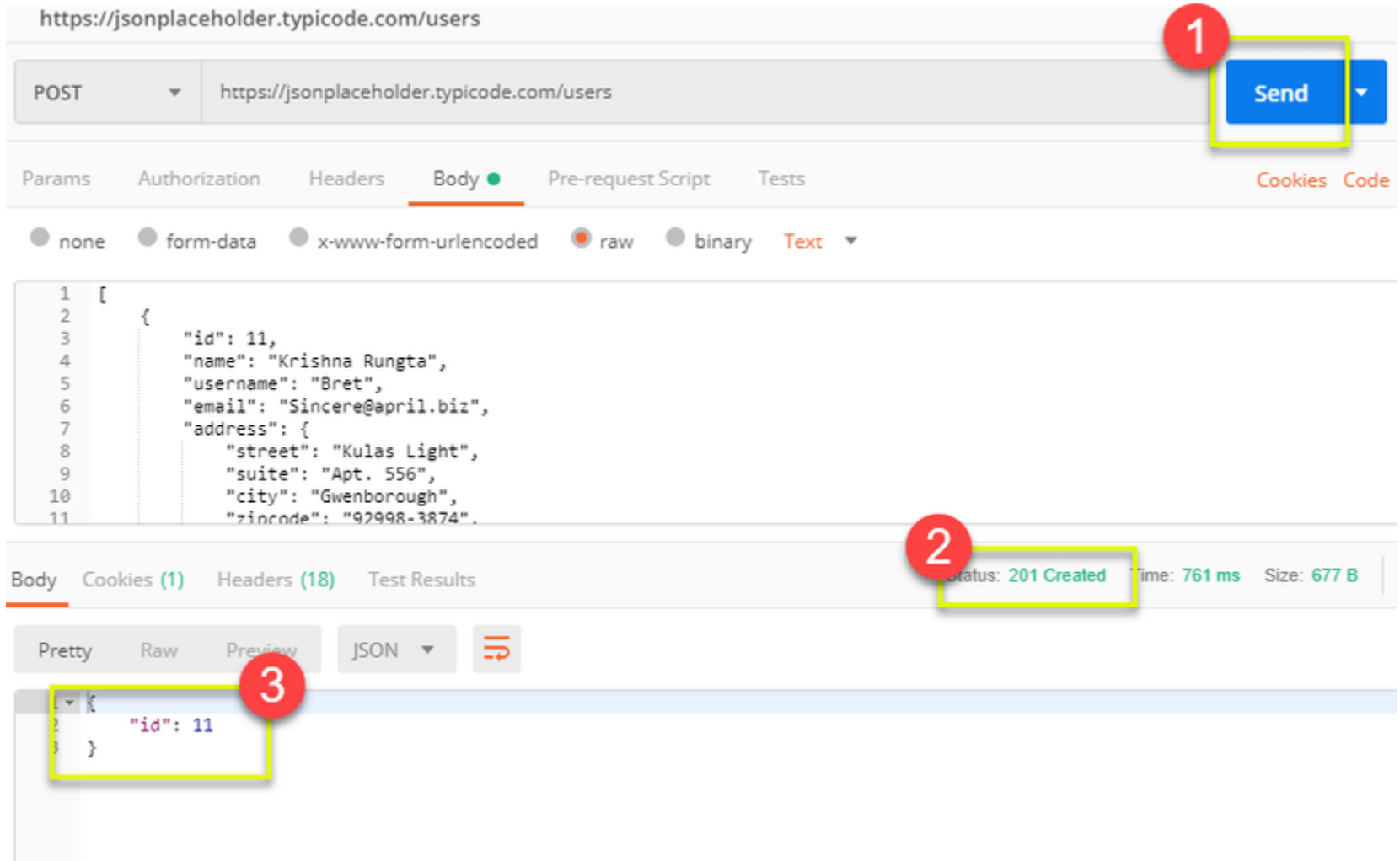
Body Cookies (1) Headers (19) Test Results

Status: 200 OK Time: 784 ms Size: 6.27 KB

Pretty Raw Preview JSON

```

1 [
2   {
3     "id": 1,
4     "name": "Leanne Graham",
5     "username": "Bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas Light",
9       "suite": "Apt. 556",
10      "city": "Gwenborough",
11      "zipcode": "92998-3874",
12      "geo": {
13        "lat": "-37.3159",
14        "lng": "81.1496"
15      }
16    },
17    "phone": "1-770-736-8031 x56442",
18    "website": "hildegard.org",
19    "company": {
20      "name": "Romaguera-Crona",
21      "catchPhrase": "Multi-layered client-server neural-net",
22      "bs": "harness real-time e-markets"
23    }
24  },
25  {
26    "id": 2,
27    "name": "Ervin Howell",
28    "username": "Carter",
29    "email": "Sincere@april.biz",
30    "address": {
31      "street": "Kulas Light",
32      "suite": "Apt. 556",
33      "city": "Gwenborough",
34      "zipcode": "92998-3874",
35      "geo": {
36        "lat": "-37.3159",
37        "lng": "81.1496"
38      }
39    },
40    "phone": "1-770-736-8031 x56442",
41    "website": "hildegard.org",
42    "company": {
43      "name": "Romaguera-Crona",
44      "catchPhrase": "Multi-layered client-server neural-net",
45      "bs": "harness real-time e-markets"
46    }
47  }
48 ]
  
```



The screenshot shows the Postman interface for a POST request to `https://jsonplaceholder.typicode.com/users`. The request body is a JSON object representing a user. The response status is `201 Created`. The response body is a JSON object with `"id": 11` highlighted.

1 Send

POST `https://jsonplaceholder.typicode.com/users`

Params Authorization Headers **Body** Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary Text

```

1 [
2   {
3     "id": 11,
4     "name": "Krishna Rungta",
5     "username": "Bret",
6     "email": "Sincere@april.biz",
7     "address": {
8       "street": "Kulas Light",
9       "suite": "Apt. 556",
10      "city": "Gwenborough",
11      "zipcode": "92998-3874"
12    }
13  }
14 ]

```

Body Cookies (1) Headers (18) Test Results Status: 201 Created Time: 761 ms Size: 677 B

Pretty Raw Preview JSON

```

{
  "id": 11
}

```

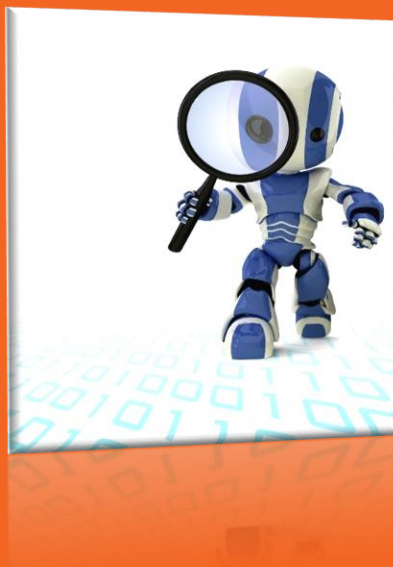
Tham khảo hướng dẫn sử dụng: <https://learning.postman.com/docs/publishing-your-api/documenting-your-api/>



DEMO

- Demo các thao tác crud với Postman





LẬP TRÌNH C# 5

BÀI 1: WEB APIs - ASP.NET CORE (P2)

- ❑ Asp.net core hỗ trợ phát triển Api thông qua template có sẵn hoặc sử dụng EF core
- ❑ Sử dụng controller kế thừa từ ControllerBase
- ❑ ControllerBase cung cấp các properties and methods quản lý các thông tin trong HTTP requests
- ❑ Dùng [ApiController] attribute cho controller
- ❑ Dùng attribute routing tạo đường dẫn Url cho Api

```
[ApiController]  
[Route("someURL/[controller]")]  
public class ExampleController : ControllerBase
```

- ❑ Tạo ứng dụng asp core empty
- ❑ Tạo model 'Reservation'
- ❑ Áp dụng DI parten, tạo Interface 'IRepository' với các Thao tác Crud tương ứng

```
public class Reservation
{
    5 references
    public int Id { get; set; }
    4 references
    public string Name { get; set; }
    4 references
    public string StartLocation { get; set; }
    4 references
    public string EndLocation { get; set; }
}
```

```
public interface IRepository
{
    2 references
    IEnumerable<Reservation> Reservations { get; }
    2 references
    Reservation this[int id] { get; }
    4 references
    Reservation AddReservation(Reservation reservation);
    2 references
    Reservation UpdateReservation(Reservation reservation);
    2 references
    void DeleteReservation(int id);
}
```

❑ Tạo class 'Repository' hiện thực interface

```
public class Repository : IRepository
{
    private Dictionary<int, Reservation> items;
    0 references
    public Repository()
    {
        items = new Dictionary<int, Reservation>();
        new List<Reservation> {
            new Reservation {Id=1, Name = "Thepv",
                StartLocation = "Ha Noi", EndLocation="Can Tho" },
            new Reservation {Id=2, Name = "Fpoly",
                StartLocation = "Sai Gon", EndLocation="Tay Nguyen" }

        }.ForEach(r => AddReservation(r));
    }
    2 references
    public Reservation this[int id] => items.ContainsKey(id) ? items[id] : null;
}
```

❑ Tạo class 'Repository' hiện thực interface

```
public IEnumerable<Reservation> Reservations => items.Values;
```

4 references

```
public Reservation AddReservation(Reservation reservation)
{
    if (reservation.Id == 0)
    {
        int key = items.Count;
        while (items.ContainsKey(key)) { key++; };
        reservation.Id = key;
    }
    items[reservation.Id] = reservation;
    return reservation;
}
```

2 references

```
public void DeleteReservation(int id) => items.Remove(id);
```

2 references

```
public Reservation UpdateReservation(Reservation reservation) =>
    AddReservation(reservation);
```

❑ Inject AddSingleton DI trong startup.cs

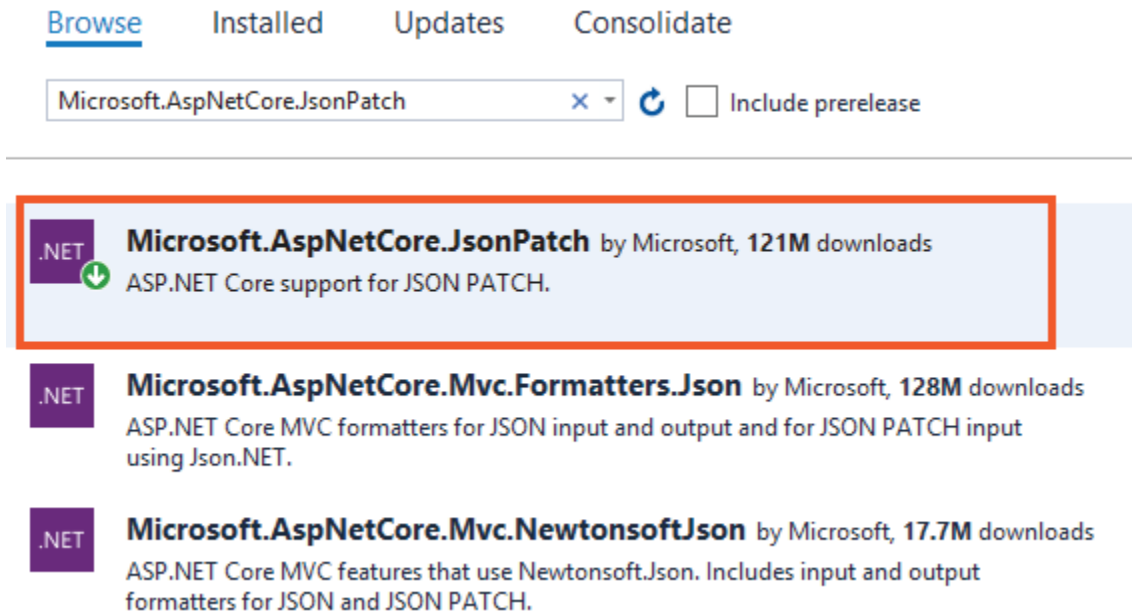
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<IRepository, Repository>();
    services.AddControllersWithViews();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();

app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

- ❑ Thiết lập các HTTP Request tương ứng cho các action method tạo ra các Url cho các thao tác như crud
- ❑ HTTP request có 2 loại được sử dụng phổ biến nhất là GET và POST
 - GET: được sử dụng để lấy thông tin từ server theo URI đã cung cấp.
 - HEAD: giống với GET nhưng response trả về không có body, chỉ có header.
 - POST: gửi thông tin tới sever thông qua các biểu mẫu http.
 - PUT: ghi đè tất cả thông tin của đối tượng với những gì được gửi lên.
 - PATCH: ghi đè các thông tin được thay đổi của đối tượng.
 - DELETE: xóa tài nguyên trên server.
 - CONNECT: thiết lập một kết nối tới server theo URI.
 - OPTIONS: mô tả các tùy chọn giao tiếp cho resource.
 - TRACE: thực hiện một bài test loop – back theo đường dẫn đến resource

- ❑ Cài đặt Microsoft.AspNetCore.JsonPatch package hỗ trợ mô tả các thay đổi trong tài liệu json



❑ Tạo controller xử lý các HTTP Request

```
[ApiController]
[Route("api/[controller]")]
1 reference
public class ReservationController : ControllerBase
{
    private IRepository repository;
    0 references
    public ReservationController(IRepository repo) => repository = repo;
    [HttpGet]
    0 references
    public IEnumerable<Reservation> Get() => repository.Reservations;
    [HttpGet("{id}")]
    1 reference
    public ActionResult<Reservation> Get(int id)
    {
        if (id == 0)
            return BadRequest("Value must be passed in the request body.");
        return Ok(repository[id]);
    }
    [HttpPost]
    0 references
    public Reservation Post([FromBody] Reservation res) =>
        repository.AddReservation(new Reservation
        {
            Name = res.Name,
            StartLocation = res.StartLocation,
            EndLocation = res.EndLocation
        });
}
```

❑ Tạo controller xử lý các HTTP Request

[HttpPut]

0 references

```
public Reservation Put([FromBody] Reservation res) => repository.UpdateReservation(res);
```

[HttpPatch("{id}")]

0 references

```
public StatusCodeResult Patch(int id, [FromBody] JsonPatchDocument<Reservation> patch)
```

```
{  
    var res = (Reservation)((OkObjectResult)Get(id).Result).Value;  
    if (res != null)  
    {  
        patch.ApplyTo(res);  
        return Ok();  
    }  
    return NotFound();  
}
```

[HttpDelete("{id}")]

0 references

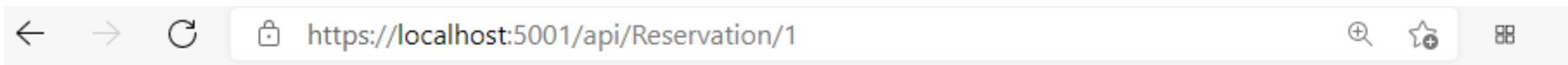
```
public void Delete(int id) => repository.DeleteReservation(id);
```

- ❑ Run ứng dụng với api/Reservation routing sẽ nhận được các đối tượng 'reservations' dưới dạng json



```
[{"id":1,"name":"Thepv","startLocation":"Ha Noi","endLocation":"Can Tho"}, {"id":2,"name":"Fpoly","startLocation":"Sai Gon","endLocation":"Tay Nguyen"}]
```

- ❑ Run ứng dụng với api/Reservation/1 routing sẽ nhận được các đối tượng 'reservations' có id = 1 dưới dạng json

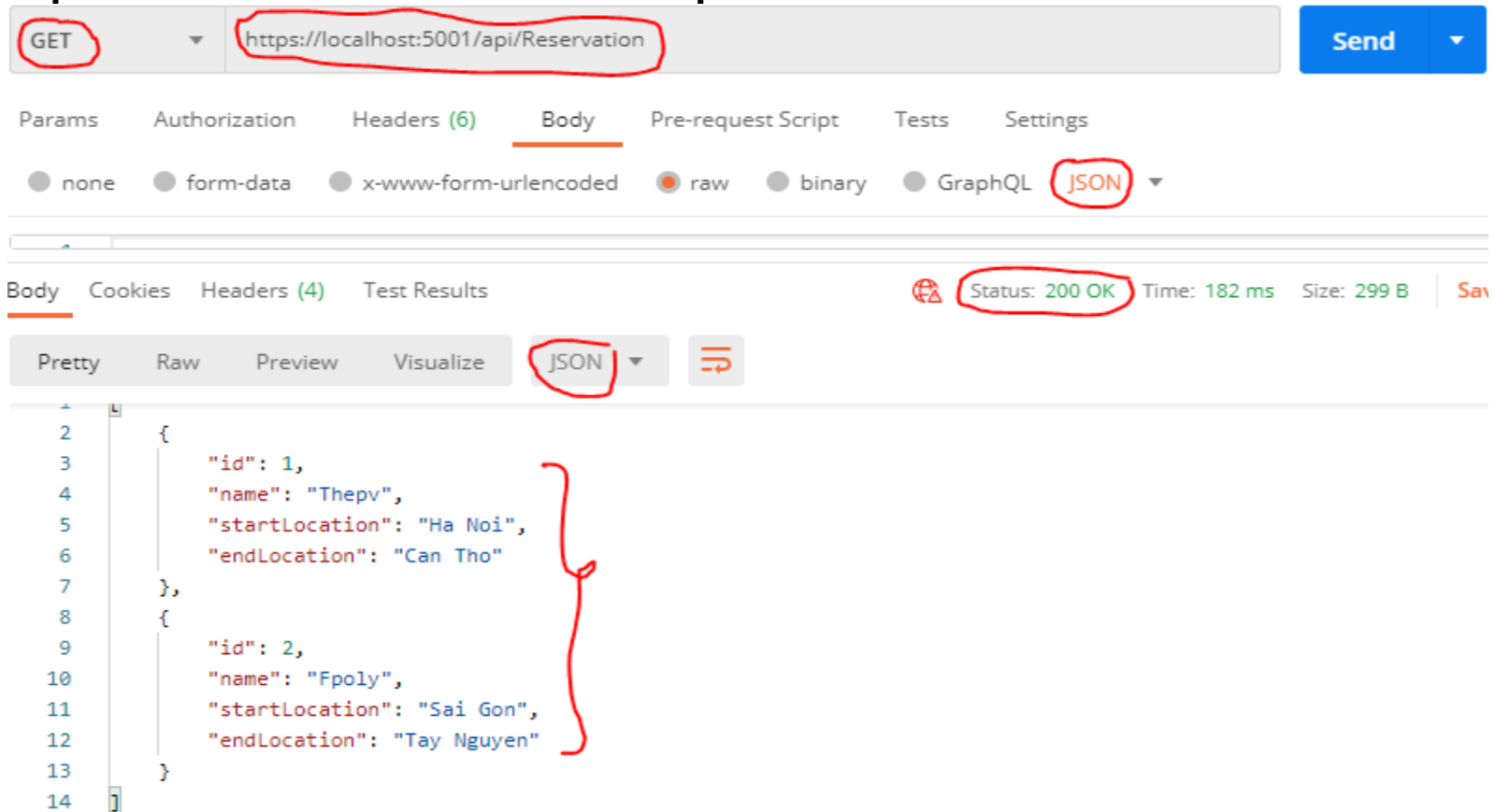


```
{"id":1,"name":"Thepv","startLocation":"Ha Noi","endLocation":"Can Tho"}
```

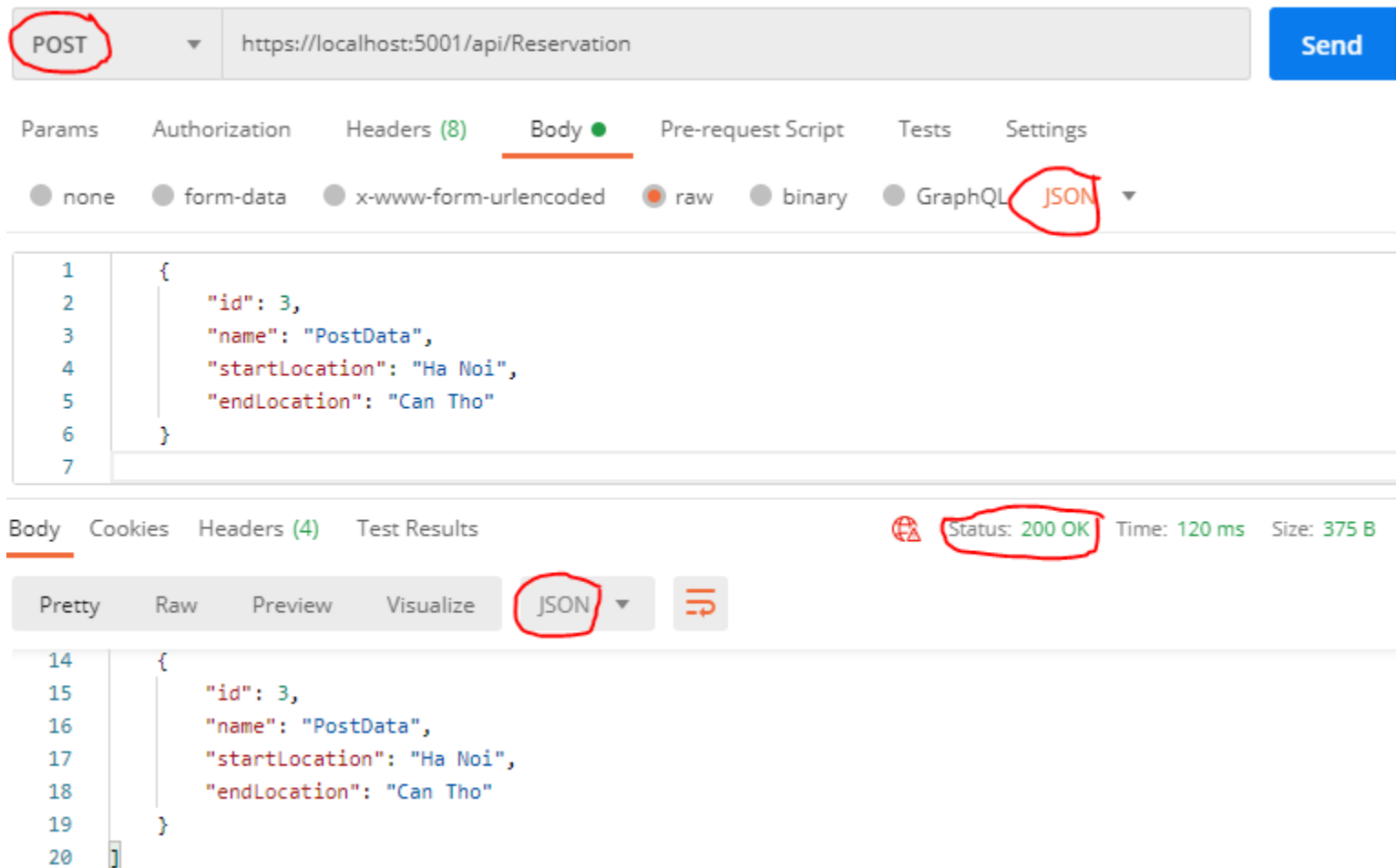
❑ Status code: Khi chúng ta request một API nào đó thường thì sẽ có vài status code để nhận biết

- 200 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.
- 201 Created – Trả về khi một Resource vừa được tạo thành công.
- 204 No Content – Trả về khi Resource xóa thành công.
- 304 Not Modified – Client có thể sử dụng dữ liệu cache.
- 400 Bad Request – Request không hợp lệ
- 401 Unauthorized – Request cần có auth.
- 403 Forbidden – bị từ chối không cho phép.
- 404 Not Found – Không tìm thấy resource từ URI
- 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.
- 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.
- 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.
- 422 Unprocessable Entity – Dữ liệu không được xác thực
- 429 Too Many Requests – Request bị từ chối do bị giới hạn

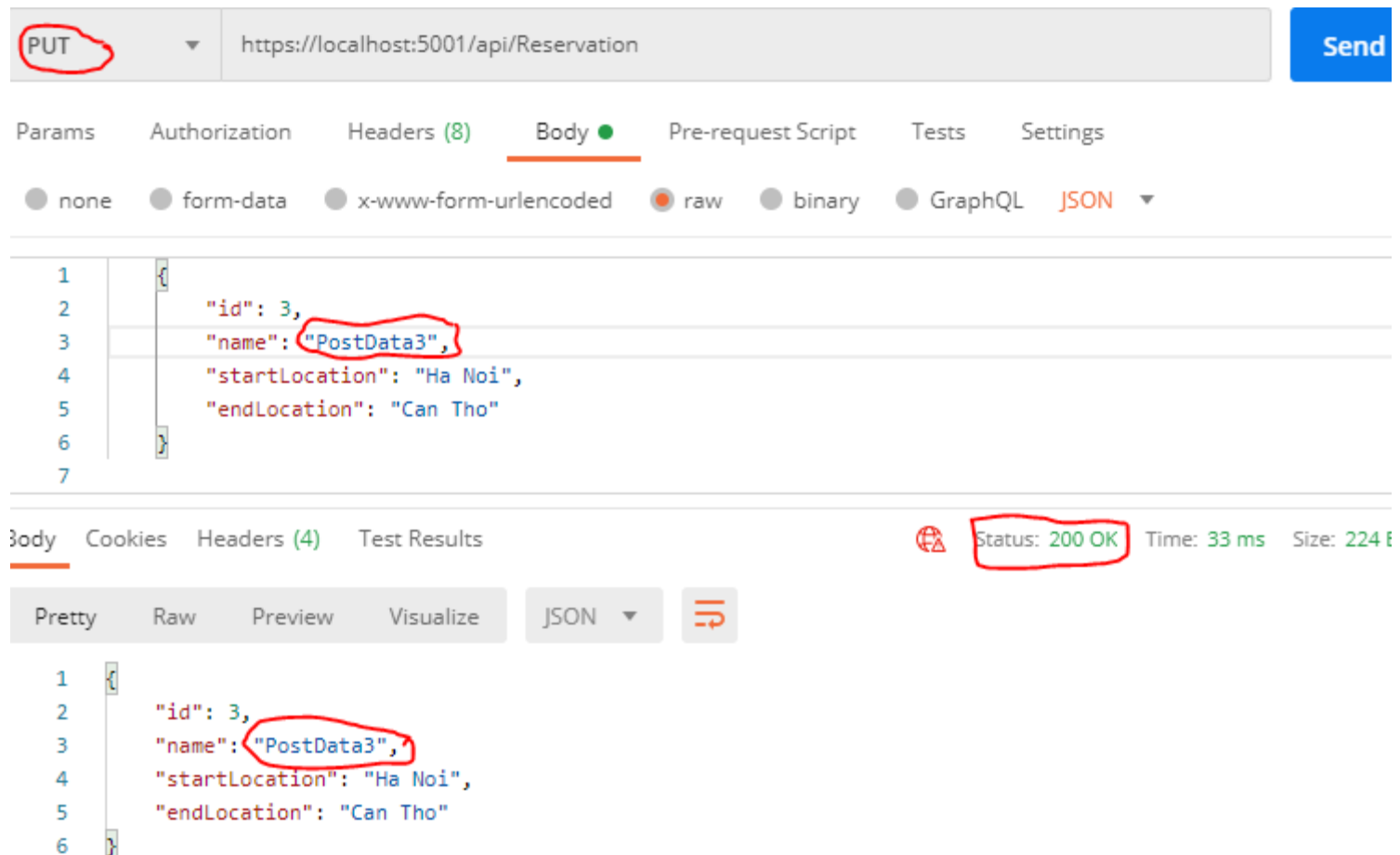
- ❑ Có thể kiểm tra các api có hoạt động hay không thông qua postman, ví dụ gọi api lấy toàn bộ Reservation thì dùng url `https://localhost:5001/api/Reservation`



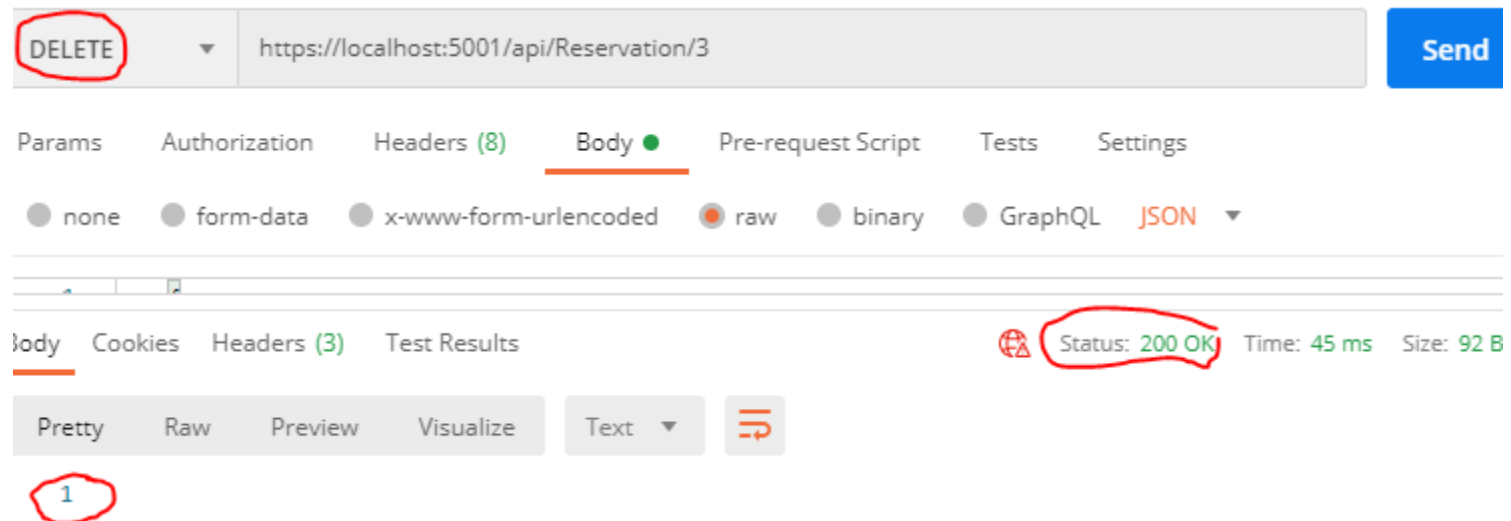
- ❑ Thêm một Reservation thì dùng url <https://localhost:5001/api/Reservation> và cung cấp giá trị cho các đối tượng Reservation

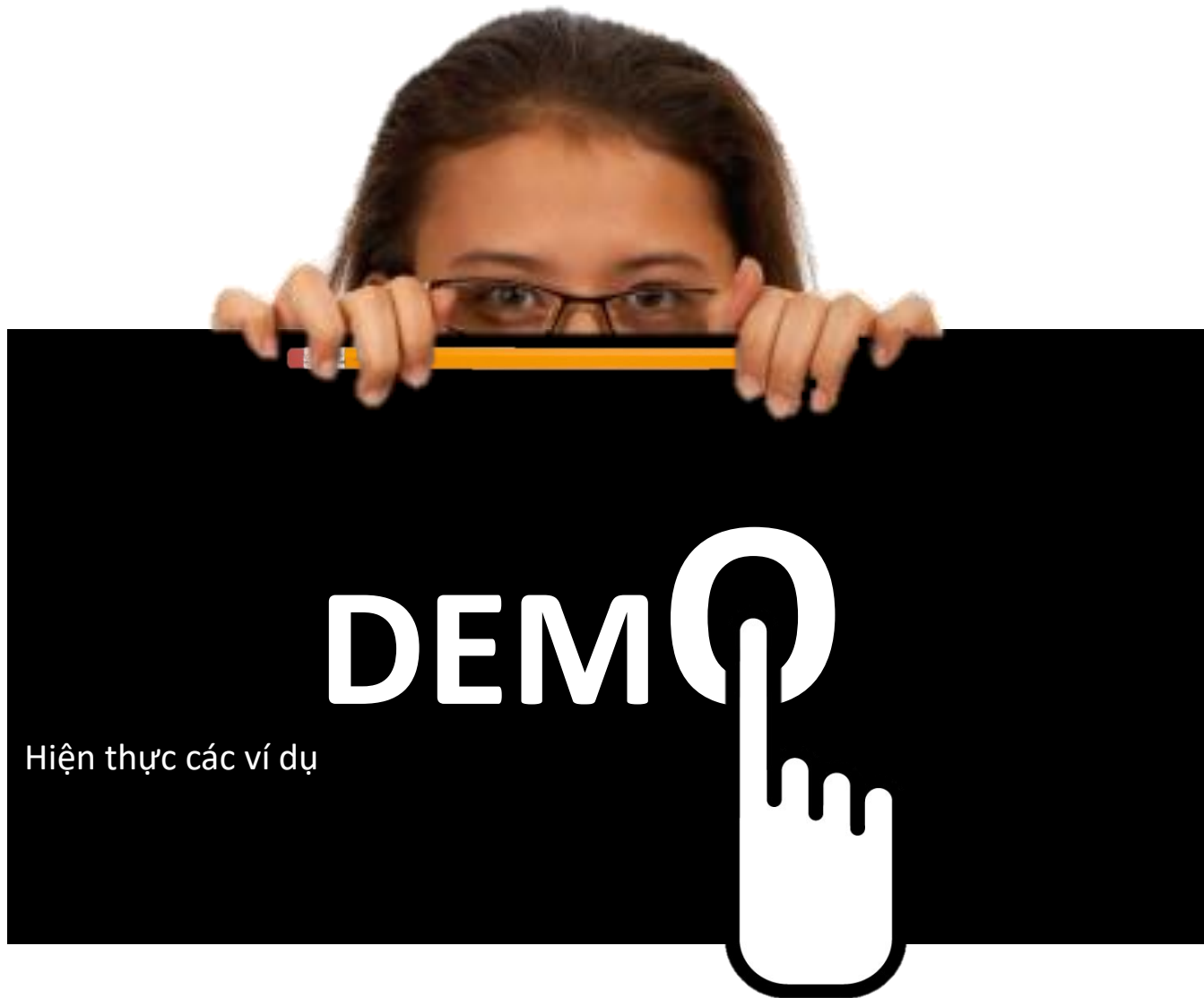


- ❑ Cập nhật một Reservation thì dùng url <https://localhost:5001/api/Reservation> và cung cấp giá trị cập nhật cho các đối tượng Reservation



- ❑ Xóa một Reservation thì dùng url <https://localhost:5001/api/Reservation/id> với id là giá trị id dữ liệu cần xóa





Tổng kết bài học

- ◎ Web Api
- ◎ Testing Api - Postman
- ◎ Web APIs in ASP.NET Core





KẾT THÚC