

# RESTful Web Service

## Xây dựng RESTful Web API trong ASP.NET

### Sử dụng Postman để kiểm thử Web API

Nguyễn Văn Mạnh

# RESTful Web Service

- Là các web service được thiết kế dựa trên kiến trúc REST
- REST( REpresentational State Transfer):
- Là một kiểu kiến trúc thống nhất giúp thiết kế các web service để có thể dễ dàng quản lý các tài nguyên
- Các web service thiết kế dựa trên kiến trúc REST phải tuân thủ 4 nguyên tắc sau:
  - Sử dụng các phương thức HTTP một cách rõ ràng
  - Phi trạng thái
  - Hiển thị các URI như cấu trúc thư mục
  - Truyền tải JSON, XML hoặc cả hai

# Nguyên tắc 1: Sử dụng các phương thức HTTP 1 cách rõ ràng

- REST đòi hỏi lập trình viên xác định rõ ý định của mình thông qua các phương thức của HTTP:
  - 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.
  - PATCH (UPDATE): Cập nhật một thành phần, thuộc tính của Resource.
  - DELETE (DELETE): Xoá một Resource.
  - HEAD: Trả về thông tin chung của một hoặc danh sách Resource.
  - OPTIONS: Trả về thông tin mà người dùng được phép với Resource

# Nguyên tắc 1:

## Sử dụng các phương thức HTTP 1 cách rõ ràng

• VD:

GET /user/adduser?name=Robert —————> Sai

POST /users

Host: myserver

Content-Type: application/xml

<?xml version="1.0"?>

<user>

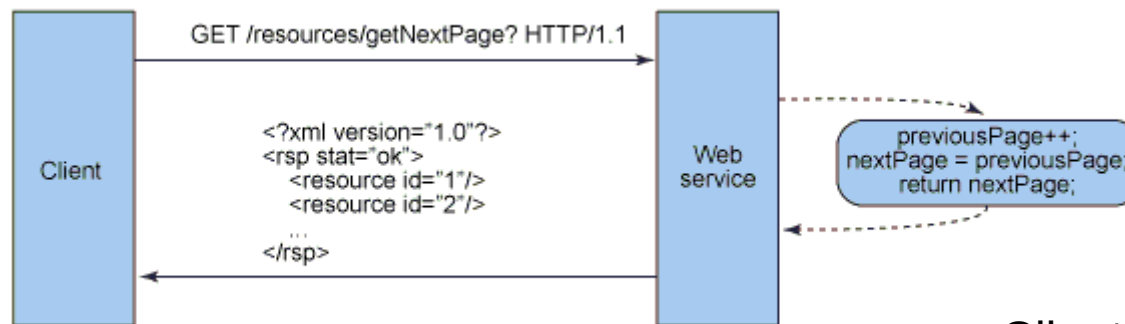
    <name>Robert</name>

</user>

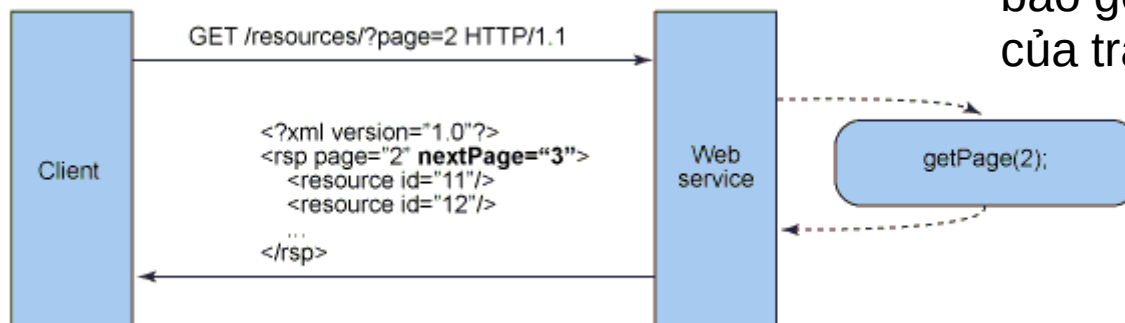
—————> Đúng

# Nguyên tắc 2: Phi trạng thái

- REST không lưu trữ thông tin của client
- VD:
  - Bạn vừa gửi yêu cầu để xem trang thứ 2 của một tài liệu, và bây giờ bạn muốn xem trang tiếp theo (sẽ là trang 3). REST không lưu trữ lại thông tin rằng trước đó nó đã phục vụ bạn trang số 2. Điều đó có nghĩa là REST không quản lý phiên làm việc (Session).



Client phải gửi yêu cầu rõ ràng, bao gồm số thự tự của trang cần xem.



# Nguyên tắc 3:

## Hiển thị các URI như cấu trúc thư mục

- URI (**U**niform **R**esource **I**dentifier): dùng để xác định một resource nào đó trên web, về mặt tên hoặc địa chỉ
- Người dùng có thể truy cập vào tài nguyên của REST thông qua các URI
- Cấu trúc của một URI nên được đơn giản, có thể dự đoán, và dễ hiểu
- VD:

<http://myservice.com/weather/hanoi/2016-11-11>

# Nguyên tắc 4:

## Truyền tải JSON, XML hoặc cả hai

- Khi Client gửi một yêu cầu tới web service, nó thường được truyền tải dưới dạng XML hoặc JSON và thông thường nhận về với hình thức tương tự
- Client cũng có thể chỉ định kiểu dữ liệu nhận về mong muốn trong Header của Request

Kiểu dữ liệu	Accept/Content-Type
JSON	application/JSON
XML	application/XML
XHTML	application/xhtml+xml

- VD:

GET /weather/hanoi/2016-08-27

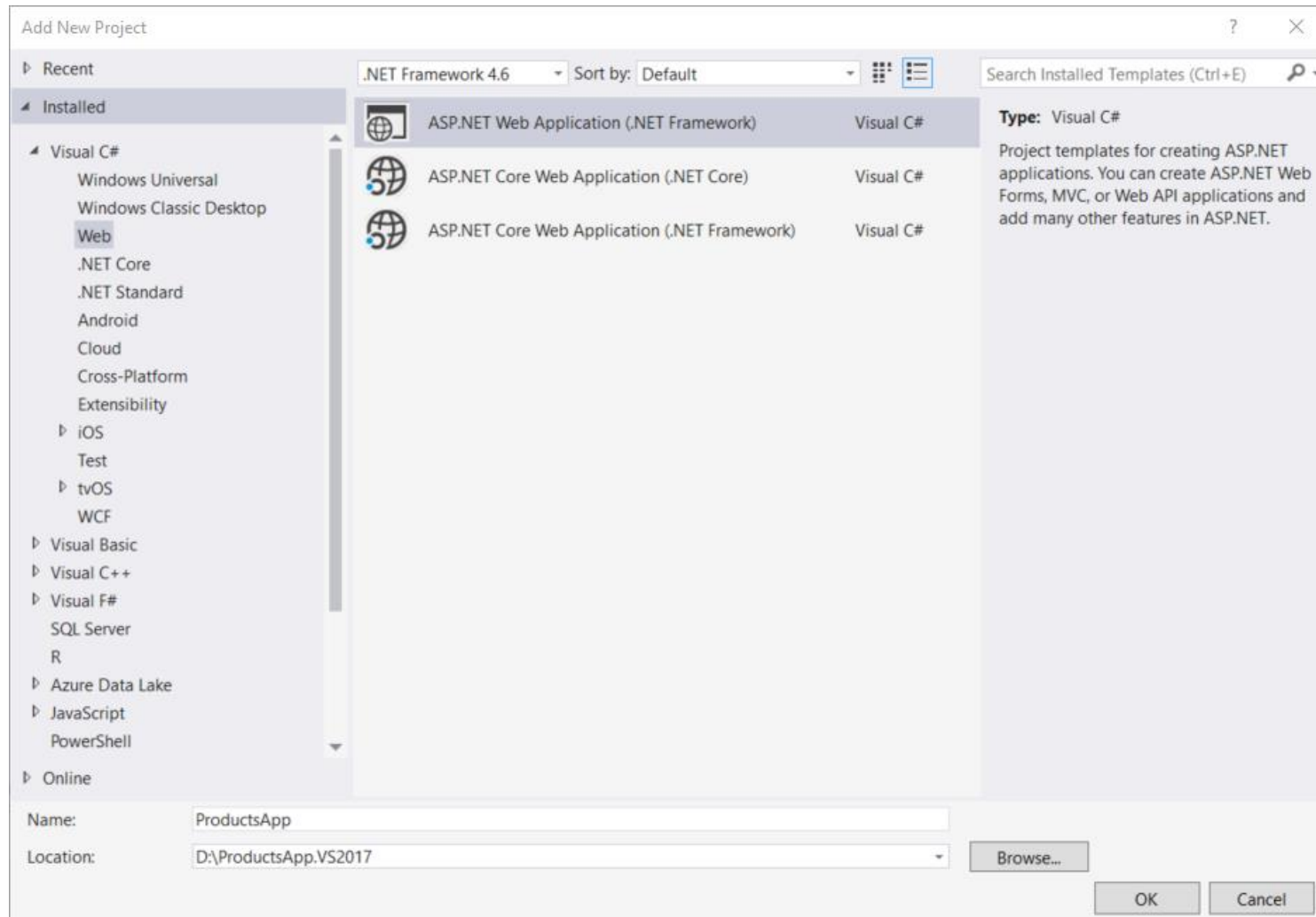
Host: myservice.com

Accept: application/xml

# Xây dựng RESTful Web API trong ASP.NET



# Tạo Web API project



# Tạo Web API project

New ASP.NET Web Application - ProductsApp

**ASP.NET 4.6 Templates**

Empty Web Forms MVC Web API Single Page Application Azure API App Azure Mobile App

An empty project template for creating ASP.NET applications. This template does not have any content in it.  
[Learn more](#)

Change Authentication

Authentication: **No Authentication**

Add folders and core references for:

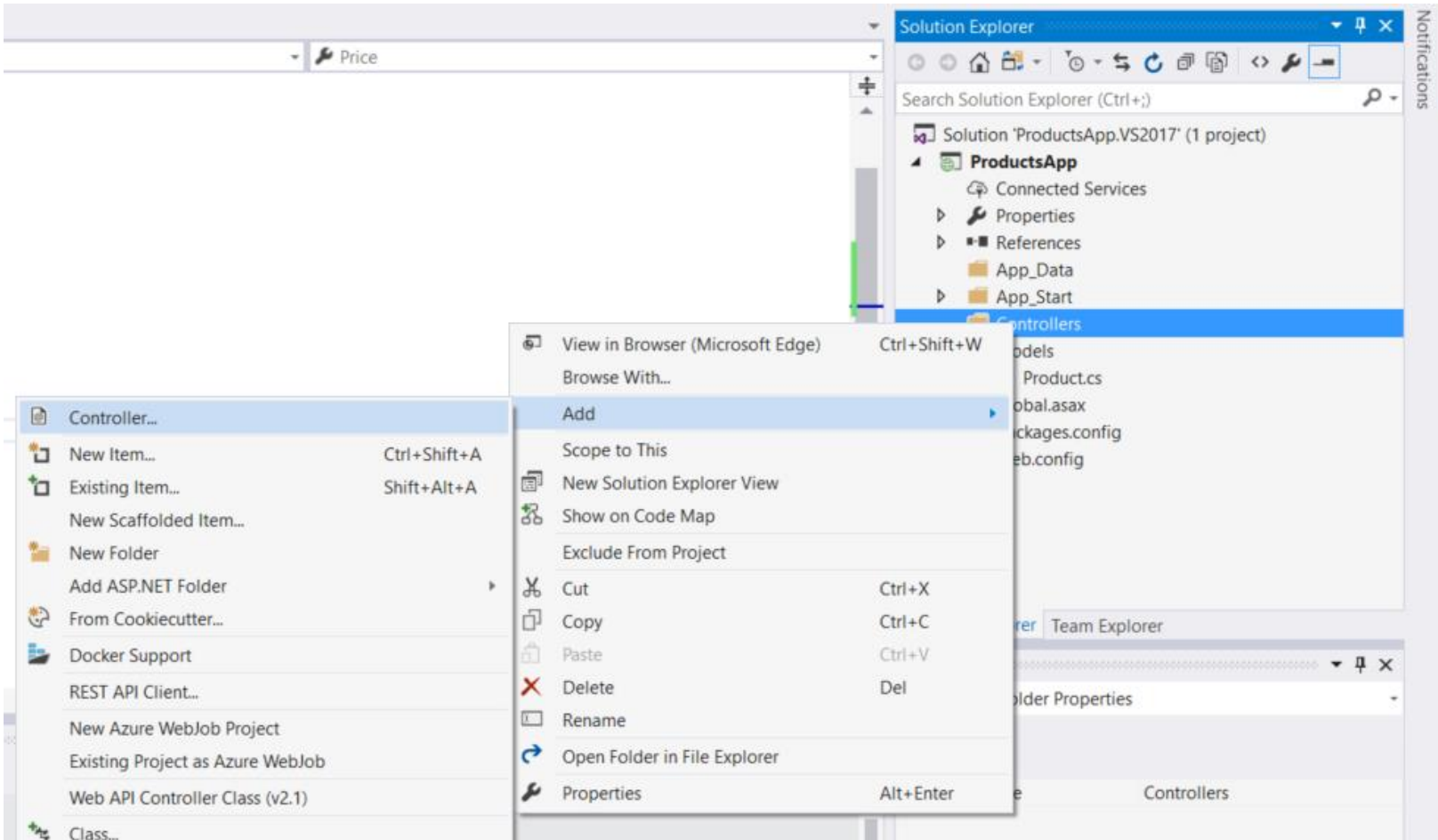
☐ Web Forms ☐ MVC ☒ Web API

☐ Add unit tests

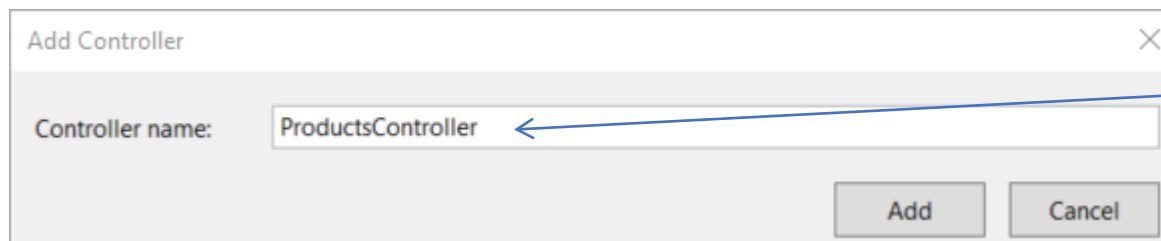
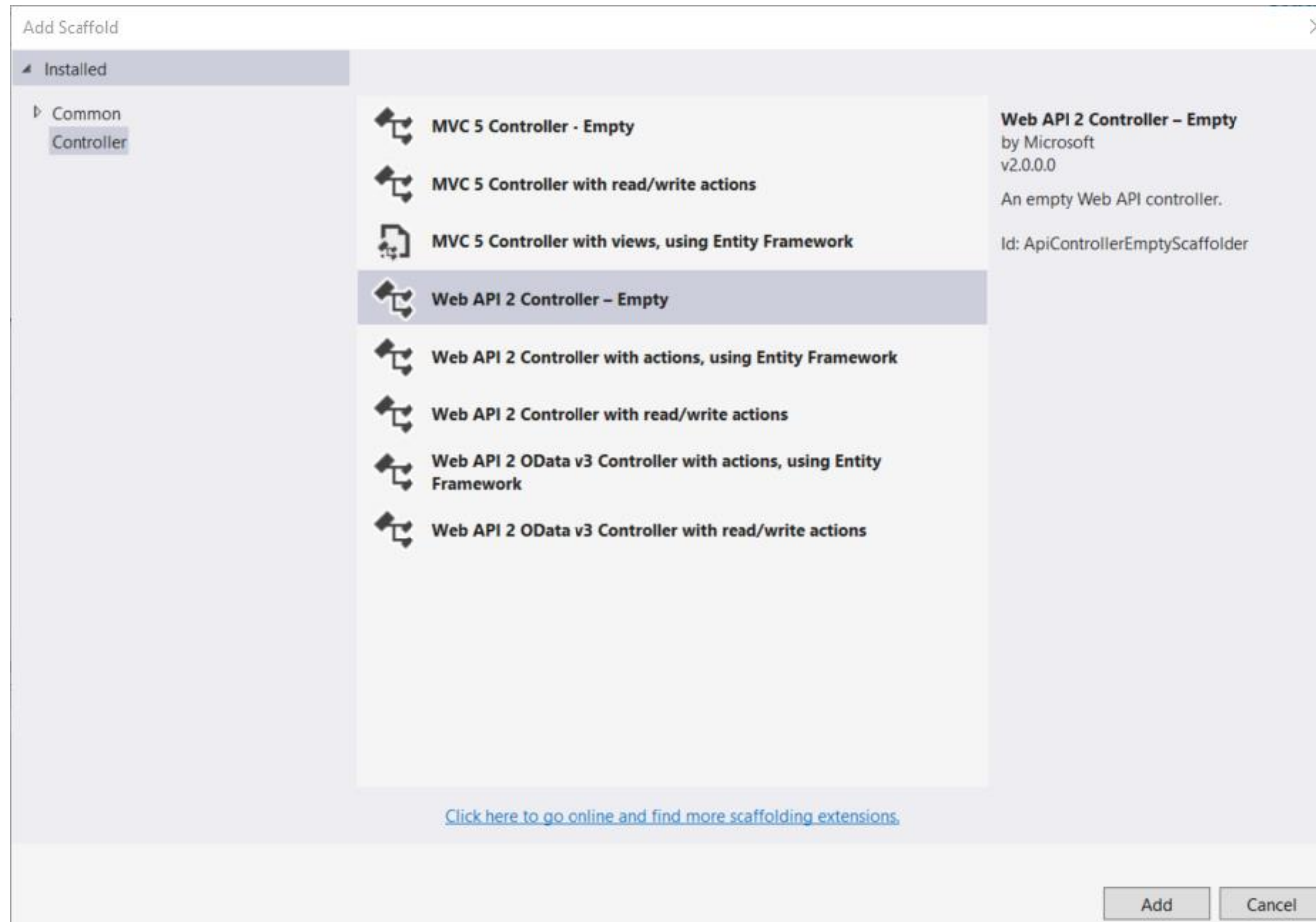
Test project name: ProductsApp.Tests

OK Cancel

# Tạo API controller



# Tạo API Controller



Tên Controller phải kết thúc bằng **Controller**

# Tạo API Controller

- Tên controller phải kết thúc bằng “Controller” và phải kế thừa lớp System.Web.Http.ApiController
- Tên action method phải bắt đầu bằng các HTTP method
- Dựa vào request URL và HTTP method (GET/POST/PUT/PATCH/DELETE), web api quyết định gọi controller và action method tương ứng
- VD: GET http://localhost:1642/api/values

```
public class ValuesController : ApiController  — Web API controller Base class
{
    // GET api/values
    public IEnumerable<string> Get()  — Handles Http GET request
    {                                http://localhost:1234/api/values
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    public string Get(int id)  — Handles Http GET request with query string
    {                          http://localhost:1234/api/values?id=1
        return "value";
    }

    // POST api/values
    public void Post([FromBody]string value)  — Handles Http POST request
    {                                          http://localhost:1234/api/values

    }

    // PUT api/values/5
    public void Put(int id, [FromBody]string value)  — Handles Http Put request
    {                                                  http://localhost:1234/api/values?id=1

    }

    // DELETE api/values/5
    public void Delete(int id)  — Handles Http DELETE request
    {                          http://localhost:1234/api/values?id=1

    }
}
```

# Action Method Naming Conventions

HTTP Request Method	Tên action method
GET	Get(), GetAllStudents(),... <i>*bắt đầu bằng <b>Get</b>*</i>
POST	Post(), PostNewStudent(),... <i>*bắt đầu bằng <b>Post</b>*</i>
PUT	Put(), PutStudent(),... <i>*bắt đầu bằng <b>Put</b>*</i>
PATCH	Patch(), PatchStudent(),... <i>*bắt đầu bằng <b>Patch</b>*</i>
DELETE	Delete(), DeleteStudent(),... <i>*bắt đầu bằng <b>Delete</b>*</i>

# Tạo API Controller

- Nếu không muốn bắt đầu bằng các phương thức Http có thể thêm **Http verb attribute** (HttpGet, HttpPost, HttpPut, HttpDelete) lên đầu các phương thức

```
public class ValuesController : ApiController
{
    [HttpGet] <-----
    public IEnumerable<string> Values()
    {
        return new string[] { "value1", "value2" };
    }

    [HttpGet] <-----
    public string Value(int id)
    {
        return "value";
    }

    [HttpPost] <-----
    public void SaveNewValue([FromBody]string value)
    {
    }

    [HttpPut] <-----
    public void UpdateValue(int id, [FromBody]string value)
    {
    }

    [HttpDelete] <-----
    public void RemoveValue(int id)
    {
    }
}
```

Http verb attribute

# Web API Routing

- Convention-based Routing:

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

        // school route
        config.Routes.MapHttpRoute(
            name: "School",
            routeTemplate: "api/myschool/{id}",
            defaults: new { controller="school", id = RouteParameter.Optional }
            constraints: new { id = "/d+" }
        );

        // default route
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

- Attribute Routing: →

```
public class StudentController : ApiController
{
    [Route("api/student/names")]
    public IEnumerable<string> Get()
    {
        return new string[] { "student1", "student2" };
    }
}
```



# Parameter Binding

- Các action method có thể có các parameter kiểu primitive hoặc kiểu complex

HTTP Method	Query String	Request Body
GET	Primitive Type, Complex Type	N/A
POST	Primitive Type	Complex Type
PUT	Primitive Type	Complex Type
PATCH	Primitive Type	Complex Type
DELETE	Primitive Type, Complex Type	N/A

- Có thể sử dụng các attribute `[FromUri]`, `[FromBody]` để thay đổi

```
public class StudentController : ApiController
{
    public Student Post([FromUri]Student stud)
    {
    }
}
```

# Kiểu trả về của action method

- Void
- Primitive type or Complex type
- HttpResponseMessage
- IHttpActionResultResult

```
public IHttpActionResultResult Get(int id)
{
    Student stud = GetStudentFromDB(id);

    if (stud == null)
    {
        return NotFound();
    }

    return Ok(stud);
}
```

# Kiểu dữ liệu

- text/html, text/xml, application/json, application/xml, image/jpeg, ...
- Có thể cài đặt trong Request Header (Accept/Content-Type)
- VD:

GET http://localhost:60464/api/student HTTP/1.1

User-Agent: Fiddler

Host: localhost:1234

Accept: application/json

POST http://localhost:60464/api/student?age=15 HTTP/1.1

User-Agent: Fiddler

Host: localhost:60464

Content-Type: application/json

Content-Length: 13

```
{  
  id:1,  
  name:'Steve'  
}
```

# Sử dụng Postman để kiểm thử Service Web API

- Postman là 1 công cụ để kiểm thử Web API: <https://www.getpostman.com/>
- Các chức năng cơ bản:
  - 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



# Sử dụng Postman để kiểm thử Service Web API

## GET

**Xem lược sử**

**Action Method**

**URI**

**Kiểu dữ liệu trả về**

**Dữ liệu trả về**

The screenshot shows the Postman application interface. The left sidebar displays a history of requests, including several GET requests to the endpoint `http://localhost:6549/api/product/sale`. The main workspace shows a GET request to `http://localhost:51748/api/product/sale`. The response is displayed in the bottom section, showing a status of 200 OK and a JSON body. The JSON body contains product information, including the product name, price, and description.

```
{  "ProductName": "Windows 10 Home (OEM)",  "PriceOriginal": 200000,  "PriceAfterSale": 0,  "SaleOffEventId": 1,  "ProductSaleOffId": "4f28a156-5f73-406f-92bc-62f886bd31f7",  "Description": "Key sử dụng kích hoạt trực tuyến (active online), chỉ sử dụng cho 1 máy. Tự",  "CalculationUnitId": 1,  "Warranty": "đổi mới ngay",  "Price": 100000,  "AvatarPath": null,  "AdvertisingImagePath": "",  "CalculationUnitName": "license",  "Id": "b6b260f5-19d1-41c7-b470-6a5bf08f132e",  "CreatedDate": "2018-05-04T02:58:02.287",  "ModifiedDate": "2018-05-04T02:58:02.287"}
```

# Sử dụng Postman để kiểm thử Service Web API

## POST

The screenshot shows the Postman interface for a POST request. The URL is `http://localhost:1642/api/product`. The action method is set to `POST`. The request body is selected, and the format is `JSON (application/json)`. The body content is a JSON object: `{ "Id": 4, "Name": "Wave RSX", "Manufacturer": "Honda" }`. The status bar at the bottom shows `Status: 200 OK`, `Time: 131 ms`, and `Size: 352 B`.

Annotations in the image:

- Chọn action method là POST**: Points to the `POST` dropdown menu.
- URI**: Points to the URL `http://localhost:1642/api/product`.
- Request body**: Points to the `Body` tab and the JSON content.
- Kiểu dữ liệu POST**: Points to the `JSON (application/json)` format selector.
- Dữ liệu POST**: Points to the JSON object content.

# Sử dụng Postman để kiểm thử Service Web API

## PUT (tương tự POST)



# Sử dụng Postman để kiểm thử Service Web API

## DELETE

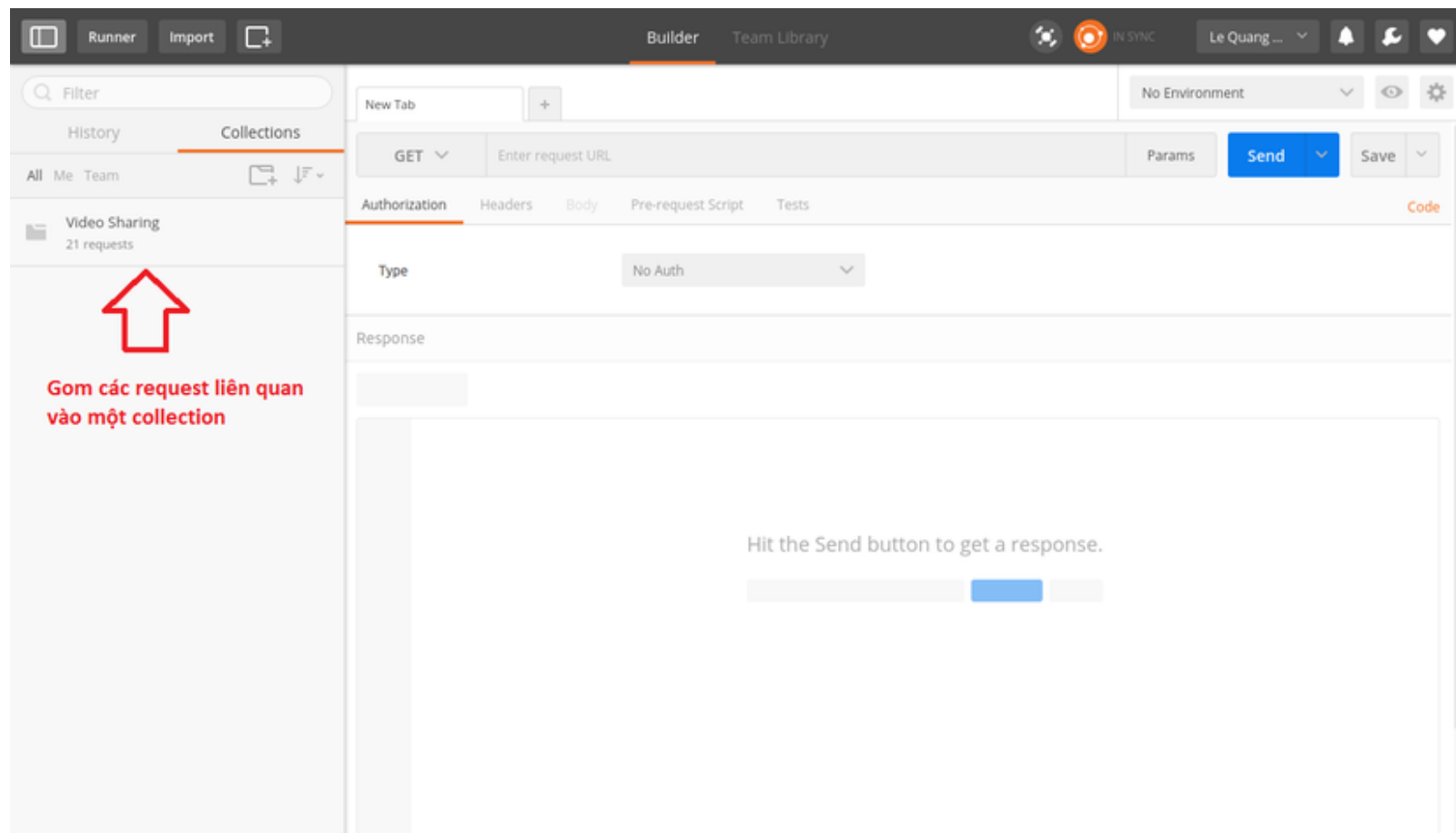
The screenshot displays the Postman application interface for configuring a DELETE request. At the top, the URL bar shows 'http://localhost:1642/' with a red status indicator and a plus icon. To the right, a dropdown menu is set to 'No Environment'. Below this, the request method is 'DELETE' and the URL is 'http://localhost:1642/api/product/1'. To the right of the URL are 'Params', 'Send' (a blue button), and 'Save' buttons. A tabbed interface below the URL bar includes 'Authorization', 'Headers' (which is selected and underlined), 'Body', 'Pre-request Script', and 'Tests'. On the far right of this tab bar are 'Cookies' and 'Code' links. The 'Headers' tab contains a table with columns 'Key', 'Value', and 'Description'. The table has one row with placeholder text: 'New key', 'Value', and 'Description'. To the right of the table are three icons: three dots, 'Bulk Edit', and 'Presets'. At the bottom of the interface, there are tabs for 'Body', 'Cookies', 'Headers (9)', and 'Test Results'. On the right side of the bottom bar, the response status is shown as 'Status: 200 OK', with 'Time: 80 ms' and 'Size: 356 B'.

Key	Value	Description
New key	Value	Description



# POSTMAN collection

- Postman có hỗ trợ collection để gom các Request có liên quan với nhau



THANK YOU