MISA®
TIN CẬY-TIỆN ÍCH-TẬN TÌNH

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>

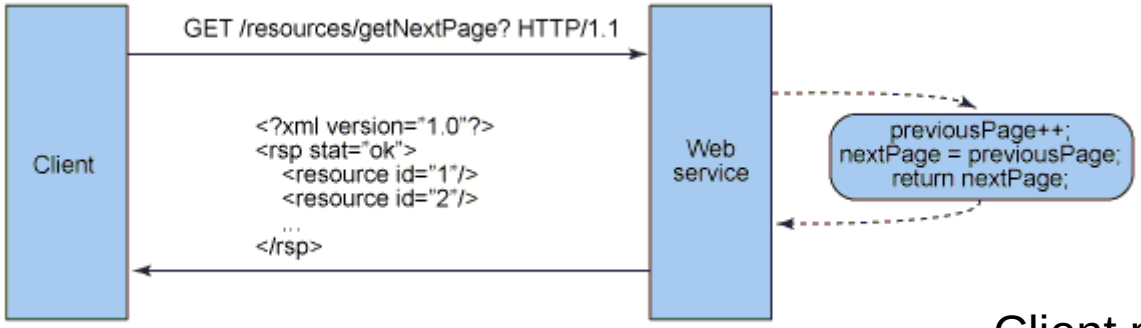
<name>Robert</name>

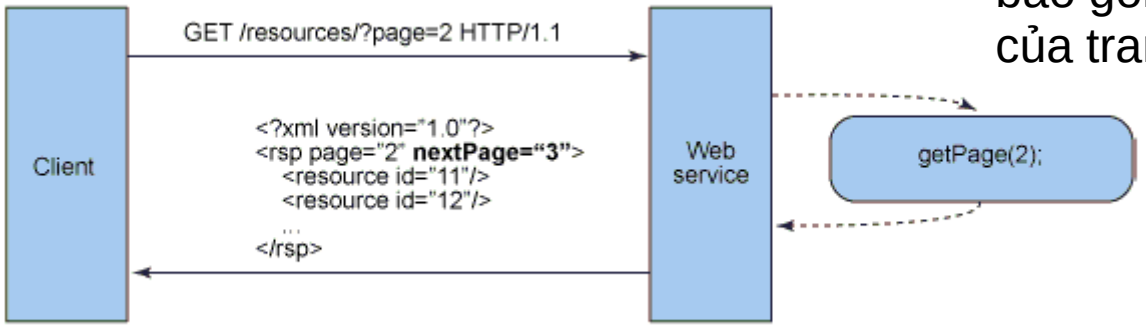
</user>

⟶ 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:

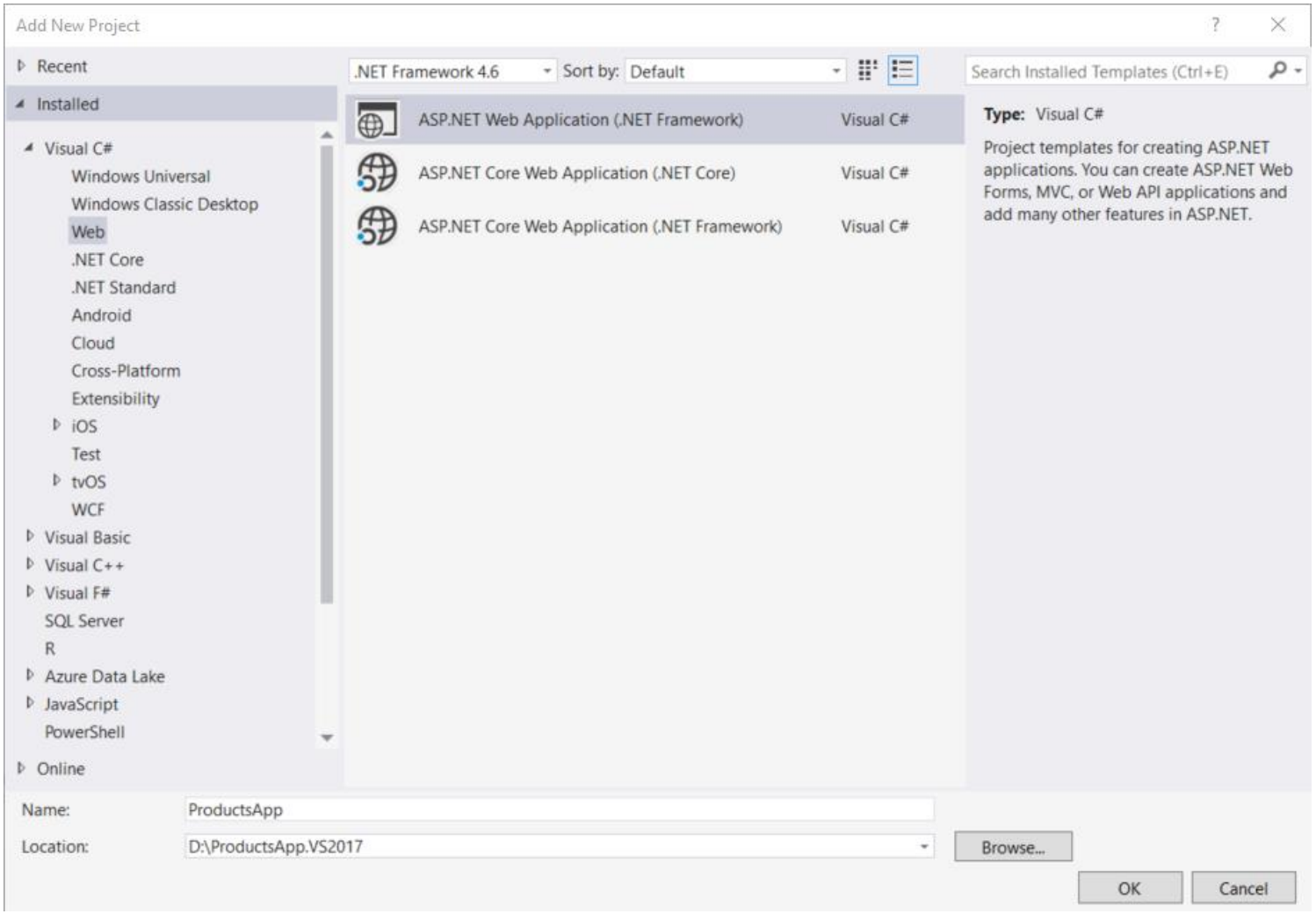
GET /weather/hanoi/2016-08-27

Host: myservice.com

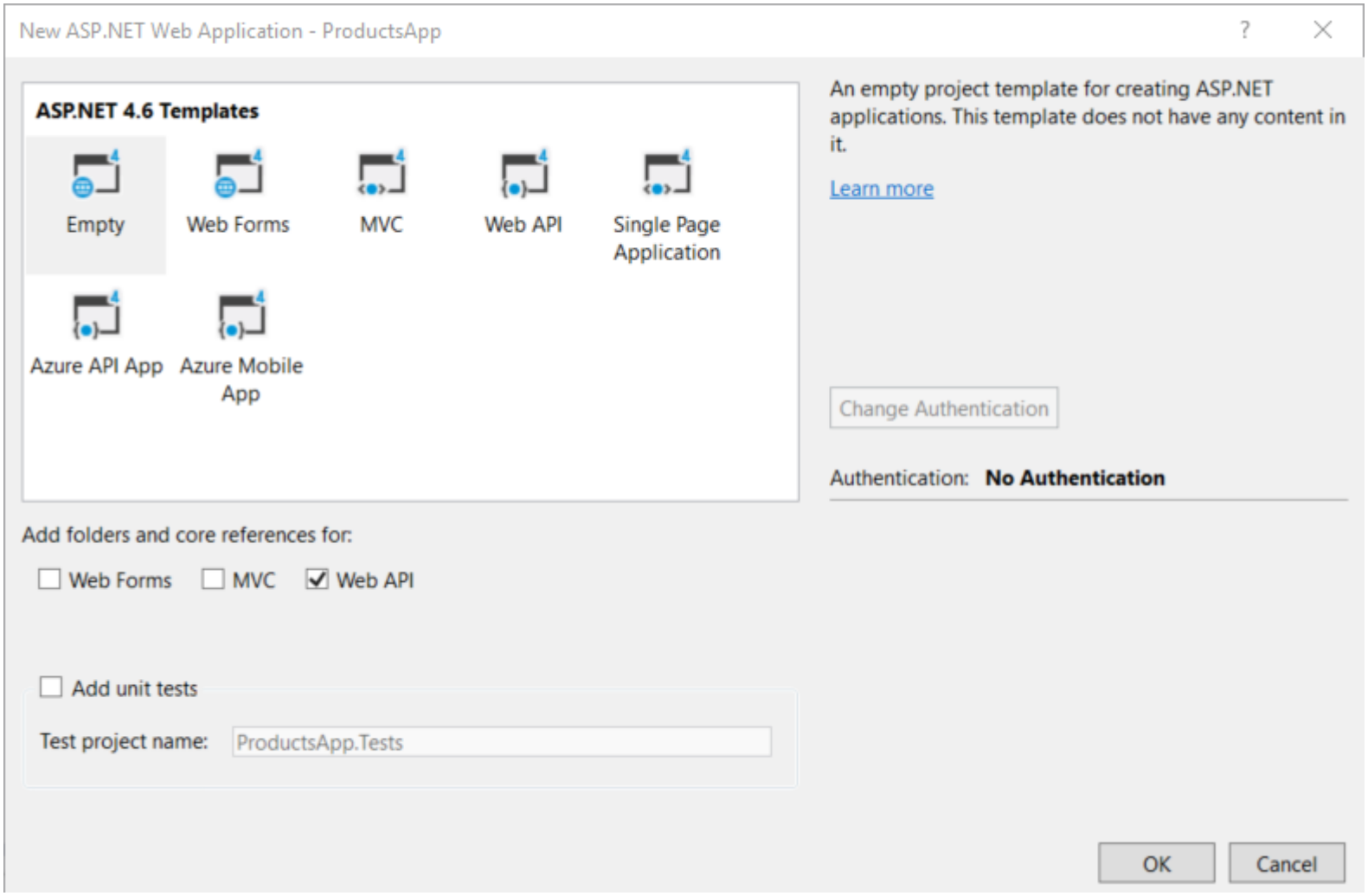
Accept: application/xml

Xây dựng RESTful Web API trong ASP.NET

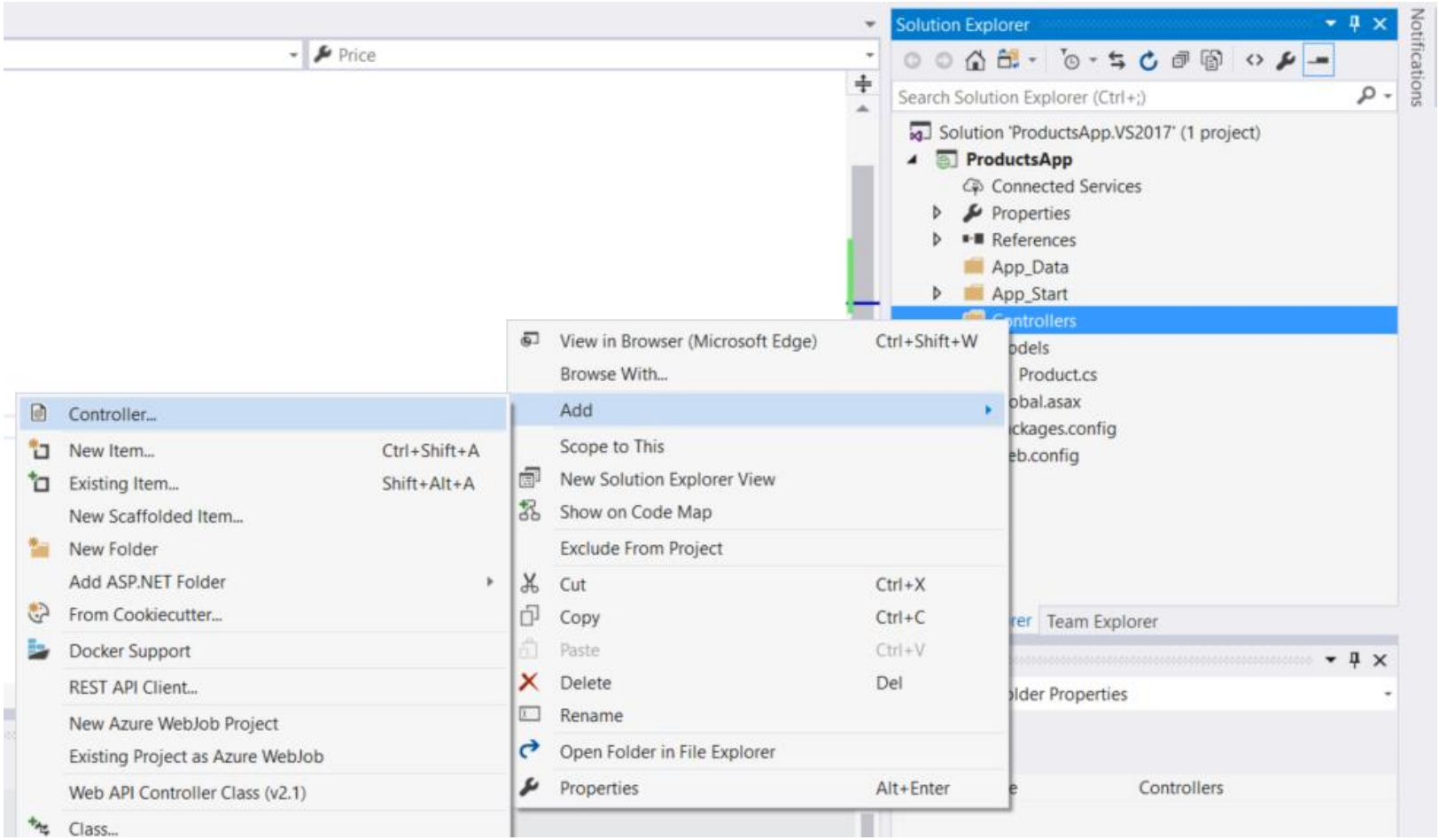
Tạo Web API project



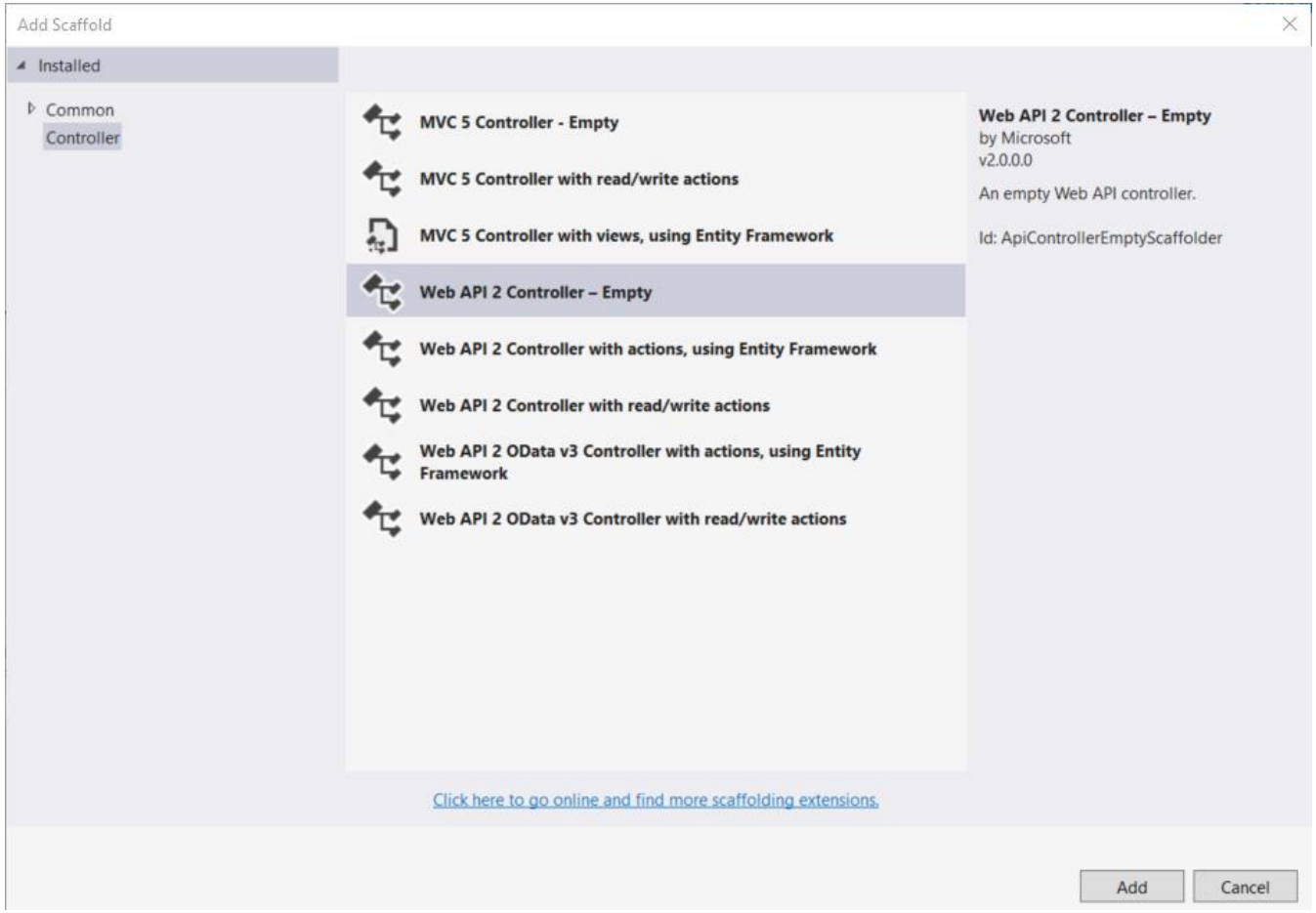
Tạo Web API project



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
{
    // GET api/values
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }

    // GET api/values/5
    public string Get(int id)
    {
        return "value";
    }

    // POST api/values
    public void Post([FromBody]string value)
    {
    }

    // PUT api/values/5
    public void Put(int id, [FromBody]string value)
    {
    }

    // DELETE api/values/5
    public void Delete(int id)
    {
    }
}
```

Web API controller Base class

Handles Http GET request
http://localhost:1234/api/values

Handles Http GET request with query string
http://localhost:1234/api/values?id=1

Handles Http POST request
http://localhost:1234/api/values

Handles Http Put request
http://localhost:1234/api/values?id=1

Handles Http DELETE request
http://localhost:1234/api/values?id=1

Action Method Naming Conventions

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

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
- IHttpActionResult

```
public IHttpActionResult 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:

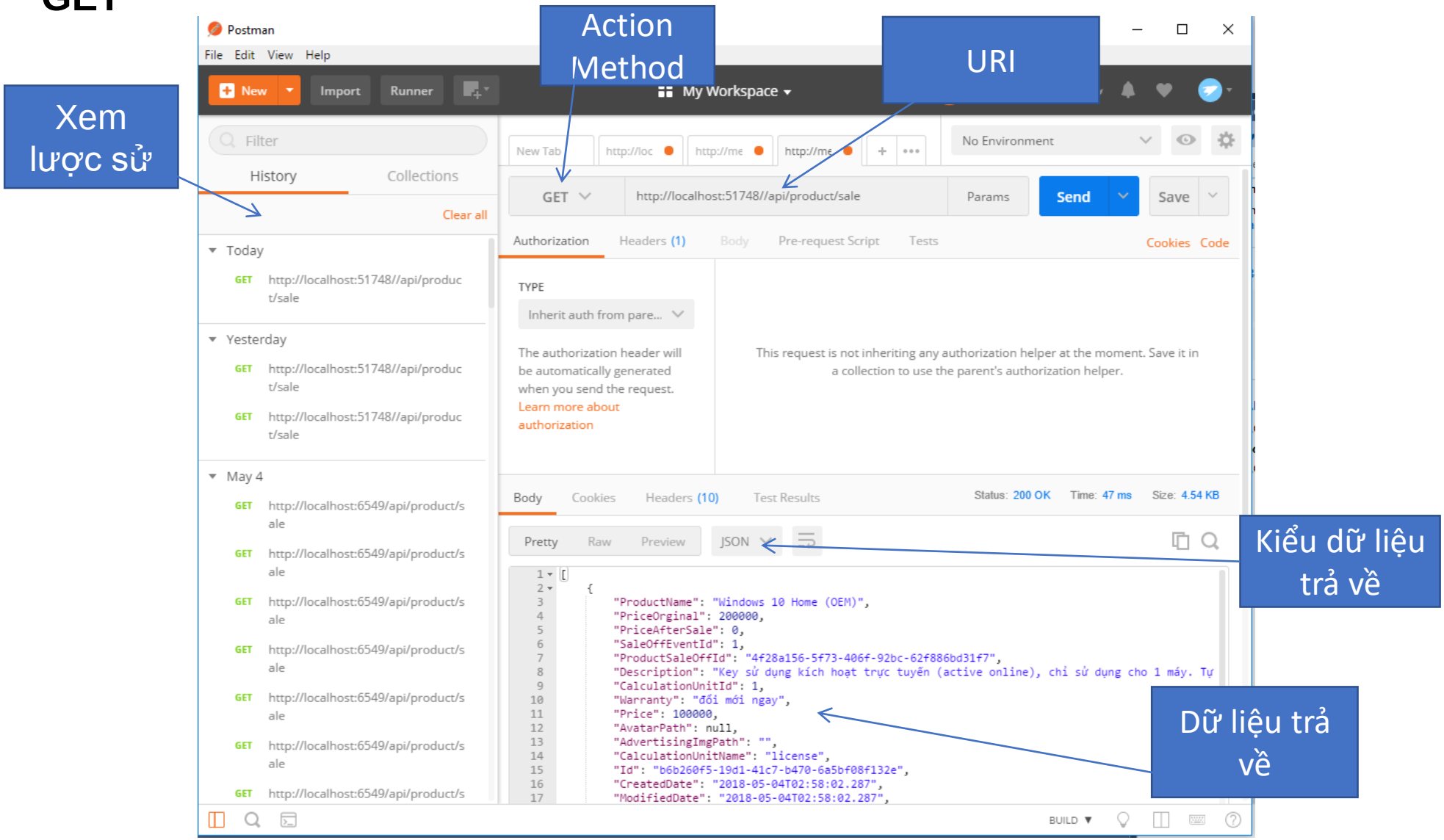
```
GET http://localhost:60464/api/student HTTP/1.1
User-Agent: Fiddler
Host: localhost:1234
Accept: application/json

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

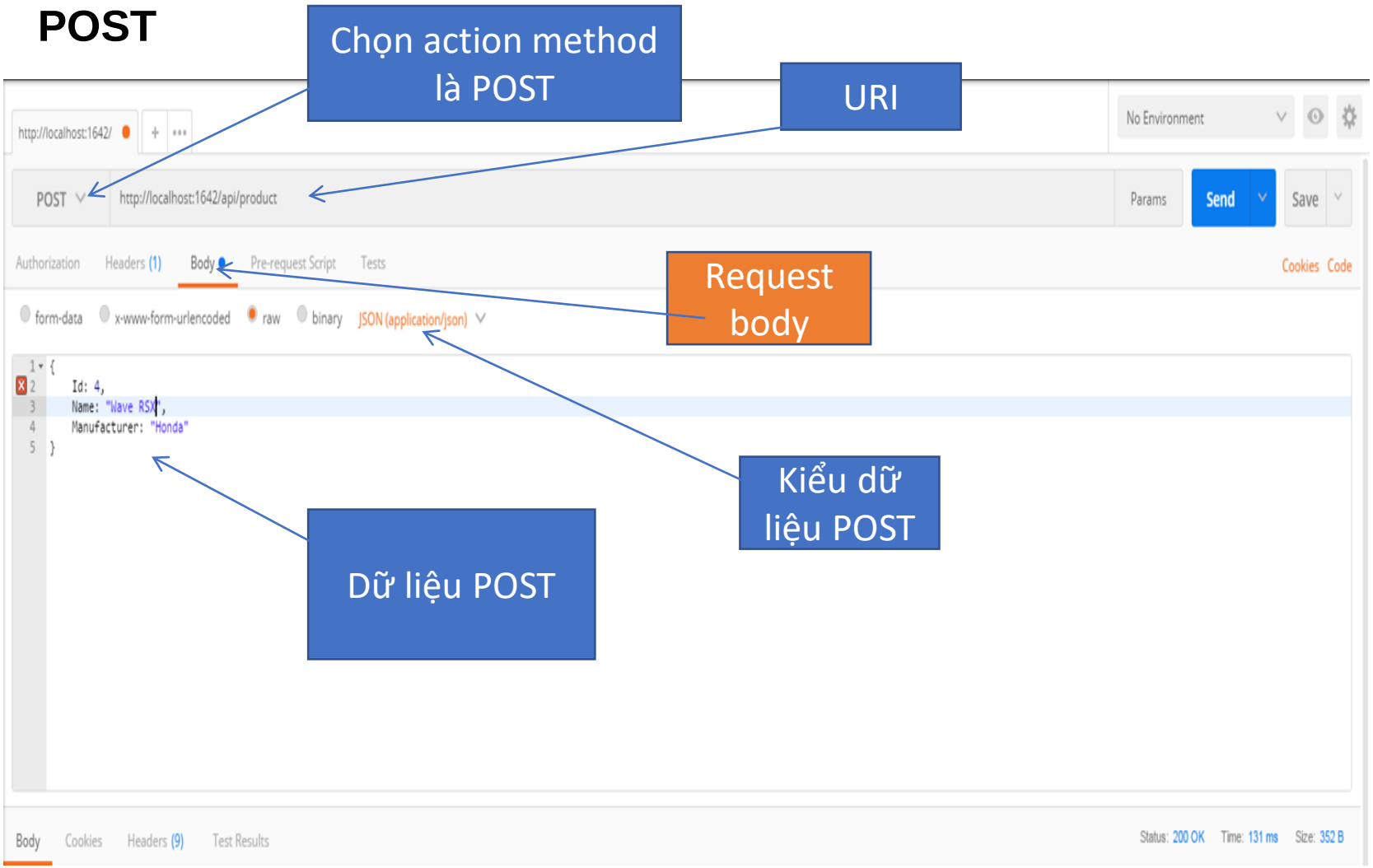
- Postman là 1 công cụ để kiểm thử Web API: https://www.getpostman.com/
- Các chức năng cơ bản:
  - Cho phép gửi HTTP Request với các method GET, POST, PUT, DELETE.
  - Cho phép post dữ liệu dưới dạng form (key-value), text, json
  - Hiện kết quả trả về dạng text, hình ảnh, XML, JSON
  - Hỗ trợ authorization (Oauth1, 2)
  - Cho phép thay đổi header của các request

Sử dụng Postman để kiểm thử Service Web API

GET



Sử dụng Postman để kiểm thử Service Web API

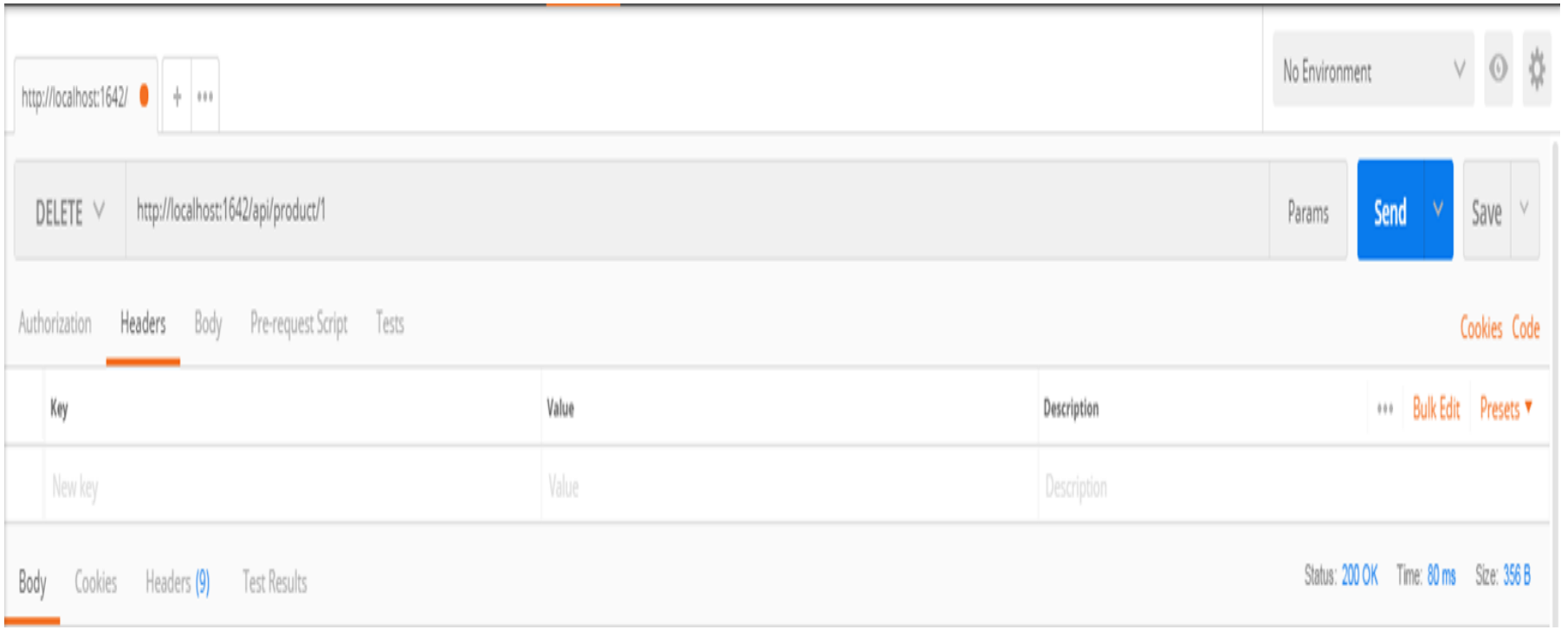


Sử dụng Postman để kiểm thử Service Web API

PUT (tương tự POST)

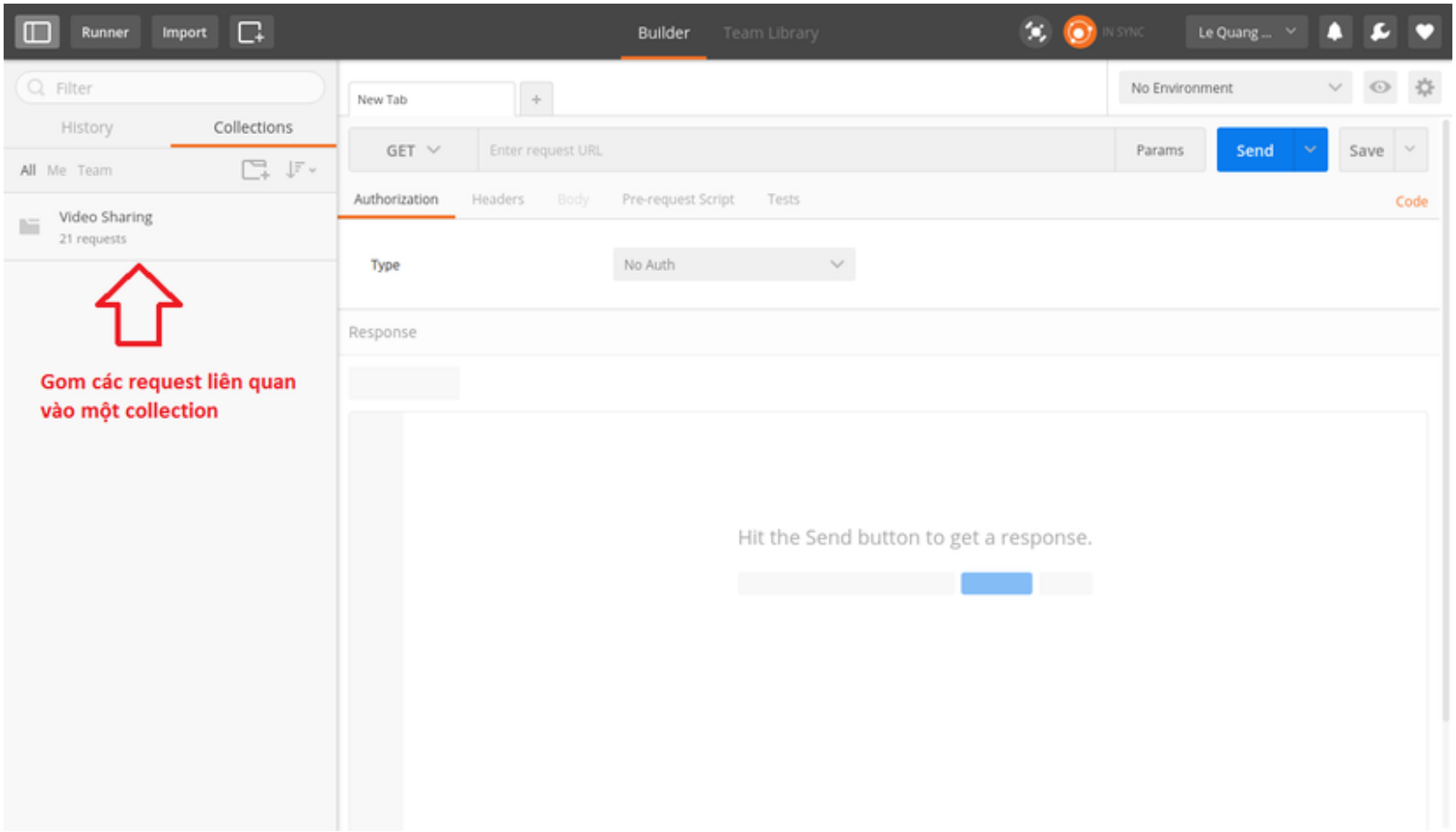
Sử dụng Postman để kiểm thử Service Web API

DELETE



POSTMAN collection

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




THANK YOU