

CHƯƠNG 3: WEB SERVICES VÀ REST API

3.1. Khái niệm về Web và Web services

3.1.1. Các khái niệm

Khái niệm Website

Khi bạn gõ vào một URL vào trình duyệt và bạn nhận được một trang web. Đây là một nội dung mà thông thường bạn có thể đọc được, nó là nội dung dành cho người dùng ở đầu cuối.



Website là một tập hợp các trang web (web pages) bao gồm văn bản, hình ảnh, video, flash v.v... thường chỉ nằm trong một tên miền (domain name) hoặc tên miền phụ (subdomain). Trang web được lưu trữ (web hosting) trên máy chủ web (web server) có thể truy cập thông qua Internet.

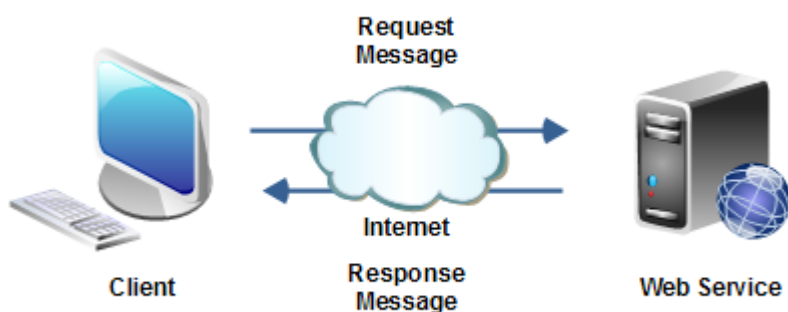
Website đóng vai trò là một văn phòng hay một cửa hàng trên mạng Internet – nơi giới thiệu thông tin về doanh nghiệp, sản phẩm hoặc dịch vụ do doanh nghiệp cung cấp... Có thể coi website chính là bộ mặt của doanh nghiệp, là nơi để đón tiếp và giao dịch với các khách hàng, đối tác trên Internet.

Khái niệm Web service

Theo định nghĩa của W3C (World Wide Web Consortium) thì Web service (Dịch vụ web) là hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng Internet.

Web service là một khái niệm rộng hơn so với khái niệm web thông thường. Nó là sự kết hợp các máy tính cá nhân với các thiết bị khác, các cơ sở dữ liệu và các mạng máy tính để tạo thành một cơ cấu tính toán ảo mà người sử dụng có thể làm việc thông qua các trình duyệt mạng. Các Web Service thường cung cấp các dữ liệu thô mà nó khó hiểu đối với đa số người dùng thông thường, chúng thường được trả về dưới dạng XML hoặc JSON.

Hiện nay, web service trở thành dịch vụ mạnh mẽ, cung cấp lợi ích cho cả doanh nghiệp, khách hàng, cá nhân, trong nhiều lĩnh vực thực tế: thông tin thương mại, dịch vụ du lịch, tỉ giá, chứng khoán...



Đặc điểm của web service

Tính độc lập (độc lập về ngôn ngữ, nền tảng): Dịch vụ Web cho phép Client và Server tương tác được với nhau ngay cả trong những môi trường khác nhau. Ví dụ , đặt Web server cho ứng dụng chạy trên một máy chủ chạy hệ điều hành Linux trong khi người dùng sử dụng máy tính chạy hệ điều hành Windows, ứng dụng vẫn có thể chạy và xử lý bình thường mà không cần thêm yêu cầu đặc biệt để tương thích giữa hai hệ điều hành này. Webservice cho phép Client và Server liên kết được với nhau ngay cả trong những môi trường khác nhau. Đó là vì webservice sử dụng XML (JSON), các chuẩn mở đã được công nhận và có thể hiểu bất kì ngôn ngữ lập trình nào.

Khả năng tự mô tả: giao diện của webservice được xây dựng thông qua tài liệu WSDL (Web Service Description Language), tài liệu này định nghĩa cấu trúc thông điệp trao đổi và cấu trúc dữ liệu thông điệp đó.

Tách biệt giữa mô tả và nội dung: đây là một trong những đặc tính then chốt làm nên sự thành công cho webservice. Webservice không đòi hỏi ở phía Client phải cài đặt bất cứ một thành phần mới nào. Còn ở phía Server, để triển khai webservice thì chỉ cần có Servlet

engine, hoặc Apache hoặc .Net Runtime... khi đã được triển khai thì Client có thể sử dụng ngay các dịch vụ. Điều này khác với các công nghệ khác như Dcom hay RMI khi mà Client phải cài đặt Client Stub để có thể truy cập dịch vụ.

Webservice được dựa trên các chuẩn mở: SOAP, WSDL, XML, JSON, UDDI (Universal Description, Discovery, and Intergration).

Sự truy cập được thông qua môi trường web: webservice được phát hành, xác định và gọi đều thông qua môi trường web. Trong đó mô tả về dịch vụ được xuất bản sử dụng WSDL, người dùng tìm kiếm và xác định dịch vụ nhờ sự trợ giúp của UDDI và yêu cầu dịch vụ bằng SOAP. Tất cả các giao thức này đều dựa trên Web.

Ưu điểm của web service

- ✎ Web service cung cấp nền tảng rộng lớn chạy được trên các hệ điều hành khác nhau
- ✎ Nâng cao khả năng tái sử dụng
- ✎ Tạo mối quan hệ tương tác lẫn nhau, dễ dàng cho việc phát triển các ứng dụng phân tán.
- ✎ Cài đặt dễ ràng, chi phí thấp, hiệu quả cao.
- ✎ Sử dụng các giao thức và chuẩn mở, giao thức và định dạng dữ liệu dựa trên văn bản giúp các lập trình viên dễ dàng hiểu được.

Nhược điểm của web service

- ✎ Có nhiều chuẩn khiến người dùng khó nắm bắt.
- ✎ Có thể xảy ra thiệt hại lớn vào khoảng thời gian không hoạt động của web service như: có thể lỗi nếu như máy tính không nâng cấp, thiếu các giao tiếp trong việc vận hành.
- ✎ Phải quan tâm nhiều hơn đến vấn đề bảo mật

Hoạt động của web service

Một dịch vụ web cho phép giao tiếp giữa các ứng dụng khác nhau bằng cách sử dụng các tiêu chuẩn mở như HTML, XML, WSDL và SOAP.

Một dịch vụ web có sự giúp đỡ của:

- XML để gắn thẻ dữ liệu.
- SOAP để chuyển một tin nhắn.
- WSDL để mô tả tính khả dụng của dịch vụ.

Bạn có thể xây dựng một dịch vụ web dựa trên Java chạy trên Solaris có thể truy cập từ chương trình Visual Basic của bạn chạy trên Windows.

Bạn cũng có thể sử dụng C# để xây dựng các dịch vụ web mới trên Windows có thể được gọi từ ứng dụng web của bạn dựa trên các trang Java Server (JSP) và chạy trên Linux.

Ví dụ

Hãy xem xét một hệ thống xử lý đơn đặt hàng và quản lý tài khoản đơn giản. Nhân viên kế toán sử dụng ứng dụng khách được xây dựng với Visual Basic hoặc JSP để tạo tài khoản mới và nhập các đơn đặt hàng của khách hàng mới.

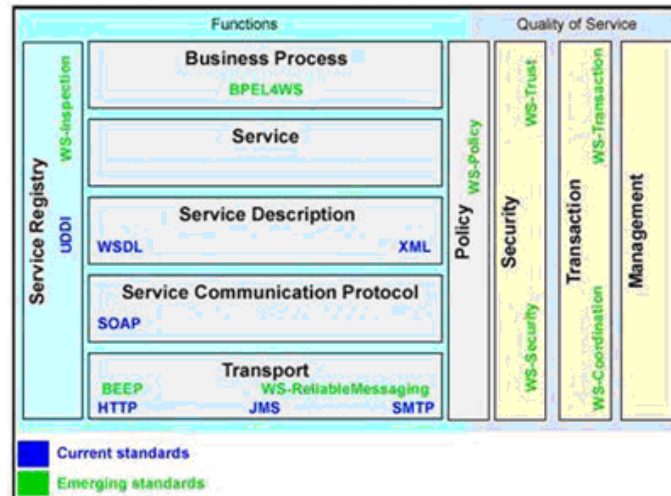
Logic xử lý cho hệ thống này được viết bằng Java và nằm trên máy Solaris, cũng tương tác với cơ sở dữ liệu để lưu trữ thông tin.

Các bước để thực hiện thao tác này như sau:

- Chương trình khách hàng kết hợp thông tin đăng ký tài khoản vào một thông điệp SOAP.
- Thông báo SOAP này được gửi đến dịch vụ web làm phần thân của yêu cầu HTTP POST.
- Dịch vụ web giải nén yêu cầu SOAP và chuyển đổi nó thành lệnh mà ứng dụng có thể hiểu được.
- Ứng dụng xử lý thông tin theo yêu cầu và phản hồi với số tài khoản duy nhất mới cho khách hàng đó.
- Tiếp theo, dịch vụ web gói trả lời vào một thông báo SOAP khác, nó gửi trở lại chương trình máy khách để đáp ứng yêu cầu HTTP của nó.
- Chương trình khách hàng mở gói thông điệp SOAP để có được kết quả của quá trình đăng ký tài khoản.

3.1.2. Kiến trúc của Web service

Kiến trúc của web service bao gồm các tầng như sau :



Kiến trúc của Web Service

Trong đó:

Tầng vận chuyển (Transport) có nhiệm vụ truyền thông điệp giữa các ứng dụng mạng, bao gồm các giao thức như HTTP, SMTP, JMS và gần đây nhất là giao thức thay đổi khối mở rộng (Blocks Extensible Exchange Protocol - BEEP).

Tầng giao thức tương tác dịch vụ (Service Communication Protocol) với công nghệ chuẩn là SOAP. SOAP cho phép người dùng triệu gọi một service từ xa thông qua một message XML. Có nhiệm vụ giải mã các thông điệp theo định dạng XML để có thể hiểu được ở mức ứng dụng tương tác với người dùng.

Tầng mô tả dịch vụ (Service Description) sử dụng để miêu tả các giao diện chung cho một dịch vụ Web cụ thể với công nghệ chuẩn là WSDL và XML. WSDL thường được sử dụng cho mục đích này, nó là một ngôn ngữ mô tả giao tiếp và thực thi dựa trên XML. Dịch vụ Web sẽ sử dụng ngôn ngữ này để truyền tham số và các loại dữ liệu cho các thao tác và chức năng mà dịch vụ Web cung cấp.

Tầng dịch vụ (Service) cung cấp các chức năng của service.

Tầng (Business Process) tiến hành các bước xử lý thông tin đầu vào để cho kết quả đầu ra theo yêu cầu, là các hoạt động có tính chất nghiệp vụ.

Tầng đăng ký dịch vụ (Service Registry) với công nghệ chuẩn là UDDI. UDDI dùng cho cả người dùng và SOAP server, nó cho phép đăng ký dịch vụ để người dùng có thể

gọi thực hiện service từ xa qua mạng , hay nói cách khác một service cần phải được đăng ký để cho phép các client có thể gọi thực hiện.

Bên cạnh đó, để cho các service có tính an toàn, toàn vẹn và bảo mật thông tin trong kiến trúc web service. Chúng ta có thêm các tầng Policy, Security, Transaction, Management giúp tăng cường tính bảo mật, an toàn và toàn vẹn thông tin khi sử dụng service.

3.1.3. Các thành phần của web service

Web service có 3 thành phần chính:

- SOAP (Simple Object Access Protocol) – giao thức truy cập đối tượng đơn giản.
- WSDL (Web Services Description Language) – ngôn ngữ định nghĩa web service.
- UDDI (Universal Description, Discovery and Integration).

SOAP

- ✎ SOAP là viết tắt của (Simple Object Access Protocol) – giao thức truy cập đối tượng đơn giản.
- ✎ SOAP là một giao thức dựa trên XML để truy cập các web service.
- ✎ SOAP là một khuyến nghị của W3C về giao tiếp giữa các ứng dụng.
- ✎ SOAP dựa trên XML, vì vậy nó độc lập với nền tảng và ngôn ngữ độc lập. Nói cách khác, nó có thể được sử dụng với ngôn ngữ Java, .Net hoặc PHP trên bất kỳ nền tảng nào.

WSDL

- ✎ WSDL là từ viết tắt của (Web Services Description Language) – ngôn ngữ định nghĩa web service.
- ✎ WSDL là một tài liệu xml chứa thông tin về các web service như tên phương thức, tham số phương thức và cách truy cập nó.
- ✎ WSDL là một phần của UDDI. Nó hoạt động như một giao diện giữa các ứng dụng web service.

UDDI

- ✎ UDDI là từ viết tắt của Universal Description, Discovery and Integration.

- ✎ UDDI là một framework dựa trên XML cho việc mô tả, khám phá và tích hợp các web service.
- ✎ UDDI là một thư mục các giao diện web service được mô tả bởi WSDL, chứa thông tin về các web service.

3.1.4. Các loại Web service

Có hai loại web service chủ yếu:

- SOAP web service.
- RESTful web service.

SOAP Web Service

SOAP là viết tắt của Simple Object Access Protocol. Nó là một giao thức dựa trên XML để truy cập các web service.

SOAP được khuyến cáo bởi W3C cho giao tiếp giữa hai ứng dụng.

SOAP là giao thức dựa trên XML. Đó là nền tảng độc lập và ngôn ngữ độc lập. Bằng cách sử dụng SOAP, bạn sẽ có thể tương tác với các ứng dụng ngôn ngữ lập trình khác.

Ưu điểm của SOAP web service

- ✎ WS Security: SOAP định nghĩa bảo mật riêng của nó được gọi là WS Security.
- ✎ Ngôn ngữ và nền tảng độc lập: các SOAP web service có thể được viết bằng bất kỳ ngôn ngữ lập trình nào và được thực thi trong bất kỳ nền tảng nào.

Nhược điểm của SOAP web service

- ✎ Chậm: SOAP sử dụng định dạng XML phải được phân tích cú pháp. Các ứng dụng SOAP phải tuân theo nhiều tiêu chuẩn. Vì vậy, nó là chậm và chiếm nhiều băng thông và tài nguyên.
- ✎ Phụ thuộc WSDL: SOAP sử dụng WSDL và không có bất kỳ cơ chế nào khác.

RESTful Web Service

REST là viết tắt của REpresentational State Transfer.

REST là một kiểu kiến trúc không phải là một giao thức.

Ưu điểm của RESTful web service

- ✎ Nhanh: RESTful web service nhanh vì không có đặc tả nghiêm ngặt như SOAP. Nó chiếm ít băng thông và tài nguyên hơn.
- ✎ Ngôn ngữ và nền tảng độc lập: RESTful web service có thể được viết bằng bất kỳ ngôn ngữ lập trình nào và được thực hiện trong bất kỳ nền tảng nào.
- ✎ Có thể sử dụng SOAP: RESTful web service có thể sử dụng các SOAP web service khi thực hiện.
- ✎ Cho phép nhiều định dạng dữ liệu khác nhau: RESTful web service cho phép định dạng dữ liệu khác nhau như Plain Text, HTML, XML và JSON.

3.1.5. Sự khác nhau giữa API và web service

API và Web service hoạt động như các phương tiện giao tiếp. Vậy giữa API và Web service có điểm gì khác nhau. Mời bạn đọc cùng tham khảo bài viết dưới đây của Tài miễn phí để cùng tìm hiểu về định nghĩa và sự khác nhau giữa API và Web Service nhé.

API và Web service hoạt động như các phương tiện giao tiếp. Điểm khác biệt duy nhất giữa API và Web service chính là Web service tạo thuận lợi cho sự tương tác giữa hai máy qua mạng. API hoạt động như một giao diện giữa hai ứng dụng khác nhau để chúng có thể giao tiếp với nhau.

API là phương pháp mà các nhà cung cấp bên thứ 3 sử dụng để có thể viết các chương trình có giao diện dễ dàng giao tiếp với các chương trình khác. Web service được thiết kế để có giao diện được mô tả ở một định dạng mà máy tính có thể xử lý được thường được chỉ định trong Web Service Description Language (WSDL).

Thông thường, "HTTP" là giao thức được sử dụng phổ biến nhất trong quá trình giao tiếp. Web service sử dụng 3 chuẩn chính là **SOAP**, **REST** và **XML-RPC** để làm phương tiện giao tiếp. API có thể sử dụng bất kỳ phương tiện giao tiếp nào để bắt đầu tương tác giữa các ứng dụng. Ví dụ, các cuộc gọi hệ thống được gọi bằng cách làm gián đoạn bởi các kernel Linux API.

API xác định chính xác các phương thức cho một chương trình phần mềm để tương tác với các chương trình khác. Khi hành động này liên quan đến việc gửi dữ liệu qua mạng, Web service sẽ xuất hiện trong đó. Một API nói chung bao gồm việc “gọi” các chức năng bên trong một chương trình phần mềm.

Trong trường hợp các ứng dụng Web, API được sử dụng dựa trên web. Các ứng dụng máy tính như bảng tính và tài liệu Word sử dụng API dựa trên VBA và COM mà không liên quan đến Web service. Một ứng dụng máy chủ như Joomla có thể sử dụng một API dựa trên PHP hiện diện trong máy chủ mà không yêu cầu Web service.

Web service chỉ đơn thuần là một API được gói trong HTTP. API không phải lúc nào cũng cần phải dựa trên web. Một API bao gồm một rule và các thông số kỹ thuật đầy đủ của một chương trình phần mềm để tạo thuận lợi cho việc tương tác. Một Web service có thể không chứa đầy đủ các thông số kỹ thuật và đôi khi không thể thực hiện tất cả các tác vụ mà một API hoàn chỉnh có thể làm được.

Các API có thể được tiếp xúc theo nhiều cách khác nhau, trong đó bao gồm: COM, file DLL và file .H trong ngôn ngữ lập trình C/C ++, file JAR hoặc RMI trong Java, XML qua HTTP, JSON qua HTTP, Phương thức mà Web service sử dụng để lộ API là hoàn toàn thông qua mạng.

Tóm lại sự khác nhau giữa API và Web service:

1. Tất cả Web services là APIs nhưng tất cả các APIs không phải là Web services.
2. Web services không thể thực hiện được tất cả các thao tác mà API sẽ thực hiện.
3. Một Web service sử dụng 3 chuẩn chính: SOAP, REST và XML-RPC trong quá trình giao tiếp, ngược lại API có thể sử dụng bất kỳ chuẩn nào để giao tiếp.
4. Một Web service đòi hỏi luôn luôn phải có mạng để nó hoạt động nhưng API thì không cần.
5. API tạo điều kiện liên kết trực tiếp với một ứng dụng trong khi Web service thì không.

3.2. Giới thiệu về Rest và Restful API

3.2.1. Vấn đề Resource (tài nguyên) của ứng dụng

Quản lý resource (tài nguyên) là một phần quan trọng và chiếm phần lớn trong việc phát triển website. Trong đó resource của các website khác nhau có thể sẽ khác nhau. Với các trang mạng xã hội như Facebook thì resource thường là danh sách người dùng (user hoặc account), danh sách các bài viết (post hoặc article), các ảnh được đăng (photo hoặc image), các trang fanpage (fanpage)... Đối với một trang chia sẻ ảnh như Instagram thì resource có thể là các ảnh được đăng (photo), danh sách người dùng (user)... Đối với các

trang bán hàng thì resource có thể là danh sách sản phẩm (product), danh sách các người bán (seller), danh sách khách hàng (user hay customer)...

Việc quản lý resource của một website bao gồm 4 tác vụ chính:

- ✎ Tạo mới một resource (create)
- ✎ Lấy thông tin một resource (read)
- ✎ Cập nhật một resource (update)
- ✎ Xoá một resource (delete)

Tương ứng với 4 tác vụ quản lý tài nguyên trên thì HTTP (Hypertext Transfer Protocol) quy định một số các phương thức để tương tác với tài nguyên như: GET (để lấy thông tin của tài nguyên); POST (để tạo mới một tài nguyên); PUT (để cập nhật thông tin của tài nguyên); DELETE (để xóa một tài nguyên); ...

Có rất nhiều cách khác nhau để xây dựng một trang web thực hiện 4 tác vụ trên. Ví dụ với một trang blog chạy dưới tên miền là `http://my-blog.xyz` thì để xem nội dung một bài viết (*post*) với ID là 123 bạn có thể làm theo một trong các cách sau:

- ✎ Gửi một request tới URL `http://my-blog.xyz/posts?id=123` với HTTP method là *GET*
- ✎ Gửi một request tới URL `http://my-blog.xyz/posts/123` với HTTP method là *GET*
- ✎ Gửi một request tới URL `http://my-blog.xyz/action=view_post&id=123` với HTTP method là *GET*
- ✎ Gửi một request tới URL `http://my-blog.xyz/view_post&id=123` với HTTP method là *GET*
- ✎ Gửi một request tới URL `http://my-blog.xyz/posts?id=123` với HTTP method là *POST*
- ✎ Gửi một request tới URL `http://my-blog.xyz/posts/123` với HTTP method là *POST*
- ✎ ...

Và sau một thời gian dài thì người ta đã thống nhất ra các tiêu chuẩn khác nhau để thực hiện việc quản lý resource. Các tiêu chuẩn này (hay còn được gọi là Web API hoặc

HTTP API) quy định một cách thống nhất việc quản lý các resource của web. Và RESTful là một trong các web API được sử dụng phổ biến ngày nay.

3.2.2. Rest và Restful API

API là viết tắt của **Application Programming Interface (Giao diện lập trình ứng dụng)** không phải là một ngôn ngữ lập trình, về cơ bản nó cho phép một chức năng của phần mềm này có thể giao tiếp với một hoặc nhiều phần mềm khác.

Hiện nay có rất nhiều loại API khác nhau: Windows API, Google API, Youtube API, ... Nhưng khi mà người ta nói đến Google API, Youtube API, Twitter API là người ta đang nói về Rest API.



Rest (Representational State Transfer) nghĩa là Chuyển trạng thái đại diện. được đưa ra vào năm 2000 trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP)

Rest là kiến trúc phần mềm phổ biến nhất hiện nay, được sử dụng trong việc giao tiếp giữa các chương trình máy tính (máy tính cá nhân và máy chủ của trang web) trong việc quản lý các tài nguyên trên internet.

REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web services chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được chuyển tải qua HTTP thông qua số lượng lớn người dùng và được viết bởi những ngôn ngữ khác nhau

REST được sử dụng rất nhiều trong việc phát triển các ứng dụng Web Services sử dụng giao thức HTTP trong giao tiếp thông qua mạng internet.

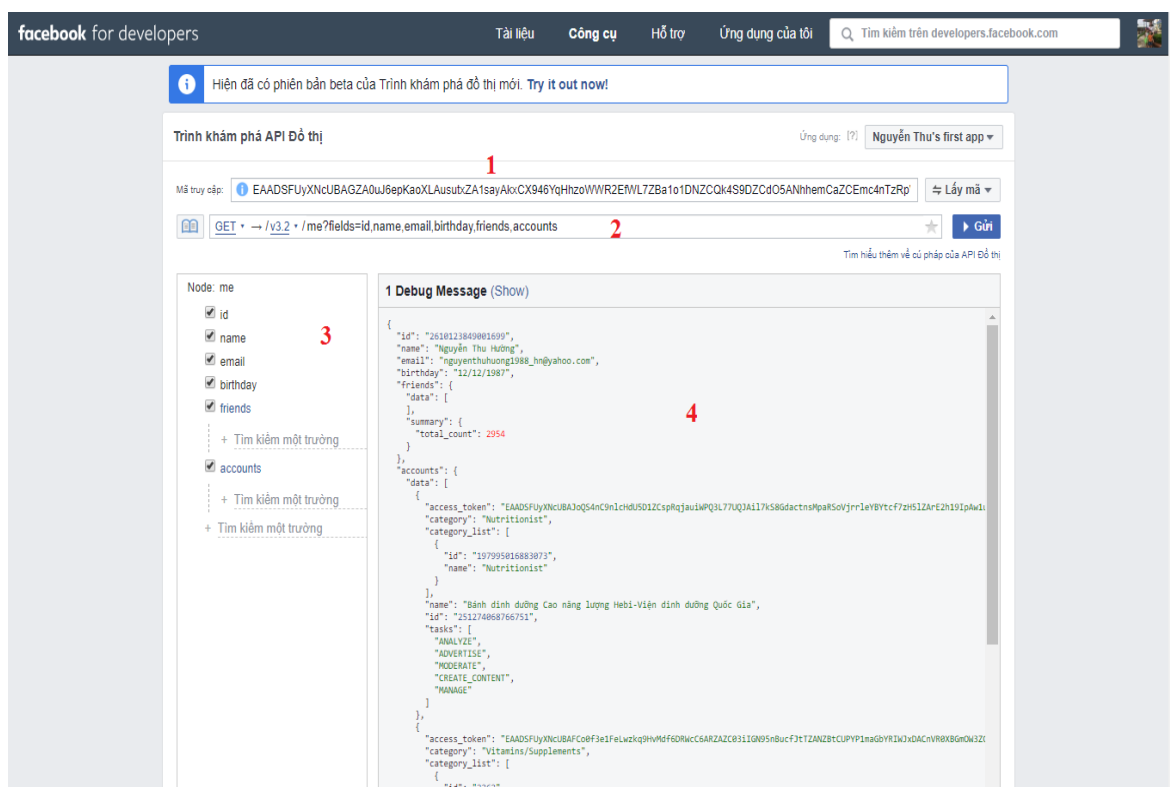
Các ứng dụng sử dụng kiến trúc REST này thì sẽ được gọi là ứng dụng phát triển theo kiểu **RESTful**.

Rest API là chỉ việc thiết kế Web API theo chuẩn Rest. Nó bao gồm các nguyên tắc, ràng buộc trong thiết kế Web API.

Không phải lúc nào cũng vậy nhưng một Rest API hoạt động khá giống một trang web. Ta thực hiện gọi hàm từ máy khách để lấy dữ liệu ở máy chủ thông qua giao thức HTTP.

Ví dụ:

Để khám phá Graph API Facebook ta vào link: <https://developers.facebook.com/>.
Chọn mục Graph API Explorer



Trong đó:

1. **Mã truy cập**: Là mã cho phép gửi đòi hỏi tới **Server**. Nếu bạn đang login vào một tài khoản facebook nào đó, giá trị này sẽ được mặc định hiển thị cho tài khoản đó.
2. **URL** gửi tới **server** để nhận về giá thông tin.

3. Giúp bạn thêm các trường (field) cần thiết vào **request**.
4. Kết quả nhận về sau khi nhấn **Gửi**.

3.3. Các nguyên tắc thiết kế Rest

Việc thiết kế một Web service Rest sẽ tuân thủ theo 4 nguyên tắc cơ bản sau:

- ✎ Sử dụng các phương thức HTTP một cách rõ ràng
- ✎ Phi trạng thái
- ✎ Hiện thị cấu trúc thư mục như URLs
- ✎ Chuyển đổi linh hoạt JavaScript Object Notation (JSON) và XML hoặc cả hai.

3.3.1. Sử dụng các phương thức HTTP một cách rõ ràng

Một đặc tính quan trọng của dịch Web được thiết kế theo chuẩn Rest là sử dụng một cách rõ ràng các phương thức HTTP theo cách một giao thức được xác định. Ví dụ HTTP GET được xác định như là một phương thức lấy thông tin về tài nguyên, dữ liệu từ một máy chủ, hoặc thực thi một truy vấn mà máy chủ sẽ tìm kiếm và phản hồi cùng với một gói thông tin tương thích.

REST yêu cầu các nhà phát triển sử dụng phương thức HTTP một cách rõ ràng theo cách tương thích với giao thức chuẩn. Nguyên lý thiết kế REST cơ bản này thiết lập một ánh xạ 1-1 giữa các hành động tạo, đọc, cập nhật và xóa (CRUD) các quá trình vận hành và các phương thức HTTP. Theo cách ánh xạ này thì:

- ✎ Để tạo một tài nguyên trên máy chủ, bạn cần sử dụng phương thức POST.
- ✎ Để truy xuất một tài nguyên, sử dụng GET.
- ✎ Để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó, sử dụng PUT.
- ✎ Để hủy bỏ hoặc xóa một tài nguyên, sử dụng DELETE.

HTTP	POST	GET	PUT	DELETE
SQL	INSERT	SELECT	UPDATE	DELETE
CRUD	CREATE	READ	UPDATE	DELETE

Bảng tương quan giữa phương thức HTTP, CRUD và các lệnh SQL

Ví dụ: Sử dụng sai phương thức

```
GET /adduser?name=Robert HTTP/1.1
```

Ở đây phương thức GET được sử dụng để tạo một bản ghi mới trong cơ sở dữ liệu. Như vậy vi phạm định nghĩa của GET chỉ để lấy và trả về dữ liệu. Theo REST để tạo một tài nguyên mới ta sử dụng POST. Resquest trên được sửa lại:

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>
  <name>Robert</name>
</user>
```

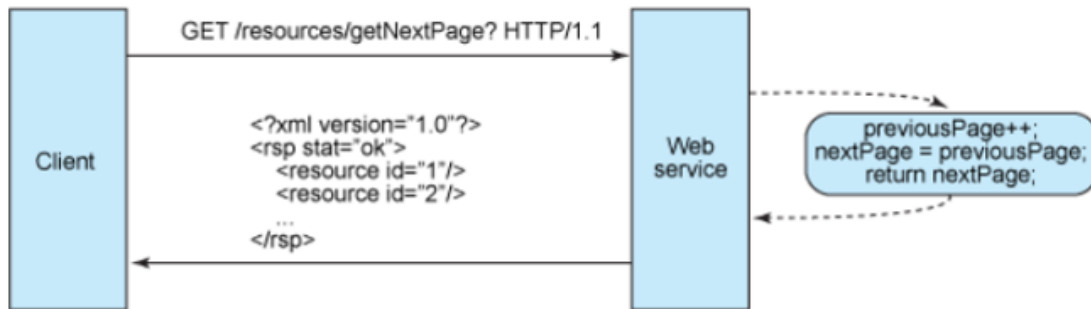
Việc sử dụng các phương thức HTTP một cách không rõ ràng ví dụ như sử dụng GET để tạo hoặc cập nhật tài nguyên, dữ liệu hệ thống sẽ gây ra vấn đề về mặt ngữ nghĩa. Web server được thiết kế để phản hồi lại các yêu cầu GET bằng việc lấy và trả về chúng bằng một dạng nào đó, chứ không phải để tạo hoặc cập nhật bản ghi trong cơ sở dữ liệu. Tương tự với các phương thức HTTP khác, theo REST chúng nên được sử dụng đúng với các chức năng nêu trên. Ngoài mặt ngữ nghĩa, còn có một vấn đề nữa có thể dẫn đến việc thay đổi dữ liệu phía server một cách không chủ tâm.

Như là một nguyên tắc thiết kế chung, các hướng dẫn sử dụng REST để sử dụng các phương thức HTTP một cách rõ ràng bằng cách sử dụng các danh từ trong URIs thay vì động từ. Trong một Web service RESTful, các động từ POST, GET, PUT, và DELETE đã được định nghĩa bởi giao thức. Và tốt nhất, để giữ giao diện được khái quát hoá và cho phép người dùng hiểu rõ các thao tác mà họ gọi thì Web service không nên đưa ra nhiều động từ hoặc các thủ tục remote từ xa, như /adduser hoặc /updateuser. Nguyên tắc thiết kế chung này cũng áp dụng đối với phần thân câu lệnh HTTP, được sử dụng có chủ ý để chuyển trạng thái tài nguyên, không mang tên của một phương thức hay thủ tục remote từ xa.

3.3.2. Phi trạng thái

Theo REST trạng thái hoặc được giữ trên client hoặc được chuyển thành trạng thái của tài nguyên, server sẽ không bao giờ giữ trạng thái thông tin trao đổi với bất kỳ client nào nó giao tiếp. Mỗi request lên server thì client phải đóng gói thông tin đầy đủ để server hiểu được. Điều này giúp hệ thống của bạn dễ phát triển, bảo trì, mở rộng vì không cần tốn công CRUD trạng thái của client. Ngoài ra còn một nguyên nhân quan trọng hơn đó là nó tách biệt client khỏi sự thay đổi của server.

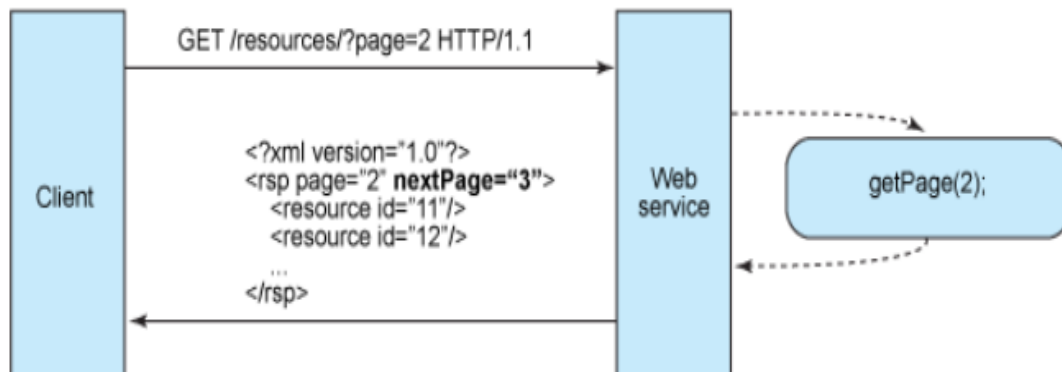
Hình sau mô tả một thiết kế trạng thái



Mô hình trạng thái

Với dịch vụ trạng thái thì một máy chủ của ứng dụng có thể yêu cầu trang sau lưu giữ trạng thái của trang trước (previousPage) để có thể phản hồi các lệnh tiếp theo. Dịch vụ trạng thái như thế này trở nên phức tạp.

Mặt khác, các thành phần máy chủ phi trạng thái ít phức tạp hơn trong thiết kế. Dịch vụ phi trạng thái không chỉ hoạt động tốt hơn, nó còn chuyển hầu hết vai trò duy trì trạng thái sang ứng dụng ở máy khách. Trong một dịch vụ mạng RESTful, máy chủ chịu trách nhiệm đưa ra các phản hồi và cung cấp một giao diện cho phép máy khách duy trì trạng thái ứng dụng của chính nó. Ví dụ, trong yêu cầu tập hợp trang kết quả, máy khách sẽ gồm số trang thực tế khi truy xuất thay vì đơn giản chỉ là yêu cầu *tiếp theo* như hình sau:



Mô hình phi trạng thái

Một dịch Web phi trạng thái sinh ra một phản hồi liên kết với số trang tiếp theo trong một tổng thể và để máy khách làm những gì mà nó cần để giữ giá trị này ở mức nhất định. Khía cạnh này của thiết kế dịch vụ Web RESTful có thể được tách thành hai phần trách nhiệm như là mức phân chia cao nhất mà chỉ rõ một dịch vụ phi trạng thái có thể được duy trì như thế nào.

3.3.3. Hiện thị cấu trúc thư mục như URIs

URI trong RESTful web service phải tự mô tả, hoặc tham chiếu được cái mà nó trở tới và các tài nguyên liên quan. Ngoài ra URI cũng phải đơn giản, có thể đoán biết được, và dễ hiểu. Để tạo ra URI với yêu cầu trên thì ta nên định nghĩa URI có cấu trúc giống thư mục. Loại URI này có phân cấp, có gốc là một đường dẫn đơn, các nhánh từ gốc là các đường dẫn phụ dẫn đến các vùng service chính.

Ví dụ về URI trong RESTful web service:

`http://www.myservice.org/discussion/topics/{topic}`

`http://www.myservice.org/discussion/2008/12/10/{topic}`

Với URI có cấu trúc như thư mục cho phép nhà phát triển dễ dàng trong việc cài đặt service của mình hướng vào một loại tài nguyên cụ thể nào đó.

Một vài hướng dẫn bổ sung để lưu ý trong khi nói về cấu trúc địa chỉ của Web service RESTful là:

- ✗ Giấu các đuôi tài liệu mở rộng của bản gốc trong máy chủ (.jsp, .php, .asp) nếu có. vì vậy bạn có thể giấu một số thứ mà không cần thay đổi địa chỉ Urls.
- ✗ Để mọi thứ là chữ thường.
- ✗ Thay thế các khoảng trống bằng gạch chân hoặc gạch nối (một trong hai loại).
- ✗ Tránh các chuỗi yêu cầu càng nhiều càng tốt.
- ✗ Thay vì sử dụng mã (404 Not Found) khi yêu cầu địa chỉ cho một phần đường dẫn, luôn luôn cung cấp một trang mặc định hoặc tài nguyên như một phản hồi.

Các địa chỉ URIs nên giữ nguyên để khi tài nguyên thay đổi hoặc khi tiến hành thay đổi dịch vụ, đường liên kết cũng sẽ giữ nguyên. Việc này cho phép đánh dấu lại vị trí đang đọc. Nó cũng rất quan trọng vì mối liên quan giữa các tài nguyên mà được mã hoá trong các địa chỉ được giữ nguyên độc lập với các mối liên quan đại diện khi chúng được lưu trữ.

3.3.4. Chuyển đổi linh hoạt JavaScript Object Notation (JSON) và XML hoặc cả hai

Điều cuối cùng trong tập các ràng buộc khi thiết kế RESTful web service phải làm là định dạng dữ liệu mà ứng dụng và web service trao đổi. Client có thể chỉ định kiểu dữ liệu trả về mà nó mong muốn (JSON, XML, ...) các chỉ định này gọi là các kiểu MIME

(*Multipurpose Internet Mail Extensions*). Cụ thể ở đây ta có thể sử dụng các một vài kiểu MIME thông dụng sau:

- JSON
- XML
- XHTML

Điều này cho phép các service sử dụng bởi các client viết bởi các ngôn ngữ khác nhau, chạy trên nhiều nền tảng và thiết bị khác nhau. Sử dụng các kiểu MIME cho phép client chọn dạng dữ liệu phù hợp với nó.

3.3.5. Các nguyên tắc khác

Ngoài 4 nguyên tắc cơ bản khi thiết kế Rest API ở trên, chúng ta cũng cần lưu ý thêm một vài các nguyên tắc khác như:

Authentication (xác thực)

Authentication thì mặc định là không bắt buộc, nhưng bạn nên cài đặt đối với các request liên quan tới POST/PUT/DELETE. Đối với 1 số dịch vụ nhiều người dùng như Twitter, họ bắt buộc authentication đối với tất cả các endpoint để tránh việc sử dụng sai mục đích.

Return type (kiểu trả về)

Dữ liệu trả về (return type) thường là kiểu JSON. Ngoài ra XML cũng là 1 lựa chọn không tệ.

API Rate Limit

Là giới hạn số lần request tới API server. Số lần giới hạn này có thể được tính trên user, application hoặc cụ thể hơn là trên 1 token. Mục đích của con số này nhằm để hạn chế việc lạm dụng API, hoặc 1 hướng khác là nhằm thu phí sử dụng dịch vụ. Ví dụ như Google Maps API, hoàn toàn free nhưng nếu bạn muốn request nhiều hơn thì cần trả thêm tiền.

URL Endpoint

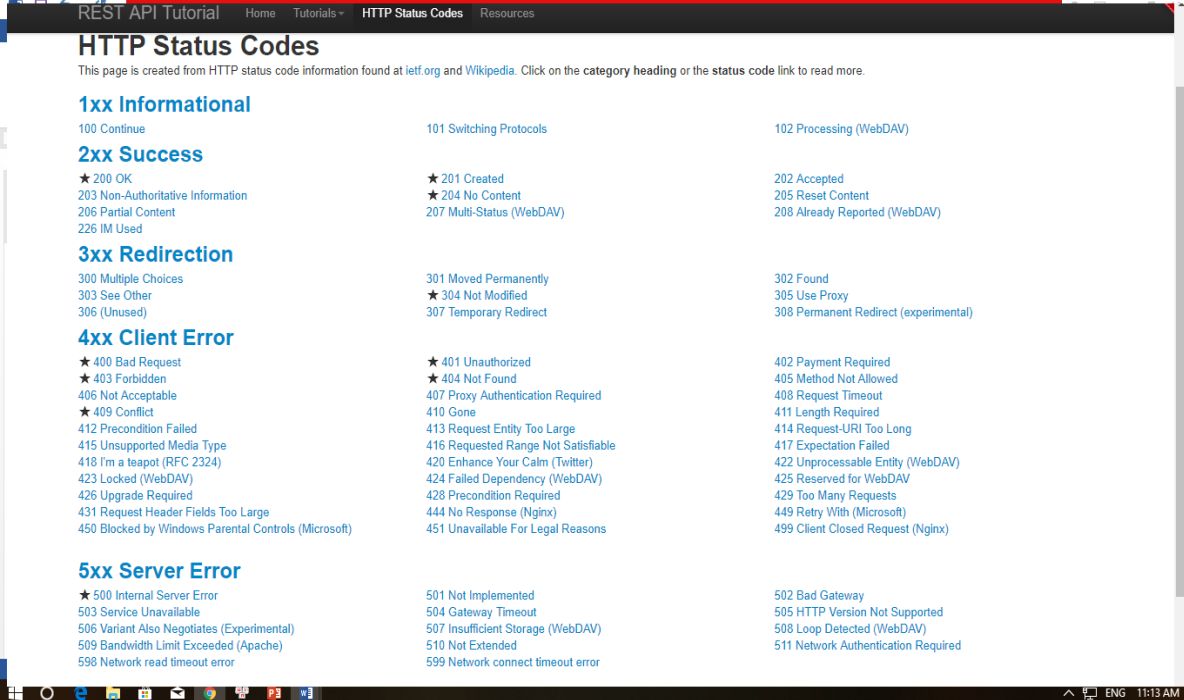
URL Endpoint là URL để client gửi request lên server. Thông thường endpoint sẽ đánh version bằng những cụm như `/v2/` hoặc `/2.0/`. Ví dụ như API search của Twitter có dạng: `https://api.twitter.com/1.1/search/tweets.json?q=%40twittera`

pi. Bạn cần lưu ý rằng với cùng 1 endpoint, nếu phương thức request (GET/DELETE..) khác nhau vẫn có thể trả về các kết quả khác nhau.

Các mã trạng thái của HTTP

Trong khi client request dữ liệu lên server ta nên lấy được mã trạng thái của hành động. Từ đó ta có thể biết được hành động có thành công hay không, nếu không thành công thì lỗi do server hay client?

Một số mã trạng thái của HTTP như sau:



HTTP Status Codes		
This page is created from HTTP status code information found at ietf.org and Wikipedia . Click on the category heading or the status code link to read more.		
1xx Informational		
100 Continue	101 Switching Protocols	102 Processing (WebDAV)
2xx Success		
★ 200 OK	★ 201 Created	202 Accepted
203 Non-Authoritative Information	★ 204 No Content	205 Reset Content
206 Partial Content	207 Multi-Status (WebDAV)	208 Already Reported (WebDAV)
226 IM Used		
3xx Redirection		
300 Multiple Choices	301 Moved Permanently	302 Found
303 See Other	★ 304 Not Modified	305 Use Proxy
306 (Unused)	307 Temporary Redirect	308 Permanent Redirect (experimental)
4xx Client Error		
★ 400 Bad Request	★ 401 Unauthorized	402 Payment Required
★ 403 Forbidden	★ 404 Not Found	405 Method Not Allowed
406 Not Acceptable	407 Proxy Authentication Required	408 Request Timeout
★ 409 Conflict	410 Gone	411 Length Required
412 Precondition Failed	413 Request Entity Too Large	414 Request-URI Too Long
415 Unsupported Media Type	416 Requested Range Not Satisfiable	417 Expectation Failed
418 I'm a teapot (RFC 2324)	420 Enhance Your Calm (Twitter)	422 Unprocessable Entity (WebDAV)
423 Locked (WebDAV)	424 Failed Dependency (WebDAV)	425 Reserved for WebDAV
426 Upgrade Required	428 Precondition Required	429 Too Many Requests
431 Request Header Fields Too Large	444 No Response (Nginx)	449 Retry With (Microsoft)
450 Blocked by Windows Parental Controls (Microsoft)	451 Unavailable For Legal Reasons	499 Client Closed Request (Nginx)
5xx Server Error		
★ 500 Internal Server Error	501 Not Implemented	502 Bad Gateway
503 Service Unavailable	504 Gateway Timeout	505 HTTP Version Not Supported
506 Variant Also Negotiates (Experimental)	507 Insufficient Storage (WebDAV)	508 Loop Detected (WebDAV)
509 Bandwidth Limit Exceeded (Apache)	510 Not Extended	511 Network Authentication Required
598 Network read timeout error	599 Network connect timeout error	

Mã trạng thái HTTP

Mã trạng thái HTTP là loại mã bao gồm 3 chữ số được server phản hồi lại để biểu thị tình trạng của một web. Một điều rất quan trọng cần phải hiểu là: chữ số đầu tiên của mỗi mã là từ 1 đến 5, Trong khoảng từ 100 đến 500, các mã được phân loại như sau:

- 1xx: Informational – yêu cầu (request) đã được nhận, tiếp tục tiến trình xử lý
- 2xx: Success – thành công
- 3xx: Redirection – chuyển hướng
- 4xx: Client Error – yêu cầu sai cú pháp hoặc không thỏa đáng
- 5xx: Server Error – máy chủ gặp lỗi

Mặc dù có rất nhiều mã trạng thái, nhưng ta thường gặp các mã trạng thái sau:

Mã trạng thái có dạng 1xx (phản ứng tạm thời)

Các mã trạng thái có dạng 1XX là những mã trạng thái thể hiện một phản ứng tạm thời và yêu cầu người yêu cầu truy cập cần phải có hành động nào đó để có thể tiếp tục truy cập.

Mã	Mô tả
100: (Continue)	Người truy cập cần tiếp tục với yêu cầu truy cập. Các máy chủ sẽ trả về mã này để thông báo rằng nó đã nhận được phần đầu của một yêu cầu truy cập và đang chờ đợi nhận được phần còn lại.
101: (Switching protocols)	Khách truy cập đã yêu cầu máy chủ chuyển đổi giao thức truy cập và máy chủ phản hồi rằng đã nhận được yêu cầu và xác nhận chuyển đổi giao thức truy cập.

Mã trạng thái có dạng 2xx (Truy cập thành công)

Khi nhận được mã này, khách truy cập (người dùng hoặc các bot của công cụ tìm kiếm) sẽ truy cập thành công liên kết đã yêu cầu trước đó.

Mã	Mô tả
200: (Successful)	Các máy chủ xử lý yêu cầu thành công. Điều này có nghĩa rằng các máy chủ sẽ cung cấp cho khách truy cập trang mà họ yêu cầu.
201 : (Created)	Các máy chủ xử lý yêu cầu thành công và tạo ra một tài nguyên mới cho người truy cập.
202 : (Accepted)	Người truy cập cần tiếp tục với yêu cầu truy cập. Các máy chủ sẽ trả về mã này để thông báo rằng nó đã nhận được phần đầu của một yêu cầu truy cập và đang chờ đợi nhận được phần còn lại.

203 : (Non-authoritative information)	Các máy chủ xử lý yêu cầu thành công, nhưng thông tin phản hồi tới người dùng có thể xuất phát từ một nguồn khác.
204 : (No content)	Các máy chủ xử lý yêu cầu thành công, nhưng không phản hồi bất kỳ nội dung gì tới người dùng.
205: (Reset content)	Các máy chủ xử lý yêu cầu thành công, nhưng không phản hồi bất kỳ nội dung gì tới người dùng. Tuy nhiên, mã này sẽ xuất hiện khi máy chủ đòi hỏi người yêu cầu truy cập phải thiết lập lại chế độ xem tài liệu.
206: (Partial content)	Các máy chủ xử lý thành công một phần của yêu cầu truy cập.

Mã trạng thái có dạng 3xx (truy cập bị chuyển hướng)

Thông thường, các mã trạng thái có dạng 3xx được sử dụng để thông báo chuyển hướng cho khách truy cập. Các mã chuyển hướng này không được Google đánh giá tốt bởi nó cho thấy rằng khách truy cập sẽ không thể đến với trang web mà họ yêu cầu truy cập ban đầu. Và việc chuyển hướng này có thể bị coi là spam.

Mã	Mô tả
300: (Multiple choices)	Các máy chủ có một loạt các hành động có sẵn để lựa chọn dựa trên những yêu cầu mà nó nhận được. Máy chủ web có thể tự động lựa chọn một hành động dựa trên thông tin của người yêu cầu truy cập hoặc thể hiện một danh sách các hành động để người yêu cầu lựa chọn.
301: (Moved permanently)	Trang web được yêu cầu truy cập đã bị chuyển vĩnh viễn sang một địa chỉ mới. Khi máy chủ trả về mã phản ứng này, nó sẽ tự động chuyển tiếp khách truy cập tới địa chỉ mới của liên kết.

302 : (Moved temporarily)	Mã này chỉ thể hiện rằng máy chủ đang tạm thời bị di chuyển sang một địa chỉ mới và người dùng vẫn nên tiếp tục truy cập địa chỉ cũ. Máy chủ hiện đang đáp ứng yêu cầu truy cập trang tại một địa chỉ khác với địa chỉ mà họ đang yêu cầu, đồng thời tự động chuyển tiếp yêu cầu tới một vị trí khác.
303 : (See other location)	Máy chủ trả về mã này khi người truy cập gửi yêu cầu truy cập cho một vị trí khác. Các máy chủ sẽ tự động chuyển yêu cầu truy cập này của người dùng đến vị trí khác đó.
304 : (Not modified)	Trang mà khách truy cập yêu cầu không hề thay đổi kể từ lần yêu cầu truy cập cuối cùng. Khi máy chủ trả về mã này, nó sẽ không gửi lại các nội dung của trang đó.
305 : (Use proxy)	Bên yêu cầu chỉ có thể truy cập vào các trang yêu cầu có sử dụng máy chủ proxy. (Proxy là một Internet server làm nhiệm vụ chuyển tiếp thông tin và kiểm soát tạo sự an toàn cho việc truy cập Internet của các máy khách)
307: (Temporary redirect)	Mã này cho thấy máy chủ đang bị chuyển hướng tạm thời sang một địa chỉ khác, và khách truy cập nên tiếp tục sử dụng địa chỉ này để truy cập trong tương lai.

Mã trạng thái có dạng 4xx (Phát sinh lỗi trong yêu cầu truy cập)

Các mã trạng thái 4xx cho thấy đã có thể phát sinh lỗi trong yêu cầu truy cập của người dùng và máy chủ web đã không thể xử lý yêu cầu này.

Mã	Mô tả
400: (Bad request)	Máy chủ không hiểu được cú pháp của yêu cầu.

401: (Not authorized)	(Không được uỷ quyền) Yêu cầu truy cập cần được chứng thực. Máy chủ có thể gửi về phản hồi này cho một trang web yêu cầu cần đăng nhập.
403 : (Forbidden)	Máy chủ từ chối các yêu cầu truy cập. Nếu Googlebot nhận mã trạng thái này khi đang cố gắng thu thập dữ liệu của trang web, có thể là do máy chủ hoặc website đã chặn truy cập của Googlebot.
404: (Not found)	Máy chủ không thể tìm thấy trang yêu cầu. Thông thường, mã này sẽ xuất hiện nếu yêu cầu truy cập một trang không tồn tại trên máy chủ.
405: (Method not allowed)	Các phương pháp quy định trong yêu cầu không được máy chủ cho phép.
406: (Not acceptable)	Trang mà khách truy cập yêu cầu không thể đáp ứng với những đặc điểm của nội dung đã yêu cầu.
407: (Proxy authentication required)	Mã trạng thái này cũng tương tự như 401 (Không được uỷ quyền); nhưng quy định rằng người yêu cầu phải xác thực bằng cách sử dụng một proxy.
408: (Request timeout)	Máy chủ đã hết thời gian để chờ nhận yêu cầu truy cập.
409 : (Conflict)	Máy chủ gặp phải một cuộc xung đột giữa các yêu cầu truy cập. Các máy chủ có thể trả về mã này khi có một yêu cầu truy cập mới xung đột với một yêu cầu trước đó.

410 : (Gone)	Máy chủ trả về phản ứng này khi các tài nguyên yêu cầu đã bị xoá vĩnh viễn. (tương tự 404 nhưng đôi khi được sử dụng để chỉ một nguồn lực đã được sử dụng để tồn tại nhưng hiện không còn nữa)
411: (Length required)	ác máy chủ sẽ không chấp nhận các yêu cầu mà không có độ dài của nội dung truy cập hợp lệ.
412 : (Precondition failed)	Các máy chủ không đáp ứng được một trong các điều kiện tiên quyết mà người yêu cầu đặt ra trong yêu cầu truy cập.
413 : (Request entity too large)	Máy chủ không thể xử lý yêu cầu bởi vì nó quá lớn để các máy chủ có thể xử lý.
414 : (Requested URI is too long)	URI được yêu cầu quá dài để máy chủ có thể xử lý.
415 : (Unsupported media type)	Khách truy cập đang yêu cầu một định dạng không được hỗ trợ bởi máy chủ web.
416 : (Requested range not satisfiable)	Các máy chủ trả về mã trạng thái này nếu bạn yêu cầu cho một phạm vi không có sẵn cho trang.
417 : (Expectation failed)	Các máy chủ không thể đáp ứng những yêu cầu theo như mong đợi của người dùng được thể hiện trong header.

Mã trạng thái có dạng 5xx (Lỗi server)

Các mã trạng thái chỉ ra rằng đã phát sinh lỗi bên trong máy chủ khi đang cố gắng để xử lý yêu cầu của khách truy cập. Những lỗi này có xu hướng xuất phát từ phía máy chủ web, chứ không phải từ yêu cầu truy cập.

Mã	Mô tả
----	-------

500: (Internal server error)	Các máy chủ đã gặp lỗi và không thể hoàn tất yêu cầu.
501: (Not implemented)	Các máy chủ không có các chức năng để hoàn tất yêu cầu. Ví dụ, máy chủ có thể trả về mã này khi nó không nhận ra phương pháp yêu cầu.
502: (Bad gateway)	Các máy chủ đã hoạt động như một gateway hoặc proxy và nhận được phản hồi không hợp lệ từ các máy chủ ở tuyến trên.
503 : (Service unavailable)	Máy chủ hiện tại không thể thực hiện yêu cầu (vì bị quá tải hoặc đang phải bảo trì). Đây là một thông báo về tình trạng tạm thời của máy chủ.
504 : (Gateway timeout)	Máy chủ đã hoạt động như một gateway hoặc proxy và không nhận được một yêu cầu kịp thời từ các máy chủ ở tuyến trên.
505 : (HTTP version not supported)	Các máy chủ không hỗ trợ phiên bản giao thức HTTP được sử dụng trong các yêu cầu.

3.4. Truy xuất tài nguyên qua Rest API

Các API công khai thường được truy xuất trực tiếp từ các địa chỉ website, do vậy cấu trúc đường dẫn là rất quan trọng và các URL kiểu này nên được thiết kế chỉ cho các yêu cầu API. Nhiều URL có thể đưa vào các tiền tố như /v2/ ám chỉ rằng đây là phiên bản thứ 2 của API. Nó là thông tin hữu ích cho các nhà phát triển để phân biệt với các phiên bản API 1.x mà cấu trúc dữ liệu đã thay đổi.

Với các phương thức HTTP khác nhau dữ liệu trả về của cùng một địa chỉ API có thể thay đổi, ví dụ GET để truy xuất nội dung trong khi POST để tạo nội dung mới. Yêu cầu có thể đến cùng một nơi nhưng kết quả rất khác nhau, do vậy cần thiết lập phương thức HTTP cho chuẩn xác.

3.5. Xây dựng một Restful API

Xây dựng một hệ thống API cung cấp dữ liệu cho website của bạn không phải là công việc quá phức tạp nếu bạn có những hiểu biết về các mẫu thiết kế API. Với các framework hiện nay như Laravel, Visual studio .NET ... API được xây dựng thực sự nhanh chóng, bạn cũng có thể sử dụng API để tạo ra các ứng dụng đa tầng phức tạp. Mỗi API cần được kết nối đến máy chủ để trả về một vài loại dữ liệu, bạn phải viết code để làm điều đó và cũng cần định dạng dữ liệu trả về.

Xây dựng đường dẫn

Một trong những công việc quan trọng của phát triển API là xây dựng các đường dẫn, khi tạo ra các tài nguyên chúng ta muốn sử dụng các danh từ chứ không phải động từ. Điều đó có nghĩa là dữ liệu API phải trả về một đối tượng như một người nào đó, một nơi nào đó ... Thật khó để đưa ra các danh từ mô tả đầy đủ trong thiết kế đường dẫn, tuy nhiên nên đơn giản nhất có thể. Có một vài các quy tắc chọn danh từ trong đường dẫn như sau:

Nên chọn danh từ số nhiều, ví dụ `/products/12345678`
Nên chọn các danh từ cụ thể tránh các từ trừu tượng, ví dụ `/products/12345678` thay vì `/items/12345678`.

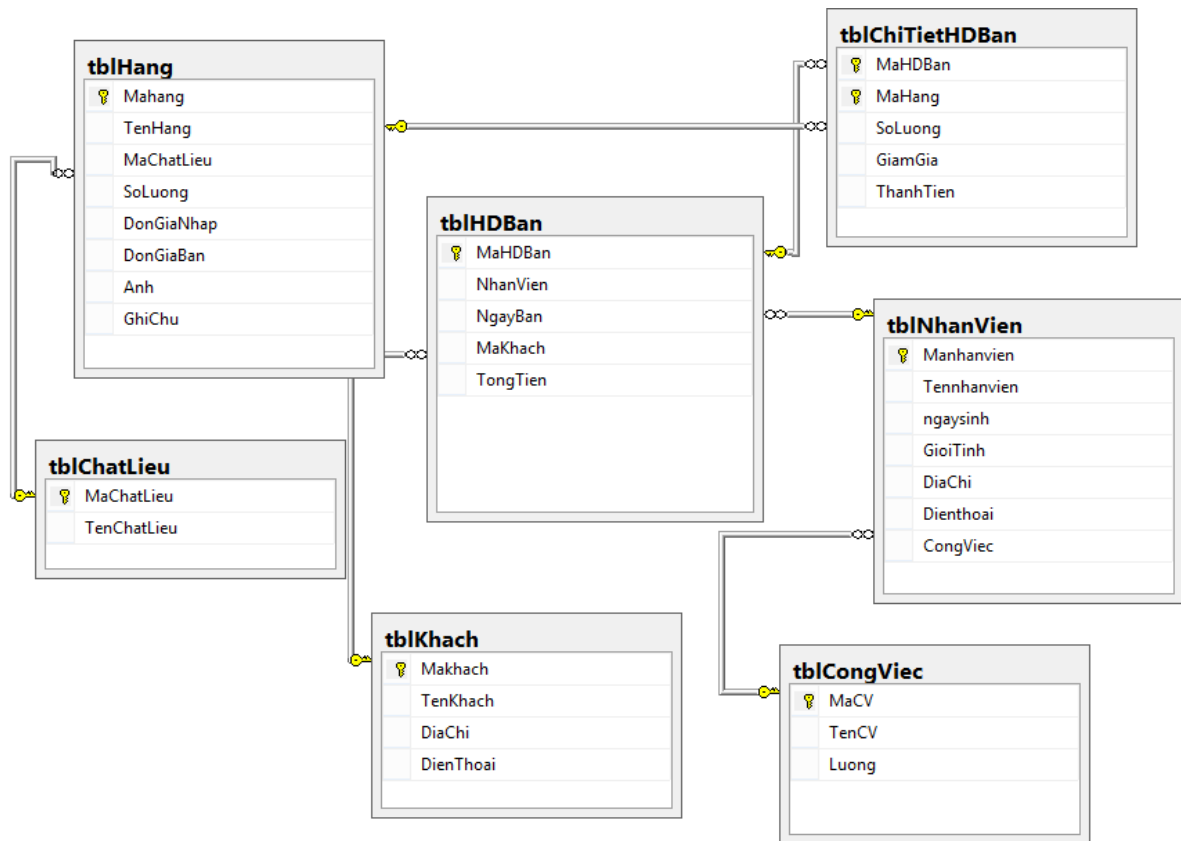
Thiết lập dạng dữ liệu trả về

Một vấn đề cần quan tâm là loại dữ liệu trả về, hầu hết người dùng API mong đợi dữ liệu trả về dạng JSON, tuy nhiên XML cũng là một sự lựa chọn tốt. Nhưng các bạn cũng đừng quá lo, nếu bạn sử dụng Visual studio .NET thì việc trả về dữ liệu kiểu gì cũng chỉ là thiết lập trong một câu lệnh đơn giản thôi.

Có rất nhiều vấn đề cần quan tâm trong khi phát triển các API, vậy để đơn giản hơn trong tiếp cận thì chúng ta hãy cùng xây dựng lấy một API đầu tiên.

3.5.1. Bài toán

Trên server có một Cơ sở dữ liệu có tên là **DuLieu** như sau:



Yêu cầu: Xây dựng Web API theo chuẩn Rest cung cấp 5 dịch vụ như sau:

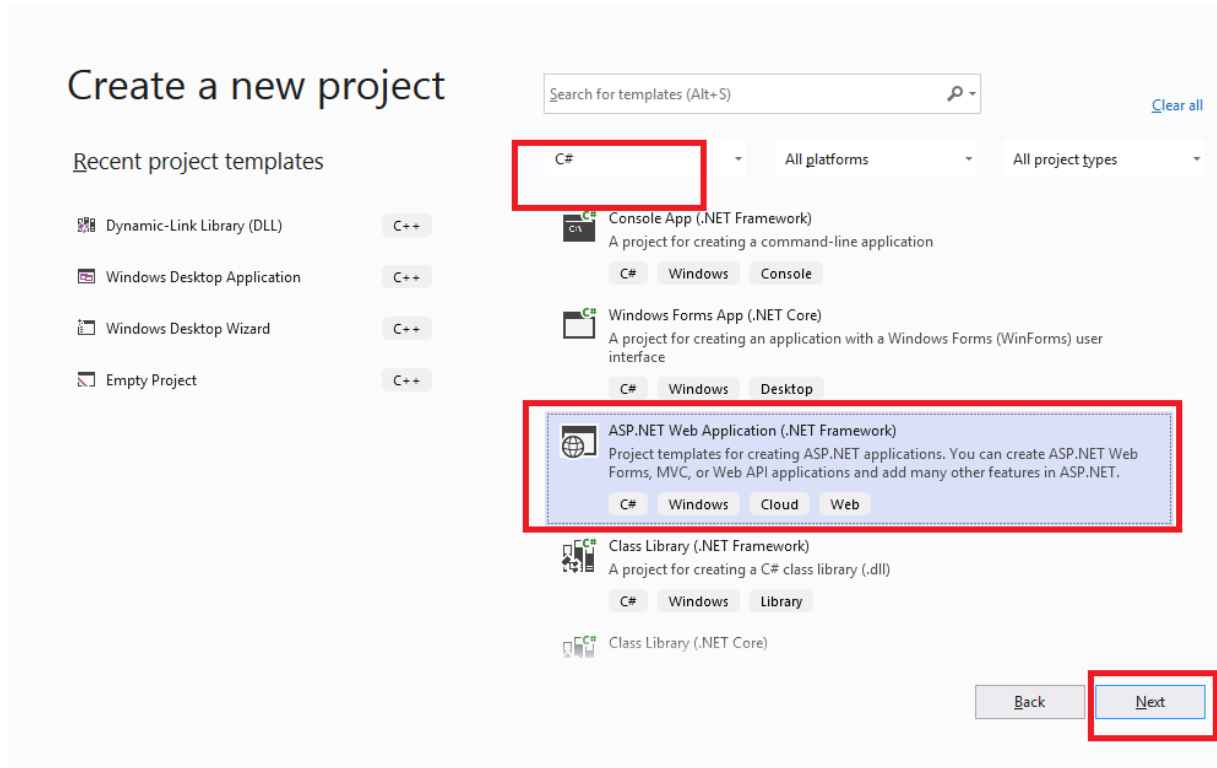
1. Lấy thông tin tất cả các khách hàng
2. Lấy thông tin một khách hàng với mã cụ thể
3. Xóa một khách hàng
4. Sửa thông tin một khách hàng
5. Thêm mới một khách hàng

3.5.2. Xây dựng Rest API

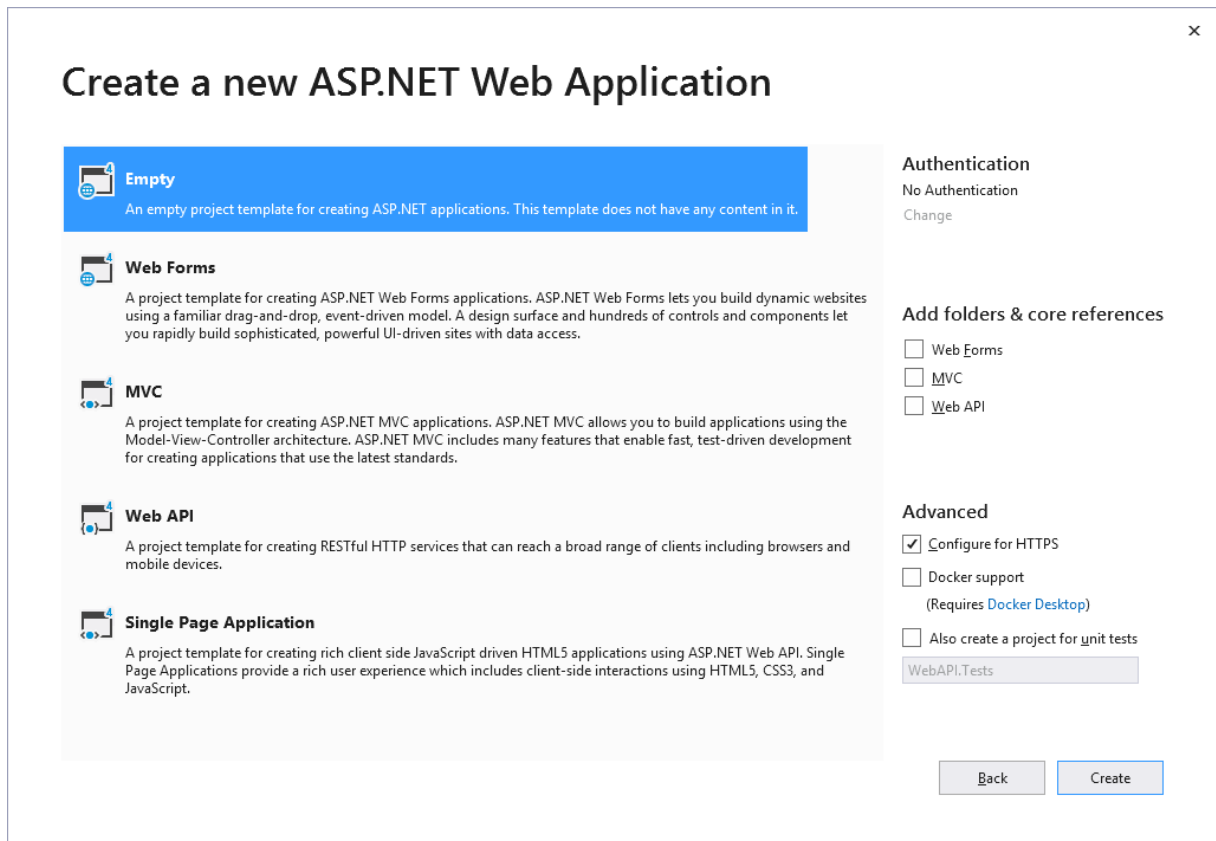
Ở đây, tôi dùng Visual studio 2019 và ngôn ngữ C# để hướng dẫn các bạn tạo một web API.

Giả sử rằng trên Server đã có sẵn Cơ sở dữ liệu **DuLieu** rồi. Bây giờ ta chỉ tập chung vào việc xây dựng Rest API.

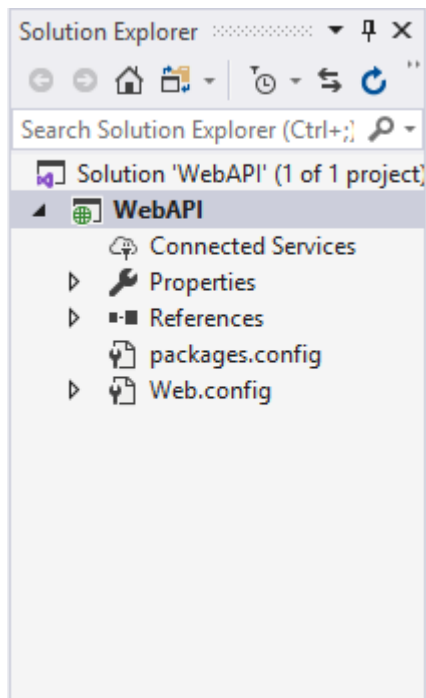
Bước 1: Tạo project mới. Khởi động Visual studio lên, khởi động Visual studio 2019 chọn **Create a new Project...**, ta chọn các cấu hình như sau:



Sau đó đặt tên dự án là WebAPI, chọn kiểu mẫu là Empty như hình sau:

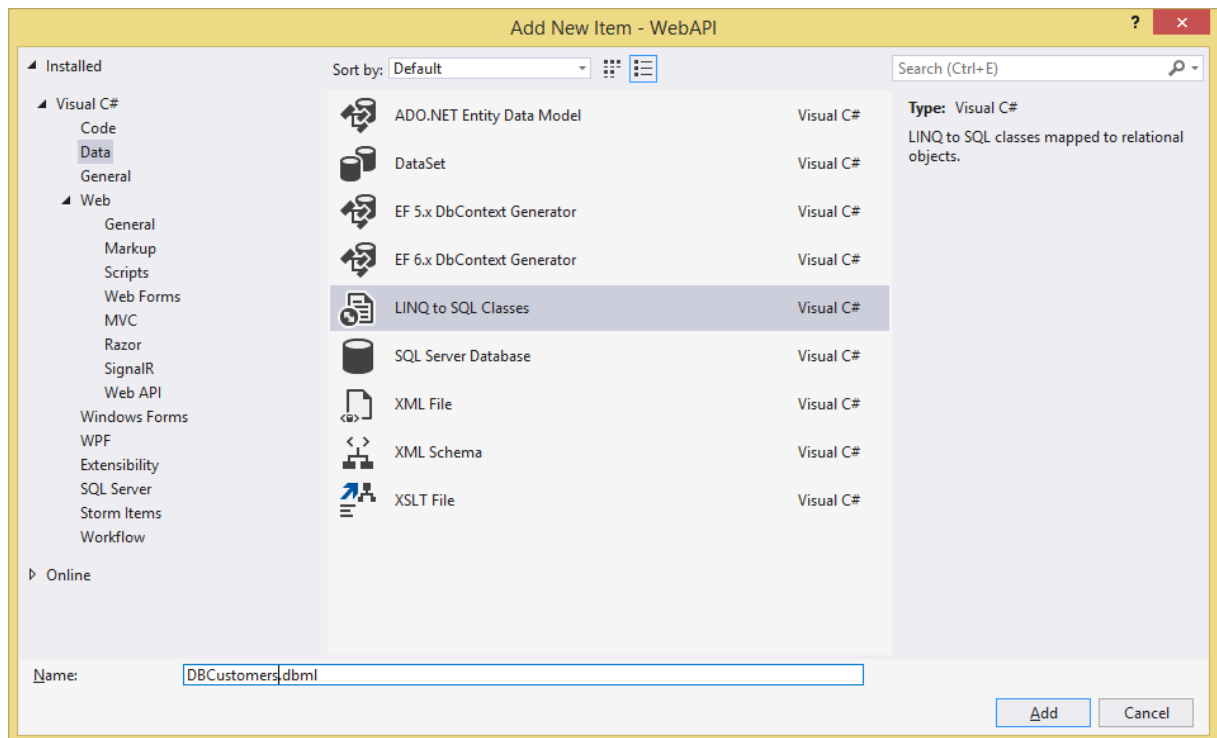


Project mặc định ban đầu như sau:

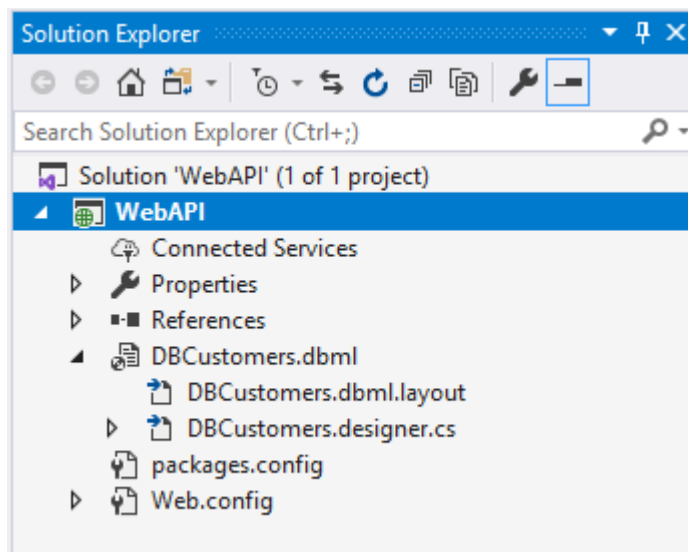


Bước 2: Tạo LINQ to SQL cho Cơ sở dữ liệu DuLieu (làm việc với bảng tblKhach).

- Bấm chuột phải vào tên Project, chọn Add/new items.../LINQ to SQL Class màn hình đặt tên cho LINQ xuất hiện ta đặt là DBCustomers

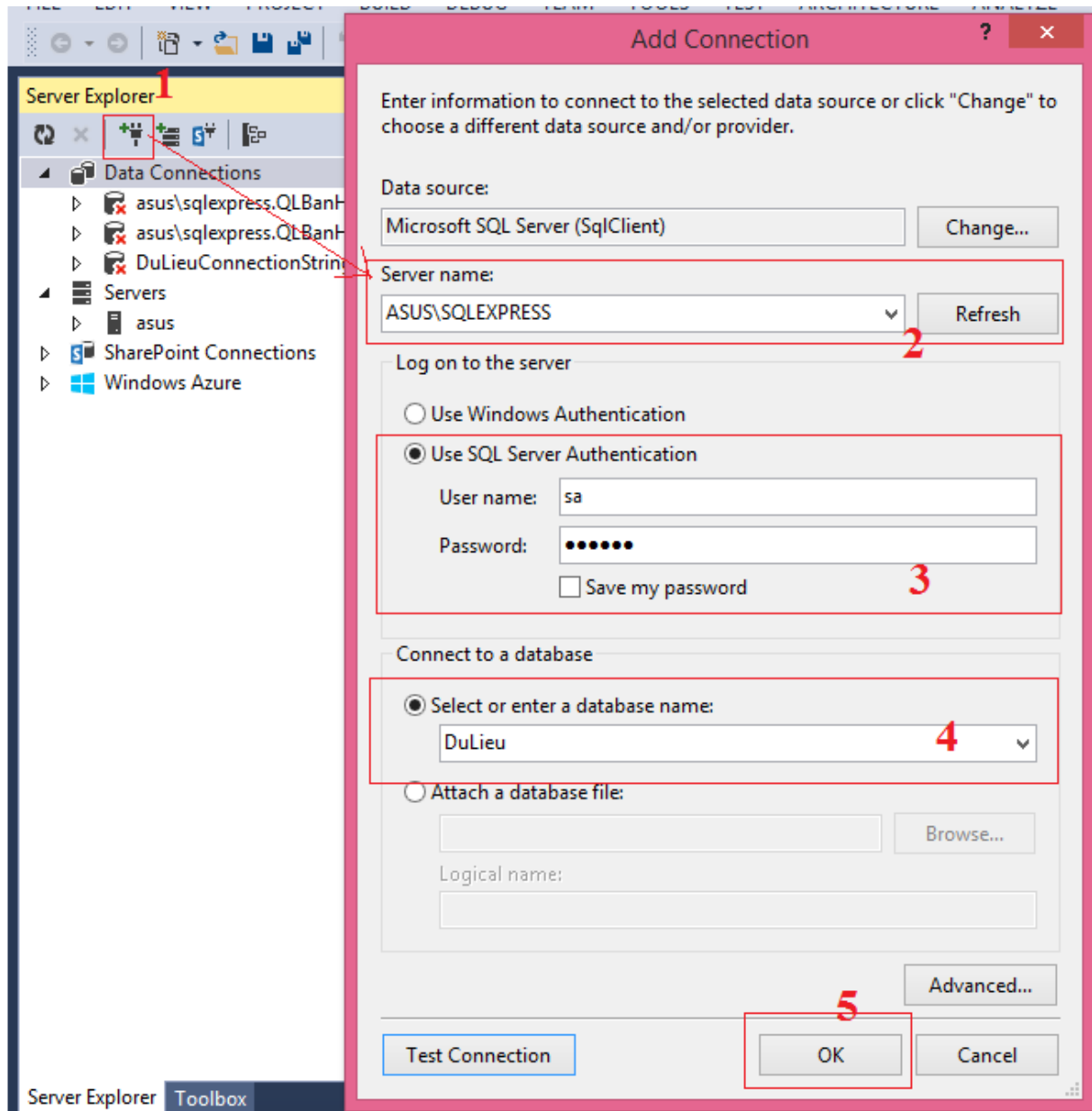


Sau khi nhấn OK ta thu được Kết quả như sau:

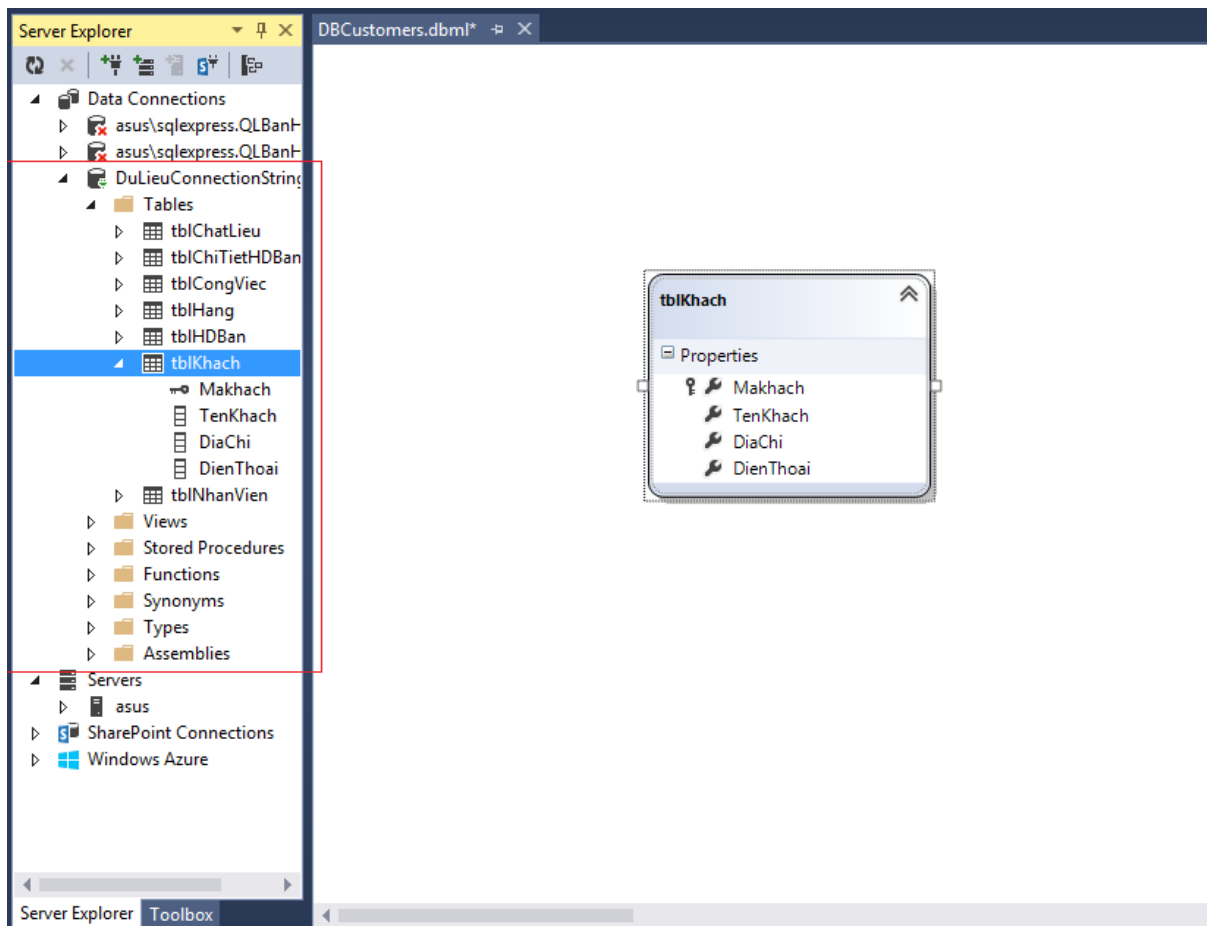


– Ta tiến hành cấu hình để kéo Cơ sở dữ liệu vào làm các lớp tương tác:

Bạn mở Server Explorer (vào menu View/Server Explorer), bấm chọn theo các bước như bên dưới (lệ thuộc vào Server của bạn mà chọn Server name, User đăng nhập cho phù hợp):

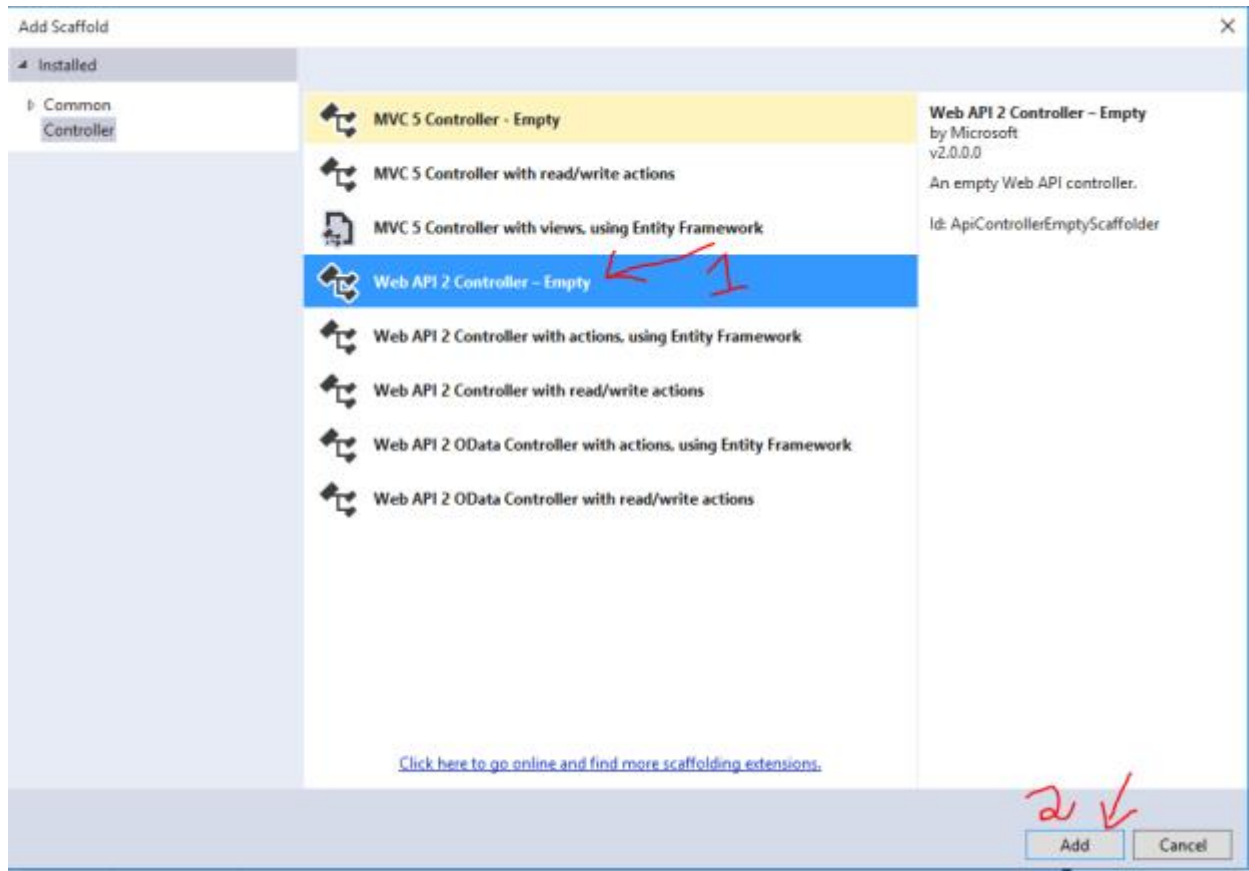


Khi bấm OK, bạn sẽ thấy Server Explorer có thêm mục **DuLieuConnectionString**, ta kéo bảng tblKhach vào mục bên phải như hình:

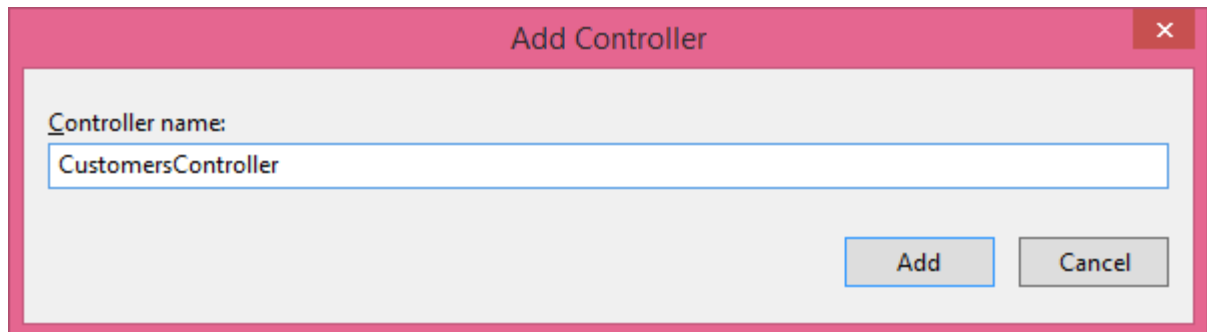


Tiếp theo bạn tạo 1 thư mục (tên gì cũng được), ở đây Tui đặt đại tên Controllers (bấm chuột phải vào Project/chọn Add/chọn New Folder):

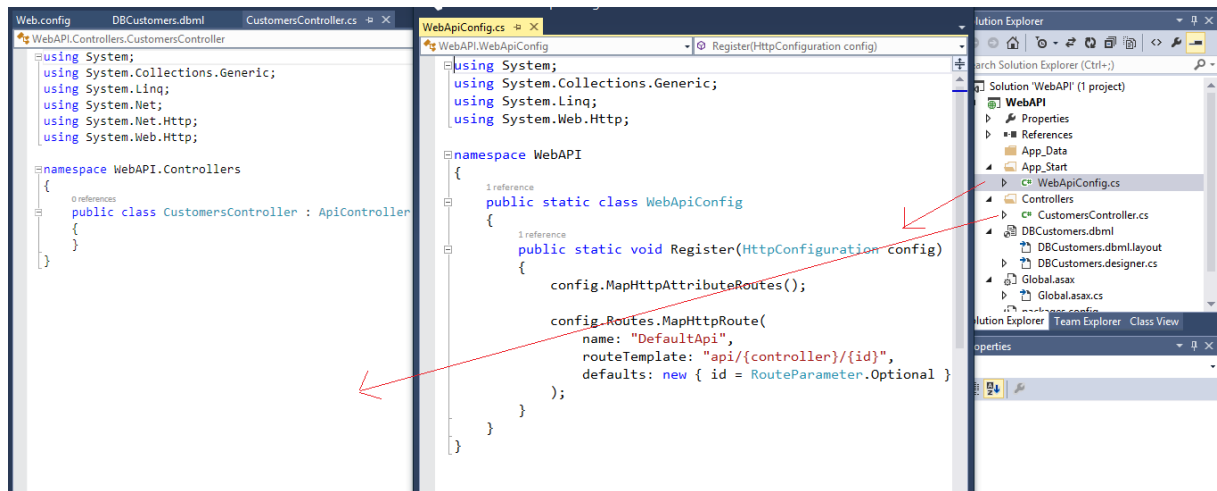
Sau khi thư mục Controllers được tạo ra, bạn bấm chuột phải vào Thư Mục này rồi chọn **Add/Controller..** màn hình sau xuất hiện:



Ta chọn Web API 2 Controller – Empty rồi bấm Add, Visual sẽ hiển thị màn hình đặt tên cho Controller, ta đặt lại tên là CustomersController



Tập tin **CustomersController.cs** sẽ được tạo ra, nhưng ta chỉ lấy **Customers**(bỏ chữ Controller đằng sau đi) để tương tác (đây là cơ chế hoạt động). Ta xem cấu trúc Controller được tạo ra như sau:



Cấu trúc này bao gồm:

File **WebApiConfig.cs** được sinh ra trong thư mục **App_Start**, bạn để ý routeTemplate: “api/{controller}/{id}”, tức là khi ta dùng thì viết: “**api/Customers**” để lấy toàn bộ danh sách, hay “**api/Customers/K01**” để lấy chi tiết thông tin của một khách hàng có mã là K01.

File **CustomersController.cs** kế thừa từ ApiController

Bước 3: Ta tiến hành viết các chức năng cho Web API cung cấp 5 dịch vụ như bài toán yêu cầu (viết trong file CustomerController.cs), ta tuân thủ theo các quy tắc sử dụng giao thức HTTP của Rest như sau:

HttpGet: Truy vấn thông tin về Khách hàng

HttpPost: Thêm Khách hàng mới

HttpPut: Thay đổi thông tin khách hàng

HttpDelete: Xóa 1 khách hàng

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
```

```
namespace WebAPI_new.Controller
{
    public class CustomersController : ApiController
    {
```

```
//httpGet: dùng để lấy thông tin khách hàng
//1. Dịch vụ lấy thông tin của toàn bộ khách hàng
[HttpGet]
public List<tblKhach> GetCustomerLists()
{
    DBCustomerDataContext dbCustomer = new
DBCustomerDataContext();
    return dbCustomer.tblKhaches.ToList();
}
//2. Dịch vụ lấy thông tin một khách hàng với mã nào
đó
[HttpGet]
public tblKhach GetCustomer(string id)
{
    DBCustomerDataContext dbCustomer = new
DBCustomerDataContext();
    return dbCustomer.tblKhaches.FirstOrDefault(x =>
x.Makhach == id);
}
//3. httpPost, dịch vụ thêm mới một khách hàng
[HttpPost]
public bool InsertNewCustomer(string id, string name,
string adress, string phoneNumber)
{
    try
    {
        DBCustomerDataContext dbCustomer = new
DBCustomerDataContext();
        tblKhach customer = new tblKhach();
        customer.Makhach = id;
        customer.TenKhach = name;
        customer.DiaChi = adress;
        customer.DienThoai = phoneNumber;

        dbCustomer.tblKhaches.InsertOnSubmit(customer);
        dbCustomer.SubmitChanges();
        return true;
    }
    catch
    {
        return false;
    }
}
//4. httpPut để chỉnh sửa thông tin một khách hàng
[HttpPut]
public bool UpdateCustomer(string id, string name,
string adress, string phoneNumber)
```

```
{
    try
    {
        DbContext dbCustomer = new
DbContext();
        //Lấy mã khách đã có
tblKhach customer =
dbCustomer.tblKhaches.FirstOrDefault(x => x.Makhach == id);
        if (customer == null) return false;
        customer.Makhach = id;
        customer.TenKhach = name;
        customer.DiaChi = adress;
        customer.DienThoai = phoneNumber;
        dbCustomer.SubmitChanges();//Xác nhận chỉnh
sửa
        return true;
    }
    catch
    {
        return false;
    }
}
//5.httpDelete để xóa một Khách hàng
[HttpDelete]
public bool DeleteCustomer(string id)
{
    try
    {
        DbContext dbCustomer = new
DbContext();
        //Lấy mã khách đã có
tblKhach customer =
dbCustomer.tblKhaches.FirstOrDefault(x => x.Makhach == id);
        if (customer == null) return false;

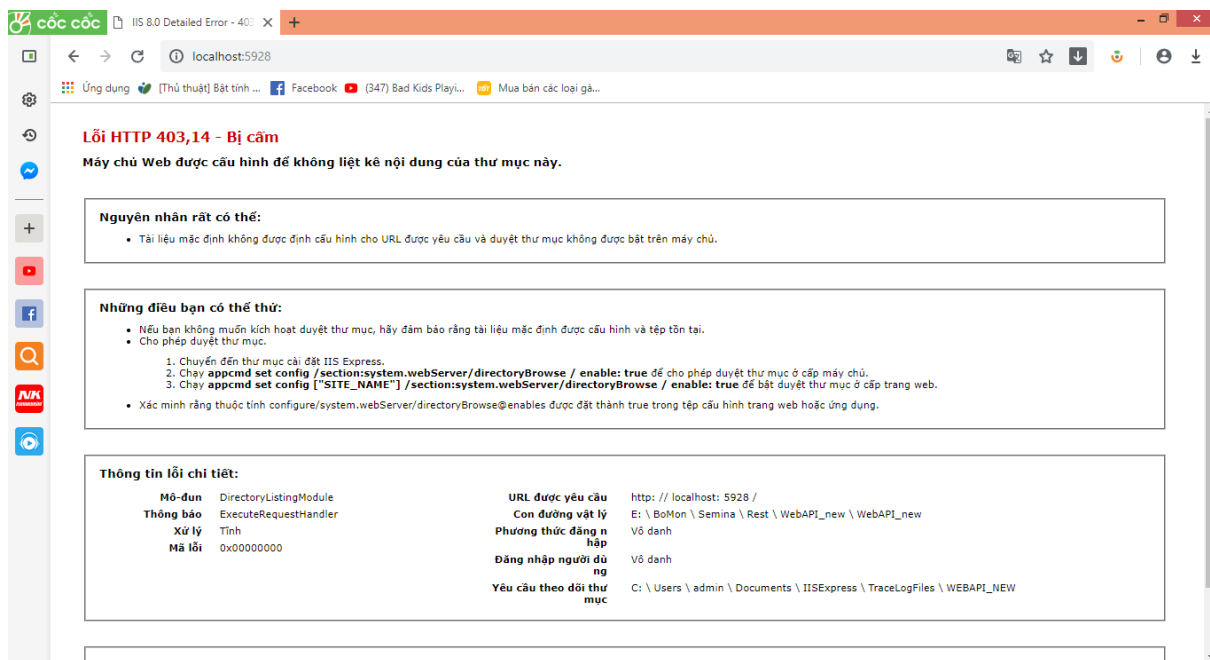
        dbCustomer.tblKhaches.DeleteOnSubmit(customer);
        dbCustomer.SubmitChanges();
        return true;
    }
    catch
    {
        return false;
    }
}
}
```

Chú ý: Web API nó không quan tâm tới tên phương thức (viết tên gì cũng được, nó tự động lấy chính xác Service yêu cầu), tuy nhiên nó không cho phép trùng tên biến. Ví dụ nếu bạn cố tình tạo thêm 1 hàm:

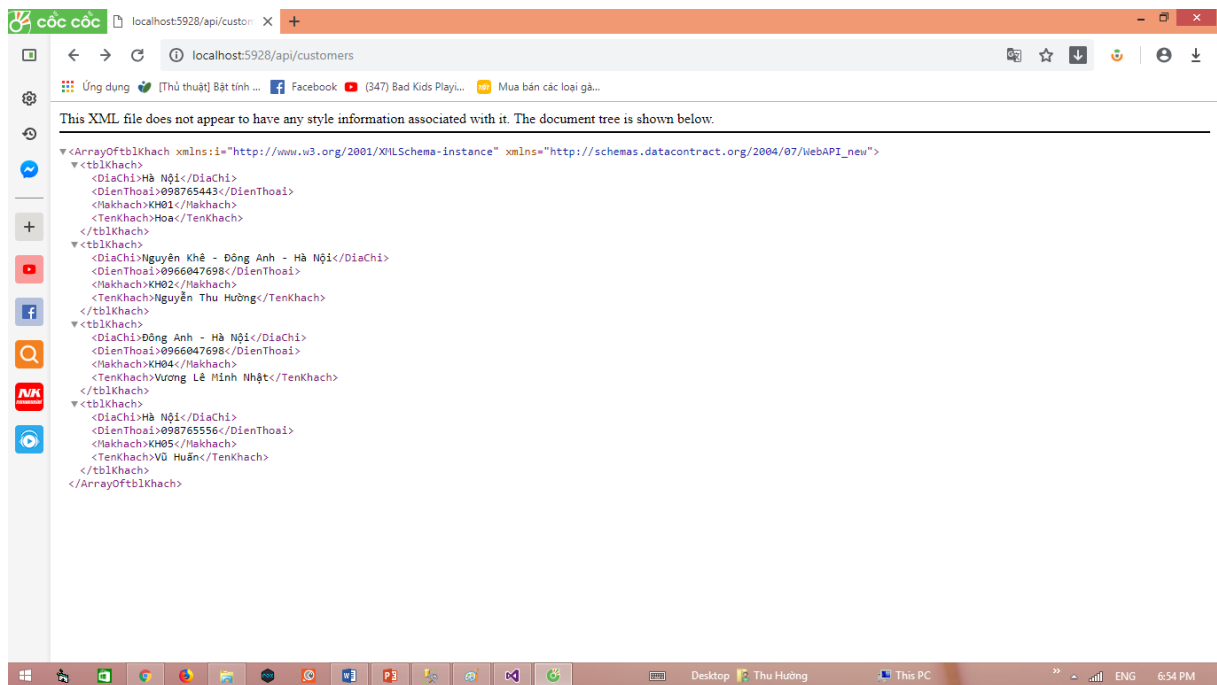
```
[HttpGet]
public Food GetCustomer_test(string id)
{
    DBCustomerDataContext db = new DBFoodDataContext();
    return db.tblKhaches.FirstOrDefault(x => x.id == id);
}
```

Khi chạy sẽ báo lỗi ngay (vì hệ thống không quan tâm tên hàm), nó thấy 2 biến **id** giống nhau ở trên 2 hàm nó sẽ không biết dùng cái nào (vì chúng cùng nhóm HttpGet).

Bước 4: Chạy web API. Ta bấm F5 để chạy Project:



Bạn chỉnh lại URL là: <http://localhost:5928/api/customers>



3.5.3. Kiểm thử Rest API bằng Postman

Tại sao phải Test API

Trong quá trình triển khai dự án, phần server và client làm việc độc lập với nhau nên nhiều khi Client chưa làm xong. Ta không thể đợi làm xong phần client mới test được các API phía server. Khi ấy ta cần test API bằng công cụ khác luôn và việc test hoàn toàn không phụ thuộc gì vào client.

Kể cả khi client làm xong rồi, nếu mình test trên client mà thấy lỗi liên quan đến logic và dữ liệu thì cũng cần test thêm cả API để biết chính xác là server sai hay client sai. Đây là việc sẽ giúp tester tìm ra lỗi nhanh hơn.

Khi làm hệ thống web services, dự án của mình chỉ viết API cho bên khác dùng, mình sẽ không có client để test giống như các dự án khác. Vì vậy phải dùng công cụ để test API hoàn toàn.

Công cụ kiểm thử Postman

Postman là một công cụ dùng để Test API.

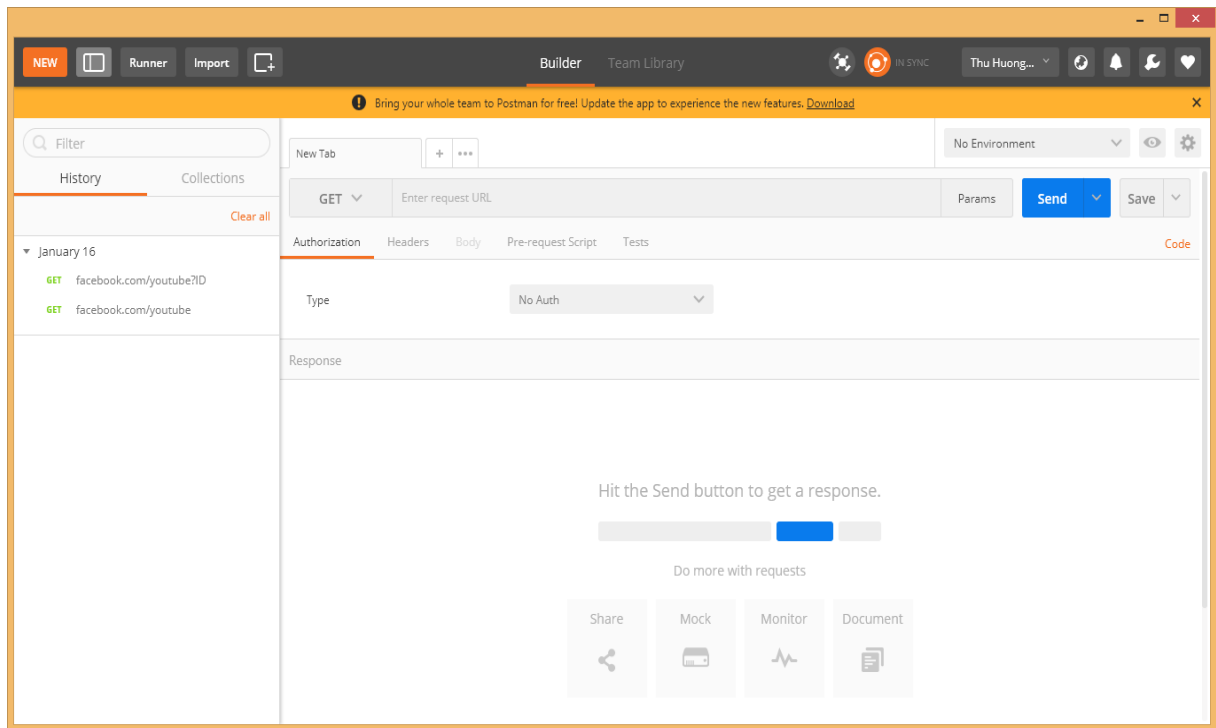
Ưu điểm:

- ✎ Dễ sử dụng, hỗ trợ cả chạy bằng UI và non-UI.
- ✎ Hỗ trợ viết code cho assert tự động bằng Javascript.

- ✎ Hỗ trợ cả RESTful services và SOAP services.
- ✎ Có chức năng tạo API document.

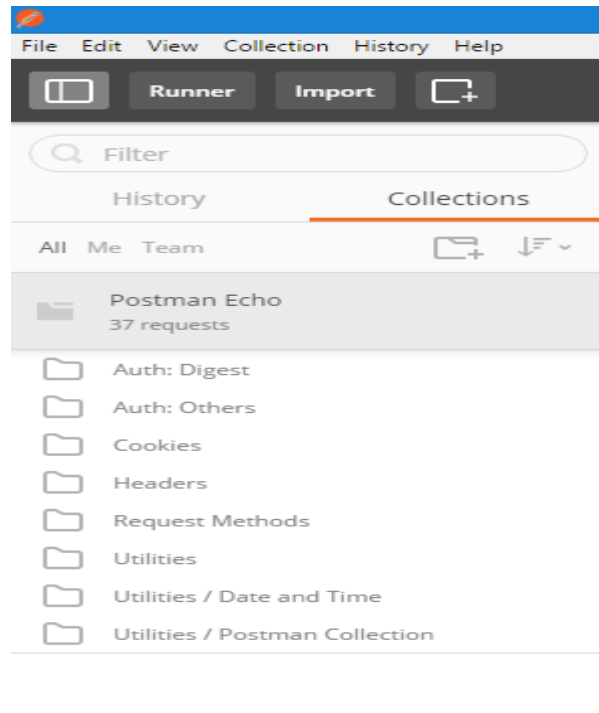
Nhược điểm: Các tính năng nâng cao như: Làm việc theo team, support trực tiếp...thì chỉ có bản tính phí mới hỗ trợ.

Sau khi cài đặt Postman, ta khởi động chương trình có giao diện như sau:



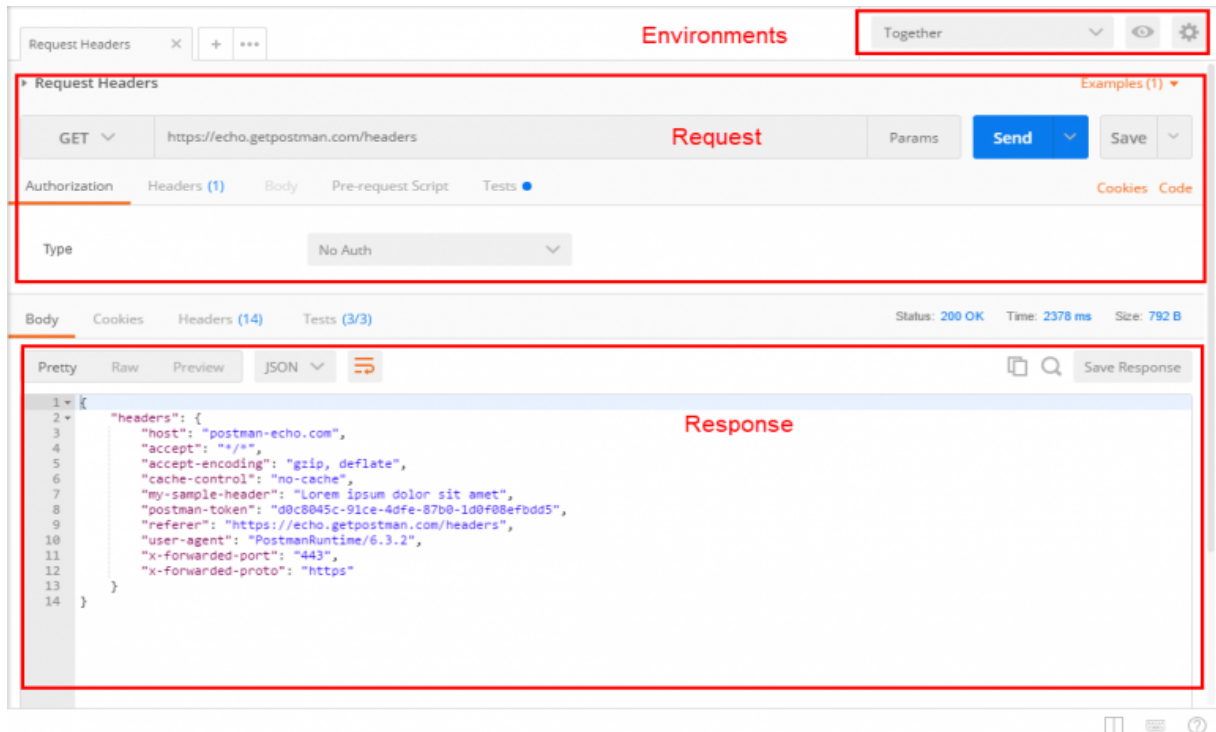
Các thành phần của Postman

- ✎ Settings: chứa các thông tin về cài đặt chung
 - Thông tin Account: dùng để Login, logout và sync data.
 - Settings tùy chỉnh: themes, shortcut, format...
 - Import data từ ngoài vào
- ✎ Collections: Lưu trữ thông tin của các API theo folder hoặc theo thời gian.



✎ API content: Hiển thị nội dung chi tiết API và các phần hỗ trợ giúp thực hiện test API. Đây là phần mà tester phải làm việc nhiều nhất. Phần này có 3 thành phần chính:

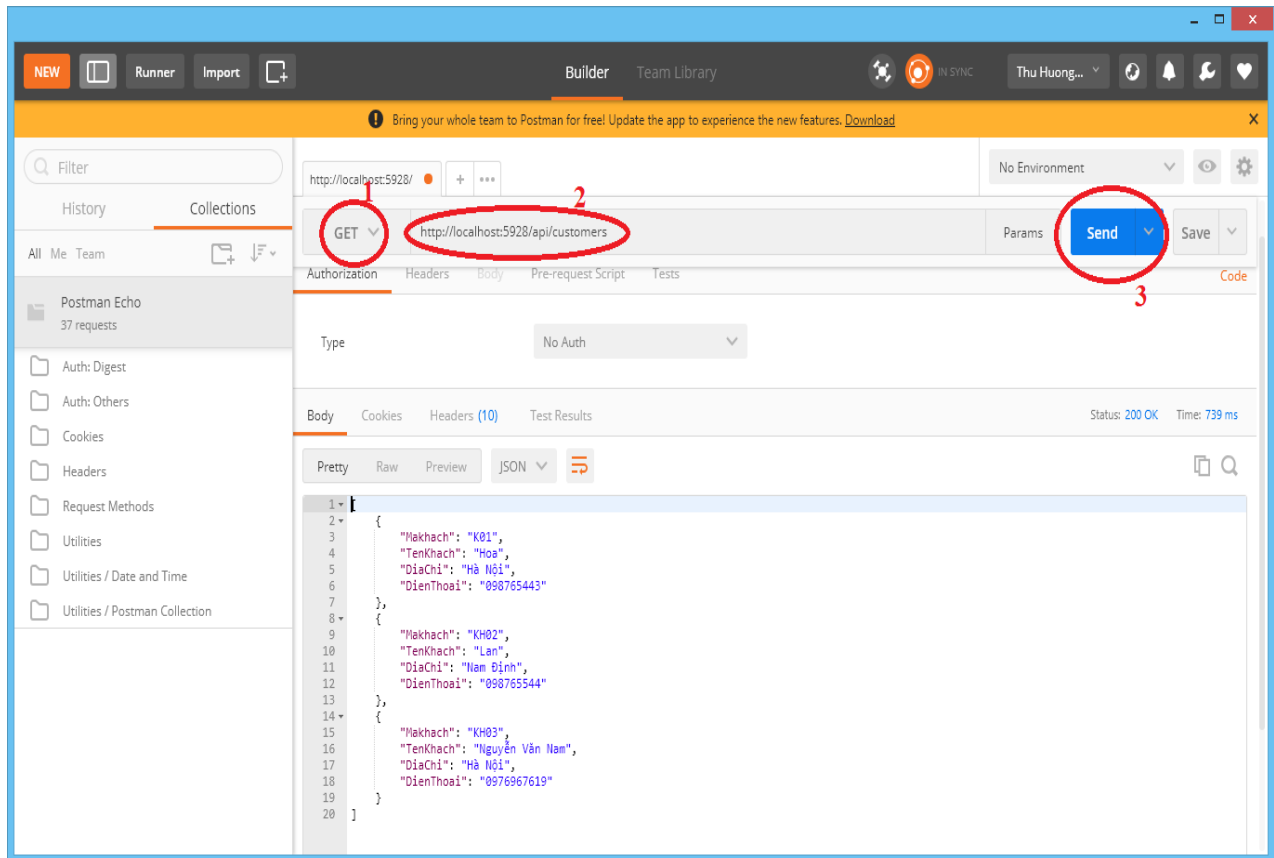
- Enviroments: Chứa các thông tin môi trường.
- Request: Phần chứa các thông tin chính của API. (URL, Method, Headers và Body)
- Reponse: Chứa các thông tin trả về sau khi Send Request.



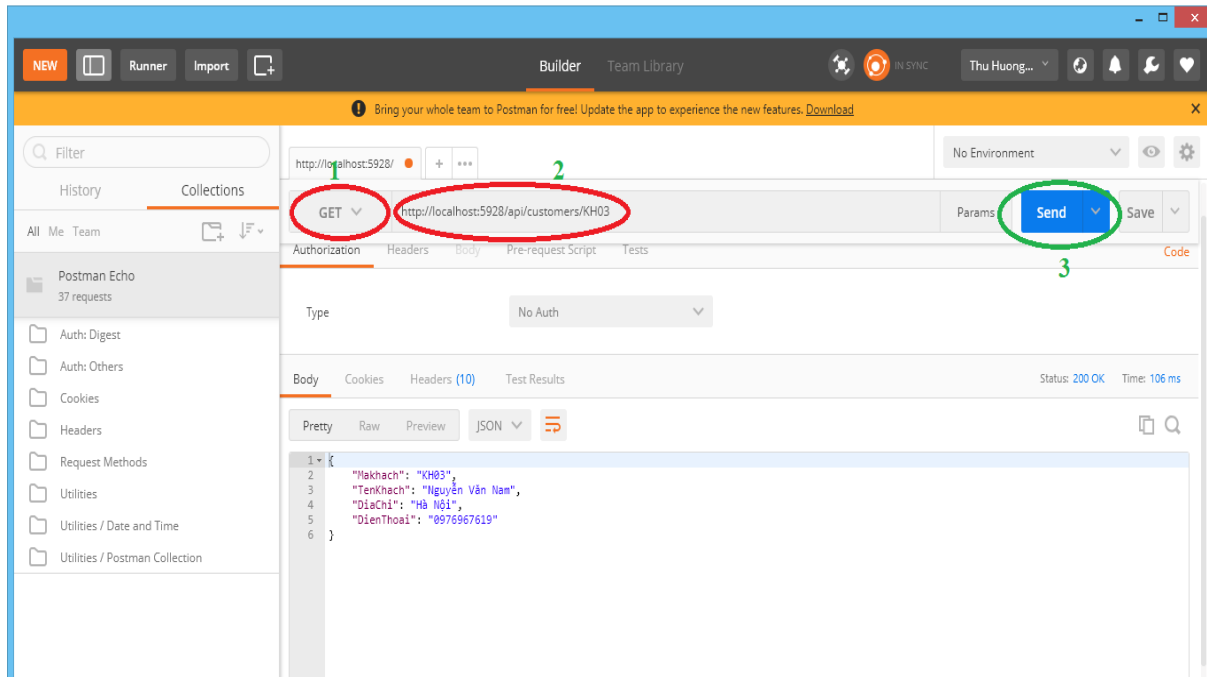
Các thành phần của API Content

Ví dụ: Dùng Postman kiểm tra sự vận hành của các dịch vụ được cung cấp ở phần 3.5.2.

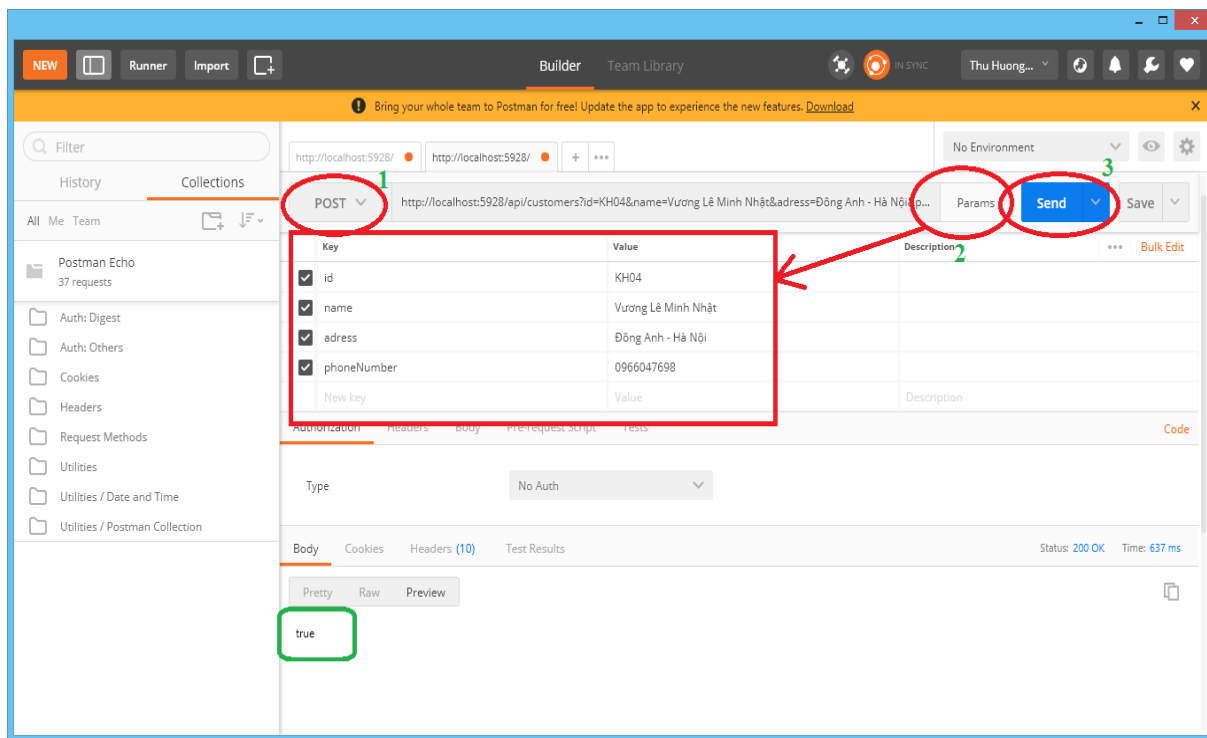
Kiểm thử chức năng lấy toàn bộ dữ liệu khách hàng với giao thức HTTP Get:



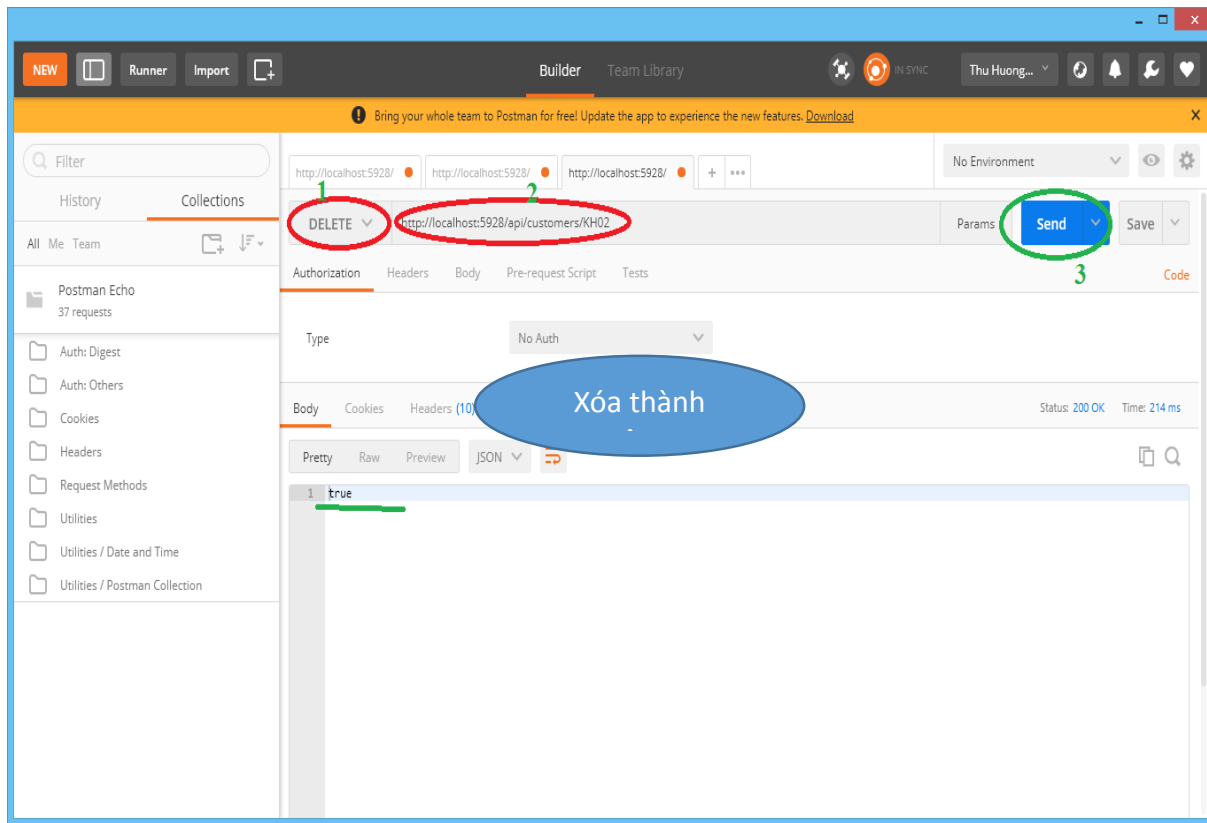
Kiểm thử chức năng lấy thông tin của một khách hàng có mã bất kỳ dùng giao thức HTTP Get



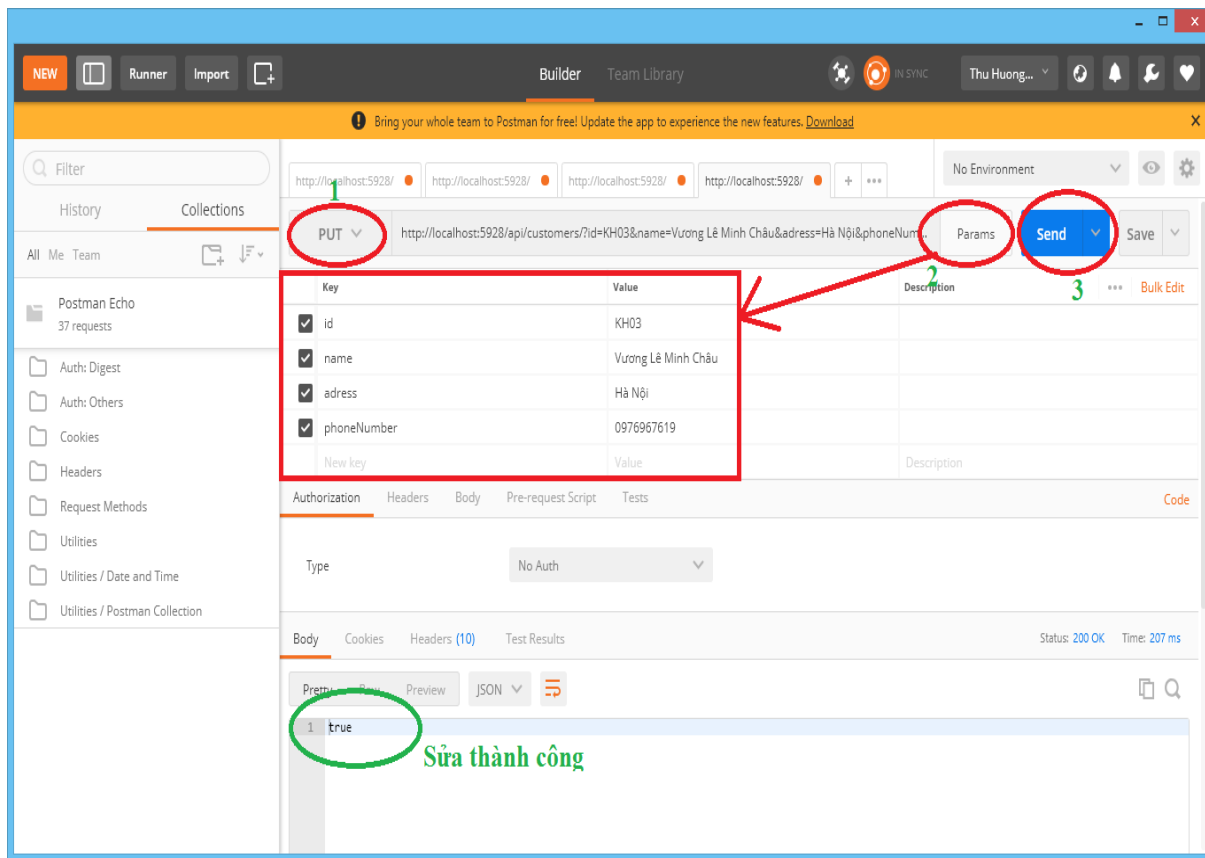
Kiểm tra HTTP Post để tạo một khách hàng mới



Xóa 1 KH với mã bất kỳ (với HTTP Delete)



HTTP Put để sửa thông tin 1 KH



3.6. 3.6. Cách gọi WEB API từ ứng dụng client

3.6.1. Giới thiệu AJAX

AJAX là chữ viết tắt của Asynchronous JavaScript and XML, AJAX = Asynchronous JavaScript and XML. Đây là một công nghệ giúp chúng ta tạo ra những Web động mà hoàn toàn không reload lại trang nên rất mượt và đẹp. Vậy Asynchronous, JavaScript, XML trong từ AJAX là gì:

- **Asynchronous**, hay nói ngắn hơn là Async – bất đồng bộ. Bất đồng bộ có nghĩa là một chương trình có thể xử lý không theo tuần tự các hàm. Sẽ không có quy trình, có thể nhảy đi bỏ qua bước nào đó. Ích lợi dễ thấy nhất của bất đồng bộ là chương trình có thể xử lý nhiều công việc một lúc.
- **JavaScript** là một ngôn ngữ lập trình nổi tiếng. Trong số rất nhiều chức năng của nó là khả năng quản lý nội dung động của website và hỗ trợ tương tác với người dùng.

- **XML** là một dạng của ngôn ngữ markup như HTML, chữ đầy đủ của nó là eXtensible Markup Language. Nếu HTML được dùng để hiển thị dữ liệu, XML được thiết kế để chứa dữ liệu.

Ajax có một số phương thức được dùng phổ biến như: `$.get()`, `$.post()` và `$.load()` được sử dụng để tạo các request Ajax chỉ với vài dòng code.

Tuy nhiên để kiểm soát được các Request Ajax được tốt hơn ví dụ như ta muốn chỉ rõ hành động nào sẽ được thực hiện (get, post, put hay delete), xử lý như thế nào trong trường hợp một Request thất bại, ... khi đó ta dùng hàm `$.ajax()` được cung cấp bởi JQuery.

3.6.2. Hàm `$.ajax()`

Hàm `$.ajax()` của JQuery được sử dụng để thực hiện các request HTTP bất đồng bộ. Dưới đây là cú pháp hàm `$.ajax()`

```
$.ajax(url[, options])  
$.ajax([options])
```

Tham số url là một chuỗi chứa URL muốn sử dụng AJAX để thực hiện request, trong khi đó tham số options là một object chứa các thiết lập cho request AJAX đó.

Ở dạng đầu tiên, phương thức này thực hiện một request AJAX sử dụng tham số **url** và các cài đặt được chỉ định ở **options**.

Ở dạng thứ hai, URL được chỉ định trong tham số **options**, hoặc có thể được lược bỏ trong trường hợp request này được gửi đến chính đường dẫn của trang hiện tại.

Một số option trong hàm `$.ajax()`

STT	Option	Ý nghĩa
1	url	Chuỗi chứa URL mà request được gửi đến
2	method	Là phương thức muốn thực thi, ví dụ như: get/post/put/delete
3	content type	Kiểu nội dung của dữ liệu được gửi lên server, ví dụ json/xml
4	success	Một hàm được gọi khi Request thành công

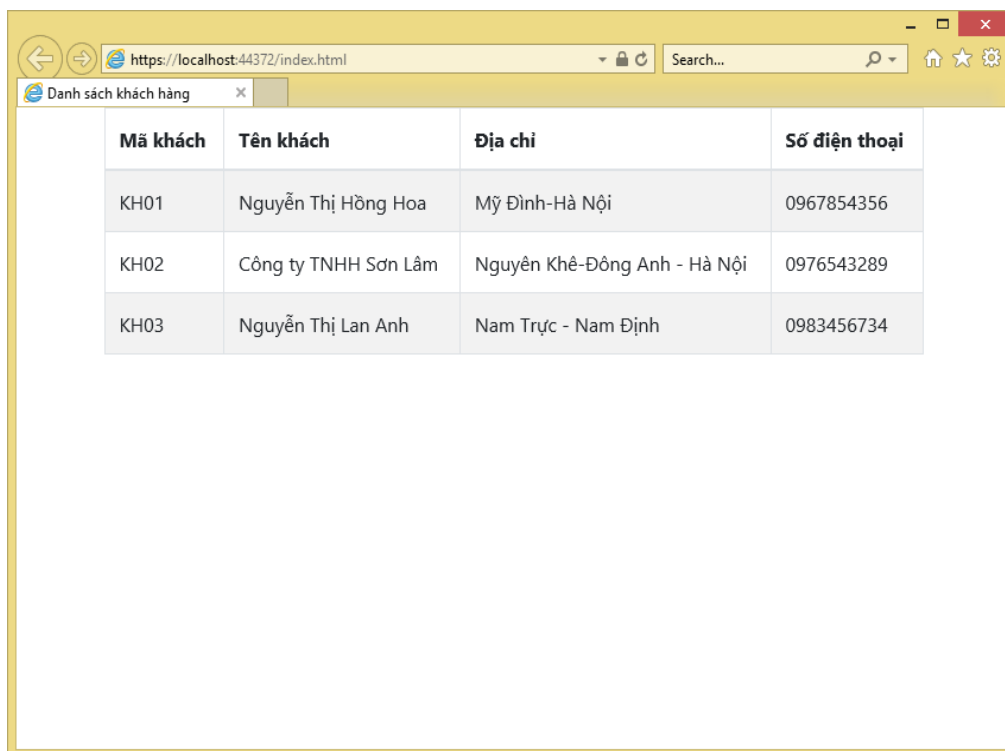
5	fail	Khi kết nối thất bại
6	data	Dữ liệu được gửi lên Server khi thực thi một Request ajax
7	datatype	Kiểu của dữ liệu mong muốn được trả về từ Server, ví dụ: xml/json
8	Accepts	Nội dung được gửi trong request header giúp server biết được kiểu response server sẽ chấp nhận khi trả về.
9	Async	Thiết lập giá trị <code>false</code> để thực hiện một request đồng bộ.
10	Cache	Thiết lập giá trị <code>false</code> để buộc browser không lưu cache các trang được request.
11	beforeSend	Một hàm pre-request gọi lại có thể dùng để điều chỉnh object <code>jqXHR</code> trước khi nó được gửi
12	complete	Một hàm được thực thi khi request kết thúc (sau khi hàm gọi lại <code>success</code> và <code>error</code> được thực thi).
13	Contents	Một object của string hoặc REGEX dùng để xác định xem JQuery sẽ phân tích response như thế nào.
14	Context	Một object được dùng làm ngữ cảnh (<code>this</code>) của tất cả các hàm gọi lại liên quan đến Ajax.
15	crossDomain	Thiết lập thuộc tính này là <code>true</code> để buộc thực hiện request chéo giữa các domain (như là JSONP) trên cùng một domain.
16	dataFilter	Một hàm được dùng để xử lý các dữ liệu response thuần của một XMLHttpRequest.
17	error	Một hàm sẽ được gọi khi request fails.

18	Global	Dùng để thiết lập xem có gọi các hàm xử lý sự kiện Ajax toàn cục cho request này hay không.
19	Headers	Một object để viết thêm vào các header gửi lên server.
20	ifModified	Thiết lập giá trị này là <code>true</code> nếu bạn muốn buộc JQuery nhận diện môi trường hiện tại là " local ".
21	Jsonp	Một chuỗi dùng để override tên hàm gọi lại trong một request JSONP.
22	jsonpCallback	Chỉ định tên hàm gọi lại cho một request JSONP.
23	mimeType	Một chuỗi chỉ định kiểu mime dùng để override lại kiểu mime của XHR.
24	Password	Mật khẩu được sử dụng với XMLHttpRequest cho response của một request yêu cầu xác thực truy nhập HTTP.
25	processData	Set giá trị này là <code>false</code> nếu bạn không muốn dữ liệu được truyền vào thiết lập <code>data</code> sẽ được xử lý và biến thành một query kiểu chuỗi.
26	scriptCharset	Thiết lập thuộc tính charset của một thẻ script dùng cho một request nhưng chỉ áp dụng khi transport script (ví dụ: request chéo giữa các domain với jsonp) được sử dụng.
27	statusCode	Một object chứa các mã HTTP ở dạng số và các hàm được gọi khi response trả về có chứa một mã tương ứng.
28	timeOut	Số được thiết lập chỉ định thời gian hết hạn cho một request.
29	traditional	Thiết lập giá trị <code>true</code> nếu bạn mong muốn param được serialize theo kiểu truyền thống.

30	username	Tên người dùng được sử dụng với XMLHttpRequest cho response của một request yêu cầu xác thực truy nhập HTTP.
31	Xhr	Một hàm gọi lại dùng để tạo một object XMLHttpRequest.
32	xhrFields	Một object các key-value được thiết lập cho object XHR native

3.6.3. Ví dụ dùng \$.ajax gọi đến WEB API dạng HTTP GET

Ở các ví dụ dưới đây, chúng ta sẽ sử dụng 5 WEB Services đã xây dựng ở mục 3.5. Ở ví dụ này ta sẽ dùng \$.jax() để gọi đến dịch vụ lấy danh sách tất cả các khách hàng trình bày như giao diện sau:



The screenshot shows a web browser window with the address bar displaying 'https://localhost:44372/index.html'. The page title is 'Danh sách khách hàng'. The main content area contains a table with the following data:

Mã khách	Tên khách	Địa chỉ	Số điện thoại
KH01	Nguyễn Thị Hồng Hoa	Mỹ Đình-Hà Nội	0967854356
KH02	Công ty TNHH Sơn Lâm	Nguyễn Khê-Đông Anh - Hà Nội	0976543289
KH03	Nguyễn Thị Lan Anh	Nam Trực - Nam Định	0983456734

Lưu ý: Ở đây để thuận tiện trong việc thiết kế giao diện và lấy dữ liệu tôi có sử dụng các thư viện bootstrap và JQuery.

Hướng dẫn:

Bước 1: Tạo Project dạng Website trên Visual studio 2019.

Bước 2: Thêm file index.html (đây là file dùng để viết giao diện và dùng ajax để đọc WEB API)

Bước 3: Cài đặt thêm Bootstrap vào project (khi cài Bootstrap thì JQuery cũng được tự động add vào, ta có thể nâng cấp JQuery lên version cao hơn. Nếu chưa có JQuery ta có thể chèn thêm vào).

Bước 4: Trong cặp thẻ <header> </header> của file index.html, ta thêm các dòng lệnh để khai báo sử dụng Bootstrap và JQuery như sau:

```
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width" />
<link rel="stylesheet" href="Content/bootstrap.min.css" />
<script type="text/javascript" src="Scripts/jquery-3.6.0.min.js"></script>
<script src="Scripts/bootstrap.min.js"></script>
```

Bước 5: Trong cặp thẻ <body></body> thêm đoạn HTML định nghĩa các thẻ chứa dữ liệu lấy ra và thẻ <script> để lấy dữ liệu như sau:

```
<div class="container">
  <table id="tblKhachHang" class="table table-striped table-bordered">
    <thead>
      <tr>
        <th>Mã khách</th>
        <th>Tên khách</th>
        <th>Địa chỉ</th>
        <th>Số điện thoại</th>
      </tr>
    </thead>
    <tbody id="allKH">
    </tbody>
  </table>
</div>
```

Đoạn HTML trên ta để ý thẻ <tbody id="allKH"> bên trong thẻ <table> là nơi chứa dữ liệu Khách hàng.

Đoạn <script> lấy dữ liệu được viết như sau:

```
<script type="text/javascript">
  $(document).ready(function () {
    GetAllCustomers();
  });
  //Hàm lấy ra toàn bộ danh sách Khách Hàng. Dùng $.ajax() thực hiện
  method HTTP GET
  function GetAllCustomers() {
    $.ajax({
      url: 'https://localhost:44373/api/customers/',
      method: 'GET',
      contentType: 'json',
      dataType: 'json',
      error: function (response) {

      },
    },
```

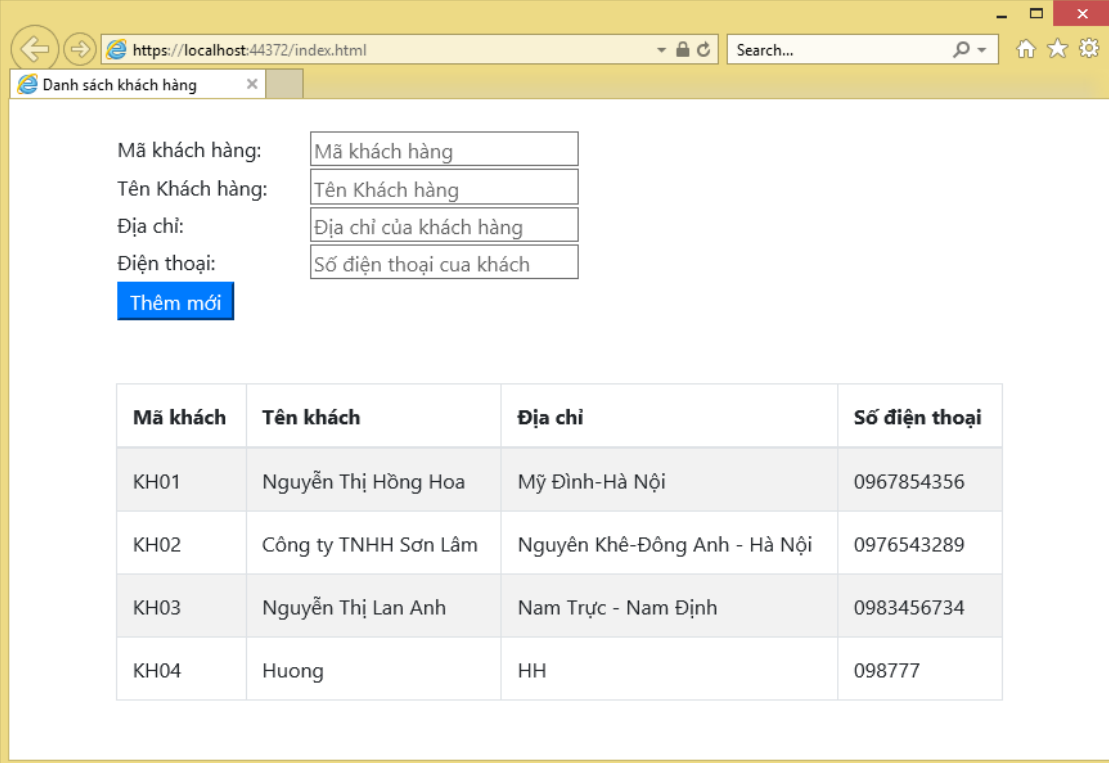
```
success: function (reponse) {
    const len = reponse.length;
    console.log(reponse);
    let table = '';
    for (var i = 0; i < len; ++i) {
        table = table + '<tr>';
        table = table + '<td>' + reponse[i].MaKH + '</td>';
        table = table + '<td>' + reponse[i].TenKH + '</td>';
        table = table + '<td>' + reponse[i].DiaChi + '</td>';
        table = table + '<td>' + reponse[i].DienThoai + '</td>';
        table = table + '</tr>';
    }
    document.getElementById('allKH').innerHTML = table;
},
fail: function (response) {
}
});
}
</script>
```

Chạy thử chương trình ta thu được giao diện như yêu cầu.

Chú ý: Khi chạy cần đảm bảo là server chứa Web API phải đang được chạy.

3.6.3. Ví dụ dùng \$.ajax gọi đến WEB API dạng HTTP POST

Xây dựng giao diện gọi đến một web api sử dụng giao thức HTTP POST để thêm mới một Khách hàng. Giao diện gợi ý như sau:



The screenshot shows a web browser window with the address `https://localhost:44372/index.html`. The page title is "Danh sách khách hàng". It contains a form with four input fields: "Mã khách hàng", "Tên Khách hàng", "Địa chỉ", and "Điện thoại", each with a placeholder text. Below the form is a blue button labeled "Thêm mới". Below the form is a table with four columns: "Mã khách", "Tên khách", "Địa chỉ", and "Số điện thoại". The table contains four rows of customer data.

Mã khách	Tên khách	Địa chỉ	Số điện thoại
KH01	Nguyễn Thị Hồng Hoa	Mỹ Đình-Hà Nội	0967854356
KH02	Công ty TNHH Sơn Lâm	Nguyễn Khê-Đông Anh - Hà Nội	0976543289
KH03	Nguyễn Thị Lan Anh	Nam Trục - Nam Định	0983456734
KH04	Huong	HH	098777

Khi người dùng nhập đủ thông tin về Mã khách hàng, tên khách hàng, Địa chỉ và số điện thoại của khách thì chương trình sẽ gọi đến URL chứa web api thực hiện thêm mới khách hàng và khách hàng đó được thêm vào CSDL đồng thời hiển thị luôn trên bảng danh sách Khách hàng phía dưới.

Hướng dẫn

Tiếp tục sử dụng project của phần 3.6.2, ta bổ sung thêm vào thẻ <body> đoạn HTML định nghĩa các thẻ nhập liệu và nút Thêm mới như sau:

```
<div class="container">
  <table style="border:none">
    <tbody>
      <tr>
        <td style="width:150px">Mã khách hàng: </td>
        <td>
          <input id="txtMK" type="text" name="MaKhach"
placeholder="Mã khách hàng" />
        </td>
      </tr>

      <tr>
        <td style="width:30px">Tên Khách hàng: </td>
        <td>
          <input type="text" name="TenKhach" placeholder="Tên
Khách hàng" />
        </td>
      </tr>

      <tr>
        <td>Địa chỉ: </td>
        <td>
          <input type="text" name="DiaChiKhach"
placeholder="Địa chỉ của khách hàng" />
        </td>
      </tr>

      <tr>
        <td>Điện thoại: </td>
        <td>
          <input id="txtDT" type="text" name="DienThoaiKhach"
placeholder="Số điện thoại của khách" />
        </td>
      </tr>

      <tr>
```

```
        <td>        <button        class="btn-primary        update-button"
onclick="insertKhachHang();" >Thêm mới</button> </td>
    </tr>
</tbody>
</table>
</div>
```

Trong đó nút **Thêm mới** có thuộc tính `onclick="insertKhachHang();"`. Hàm `insertKhachHang()` được viết trong thẻ `<script>` như sau:

```
<script type="text/javascript">
    //Hàm lấy Thêm mới Khách hàng. Dùng $.ajax() thực hiện method HTTP
    POST
    function insertKhachHang() {
        var url = 'https://localhost:44373/api/customers?id=' +
        $('input').eq(0).val() + '&name=' + $('input').eq(1).val() +
        '&address=' + $('input').eq(2).val() + '&phoneNumber=' +
        $('input').eq(3).val();
        $.ajax({
            url: url,
            method: 'POST',
            contentType: 'json',
            dataType: 'json',
            error: function (response) {
                alert("Thêm mới không thành công");
            },
            success: function (reponse) {
                alert("Thêm mới thành công");
                GetAllCustomers(); //Gọi đến hàm lấy dữ liệu lên bảng
            }
        });
    }
</script>
```

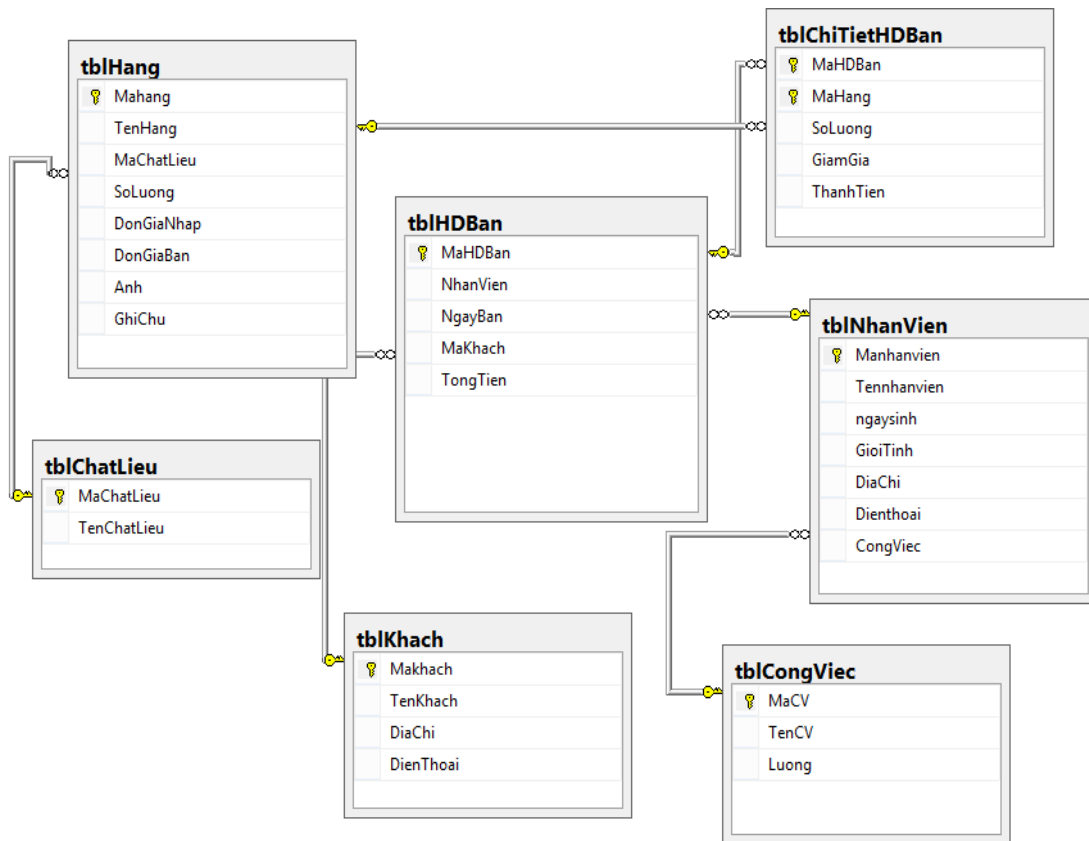
Trong đoạn mã trên hàm `$.ajax()` thực hiện một WEB API được chỉ định trong tùy chọn url và thực hiện phương thức HTTP POST được chỉ định trong tùy chọn method.

Cách gọi WEB API dạng HTTP PUT và HTTP DELETE cũng được thực hiện tương tự.

3.7. Bài tập chương 3

Bài 1.

Giả sử trên Server có một cơ sở dữ liệu có tên là *Dulieu* như sau:



Xây dựng Web API tuân theo chuẩn Rest để cung cấp các dịch vụ như sau:

1. Tra cứu giá bán của một sản phẩm bất kỳ.
2. Thống kê doanh thu của một sản phẩm bất kỳ trong tháng cụ thể.
3. Lấy ra danh sách hàng tồn.
4. Lấy ra top 5 khách hàng lớn trong năm.
5. Thêm mới một nhân viên
6. Sửa thông tin một nhân viên khi biết mã nhân viên
7. Xóa thông tin một nhân viên khi biết mã nhân viên
8. Lấy ra danh sách toàn bộ nhân viên
9. Sửa thông tin giá bán của một mặt hàng.
10. Lấy ra nhân viên có doanh thu lớn nhất tháng.
11. Tra cứu danh sách các mặt hàng có chất liệu cụ thể

Bài 2. Dùng Postman để kiểm tra sự vận hành của các Web API đã được xây dựng ở **Bài 1.**

Bài 3: Xây dựng giao diện web để thực thi các Web API số 5,6,7,8 ở **Bài 1.**