**Tìm hiểu về Web Service**

**https://gpcoder.com/5572-tim-hieu-ve-web-service/**

Trong bài này, chúng ta sẽ cùng tìm hiểu về Web service là gì, các thành phần của một web service, các loại web service, so sánh SOAP với REST web service.

Bài này khá nặng về lý thuyết, mình xin phép tổng hợp lại từ các tài liệu mình tham khảo được từ các trang khác cũng như kinh nghiệm thực tế của mình để giúp các bạn có cái nhìn đầy đủ nhất về web service.

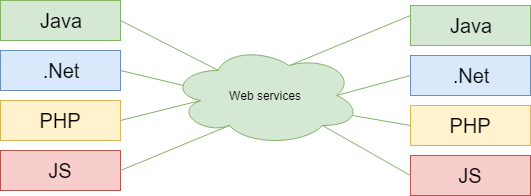
**Nội dung**[[Ẩn](https://gpcoder.com/5572-tim-hieu-ve-web-service/)]

* [1 Web service – Dịch vụ web là gì?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Web_service_8211_Dich_vu_web_la_gi)
* [2 Các thành phần của web service](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Cac_thanh_phan_cua_web_service)
  + [2.1 XML – eXtensible Markup Language](https://gpcoder.com/5572-tim-hieu-ve-web-service/#XML_eXtensible_Markup_Language)
  + [2.2 WSDL – Web Service Description Language](https://gpcoder.com/5572-tim-hieu-ve-web-service/#WSDL_Web_Service_Description_Language)
  + [2.3 Universal Description, Discovery, and Integration (UDDI)](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Universal_Description_Discovery_and_Integration_UDDI)
  + [2.4 SOAP – Simple Object Access Protocol](https://gpcoder.com/5572-tim-hieu-ve-web-service/#SOAP_Simple_Object_Access_Protocol)
* [3 Web Services Architecture](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Web_Services_Architecture)
* [4 Web service hoạt động như thế nào?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Web_service_hoat_dong_nhu_the_nao)
* [5 Các loại Web service](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Cac_loai_Web_service)
  + [5.1 SOAP Web Service là gì?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#SOAP_Web_Service_la_gi)
  + [5.2 RESTful web service là gì?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#RESTful_web_service_la_gi)
    - [5.2.1 Sử dụng các phương thức HTTP một cách rõ ràng](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Su_dung_cac_phuong_thuc_HTTP_mot_cach_ro_rang)
    - [5.2.2 Phi trạng thái (Stateless)](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Phi_trang_thai_Stateless)
    - [5.2.3 Đưa ra cấu trúc thư mục giống các URI](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Dua_ra_cau_truc_thu_muc_giong_cac_URI)
    - [5.2.4 Truyền tải XML, JSON hoặc cả hai](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Truyen_tai_XML_JSON_hoac_ca_hai)
  + [5.3 Sự khác nhau giữa REST và SOAP](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Su_khac_nhau_giua_REST_va_SOAP)
  + [5.4 Lý do sử dụng REST thay vì SOAP?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Ly_do_su_dung_REST_thay_vi_SOAP)
  + [5.5 Lý do sử dụng SOAP?](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Ly_do_su_dung_SOAP)
* [6 Một số bài viết về Web service](https://gpcoder.com/5572-tim-hieu-ve-web-service/#Mot_so_bai_viet_ve_Web_service)

**Web service – Dịch vụ web là gì?**

Web service (dịch vụ web) là tập hợp các giao thức và tiêu chuẩn mở được sử dụng để trao đổi dữ liệu giữa các ứng dụng hoặc giữa các hệ thống.

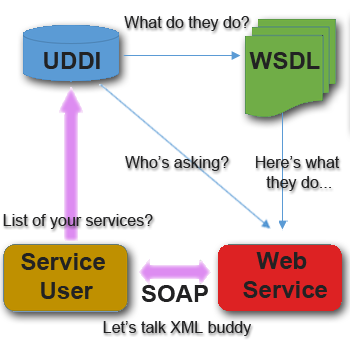
Các ứng dụng phần mềm được viết bằng các ngôn ngữ lập trình khác nhau hoặc chạy trên các nền tảng khác nhau, chúng có thể sử dụng các web service để trao đổi dữ liệu qua lại theo cách tương tự như liên lạc giữa các quá trình trên một máy tính.

[](https://gpcoder.com/wp-content/uploads/2019/05/Web-service.png)

**Các thành phần của web service**

Nền tảng cơ bản của web service (WS) là XML + HTTP. Nó bao gồm các thành phần chính sau:

* **UDDI** – Universal Description, Discovery, and Integration (Mô tả, Khám phá và Tích hợp Toàn cầu): UDDI là một tiêu chuẩn dựa trên XML để mô tả, xuất bản và tìm kiếm các dịch vụ web.
* **WSDL** – Web Service Description Language (Ngôn ngữ mô tả web service): WSDL là một ngôn ngữ dựa trên XML để mô tả các dịch vụ web và cách truy cập chúng. WSDL mô tả một dịch vụ web, cùng với định dạng thông báo và các chi tiết giao thức cho dịch vụ web.
* **SOAP** – Simple Object Access Protocol (Giao thức truy cập đối tượng đơn giản): SOAP là một giao thức dựa trên XML đơn giản cho phép các ứng dụng trao đổi thông tin qua HTTP.

[](https://gpcoder.com/wp-content/uploads/2019/05/web-service-diagram.png)

**XML – eXtensible Markup Language**

Là một chuẩn mở do W3C đưa ra cho cách thức mô tả dữ liệu, nó được sử dụng để định nghĩa các thành phần dữ liệu trên trang web và cho những tài liệu B2B. Về hình thức, XML hoàn toàn có cấu trúc thẻ giống như ngôn ngữ HTML nhưng **HTML định nghĩa thành phần được hiển thị như thế nào thì XML lại định nghĩa những thành phần đó chứa cái gì**. Với XML, các thẻ có thể được lập trình viên tự tạo ra trên mỗi trang web và được chọn là định dạng thông điệp chuẩn bởi tính phổ biến và hiệu quả mã nguồn mở.

Do dịch vụ Web là sự kết hợp của nhiều thành phần khác nhau nên nó sử dụng các tính năng và đặc trưng của các thành phần đó để giao tiếp. XML là công cụ chính để giải quyết vấn đề này và là kiến trúc nền tảng cho việc xây dựng một dịch vụ Web, tất cả dữ liệu sẽ được chuyển sang định dạng thẻ XML. Khi đó, các thông tin mã hóa sẽ hoàn toàn phù hợp với các thông tin theo chuẩn của SOAP hoặc XML-RPC và có thể tương tác với nhau trong một thể thống nhất.

**WSDL – Web Service Description Language**

WSDL định nghĩa cách mô tả dịch vụ Web theo cú pháp tổng quát của XML, bao gồm các thông tin:

* Tên dịch vụ.
* Giao thức và kiểu mã hóa sẽ được sử dụng khi gọi các hàm của dịch vụ Web.
* Loại thông tin: thao tác, tham số, những kiểu dữ liệu (có thể là giao diện của dịch vụ Web cộng với tên cho giao diện này).

Một WSDL hợp lệ gồm hai phần: *phần giao diện* mô tả giao diện và phương thức kết nối và *phần thi hành* mô tả thông tin truy xuất CSDL. Cả hai phần này sẽ được lưu trong 2 tập tin XML tương ứng là tập tin giao diện dịch vụ và tập tin thi hành dịch vụ. Giao diện của một dịch vụ Web được miêu tả trong phần này đưa ra cách thức làm thế nào để giao tiếp qua dịch vụ Web. Tên, giao thức liên kết và định dạng thông điệp yêu cầu để tương tác với dịch vụ Web được đưa vào thư mục của WSDL.

WSDL thường được sử dụng kết hợp với XML schema và SOAP để cung cấp dịch vụ Web qua Internet. Một client khi kết nối tới dịch vụ Web có thể đọc WSDL để xác định những chức năng sẵn có trên server. Sau đó, client có thể sử dụng SOAP để lấy ra chức năng chính xác có trong WSDL.

**Universal Description, Discovery, and Integration (UDDI)**

Để có thể sử dụng các dịch vụ, trước tiên client phải tìm dịch vụ, ghi nhận thông tin về cách sử dụng và biết được đối tượng nào cung cấp dịch vụ. UDDI định nghĩa một số thành phần cho biết các thông tin này, cho phép các client truy tìm và nhận những thông tin được yêu cầu khi sử dụng dịch vụ Web.

Cấu trúc UDDI :

* Trang trắng – White pages: chứa thông tin liên hệ và các định dạng chính yếu của dịch vụ Web, chẳng hạn tên giao dịch, địa chỉ, thông tin nhận dạng… Những thông tin này cho phép các đối tượng khác xác định được dịch vụ.
* Trang vàng – Yellow pages: chứa thông tin mô tả dịch vụ Web theo những loại khác nhau. Những thông tin này cho phép các đối tượng thấy được dịch vụ Web theo từng loại với nó.
* Trang xanh – Green pages: chứa thông tin kỹ thuật mô tả các hành vi và các chức năng của dịch vụ Web.
* Loại dịch vụ – tModel:  chứa các thông tin về loại dịch vụ được sử dụng.

Những thông tin về dịch vụ Web được sử dụng và công bố lên mạng sử dụng giao thức này. Nó sẽ kích hoạt các ứng dụng để tìm kiếm thông tin của dịch vụ Web khác nhằm xác định xem dịch vụ nào sẽ cần đến nó.

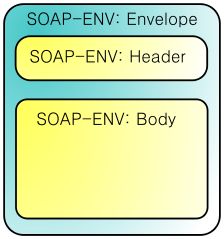
**SOAP – Simple Object Access Protocol**

Chúng ta đã hiểu cơ bản dịch vụ Web như thế nào nhưng vẫn còn một vấn đề khá quan trọng. Đó là làm thế nào để truy xuất dịch vụ khi đã tìm thấy? Câu trả lời là các dịch vụ Web có thể truy xuất bằng một giao thức là Simple Object Access Protocol – SOAP. Nói cách khác chúng ta có thể truy xuất đến UDDI registry bằng các lệnh gọi hoàn toàn theo định dạng của SOAP.

SOAP là một giao thức giao tiếp có cấu trúc như XML. Nó được xem là cấu trúc xương sống của các ứng dụng phân tán được xây dựng từ nhiều ngôn ngữ và các hệ điều hành khác nhau. SOAP là giao thức thay đổi các thông điệp dựa trên XML qua mạng máy tính, thông thường sử dụng giao thức HTTP.

Một client sẽ gửi thông điệp yêu cầu tới server và ngay lập tức server sẽ gửi những thông điệp trả lời tới client. Cả SMTP và HTTP đều là những giao thức ở lớp ứng dụng của SOAP nhưng HTTP được sử dụng và chấp nhận rộng rãi hơn bởi ngày nay nó có thể làm việc rất tốt với cơ sở hạ tầng Internet.

**Cấu trúc một thông điệp theo dạng SOAP**

[](https://gpcoder.com/wp-content/uploads/2019/05/6-soap-structure.jpg)

Thông điệp theo định dạng SOAP là một văn bản XML bình thường bao gồm các phần tử sau:

* Phần tử gốc – **envelop**: phần tử bao trùm nội dung thông điệp, khai báo văn bản XML như là một thông điệp SOAP.
* Phần tử đầu trang – **header**: chứa các thông tin tiêu đề cho trang, phần tử này không bắt buộc khai báo trong văn bản. Header còn có thể mang những dữ liệu chứng thực, những chứ ký số, thông tin mã hóa hay cài đặt cho các giao dịch khác.
* Phần tử khai báo nội dung chính trong thông điệp – **body**, chứa các thông tin yêu cầu và thông tin được phản hồi.
* Phần tử đưa ra các thông tin về lỗi – **fault**, cung cấp thông tin lỗi xảy ra trong qúa trình xử lý thông điệp.

Một SOAP đơn giản trong body sẽ lưu các thông tin về tên thông điệp, tham chiếu tới một thể hiện của dịch vụ, một hoặc nhiều tham số. Có 3 kiểu thông báo sẽ được đưa ra khi truyền thông tin: request message(tham số gọi thực thi một thông điệp), respond message (các tham số trả về, được sử dụng khi yêu cầu được đáp ứng) và cuối cùng là fault message (thông báo tình trạng lỗi).

Ví dụ:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?xml version="1.0"?>  <soap:Envelope xmlns:soap="<http://www.w3.org/2001/12/soap-envelope>" soap:encodingStyle="<http://www.w3.org/2001/12/soap-encoding>">    <soap:Header>  ...  </soap:Header>    <soap:Body>  ...    <soap:Fault>    ...    </soap:Fault>  </soap:Body>    </soap:Envelope> |

**Kiểu truyền thông**

Có 2 kiểu truyền thông

* Remote procedure call (RPC): cho phép gọi hàm hoặc thủ tục qua mạng. Kiểu này được khai thác bởi nhiều dịch vụ Web.
* Document: được biết đến như kiểu hướng thông điệp, nó cung cấp giao tiếp ở mức trừu tượng thấp, khó hiểu và yêu cầu lập trình viên mất công sức hơn.

Hai kiểu truyền thông này cung cấp các định dạng thông điệp, tham số, lời gọi đến các API khác nhau nên việc sử dụng chúng tùy thuộc vào thời gian và sự phù hợp với dịch vụ Web cần xây dựng.

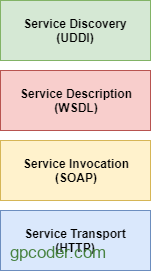
**Cấu trúc dữ liệu**

Cung cấp những định dạng và khái niệm cơ bản giống như trong các ngôn ngữ lập trình khác như kiểu dữ liệu (int, string, date…) hay những kiều phức tạp hơn như struct, array, vector… Định nghĩa cấu trúc dữ liệu SOAP được đặt trong namespace SOAP-ENC.

**Mã hóa**

Giả sử service requester và service provider được phát triển trong Java, khi đó mã hóa SOAP là làm thế nào chuyển đổi từ cấu trúc dữ liệu Java sang SOAP XML và ngược lại, bởi vì định dạng cho Web Service chính là XML. Bất kỳ một môi trường thực thi SOAP nào cũng phải có một bảng chứa thông tin ánh xạ nhằm chuyển đổi từ ngôn ngữ Java sang XML và từ XML sang Java – bảng đó được gọi là SOAPMappingRegistry. Nếu một kiểu dữ liệu được sử dụng dưới một dạng mã hóa thì sẽ có một ánh xạ tồn tại trong bộ đăng ký của môi trường thực thi SOAP đó.

**Web Services Architecture**

[](https://gpcoder.com/wp-content/uploads/2019/05/Web-Services-Architecture-1.png)

* **Service Discovery** : Phần kiến ​​trúc này chịu trách nhiệm tập trung các dịch vụ vào một nơi đăng ký chung và cung cấp chức năng publish/ search dễ dàng. Điều này được xử lý bởi **UDDI**.
* **Service Description** : Một trong những tính năng thú vị nhất của Dịch vụ web là chúng tự mô tả. Điều này có nghĩa là, một khi Dịch vụ web được định vị, nó sẽ cho chúng ta biết những hoạt động mà nó hỗ trợ và cách gọi nó. Điều này được xử lý bởi **WSDL**.
* **Service Invocation** : Gọi một dịch vụ web liên quan đến việc truyền tin nhắn giữa Client và Server. **SOAP** chỉ định cách chúng ta nên định dạng các thông báo yêu cầu (request) đến Server và cách Server định dạng các thông điệp phản hồi (response) của nó.
* **Service Transport** : Cuối cùng, tất cả các thông báo này phải được truyền đi bằng cách nào đó giữa Client và Server. Giao thức được lựa chọn cho phần kiến ​​trúc này là **HTTP** – giao thức được sử dụng để truy cập các trang web thông thường trên Internet. Chúng ta cũng có thể sử dụng các giao thức khác, nhưng HTTP hiện là giao thức được sử dụng nhiều nhất.

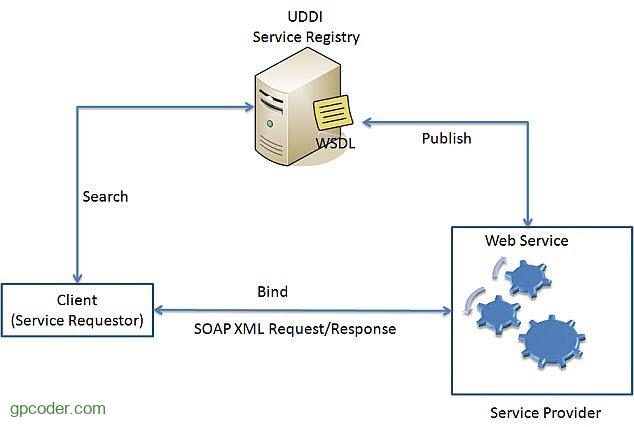
**Web service hoạt động như thế nào?**

Một ứng dụng WS bao gồm 2 thành phần: Client và Server giao tiếp với nhau qua giao thức HTTP.

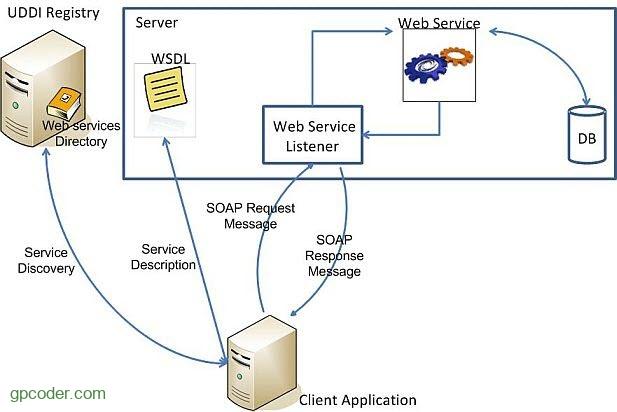
* Client gửi yêu cầu qua các lời gọi hàm thông qua HTTP Request đến Server
* Server gửi các kết quả được thực thi các ở hàm thông qua HTTP Request

Mô hình hoạt động của ứng dụng WebService gồm 3 thành phần chính:

* **UDDI service registry**: Công cụ giúp nhà phát triển WS công bố những thông tin về WebService của mình cho cộng đồng các nhà phát triển ứng dụng. Người dùng sẽ dựa vào các thông tin này để sử dụng WebService trong ứng dụng riêng của minh.
* **Web service**: Chứa giao thức SOAP định dạng dữ liệu, tài liệu WSDL định nghĩa các hàm trong WebService, XML để xây dựng ứng dụng phân tán.
* **Applicantion Client**: Ứng dụng phía Client sử dụng WebService xây dựng riêng cho mình  
  Cách thức hoạt động có thể mô tả như sau: Đầu tiên, Applicantion Client cần truy vấn các mẫu tin.

[](https://gpcoder.com/wp-content/uploads/2019/05/5-ws-platform.jpg)

UDDI theo 1 thông tin nào đó (chẳng hạn tên loại) để xác định WebService cần tìm. Khi đã xác định được WebService cần cho ứng dụng, Client có thế lấy thông tin về địa chỉ của tài liệu WSDL của WebService này dựa trên mẫu tin UDDI. Tài liệu WSDL sẽ mô tả cách thức liên lạc với WebService, định dạng gói tin truy vấn và phản hồi. Dựa vào những thông tin này, Client có thể tạo những gói tin SOAP tương ứng để liên lạc với Service.

[](https://gpcoder.com/wp-content/uploads/2019/05/3-ws-working.jpg)

**Các loại Web service**

Có 2 loại web service chính:

* SOAP web services.
* RESTful web services.

**SOAP Web Service là gì?**

SOAP là viết tắt của **S**imple **O**bject **A**ccess **P**rotocol.

SOAP (Simple Object Access Protocol) là giao thức sử dụng **XML** để định nghĩa dữ liệu dạng thuần văn bản (plain text) thông qua HTTP. SOAP là cách mà Web Service sử dụng để truyền tải dữ liệu. Vì dựa trên XML nên SOAP là một giao thức không phụ thuộc platform cũng như bất kì ngôn ngữ lập trình nào. Chúng ta có thể viết bằng Java, PHP, .NET, … và triển khai trên Window, Linux, …

**RESTful web service là gì?**

REST là viết tắt của **RE**presentational **S**tate **T**ransfer.

REST là một loại kiến trúc phần mềm (architectural style), không phải là một protocol.

RESTful Web Service là các Web Service được viết dựa trên kiến trúc REST. REST đã được sử dụng rộng rãi để thay thế cho các Web Service dựa trên SOAP và WSDL.

Tương tự SOAP, REST cũng không phụ thuộc platform cũng như bất kì ngôn ngữ lập trình nào.

REST có thể sử dụng SOAP web service như một implement của nó.

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 truyền tải qua HTTP, và được viết bởi nhiều ngôn ngữ khác nhau. Nếu tính theo số dịch vụ mạng sử dụng, REST đã nổi lên trong vài năm qua như là một mô hình thiết kế dịch vụ chiếm ưu thế. Trong thực tế, REST đã có những ảnh hưởng lớn và gần như thay thế SOAP và WSDL vì nó đơn giản và dễ sử dụng hơn rất nhiều.

REST tuân thủ 4 nguyên tắc thiết kế cơ bản sau:

* Sử dụng các phương thức HTTP một cách rõ ràng.
* Phi trạng thái.
* Hiển thị cấu trúc thư mục như các URls.
* Truyền tải JavaScript Object Notation (JSON), XML hoặc cả hai.

**Sử dụng các phương thức HTTP một cách rõ ràng**

REST đặt ra một quy tắc đòi hỏi lập trình viên xác định rõ hành động của service thông qua các phương thức của HTTP.

Các hành động của một webservice thông thường bao gồm tạo dữ liệu (**C**reate), lấy dữ liệu (**R**ead),cập nhập dữ liệu (**U**pdate) hoặc xóa dữ liệu (**D**elete). Các hành động này còn được gọi là **CRUD**.

Thiết lập một ánh xạ 1-1 giữa các hành động (CRUD) và các phương thức **HTTP**:

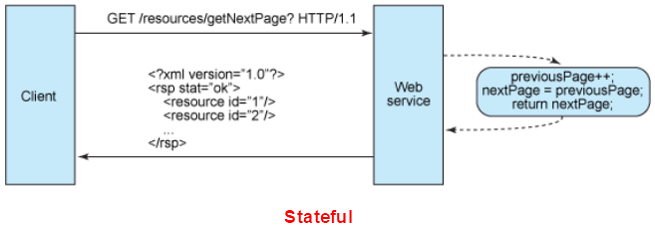
* POST : để tạo một tài nguyên trên Server.
* GET : để truy xuất một tài nguyên.
* PUT : để thay đổi trạng thái một tài nguyên hoặc để cập nhật nó.
* DELETE : để huỷ bỏ hoặc xoá một tài nguyên.

Lưu ý: các nguyên tắc ở trên là không bắt buộc, trên thực tế chúng ta có thể sử dụng phương thức GET để yêu cầu lấy, tạo, sửa hoặc xóa dữ liệu trên Server. Tuy nhiên, REST đưa ra các nguyên tắc ở trên mục đích đưa mọi thứ trở lên rõ ràng và dễ hiểu.

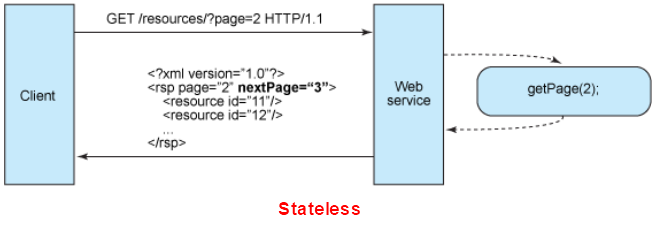
**Phi trạng thái (Stateless)**

Một đặc điểm của REST là phi trạng thái (stateless), có nghĩa là nó không lưu giữ thông tin của client. Chẳng hạn 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).

Hình dưới đây minh họa một ứng dụng có lưu trữ trạng thái, nó biết người dùng đang xem trang số mấy. Và người dùng chỉ cần yêu cầu “Trang Tiếp theo” để nhận được trang mong muốn.

[](https://gpcoder.com/wp-content/uploads/2019/05/ws-stateful.png)

Với các thiết kế phi trạng thái, Client phải gửi yêu cầu rõ ràng, bao gồm số thự tự của trang cần xem.

[](https://gpcoder.com/wp-content/uploads/2019/05/ws-stateless.png)

Như vậy, các thành phần Server phi trạng thái ít phức tạp hơn để thiết kế, viết và phân bổ thông qua Server được cân bằng tải. 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 Client. Trong một dịch vụ mạng RESTful, Server chịu trách nhiệm đưa ra các phản hồi và cung cấp một cách thức cho phép Client duy trì trạng thái ứng dụng của chính nó.

**Đưa ra cấu trúc thư mục giống các URI**

REST đưa ra một cấu trúc để người dùng có thể truy cập vào tài nguyên của nó thông qua các URL, tài nguyên ở đây là tất cả những cái mà bạn có thể gọi tên được (Video, ảnh, báo cáo thời tiết,..)  
Bạn cần tạo ra các REST serivce để nó trả về cho người dùng các nguồn tài nguyên tương ứng.

Các địa chỉ REST service cần phải thật trực quan đến mức người dùng dễ đoán. Hãy nghĩ về một địa chỉ (URI) giống như một gợi ý rõ ràng, dễ đoán rằng nó đang trỏ tới cái gì và cung cấp tài nguyên gì. Tóm lại, cấu trúc của một URI nên được đơn giản, có thể dự đoán, và dễ hiểu.

Hãy xem một URL dưới đây, nó cung cấp danh sách bài viết của một ngày cụ thể, và nó dễ hiểu đối với người dùng.

|  |  |
| --- | --- |
| 1 | https://gpcoder.com/posts/2019-05-20 |

Một vài nguyên tắc bổ sung để lưu ý trong khi nói về cấu trúc địa chỉ của RESTful Web service 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 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.

**Truyền tải XML, JSON 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.

Đôi khi Client cũng có thể chỉ định kiểu dữ liệu nhận về mà nó mong muốn (JSON, hoặc XML,..), các chỉ định này được gọi là các kiểu MINE, nó được gửi kèm trên phần HEADER của request.

Dưới đây là các kiểu MIME phổ biến thường sử dụng với REST service:

|  |  |
| --- | --- |
| Extention | Content-Type |
| .json | application/json |
| .xml | application/xml |

Tham khảo thêm các MIME type khác: <https://www.freeformatter.com/mime-types-list.html>

Ví dụ: Client gửi một yêu cầu để lấy thông tin danh sách bài viết, và yêu cầu dữ liệu trả về là định dạng XML.

|  |  |
| --- | --- |
| 1  2  3 | GET https://gpcoder.com/posts  authority: gpcoder.com  Accept: application/xml;q=0.9 |

Và dữ liệu nhận được:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <posts>      <post>          <date>2019/05/20</date>          <title>Java Web service tutorial</title>          <content>...</content>      </post>      <post>          <date>2019/05/25</date>          <title>SOAP Web Service</title>          <content>...</content>      </post>      <post>          <date>2019/05/30</date>          <title>RESTful Web Service</title>          <content>...</content>      </post>  <posts> |

**Sự khác nhau giữa REST và SOAP**

| **#** | **TIÊU CHÍ** | **SOAP** | **REST** |
| --- | --- | --- | --- |
| 1 | Viết tắt | SOAP là viết tắt của Simple Object Access Protocol (Giao thức truy cập đối tượng đơn giản) | REST là viết tắt của REpresentational State Transfer (Chuyển giao trạng thái phản hồi) |
| 2 | Kiến trúc và giao thức | Một giao thức gửi nhận message có dạng XML. | Một loại kiến trúc bao gồm các quy tắc để thao tác với server. |
| 3 | Các dạng format được support | SOAP là một chuẩn được tạo ra để chuẩn hóa giao tiếp giữa client và server về format, structure và method. Sử dụng WSDL để giao tiếp giữa máy chủ và máy khách. SOAP sử dụng các message dạng XML để giao tiếp với server. | Sử dụng XML hoặc JSON để gửi nhận dữ liệu. |
| 4 | Tài nguyên | SOAP sử dụng các message dạng XML để xác định các thủ tục hay tài nguyên yêu cầu. Gọi các service thông qua method RPC. | RESTful sử dụng **URL** để xác định tài nguyên mong muốn được truy cập. |
| 5 | Kết quả trả về | Không dễ đọc. | Dễ đọc vì đơn giản chỉ là text XML hoặc JSON. |
| 6 | Sử dụng giao thức HTTP | Có thể truyền qua nhiều giao thức khác nhau như HTTP, SMTP, FTP,… Tuy nhiên trong thực tết chúng ta cũng sử dụng giao thức HTTP nhiều hơn nên lợi thế này của SOAP cũng không mấy được coi trọng. | REST chỉ có thể truyền qua HTTP và nó thừa hưởng tất cả những lợi ích của giao thức HTTP, bao gồm các method như GET, POST, PUT và DELETE để thực hiện các hành động truy vấn, thêm, sửa, xóa dữ liệu. |
| 7 | Client | JS có thể dùng để gọi SOAP, nhưng rất khó để làm. | Quá đơn giản nếu dùng JS. |
| 8 | Performance | Performance không tốt bằng REST. | Performance tốt hơn SOAP, tốn ít tài nguyên CPU hơn, code ngắn gọn hơn. |
| 9 | Băng thông | Các message SOAP thường có độ dài và dung lượng cao hơn so với một request của RESTful, do đó nếu dùng SOAP thì bạn sẽ tốn băng thông nhiều hơn. | Tốn ít băng thông hơn SOAP. |
| 10 | Bảo mật | SOAP cần viết thêm các cơ sở hạ tầng để bảo mật các message và giao thức vận chuyển. | RESTful web service có thể được triển khai bằng các giải pháp và tiêu chuẩn truyền thống để các truy cập được phân quyền và xác thực trước khi sử dụng tài nguyên web. |
| 11 | Caching | Các dịch vụ web SOAP hoàn toàn bỏ qua caching web. | RESTful web service tận dụng tối đa cơ chế caching vì về cơ bản chúng dựa trên URL. |
| 12 | Tiếp cận | Các dịch vụ web SOAP, mọi thực thể đều tập trung vào các interface và message. | Các dịch vụ web dựa trên REST, mọi thực thể đều tập trung vào các tài nguyên |

**Lý do sử dụng REST thay vì SOAP?**

* REST có thể được sử dụng bởi bất kỳ client nào ví dụ: Java, C ++, Python client và thậm chí là một trình duyệt web với Ajax và JavaScript.
* REST nhẹ hơn so với SOAP, nó không yêu cầu phân tích cú pháp XML và nó cũng tiêu tốn ít băng thông hơn vì không giống như SOAP, REST không yêu cầu SOAP header cho mỗi lần request.
* SOAP là một công nghệ cũ, còn tất cả những gã khổng lồ hiện đại đang sử dụng REST, ví dụ: Google, Twitter và Flickr…
* REST rất dễ học, nó chỉ cần hiểu danh từ và động từ. Nếu bạn đã biết các phương thức HTTP thì nó thậm chí còn dễ dàng hơn.
* Java hỗ trợ tuyệt vời cho RESTFul web service và cũng hỗ trợ tốt cho SOAP web service. Bạn có rất nhiều sự lựa chọn ở đây, ví dụ: Jersey, RESTLet, …

**Lý do sử dụng SOAP?**

Mặc dù REST có rất nhiều ưu điểm hơn so với SOAP. Tuy nhiên, có một số lý do có thể khiến bạn muốn lựa chọn SOAP cho dịch vụ web của mình:

* ***WS-Security*** : SOAP không chỉ hỗ trợ SSL (giống như REST) mà còn hỗ trợ WS-Security, bổ sung thêm một số tính năng enterprise security. Hỗ trợ nhận dạng thông qua các trung gian, không chỉ là point-to-point như SSL. Nó được dùng khi muốn xây dựng những web service đảm bảo và tin cậy. Web Service Security đảm bảo cho tính an toàn, sự toàn vẹn thông điệp và tính tin cậy của thông điệp.
* ***WS-AtomicTransaction*** : Khi muốn có các giao dịch ACID qua một dịch vụ, bạn sẽ phải cần SOAP. Mặc dù REST có hỗ trợ các transactions, nhưng nó không toàn diện và cũng không phù hợp với ACID. REST bị hạn chế bởi HTTP nên không thể cung cấp cam kết hai pha trên các tài nguyên giao dịch phân tán, nhưng SOAP lại có thể àm được điều này. Thật may mắn các giao dịch ACID gần như không có ý nghĩa nhiều đối với các dịch vụ internet thông thường. Nhưng đôi khi các ứng dụng doanh nghiệp lại cần mức độ tin cậy giao dịch này.
* ***WS-ReliableMessaging*** : REST không có hệ thống báo lỗi chuẩn và mong muốn khách hàng giải quyết các lỗi communicate bằng cách retry và … retry … SOAP đã thành công trong việc xử lý những tình huống này và cung cấp end-to-end một cách tin cậy thông qua các trung gian SOAP

SOAP rõ ràng là hữu ích và quan trọng. Ví dụ, Nếu bạn viết một ứng dụng để giao tiếp với ngân hàng chắc chắn bạn sẽ cần phải sử dụng SOAP. Tất cả ba tính năng trên là bắt buộc đối với các giao dịch ngân hàng. Ví dụ: nếu tôi chuyển tiền từ tài khoản này sang tài khoản khác, tôi cần phải chắc chắn rằng nó đã hoàn tất. Việc cứ cố gắng retry thực sự là quá kinh dị nếu giao dịch thành công lần đầu tiên nhưng thông báo tôi nhận được lại là thất bại.

Bài viết khá dài và nặng về lý thuyết, có thể các bạn chưa hiểu được khi lần đầu tiên tìm hiểu về web service. Trong các bài viết tiếp theo, chúng ta sẽ cùng tìm hiểu cách xây dựng về SOAP / RESTful web service và cách sử dụng nó, khi đó các bạn sẽ hiểu rõ hơn về các vấn đề đã được trình bày trong bài viết này.

**Tài liệu tham khảo:**

* <https://spf13.com/post/soap-vs-rest/>
* <https://docs.oracle.com/javaee/7/tutorial/partwebsvcs.htm>
* <https://javarevisited.blogspot.com/2015/08/difference-between-soap-and-restfull-webservice-java.html>
* <https://www.javatpoint.com/soap-vs-rest-web-services>
* <https://www.w3schools.com/xml/xml_services.asp>
* <http://theopentutorials.com/tutorials/web-services/web-services-platform/>
* <https://www.tutorialspoint.com/webservices/>
* <https://o7planning.org/vi/10773/restful-web-service-la-gi>
* [https://vovanhai.wordpress.com/web-services/](https://vovanhai.wordpress.com/web-services/gi%E1%BB%9Bi-thi%E1%BB%87u-v%E1%BB%81-web-service/)

**Một số bài viết về Web service**

* [Tìm hiểu về Web Service](https://gpcoder.com/5572-tim-hieu-ve-web-service/)
* [Java Web Services – JAX-WS – SOAP](https://gpcoder.com/5615-java-web-services-jax-ws-soap/)
* [Giới thiệu SOAP UI và thực hiện test Web Service](https://gpcoder.com/5650-gioi-thieu-soap-ui-va-thuc-hien-test-web-service/)
* [SOAP Web service: Authentication trong JAX-WS](https://gpcoder.com/5701-soap-web-service-authentication-trong-jax-ws/)
* [SOAP Web service: Upload và Download file sử dụng MTOM trong JAX-WS](https://gpcoder.com/5703-soap-web-service-upload-va-download-file-su-dung-mtom-trong-jax-ws/)
* [Java Web Services – Jersey JAX-RS – REST và sử dụng REST API testing tools với Postman](https://gpcoder.com/5677-java-web-services-jersey-jax-rs-rest/)
* [REST Web service: Tạo ứng dụng Java RESTful Client với Jersey Client 2.x](https://gpcoder.com/5728-rest-web-service-tao-ung-dung-java-restful-client-voi-jersey-client-2-x/)
* [REST Web service: Upload và Download file với Jersey 2.x](https://gpcoder.com/5726-rest-web-service-upload-va-download-file-voi-jersey-2-x/)
* [REST Web service: HTTP Status Code và xử lý ngoại lệ RESTful web service với Jersey 2.x](https://gpcoder.com/5763-rest-web-service-http-status-code-va-xu-ly-ngoai-le-restful-web-service-voi-jersey-2-x/)
* [Rest Web service: Filter và Interceptor với Jersey 2.x (P1)](https://gpcoder.com/5789-rest-web-service-filter-va-interceptor-voi-jersey-2-x-p1/)
* [Rest Web service: Filter và Interceptor với Jersey 2.x (P2)](https://gpcoder.com/5816-rest-web-service-filter-va-interceptor-voi-jersey-2-x-p2/)
* [Tìm hiểu về xác thực và phân quyền trong ứng dụng](https://gpcoder.com/5825-tim-hieu-ve-xac-thuc-va-phan-quyen-trong-ung-dung/)
* [REST Web service: Basic Authentication trong Jersey 2.x](https://gpcoder.com/5724-rest-web-service-basic-authentication-trong-jersey-2-x/)
* [Giới thiệu Json Web Token (JWT)](https://gpcoder.com/5827-gioi-thieu-json-web-token-jwt/)
* [REST Web service: JWT – Token-based Authentication trong Jersey 2.x](https://gpcoder.com/5883-rest-web-service-jwt-token-based-authentication-trong-jersey-2-x/)
* [Tạo ứng dụng Java RESTful Client không sử dụng 3rd party libraries](https://gpcoder.com/5942-tao-ung-dung-java-restful-client-khong-su-dung-3rd-party-libraries/)
* [Tạo ứng dụng Java RESTful Client với thư viện OkHttp](https://gpcoder.com/5903-tao-ung-dung-java-restful-client-voi-thu-vien-okhttp/)
* [Tạo ứng dụng Java RESTful Client với thư viện Retrofit](https://gpcoder.com/5911-tao-ung-dung-java-restful-client-voi-thu-vien-retrofit/)
* [Giới thiệu Feign – Tạo ứng dụng Java RESTful Client không thể đơn giản hơn](https://gpcoder.com/5922-gioi-thieu-feign-tao-ung-dung-java-restful-client-khong-can-code-implement/)
* [Giới thiệu HATEOAS](https://gpcoder.com/5946-gioi-thieu-hateoas/)
* [Giới thiệu Swagger – Công cụ document cho RESTfull APIs](https://gpcoder.com/5967-gioi-thieu-swagger-cong-cu-document-cho-restfull-apis/)
* [Cài đặt và sử dụng Swagger UI](https://gpcoder.com/5993-cai-dat-va-su-dung-swagger-ui/)
* [Sử dụng Swagger UI trong jersey REST WS project](https://gpcoder.com/6006-su-dung-swagger-ui-trong-jersey-rest-ws-project/)
* [Làm thế nào để Test Jersey Rest API với JUnit?](https://gpcoder.com/6028-lam-the-nao-de-test-jersey-rest-api-voi-junit/)
* [Test REST Web Service đơn giản hơn với REST Assured](https://gpcoder.com/6038-test-rest-webservice-don-gian-hon-voi-rest-assured/)
* [Triển khai ứng dụng Jersey REST Web service lên Tomcat Server](https://gpcoder.com/6084-trien-khai-ung-dung-jersey-rest-web-service-len-tomcat-server/)
* [Giới thiệu Castle Mock – Mock REST APIs và SOAP web-services](https://gpcoder.com/6070-gioi-thieu-castle-mock-mock-rest-apis-va-soap-web-services/)