

MỤC LỤC

CHƯƠNG I: CÔNG NGHỆ WEB SERVICE	4
1. Web Service là gì ?.....	5
2. Đặc điểm của Web service	6
3. Nền tảng của Web service	8
4. Các công nghệ của Web service.....	9
4.1. Ngôn ngữ XML – RPC	9
4.2. Giao thức truyền thông điệp SOAP.....	10
4.3. Ngôn ngữ mô tả Web Service - WSDL.....	13
4.4. Đăng ký dịch vụ UDDI	16
CHƯƠNG II: KIẾN TRÚC WEB SERVICE	19
1. Cơ chế hoạt động của Web Service.....	19
2. Kiến trúc phân tầng của Web Service	20
3. Kiến trúc hướng dịch vụ SOA.....	22
3.1. Khái niệm kiến trúc hướng dịch vụ SOA	22
3.2. Nguyên tắc thiết kế của SOA	23
CHƯƠNG III: XÂY DỰNG WEB SERVICE.....	24
1. Các vấn đề cần xác định rõ trước khi bắt tay xây dựng ứng dụng Web service	24
2. Xây dựng Web Service với ASP.NET	25
3. Xây dựng Web Service với Java	31
CHƯƠNG IV: KHAI THÁC WEB SERVICE.....	46
1. Ứng dụng Window Form kết nối tới Web Service.....	46
2. Ứng dụng Java Swing kết nối tới Web Service.....	50
3. Ứng dụng Web ASP.NET kết nối tới Web Service	57
CHƯƠNG V: BẢO MẬT TRONG WEB SERVICE	62
1. Tổng quan về vấn đề bảo mật.....	62
2. Một số kiểu giả mạo, đánh cắp thông tin và cách phòng chống:.....	63
2.1. Message Replay Attack.....	63
2.2. Web Spoofing.....	65
3. Bảo mật trong web service	68
4. Thực hiện bảo mật trong web service	74
4.1. Cài đặt WSE 3.0.....	75

4.2. Xây dựng ứng dụng bảo mật Web service thông qua bảo mật thông điệp SOAP với WS-Security ..	77
5. Tổng kết	84
PHỤ LỤC I: Tài nguyên và tài liệu tham khảo	86

PHÂN BỐ CHƯƠNG TRÌNH

BUỔI			NỘI DUNG
STT	Lý thuyết	Thực hành	
1	Công nghệ Web Service		
2	Kiến trúc Web Service và Xây dựng Web Service		
3		Lab 01: Xây dựng Web Service	
4		Lab 02: Xây dựng Web Service	
5	Khai thác Web Service		
6		Lab 03: Khai thác Web Service	
7		Lab 04: Khai thác Web Service	
8	Bảo mật trong Web Service		
9		Lab 05 : Bảo mật Web Service	
10	Thi kết thúc môn		

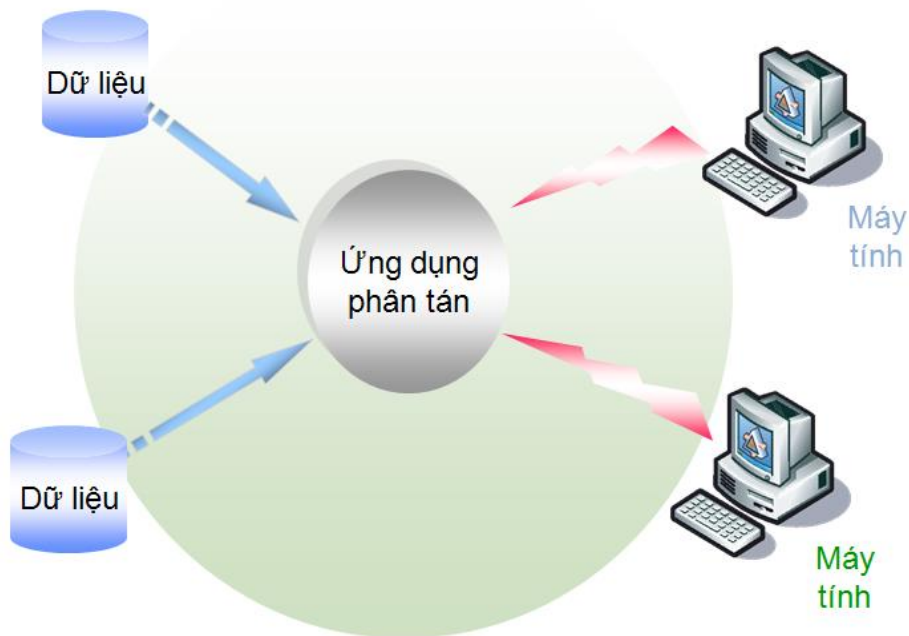
CHƯƠNG I: CÔNG NGHỆ WEB SERVICE

Mục tiêu:

Sau khi kết thúc chương này, bạn có thể:

- Hiểu về mục đích hoạt động của Web Service
- Hiểu về mô hình hoạt động client-server
- Hiểu về các công nghệ hỗ trợ được Web Service hỗ trợ
 - XML-RPC
 - SOAP
 - WSDL
 - UDDI

Sự phát triển của công nghệ thông tin cho phép ứng dụng hiệu quả vào các hoạt động kinh doanh, giải trí, quản lý cũng như một số lĩnh vực khoa học xã hội khác. Sự bùng nổ của Internet đã trở thành một điều kiện hết sức thuận lợi, đem lại hiệu suất cao trong công việc đồng thời giảm thiểu chi phí cho các doanh nghiệp. Tuy nhiên các yêu cầu về nghiệp vụ phức tạp trong hệ thống này dẫn đến các hệ thống phần mềm tương ứng cũng ngày càng trở nên phức tạp, cồng kềnh và khó kiểm soát. Rất nhiều yêu cầu nghiệp vụ đòi hỏi xử lý các vấn đề liên quan đến dữ liệu phân tán, xử lý các thông tin khác nhau do nhiều tổ chức nắm giữ.



Đã có nhiều kiến trúc phần mềm được đưa ra nhưng chưa đủ mạnh để giải quyết được vấn đề này. Qua thời gian, việc xây dựng ứng dụng như là một tập hợp thành phần phân phối vào mạng chung và làm việc với nhau như là một phần trong chương trình tổng thể ngày càng trở nên phổ biến. Hình thức này thường được gọi là kỹ thuật Distributed Component Object Model (DCOM) của Microsoft, Common Object Request Broker Architecture (CORBA) của Object Management Group, Remote Method Invocation (RMI) của Sun. Chúng cung cấp cấu trúc mềm dẻo, tin cậy để nắm bắt kịp thời sự phát triển cần thiết của các ứng dụng.

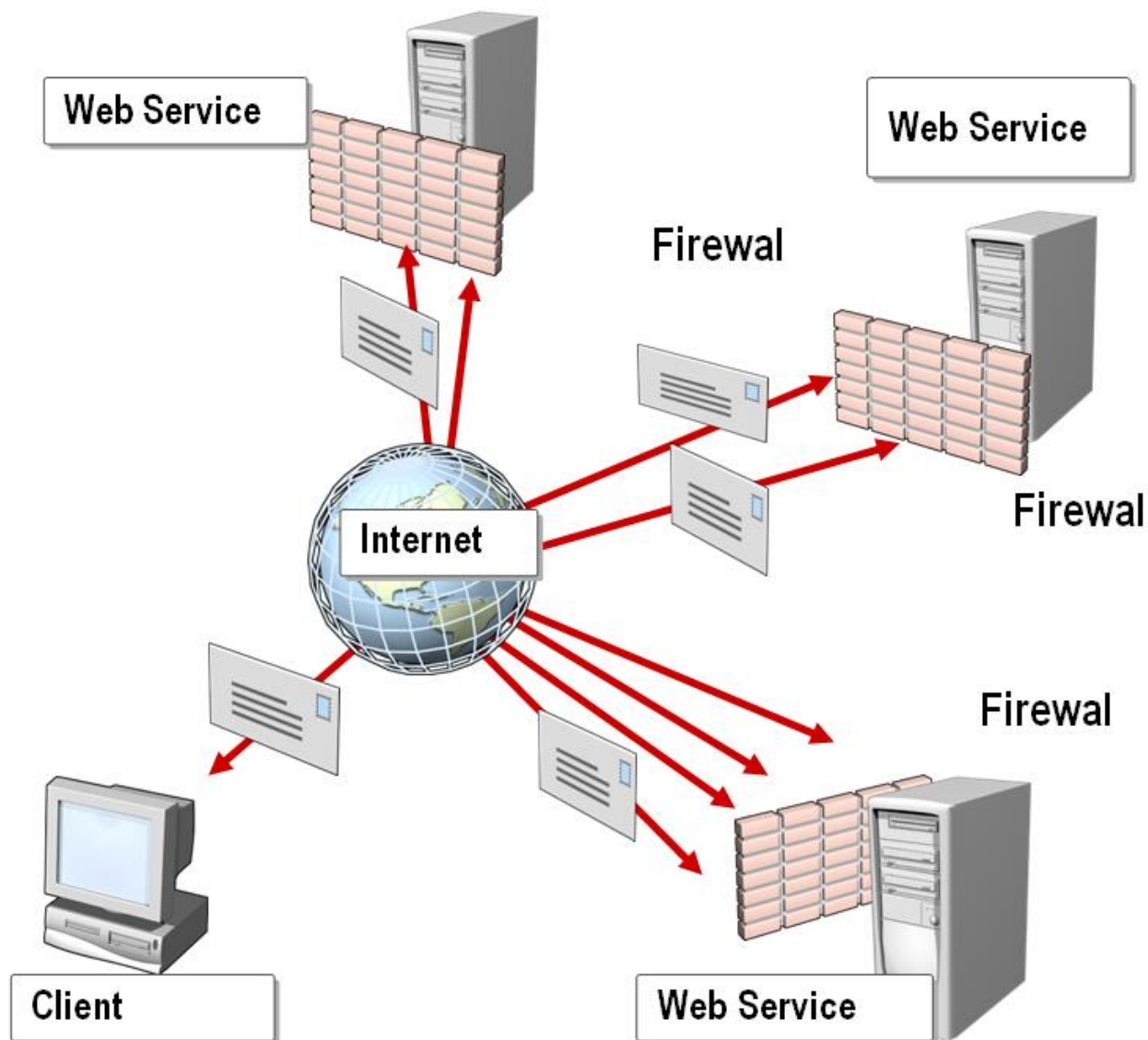
Mặc dù công nghệ dựa trên kiểu thành phần này hoạt động tốt trong môi trường Intranet, nhưng với Internet lại xuất hiện hai vấn đề đáng kể. Đầu tiên là chúng không thực hiện theo phương thức từng phần. Tất cả hoạt động xử lý dựa trên các đối tượng (object) nhưng lại không đưa ra từng chi tiết cụ thể như quản lý vòng đời hoạt động, hỗ trợ xây dựng và hỗ trợ kế thừa. Thứ hai, quan trọng hơn là quá tập trung vào hình thức truyền thông kiểu RPC điển hình dẫn đến việc xây dựng các hệ thống kết hợp chặt chẽ quanh các phương thức đối tượng.

Sự ra đời của kiến trúc phần mềm hướng dịch vụ (SOA – Service Orient Architecture) đã mở ra một hướng đi mới trong việc giải quyết các loại bài toán này. Kiến trúc SOA định nghĩa một kiểu kiến trúc cho việc xây dựng các hệ thống phân tán theo hướng dịch vụ, tức là hệ thống được phân tách thành các module chương trình, và các module này được phát triển độc lập, các module sử dụng các công nghệ khác nhau nhưng vẫn có thể giao tiếp được với nhau. Một công nghệ tiêu biểu nhất cho kiến trúc hướng dịch vụ là công nghệ Web Service. Với công nghệ Web Service, mỗi Service ở đây là một module có thể thực hiện các công việc khác nhau, ta có thể tổng hợp các Service thành phần lại để cùng thực hiện một công việc lớn, được gọi là công nghệ tích hợp Web Service, khi đó mỗi Service thành phần được gọi là một Service Composition. Sự ra đời của công nghệ Web Service đã đem lại rất nhiều lợi thế cho việc chia sẻ tài nguyên qua mạng, trợ giúp xây dựng các hệ thống phân tán đồng thời đáp ứng được tính mềm dẻo cần thiết, hệ thống có thể dễ dàng chấp nhận những thay đổi lớn so với thiết kế ban đầu mà vẫn đảm bảo cho vấn đề nâng cấp và bảo trì sau này. Web Service đem đến đầy đủ sự đáp ứng cần thiết cho các quy trình B2B – Bussiness to Bussiness và B2C – Bussiness to Customer, chính vì thế Web Service hiện tại đang là một thuật ngữ đang được nhắc đến rất nhiều và ngày càng được sử dụng rộng rãi.

1. Web Service là gì ?

Theo định nghĩa của W3C (World Wide Web Consortium), Web service là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau trong môi trường Internet thông qua các giao diện (Interface) chung và sự gắn kết được mô tả bằng XML.

Web service là tài nguyên phần mềm có thể xác định bằng địa chỉ URL để thực hiện các chức năng và đưa thông tin ra cho người dùng.

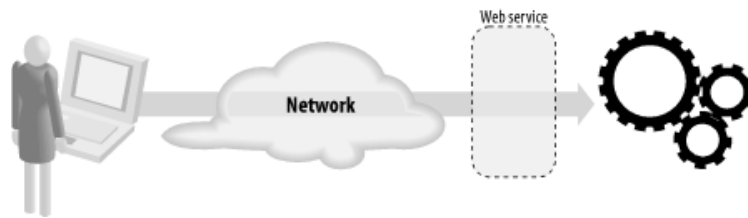


Web service được tạo ra bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng tìm thấy và truy cập tới các dịch vụ mà nó cung cấp, đồng thời vẫn có thể yêu cầu thông tin từ các dịch vụ khác. Web service bao gồm các mô đun độc lập để thực hiện yêu cầu nghiệp vụ của doanh nghiệp và được thực thi trên Server.

Ứng dụng cơ bản của Web service là tích hợp các hệ thống và là một trong những hoạt động chính khi phát triển hệ thống. Trong hệ thống này, các ứng dụng cần được tích hợp với cơ sở dữ liệu (CSDL) và các ứng dụng khác, người sử dụng sẽ giao tiếp với CSDL để tiến hành phân tích và lấy dữ liệu.

2. Đặc điểm của Web service

Web Service cho phép các ứng dụng khác nhau từ các nguồn khác nhau có thể giao tiếp với các ứng dụng khác mà không đòi hỏi nhiều thời gian coding, do tất cả các quá trình giao tiếp đều tuân theo định dạng XML, cho nên Web Service không bị phụ thuộc vào bất kỳ hệ điều hành hay ngôn ngữ lập trình nào. Web service cho phép client và server có thể tương tác được với nhau trên các nền tảng khác nhau mà không cần bất cứ thay đổi hay yêu cầu đặc biệt nào. Ví dụ, chương trình viết bằng ngôn ngữ Java cũng có thể trao đổi dữ liệu với các chương trình viết bằng Perl, các ứng dụng chạy trên nền Windows cũng có thể trao đổi dữ liệu với các ứng dụng chạy trên nền Linux. Công nghệ Web Service không yêu cầu phải sử dụng trình duyệt và ngôn ngữ HTML.

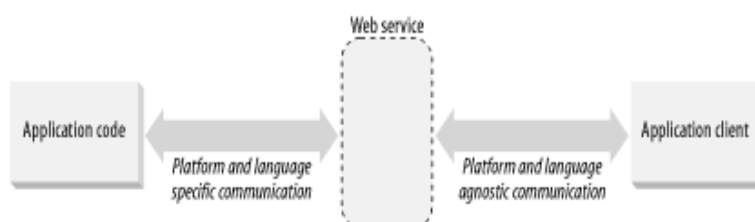


Phần lớn kỹ thuật của Web service được xây dựng trên mã nguồn mở và được phát triển từ các chuẩn đã được công nhận. Nó tích hợp các ứng dụng trên nền web lại với nhau bằng cách sử dụng các công nghệ XML, SOAP, WSDL, và UDDI trên nền tảng các giao thức Internet với mục tiêu tích hợp ứng dụng và truyền thông điệp. XML được sử dụng để đánh dấu dữ liệu, SOAP được dùng để truyền dữ liệu, WSDL được sử dụng để mô tả các dịch vụ có sẵn và UDDI được sử dụng để liệt kê những dịch vụ nào hiện tại đang có sẵn để có thể sử dụng. Web Service cho phép các tổ chức có thể trao đổi dữ liệu với nhau mà không cần phải có kiến thức hiểu biết về hệ thống thông tin đứng sau Firewall.

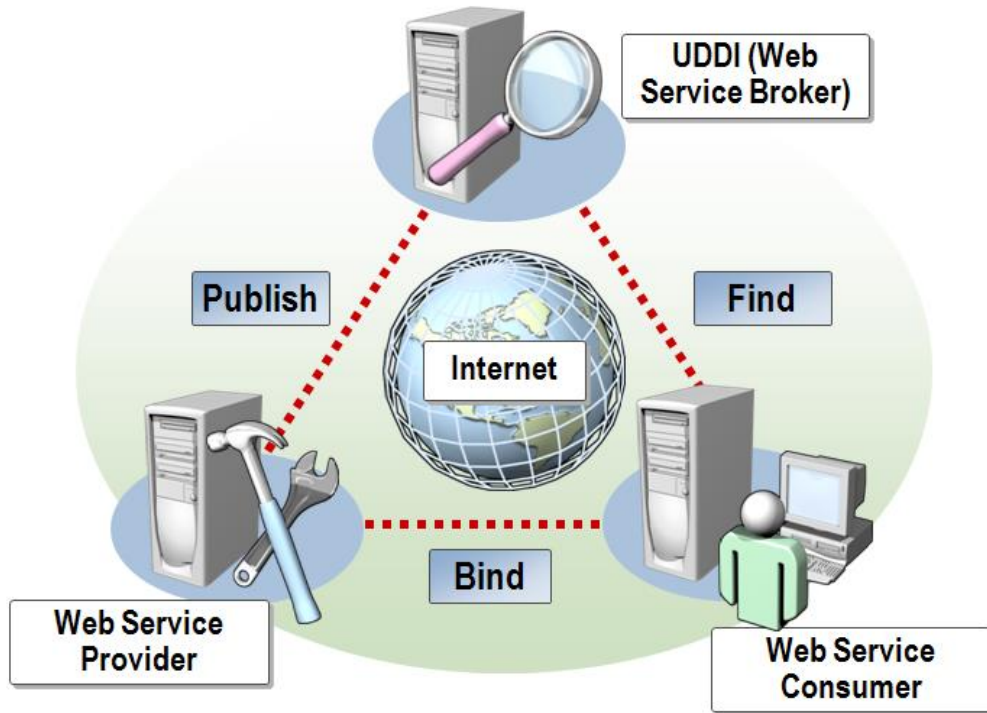
Web service có thể gồm nhiều mô đun và được công bố trên Internet.

Là sự kết hợp của việc phát triển theo hướng từng thành phần với những lĩnh vực cụ thể và cơ sở hạ tầng Web, đưa ra lợi ích cho cả doanh nghiệp, khách hàng, những nhà cung cấp dịch vụ khác và cả các cá nhân thông qua mạng Internet.

Web service khi được triển khai sẽ hoạt động theo mô hình client-server. Nó có thể được triển khai bởi một phần mềm ứng dụng phía Server như PHP, JSP, ASP.NET, ... Không giống như mô hình client-server truyền thống, chẳng hạn như hệ thống Web server-web page, Web Service không cung cấp cho người dùng một giao diện đồ họa nào, Web Service đơn thuần chỉ là việc chia sẻ các dữ liệu logic và xử lý các dữ liệu đó thông qua một giao diện chương trình ứng dụng được cài đặt xuyên suốt trên mạng máy tính.



Tính tương thích (Inteoperability) là một lợi thế vô cùng mạnh mẽ của Web Service, thông thường, các công nghệ Java và công nghệ của Microsoft rất khó có thể tích hợp được với nhau, nhưng với Web Service thì các Application và Client sử dụng 2 công nghệ trên hoàn toàn có khả năng tương tác với nhau thông qua Web Service.



3. *Nền tảng của Web service*

Dịch vụ Web cũng có thể được nói một cách khác là các khối cơ bản được xây dựng để di chuyển trong hệ thống máy tính phân tán trên Internet. Các chuẩn mở và việc tập trung vào giao tiếp và làm việc cộng tác giữa con người và các ứng dụng đã tạo nên một môi trường nơi mà Web service đang trở thành nền tảng cho việc tích hợp ứng dụng. Các ứng dụng được xây dựng sử dụng các Web service các loại từ nhiều nguồn khác nhau làm việc cùng với nhau bất kể là chúng ở đâu hoặc chúng đã được triển khai như thế nào. Có thể có các định nghĩa khác nhau về Web service khi các công ty xây dựng chúng, nhưng hầu hết tất cả các định nghĩa đều có chung các điểm sau:

- Thứ nhất, Web service đưa ra chức năng hữu dụng cho người sử dụng Web thông qua một giao thức chuẩn Web. Trong hầu hết các trường hợp, giao thức được sử dụng đó là SOAP.
- Thứ hai, Web service đưa ra cách mô tả các giao diện của chúng một cách đủ chi tiết nhằm cho phép người sử dụng xây dựng một ứng dụng máy trạm để giao tiếp được với chúng. Mô tả này thường được cung cấp ở dạng một tài liệu XML gọi là một tài liệu về ngôn ngữ mô tả Web service – WSDL (Web service Description Language).

- Thứ ba, Web service được đăng ký sao cho các khách hàng tiềm năng là người sử dụng có thể tìm thấy chúng một cách dễ dàng. Điều này được thực hiện với UDDI (Universal Discovery Description and Integration).

Web service như một dịch vụ phần mềm được trình bày trên Web thông qua giao thức SOAP, được mô tả bằng một tệp WSDL và được đăng ký trong UDDI. Các dịch vụ Web service là nguồn thông tin mà ta có thể dễ dàng kết hợp vào các ứng dụng. Dễ dàng nhận ra toàn bộ lớp ứng dụng có thể được xây dựng để phân tích và tích hợp thông tin ta quan tâm và trình bày nó theo nhiều cách khác nhau.

Việc trình bày các ứng dụng đang có như các dịch vụ Web service cho phép người sử dụng xây dựng các ứng dụng có các tính năng mạnh hơn thông qua việc sử dụng Web service như những block được xây sẵn. Ví dụ, người sử dụng có thể phát triển một ứng dụng mua bán để tự động lấy các thông tin về giá cả từ nhiều nhà cung cấp khác nhau, cho phép người dùng chọn một nhà cung cấp, chuyển đơn hàng và sau đó theo dõi việc chuyển hàng cho tới khi nhận được hàng. Ứng dụng của nhà cung cấp, khi trình bày các dịch vụ của họ trên Web, có thể quay ra sử dụng các dịch vụ Web service để kiểm tra tín dụng của khách hàng, lấy tiền từ tài khoản của khách hàng và thiết lập việc chuyển hàng với một công ty vận tải.

4. Các công nghệ của Web service

4.1. Ngôn ngữ XML – RPC

- XML : được viết tắt của cụm từ Extensible Markup Language – Ngôn ngữ đánh dấu dữ liệu.
- RPC – được viết tắt của cụm từ Remote Procedure Call – Thủ tục gọi từ xa. RPC cung cấp cho người phát triển kỹ thuật để định nghĩa ra một giao diện mà có thể được gọi từ xa thông qua môi trường mạng máy tính. Giao diện này có thể là một hàm đơn giản nhưng cũng có thể là một thư viện API khổng lồ.
- XML – RPC là một hướng tiếp cận dễ và rõ ràng nhất cho Web Service, nó cung cấp phương thức gọi một ứng dụng từ một máy tính local đến một máy tính từ xa thông qua môi trường mạng.
- XML – RPC cho phép chương trình có khả năng tạo ra các hàm hoặc các thủ tục gọi hàm thông qua mạng máy tính.
- XML – RPC sử dụng giao thức HTTP để vận chuyển thông tin từ Client đến Server.
- XML – RPC sử dụng ngôn ngữ XML để mô tả các thông điệp yêu cầu và các thông điệp đáp ứng gắn gũi với ngôn ngữ tự nhiên.

- XML – RPC Client chỉ ra cụ thể các thông tin về tên thủ tục, các tham biến trong thông điệp XML request, và Server trả về lỗi hoặc trả về thông điệp response trong thông điệp XML response.
- Các tham số của XML-RPC đơn giản chỉ là kiểu dữ liệu và nội dung – tuy nhiên các cấu trúc dữ liệu phức tạp như struct, array cũng được hỗ trợ bởi XML –RPC

4.2. Giao thức truyền thông điệp SOAP

SOAP viết tắt cho cụm từ - Simple Object Access Protocol. Trong kiến trúc phân tầng của Web Service, SOAP nằm ở tầng Packaging, SOAP là một giao thức đóng gói cho các dữ liệu chia sẻ giữa các ứng dụng. Xét về cơ bản, SOAP là XML, chính vì thế SOAP là một ứng dụng cụ thể của XML. SOAP được xây dựng lên từ các chuẩn XML như XML Schema và XML Namespaces dùng cho việc định nghĩa SOAP và các chức năng của nó.

Các thành phần cơ bản của SOAP gồm có:

a. Thông điệp XML

Thông điệp XML đó là các tài liệu XML được dùng để trao đổi thông tin giữa các ứng dụng. Nó cung cấp tính mềm dẻo cho các ứng dụng trong quá trình giao tiếp với nhau và là một dạng cơ bản của SOAP.

Các thông điệp này có thể là bất cứ thứ gì: Hóa đơn thanh toán, yêu cầu về giá cổ phiếu, một truy vấn tới một công cụ tìm kiếm hoặc có thể là bất kì thông tin nào có quan hệ tới từng thành phần của ứng dụng.



Hình 1: Mô tả cấu trúc của một thông điệp XML

Bởi vì XML không phụ thuộc vào một ứng dụng cụ thể, hệ điều hành hay ngôn ngữ lập trình nào, cho nên các thông điệp XML có thể sử dụng trong tất cả các môi trường. Một chương trình Windows Perl có thể tạo ra một thông điệp XML, trình bày thông điệp đó và gửi đến cho một chương trình cài đặt bằng ngôn ngữ Java được triển khai trên nền Unix.

b. RPC và EDI

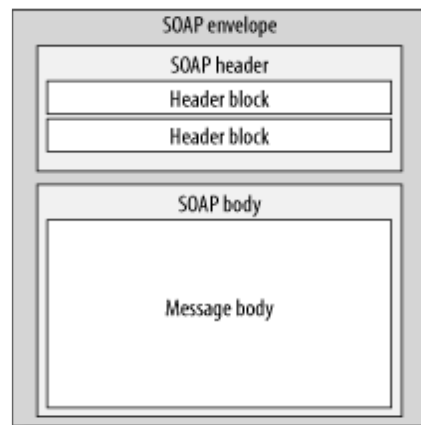
Sử dụng thông điệp XML, đương nhiên SOAP có 2 ứng dụng liên quan: RPC và EDI. Thủ tục gọi hàm từ xa RPC - Remote Procedure Call là một dạng tính toán phân tán cơ bản, mô tả cách thức để một chương trình tạo ra một thủ tục gọi hàm hoặc phương thức tới một máy tính khác, truyền đối số và lấy giá trị trả về. Trao đổi tài liệu điện tử EDI Electronic Document Interchange là một dạng transaction cơ bản cho quy trình thương mại nó định

nghĩa các chuẩn định dạng và thông dịch của các tài liệu, thông điệp tài chính và thương mại.

Nếu bạn sử dụng SOAP cho EDI, khi đó thông điệp XML có thể là các hóa đơn thanh toán, trả tiền thuế, hoặc các tài liệu tương tự. Nếu bạn sử dụng SOAP cho RPC khi đó thông điệp XML có thể trình bày các đối số hoặc các giá trị trả về.

c. Thông điệp SOAP

Thông điệp SOAP bao gồm phần tử gốc envelope bao trùm toàn bộ nội dung thông điệp SOAP, và các phần tử header và body. Phần tử header chứa các khối thông tin có liên quan đến cách thức các thông điệp được xử lý như thế nào. Nó bao gồm việc định tuyến và các thiết lập cho việc phân phối các thông điệp. Ngoài ra phần tử Header còn có thể chứa các thông tin về việc thẩm định quyền, xác minh và các ngữ cảnh cho các transaction. Các dữ liệu thực sự được lưu trữ tại phần tử body. Bất cứ thứ gì có thể trình bày cú pháp XML đều nằm trong phần tử body của một thông điệp SOAP.



Hình 2: Mô tả cấu trúc của một thông điệp SOAP

Tất cả các phần tử envelope đều chứa chính xác một phần tử body. Phần tử body có thể chứa các nút con theo yêu cầu. Nội dung của phần tử body là các thông điệp. Nếu phần tử envelope mà chứa phần tử header, nó chỉ chứa không nhiều hơn một phần tử header và phần tử header này bắt buộc phải là phần tử con đầu tiên của phần tử envelope. Mỗi một phần tử chứa header đều được gọi là header block. Mục đích của header block cung cấp giao tiếp các thông tin theo ngữ cảnh có liên quan đến quy trình xử lý các thông điệp SOAP.

d. SOAP Faults

SOAP faults là một dạng thông điệp SOAP đặc biệt được dùng để thông báo lỗi trong quá trình trao đổi thông tin, SOAP faults có thể xuất hiện trong quá trình xử lý các thông điệp SOAP[1].

```
<s:Envelope xmlns:s="...">
  <s:Body>
    <s:Fault>
      <faultcode>Client.Authentication</faultcode>
      <faultstring>
        Invalid credentials
      </faultstring>
      <faultactor>http://acme.com</faultactor>
      <details>
        <!-- application specific details -->
      </details>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

Hình 3: Mô tả thông điệp SOAP faults

Các thông tin về SOAP faults được diễn tả dưới đây:

- **Fault code**: Các thuật toán phát hiện lỗi sẽ tự sinh ra các giá trị dùng để phân biệt các kiểu lỗi xuất hiện. Các giá trị này phải là các XML Qualified Name, điều đó có nghĩa là các tên của các mã lỗi chỉ có ý nghĩa duy nhất trong vùng định nghĩa XML Namespace.
- **Fault string**: Diễn tả các lỗi mà người dùng có thể đọc hiểu được.
- **Fault actor**: Đây là dấu hiệu nhận dạng duy nhất của các nốt xử lý các thông điệp nơi mà các lỗi có khả năng xuất hiện.
- **Fault details**: Được sử dụng để trình bày các thông tin cụ thể của ứng dụng về lỗi mà nó xuất hiện. Nó phải được trình bày nếu có lỗi xuất hiện trực tiếp có liên quan đến các vấn đề về phần thân của thông điệp. Fault details có thể không cần sử dụng, tuy nhiên sẽ cần thiết khi ta cần trình bày cụ thể về thông tin lỗi xuất hiện trong mối quan hệ tới các phần còn lại của quy trình xử lý các thông điệp SOAP.

e. Vận chuyển SOAP

Như chúng tôi đã trình bày ở trên, SOAP được đặt ở tầng Packaging trong kiến trúc phân tầng của Web Service, SOAP đứng phía trên tầng Network và tầng Transport. Vì thế SOAP không quan tâm đến việc giao thức vận chuyển nào được sử dụng trong quá trình trao đổi các thông điệp, điều đó làm cho giao thức thực sự mềm dẻo tại bất kỳ môi trường SOAP được triển khai nào. Tính mềm dẻo của SOAP được thể hiện qua việc SOAP có thể sử dụng các giao thức vận chuyển khác nhau để trao đổi các thông điệp, như HTTP, FTP, SMTP, POP3, MQUERY và Jabber.

Hiện nay, HTTP được sử dụng phổ biến trên Internet, chính vì tính phổ biến của nó, cho nên HTTP hiện tại đang là giao thức vận chuyển phổ biến nhất cho việc vận chuyển các thông điệp SOAP.

SOAP thông qua HTTP rất thuận tiện cho SOAP RPC trong việc gọi yêu cầu và nhận các thông điệp đáp ứng bởi vì bản chất HTTP chính là giao thức dựa trên nền tảng gọi các

yêu cầu và nhận các đáp ứng (request-response-base protocol). Các SOAP request được gửi tới HTTP server thông qua phương thức POST và HTTP Server trả lại giá trị SOAP response thông qua các HTTP response.



Hình 4: Mô tả việc trao đổi thông điệp SOAP thông qua giao thức HTTP

4.3. Ngôn ngữ mô tả Web Service - WSDL

4.3.1. Tổng quan về WSDL

WSDL viết tắt của cụm từ Web Service Description Language – Ngôn ngữ mô tả Web Service. WSDL ra đời dưới sự phát triển của IBM và Microsoft.

WSDL dựa trên giao thức XML để trao đổi thông tin trong môi trường tập trung hoặc phân tán. WSDL mô tả cách thức truy cập tới Web Service và các hành động thực thi trên Web Service đó.

WSDL là ngôn ngữ cho việc mô tả các giao diện Web Service dựa trên nền tảng XML. WSDL là ngôn ngữ mà UDDI sử dụng.

4.3.2. Các thành phần của WSDL

Một tài liệu WSDL thường bao gồm các thành phần chính sau đây:

Thành phần	Mô tả
<type>	Định nghĩa kiểu dữ liệu được dùng trong Web Service
<message>	Các thông điệp được sử dụng trong Web Service
<porttype>	Các thao tác được thực thi bởi Web Service
<binding>	Các giao thức giao tiếp dùng cho Web Service

Giải thích ý nghĩa các thành phần:

- *Type* : Thành phần type định nghĩa kiểu dữ liệu được sử dụng cho Web Service Để đảm bảo tính không phụ thuộc vào platform, WSDL sử dụng cấu trúc của lược đồ XML để định nghĩa kiểu dữ liệu.
- *Message* : Thành phần message dùng để định nghĩa các thành phần dữ liệu và các thông điệp mà nó được gọi tới. Mỗi thông điệp có thể bao gồm một hoặc nhiều phần, các thành phần này có thể so sánh với các câu lệnh của các lời gọi hàm trong các ngôn ngữ lập trình truyền thống.
- *PortType* : Đây là thành phần quan trọng nhất trong một tài liệu WSDL. Nó được sử dụng để mô tả Web Service, các thao tác được thực thi và các lời gọi thông điệp. Thành phần port type có thể được so sánh với các thư viện hàm (hoặc các module, các lớp) trong các ngôn ngữ lập trình.

Trong thành phần <port Type>, ta thường gặp 4 kiểu thao tác được WSDL định nghĩa dưới đây:

Kiểu thao tác	Mô tả
One-way	Thao tác này thể hiện rằng nó chỉ nhận các lời gọi thông điệp nhưng không trả lại thông điệp đáp ứng
Request-response	Thao tác này bao gồm việc nhận các thông điệp yêu cầu và trả về các thông điệp đáp ứng
Solicit-response	Thao tác này sẽ gửi đi các yêu cầu và đợi các đáp ứng
Notification	Thao tác này sẽ gửi đi các yêu cầu nhưng không đợi để nhận các đáp ứng

- *Binding*: Thành phần này định nghĩa các định dạng thông điệp, các mô tả cụ thể về các giao thức cho mỗi port.

```
<binding type="glossaryTerms" name="b1">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation soapAction="http://example.com/getTerm"/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

Hình 5: Mô tả thành phần binding trong tài liệu WSDL

Một thành phần binding thông thường bao gồm 2 thuộc tính: name và type.

Thuộc tính “name” định nghĩa tên của “binding”, và thuộc tính “type” trỏ đến “port” của binding, trong ví dụ này port của binding là “glossaryTerms”.

Thành phần soap:binding có 2 thuộc tính là “style” và “transport”.

Thuộc tính style có thể là “rpc” hoặc “document”. Trong ví dụ trên chúng ta sử dụng “document”. Thuộc tính transport định nghĩa giao thức vận chuyển thông điệp SOAP. Trong ví dụ trên sử dụng giao thức HTTP.

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

Hình 6: Minh họa ví dụ của một tài liệu WSDL

Trong ví dụ trên, thành phần <portType> định nghĩa “glossaryTerm” như là tên của một Port, và “getTerm” như tên của một thao tác.

Thao tác “getTerm” có thông điệp nhập vào gọi là “getTermRequest” và có thông điệp xuất ra gọi là “getTermResponse”.

Thành phần <message> định nghĩa các phần của mỗi thông điệp và kiểu dữ liệu kết hợp với các thông điệp đó. Nếu so sánh với các ngôn ngữ lập trình truyền thống,

glossaryTerm có thể được coi như là một thư viện hàm, “getTerm” là một hàm với đối số truyền vào là “getTermRequest” và trả lại kết quả là getTermResponse.

4.4. Đăng ký dịch vụ UDDI

4.4.1. Tổng quan về UDDI

UDDI là một chuẩn dựa trên XML dùng cho việc mô tả, công bố và tìm kiếm Web Service. UDDI được viết tắt của Universal Description, Discovery and Integration.

UDDI là thư mục dùng cho việc lưu trữ các thông tin về Web Service. UDDI là thư mục của một giao diện Web Service được mô tả bởi WSDL.

UDDI giao tiếp thông qua SOAP. UDDI cùng với SOAP và WSDL được xem là 3 chuẩn của Web Service. UDDI là một kỹ thuật mở đầu tiên cho phép các quy trình thương mại điện tử có thể khám phá lẫn nhau và định nghĩa cách thức tương tác với nhau qua Internet.

4.4.2. Các thành phần của UDDI

UDDI gồm 2 thành phần chính:

- Phần đăng ký của tất cả các Web Service’s metadata, bao gồm cả việc trỏ đến tài liệu WSDL mô tả dịch vụ[16].
- Phần thiết lập WSDLPort type định nghĩa cho các thao tác và tìm kiếm thông tin đăng ký.

UDDI xây dựng dựa trên các giao thức chuẩn Internet được công bố bởi W3C và IETF như XML, HTTP, và DNS. UDDI sử dụng WSDL để mô tả giao diện của Web Service. Thêm nữa tính năng độc lập với nền tảng ngôn ngữ lập trình đã được điều hợp cùng với giao thức SOAP.

4.4.3. Mô hình dữ liệu của UDDI

UDDI bao gồm lược đồ XML, mô tả bốn kiểu cấu trúc dữ liệu dưới đây:

- BusinessEntity
- BusinessService
- BindingTemplate
- tModel
- publisherAssertion

a. Cấu trúc dữ liệu businessEntity

Cấu trúc dữ liệu businessEntity trình bày nhà cung cấp Web Service. Cấu trúc này chứa các thông tin về công ty, bao gồm danh sách liên lạc, thông tin, phân biệt các tổ chức thương mại, và danh sách các nhà cung cấp dịch vụ web.

b. Cấu trúc dữ liệu businessService

Cấu trúc dữ liệu business service trình bày một Web Service độc lập được cung cấp bởi business entity. Nó mô tả các thông tin về cách thức gắn kết với Web Service, định nghĩa kiểu Web Service và phân loại danh mục được liệt kê trong đó.

```
<businessService serviceKey="uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
  businessKey="uuid:COE6D5A8-C446-4f01-99DA-70E212685A40">
  <name>Hello World Web Service</name>
  <description>A friendly Web service</description>
  <bindingTemplates>
    ...
  </bindingTemplates>
  <categoryBag />
</businessService>
```

Hình 7: Minh họa cấu trúc dữ liệu businessService

Chú ý rằng sử dụng dấu hiệu nhận dạng duy nhất – Universally Unique Identifiers trong thuộc tính BusinessKey và serviceKey. Tất cả các business entity và business service đều là dấu hiệu nhận dạng duy nhất trong UDDI registries thông qua việc chỉ định UDDI bởi việc đăng ký khi thông tin được nhập vào lần đầu.

c. Cấu trúc dữ liệu bindingTemplate

BindingTemplate là kỹ thuật mô tả của Web Service được trình bày bởi cấu trúc dữ liệu Business Service. Binding template trình bày sự hoạt động thực tế của Web Service, mô tả công nghệ sử dụng để giao tiếp với Web Service. Một Business Service có thể có thể có nhiều binding template, cho nên dịch vụ phải chỉ rõ các hành động cụ thể khác nhau trong cùng một dịch vụ.

d. Cấu trúc dữ liệu tModel

tModel là lõi trong cùng của kiểu dữ liệu, nhưng rất khó có khả năng để có thể nắm bắt được hết. tModel là chuẩn cho mô hình kỹ thuật

tModel là phương pháp để mô tả một vài quy trình thương mại, dịch vụ và các cấu trúc mẫu lưu trữ trong UDDI registry. Bất kỳ một khái niệm trừu tượng nào đều có thể được đăng

ký trong UDDI như là một tModel. Ví dụ: chúng ta có thể định nghĩa ra một kiểu cổng (port type) WSDL mới, và đồng nghĩa với đó ta có thể định nghĩa ra một tModel mới mà trình bày kiểu cổng đó trong UDDI. Sau đó, ta có thể chỉ định ra dịch vụ thương mại mà thực thi kiểu cổng đó bằng việc kết hợp với tModel với một business service's binding template.

e. Cấu trúc dữ liệu publisherAssertion

Đây là một cấu trúc dữ liệu quan hệ mà nó đặt sự kết hợp giữa hai hoặc nhiều cấu trúc dữ liệu businessEntity theo một kiểu quan hệ cụ thể, chẳng hạn như một công ty con hoặc một phòng ban.

Cấu trúc dữ liệu publisherAssertion bao gồm ba thành phần chính: fromKey (BusinessKey đầu tiên), toKey (businesskey thứ hai) và keyedReference.

KeyReference thiết kế ra kiểu mỗi quan hệ kết hợp trong cặp thuật ngữ keyName, keyValue trong tModel. Tham chiếu duy nhất bởi tModelkey.

Tóm tắt bài học

Web service được tạo ra bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng tìm thấy và truy cập tới các dịch vụ mà nó cung cấp, đồng thời vẫn có thể yêu cầu thông tin từ các dịch vụ khác.

Web service bao gồm các mô đun độc lập để thực hiện yêu cầu nghiệp vụ của doanh nghiệp và được thực thi trên Server.

Ứng dụng cơ bản của Web service là tích hợp các hệ thống và là một trong những hoạt động chính khi phát triển hệ thống.

Web service khi được triển khai sẽ hoạt động theo mô hình client-server. Nó có thể được triển khai bởi một phần mềm ứng dụng phía Server như PHP, JSP, ASP.NET, ...

Web service cho phép client và server có thể tương tác được với nhau trên các nền tảng khác nhau mà không cần bất cứ thay đổi hay yêu cầu đặc biệt nào tuân theo định dạng XML.

Web Service sử dụng nhiều công nghệ hỗ trợ khác nhau, đều dựa trên chuẩn mở

- Ngôn ngữ XML – RPC
- Giao thức truyền thông điệp SOAP
- Ngôn ngữ mô tả Web Service - WSDL
- Đăng ký dịch vụ UDDI

CHƯƠNG II: KIẾN TRÚC WEB SERVICE

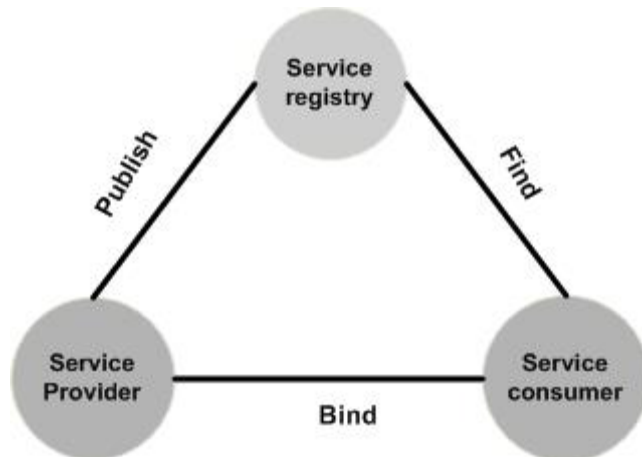
Mục tiêu:

Sau khi kết thúc chương này, bạn có thể:

- Hiểu về các tác nhân đối với Web Service: Service provider, Service registry, Service consumer
- Hiểu về kiến trúc phân tầng của Web Service
- Hiểu về mục đích, vai trò thiết kế của SOA đối với Web Service

1. Cơ chế hoạt động của Web Service

Cơ chế hoạt động của Web Service yêu cầu phải có 3 thao tác đó là : Find, Publish, Bind.



Hình 1: Mô tả cơ chế hoạt động của Web Service.

Trong kiến trúc Web Service, Service Provider công bố các mô tả về các service thông qua Service Registry. Service Consumer tìm kiếm trong các Service Registry để tìm ra các service mà họ cần sử dụng. Service Consumer có thể là một người hoặc cũng có thể là một chương trình.

Kỹ thuật mô tả dịch vụ là một trong những thành phần chủ chốt của kiến trúc Web Service. Các thông tin mô tả đầy đủ nhất về kiến trúc Web Service được thể hiện trong hai tài liệu riêng biệt, đó là NASSL – Network Accessible Service Specification Language và

WDS – Web-Defined Service. NASSL là một tài liệu dưới dạng chuẩn của XML cho các service chạy trên nền Network, nó được sử dụng để chỉ ra các thông tin hoạt động của Web Service, chẳng hạn như danh sách các service, các mô tả về service, ngày hết hạn của service và các thông tin liên quan đến các Service Provider, như tên, địa chỉ. Tài liệu WDS là một tài liệu mang tính đáp ứng đầy đủ cho tài liệu NASSL. Khi ta kết hợp hai tài liệu này với nhau ta sẽ có được sự mô tả một cách đầy đủ về các dịch vụ để cho phía yêu cầu dịch vụ có thể dễ dàng tìm kiếm và gọi các dịch vụ đó.

2. Kiến trúc phân tầng của Web Service



Hình 1: Web Service technology stack

Mô hình kiến trúc phân tầng của Web Service tương tự với mô hình TCP/IP được sử dụng để mô tả kiến trúc Internet.



Hình 2: TCP/IP network model

Các tầng truyền thống như Packaging, Description, và Discovery trong mô hình Web Service Stack là những tầng cung cấp khả năng tích hợp và cần thiết cho mô hình ngôn ngữ lập trình trung lập.

- **Tầng Discovery** : Tầng Discovery cung cấp cơ chế cho người dùng khả năng lấy các thông tin mô tả về các Service Provider. Công nghệ được sử dụng tại tầng này đó chính là UDDI – Universal Description, Discovery and Integration.
- **Tầng Description** : Khi Web Service được thực thi, nó cần phải đưa ra các quyết định về các giao thức trên các tầng Network, Transport, Packaging mà nó sẽ hỗ trợ trong quá trình thực thi. Các mô tả về dịch vụ sẽ đưa ra phương pháp để làm thế nào mà các Service Consumer có thể liên kết và sử dụng các service đó. Tại tầng

Description, công nghệ được sử dụng ở đây chính là WSDL (Web Service Description Language) – Ngôn ngữ mô tả Web Service. Ngoài ra, ít phổ biến hơn, chúng ta còn có 2 ngôn ngữ khác được định nghĩa bởi tổ chức W3C đó là ngôn ngữ mô tả tài nguyên - W3C's Resource Description Framework (RDF) và ngôn ngữ đánh dấu sự kiện DARPA.

Cả hai ngôn ngữ này đều có khả năng cung cấp việc mô tả Web Service mạnh hơn ngôn ngữ WSDL tuy nhiên do tính phức tạp của chúng nên không được phát triển rộng rãi. Chúng tôi sẽ đề cập đến ngôn ngữ WSDL một cách cụ thể hơn trong phần “Các công nghệ của Web Service” tại chương 2 của khóa luận này.

- **Tầng Packaging:** Việc thực hiện vận chuyển các dữ liệu Web Service được thực hiện bởi tầng Transport, tuy nhiên trước khi được vận chuyển, các dữ liệu cần phải được đóng gói lại theo các định dạng đã định trước để các thành phần tham gia vào mô hình Web Service có thể hiểu được, việc đóng gói dữ liệu được thi bởi tầng Packaging. Việc đóng gói dữ liệu bao gồm các công việc định dạng dữ liệu, mã hóa các giá trị đi kèm dữ liệu đó và các công việc khác.

Các dữ liệu có thể được đóng gói dưới dạng các tài liệu HTML, tuy nhiên với các tài liệu HTML thường không thuận tiện cho yêu cầu này bởi vì HTML chỉ có ưu điểm trong việc thể hiện dữ liệu hơn là trình bày ý nghĩa dữ liệu đó. XML là một định dạng cơ bản nhất cho việc trình bày dữ liệu, bởi vì XML có thể được sử dụng để trình bày ý nghĩa dữ liệu được vận chuyển, và hơn thế nữa, hiện tại đa số các ứng dụng chạy trên nền Web-Base đều hỗ trợ các bộ phân tích cú pháp XML.

SOAP là công nghệ chủ yếu được sử dụng tại tầng này, nó là một giao thức đóng gói dữ liệu phổ biến dựa trên nền tảng XML. Chúng ta sẽ đề cập sâu hơn đến giao thức đóng gói dữ liệu SOAP trong phần “Các công nghệ của Web Service” trong chương 2 của khóa luận này.

- **Tầng Transport :** Tầng Transport có vai trò đảm nhiệm việc vận chuyển các Web Service Message, tại đây bao gồm một vài dạng công nghệ khác nhau cho phép các giao tiếp trực tiếp giữa các Application – to – Application dựa trên tầng Network. Mỗi công nghệ bao gồm các giao thức như tcp, http, smtp và jabber..v.v.

Việc lựa chọn giao thức vận chuyển được dựa trên mỗi nhu cầu giao tiếp của các Web Service. ví dụ: với giao thức HTTP là một giao thức vận chuyển khá phổ biến được sử dụng cho các ứng dụng Web-Base, nhưng nó không cung cấp cơ chế giao tiếp bất đối xứng. Jabber, xét trên phương diện khác, nó không phải là một chuẩn nhưng có khả năng cung cấp tốt các kênh giao tiếp bất đối xứng.

- **Tầng Network** : Tầng Network trong công nghệ Web Service chính xác giống tầng Network trong mô hình giao thức TCP/IP. Nó cung cấp khả năng giao tiếp cơ bản, định địa chỉ và định tuyến.

3. Kiến trúc hướng dịch vụ SOA

3.1. Khái niệm kiến trúc hướng dịch vụ SOA

SOA - viết tắt của thuật ngữ Service Oriented Architecture (kiến trúc hướng dịch vụ) là “Khái niệm về hệ thống trong đó mỗi ứng dụng được xem như một nguồn cung cấp dịch vụ”.

Dịch vụ là yếu tố then chốt trong SOA. Có thể hiểu dịch vụ như là hàm chức năng (module phần mềm) thực hiện quy trình nghiệp vụ nào đó, một cách cơ bản, SOA là tập hợp các dịch vụ kết nối mềm dẻo với nhau (nghĩa là một ứng dụng có thể nói chuyện với một ứng dụng khác mà không cần biết các chi tiết kỹ thuật bên trong), có giao tiếp (dùng để gọi hàm dịch vụ) được định nghĩa rõ ràng và độc lập với nền tảng hệ thống, và có thể tái sử dụng. SOA là cấp độ cao hơn của phát triển ứng dụng, chú trọng đến quy trình nghiệp vụ và dùng giao tiếp chuẩn để giúp che đi sự phức tạp của kỹ thuật bên dưới.

Thiết kế SOA tách riêng phần thực hiện dịch vụ (phần mềm) với giao tiếp gọi dịch vụ. Điều này tạo nên một giao tiếp nhất quán cho ứng dụng khách sử dụng dịch vụ bất chấp công nghệ thực hiện dịch vụ. Thay vì xây dựng các ứng dụng đơn lẻ và đồ sộ, nhà phát triển sẽ xây dựng các dịch vụ có tính linh hoạt có thể triển khai và tái sử dụng trong toàn bộ quy trình nghiệp vụ. Điều này cho phép tái sử dụng phần mềm tốt hơn, cũng như tăng sự linh hoạt vì nhà phát triển có thể cải tiến dịch vụ mà không làm ảnh hưởng đến Client sử dụng dịch vụ.

Thực ra khái niệm SOA không hoàn toàn mới, DCOM và CORBA cũng có kiến trúc tương tự. Tuy nhiên các kiến trúc cũ ràng buộc các thành phần với nhau quá chặt, ví dụ các ứng dụng phân tán muốn làm việc với nhau phải đạt được thoả thuận về chi tiết tập hàm API, một thay đổi mã lệnh trong thành phần COM sẽ yêu cầu những thay đổi tương ứng đối với mã lệnh truy cập thành phần COM này.

Ưu điểm quan trọng nhất của SOA là khả năng kết nối mềm dẻo (nhờ sự chuẩn hoá giao tiếp) và tái sử dụng. Các dịch vụ có thể được sử dụng với trình Client chạy trên nền tảng bất kỳ và được viết bởi ngôn ngữ bất kỳ.

3.2. Nguyên tắc thiết kế của SOA

SOA dựa trên hai nguyên tắc thiết kế quan trọng:

- Mô-đun: đó là tách các vấn đề lớn thành nhiều vấn đề nhỏ hơn
- Đóng gói : Che đi dữ liệu và lô-gic trong từng mô-đun đối với các truy cập từ bên ngoài.

Hai tính chất này sẽ dẫn đến đặc điểm thiết kế của kiến trúc SOA đó là các dịch vụ tương tác với nhau qua các thành phần giao tiếp, tuy nhiên các dịch vụ đó vẫn hoạt động độc lập với nhau, chia sẻ các lược đồ dữ liệu cho nhau và tuân thủ các chính sách của kiến trúc chung nhất.

Tóm tắt bài học

Trong kiến trúc Web Service, Service Provider công bố các mô tả về các service thông qua Service Registry. Service Consumer tìm kiếm trong các Service Registry để tìm ra các service mà họ cần sử dụng. Service Consumer có thể là một người hoặc cũng có thể là một chương trình.

Các tầng truyền thống như Packaging, Description, và Discovery trong mô hình Web Service Stack là những tầng cung cấp khả năng tích hợp và cần thiết cho mô hình ngôn ngữ lập trình trung lập.

SOA - viết tắt của thuật ngữ Service Oriented Architecture (kiến trúc hướng dịch vụ).

Ưu điểm quan trọng nhất của SOA là khả năng kết nối mềm dẻo (nhờ sự chuẩn hoá giao tiếp) và tái sử dụng. Các dịch vụ có thể được sử dụng với trình Client chạy trên nền tảng bất kì và được viết bởi ngôn ngữ bất kì.

SOA dựa trên hai nguyên tắc thiết kế quan trọng:

- Mô-đun: đó là tách các vấn đề lớn thành nhiều vấn đề nhỏ hơn
- Đóng gói : Che đi dữ liệu và lô-gic trong từng mô-đun đối với các truy cập từ bên ngoài.

CHƯƠNG III: XÂY DỰNG WEB SERVICE

Mục tiêu:

Sau khi kết thúc chương này, bạn có thể:

- Hiểu về các giai đoạn chính để xây dựng Web Service
- Hiểu về các cách tiếp cận để quyết định cách thức xây dựng Web Service: bottom-up, top-down, from-scratch
- Nắm vững quy trình hoàn thiện Web Service
- Sử dụng thành thạo ngôn ngữ và công cụ IDE để phát triển

Trong thực tế, ngôn ngữ lập trình C# và Java đã được sử dụng phổ biến và thân thiện trong công việc lập trình nên để minh họa và làm rõ hơn nội dung bài học ta sẽ chọn 2 ngôn ngữ này.

1. Các vấn đề cần xác định rõ trước khi bắt tay xây dựng ứng dụng Web service

Có 4 giai đoạn chính để xây dựng một dịch vụ Web là xây dựng, triển khai, tiến hành và quản lý, trong đó:

- Giai đoạn xây dựng bao gồm phát triển và chạy thử ứng dụng dịch vụ Web, xây dựng các chức năng và định nghĩa dịch vụ. Có hai cách khác nhau để tiến hành trong giai đoạn này, đó là Red-path- solid và Blue-path-dashed. Với Red- path-solid, chúng ta sẽ xây dựng một dịch vụ Web mới từ trạng thái ban đầu hoặc với một dịch vụ đã có sẵn. Từ đó, xây dựng định nghĩa service (WSDL) với các đối tượng, hàm chức năng mà chúng ta mong muốn. Nếu theo cách Blue-path-dashed, dịch vụ Web sẽ được xây dựng từ đầu hoặc từ một định nghĩa dịch vụ WSDL. Sử dụng WSDL này, xây dựng hoặc sửa đổi lại mã để thực hiện các yêu cầu mong muốn trong dịch vụ Web.

- Giai đoạn triển khai: công bố định nghĩa dịch vụ, xây dựng WSDL và triển khai mã thực thi của dịch vụ Web. Triển khai dịch vụ Web tới một ứng dụng phía server, sau đó sẽ công bố dịch vụ Web trên mạng Internet để các client có thể nhìn thấy. Sử dụng UDDI registry để công bố lên mạng.

- Giai đoạn tiến hành: tìm kiếm và gọi thực thi dịch vụ Web bởi những người dùng muốn sử dụng dịch vụ.

- Quản lý: Quản lý và quản trị dịch vụ, duy trì sự ổn định của dịch vụ, cập nhật thông tin mới, sửa lỗi khi nó xảy ra...

Để xây dựng một dịch vụ Web, chúng ta cần hiểu được những việc phải làm và nên bắt đầu từ đâu. Có 3 cách tiếp cận chủ yếu để xây dựng nên một dịch vụ Web, có thể từ một ứng dụng đã có (bottom-up); từ một định nghĩa dịch vụ, WSDL để phát sinh một ứng dụng mới (top-down) hoặc có thể từ một nhóm các dịch vụ Web hiện có, kết hợp lại với nhau để tạo nên các chức năng mới hoặc mở rộng thêm chức năng. Những hướng tiếp cận này dựa trên những gì mà chúng ta đã có, tùy thuộc vào yêu cầu của hệ thống, trong đó tối đa việc sử dụng lại các chức năng, các thành phần, môđun đã được xây dựng.

Qui trình xây dựng một dịch vụ Web bao gồm các bước sau:

- Định nghĩa và xây dựng các chức năng, các dịch vụ mà dịch vụ sẽ cung cấp (sử dụng ngôn ngữ Java chẳng hạn).
- Tạo WSDL cho dịch vụ
- Xây dựng SOAP server
- Đăng ký WSDL với UDDI registry để cho phép các client có thể tìm thấy và truy xuất.
- Client nhận file WSDL và từ đó xây dựng SOAP client để có thể kết nối với SOAP server
- Xây dựng ứng dụng phía client (chẳng hạn sử dụng Java) và sau đó gọi thực hiện dịch vụ thông qua việc kết nối tới SOAP server.

Lựa chọn một ngôn ngữ, xây dựng các tiến trình nghiệp vụ và chúng ta bắt đầu tạo nên một dịch vụ Web như ý muốn. Sau đó là cung cấp dịch vụ Web này trên Internet.

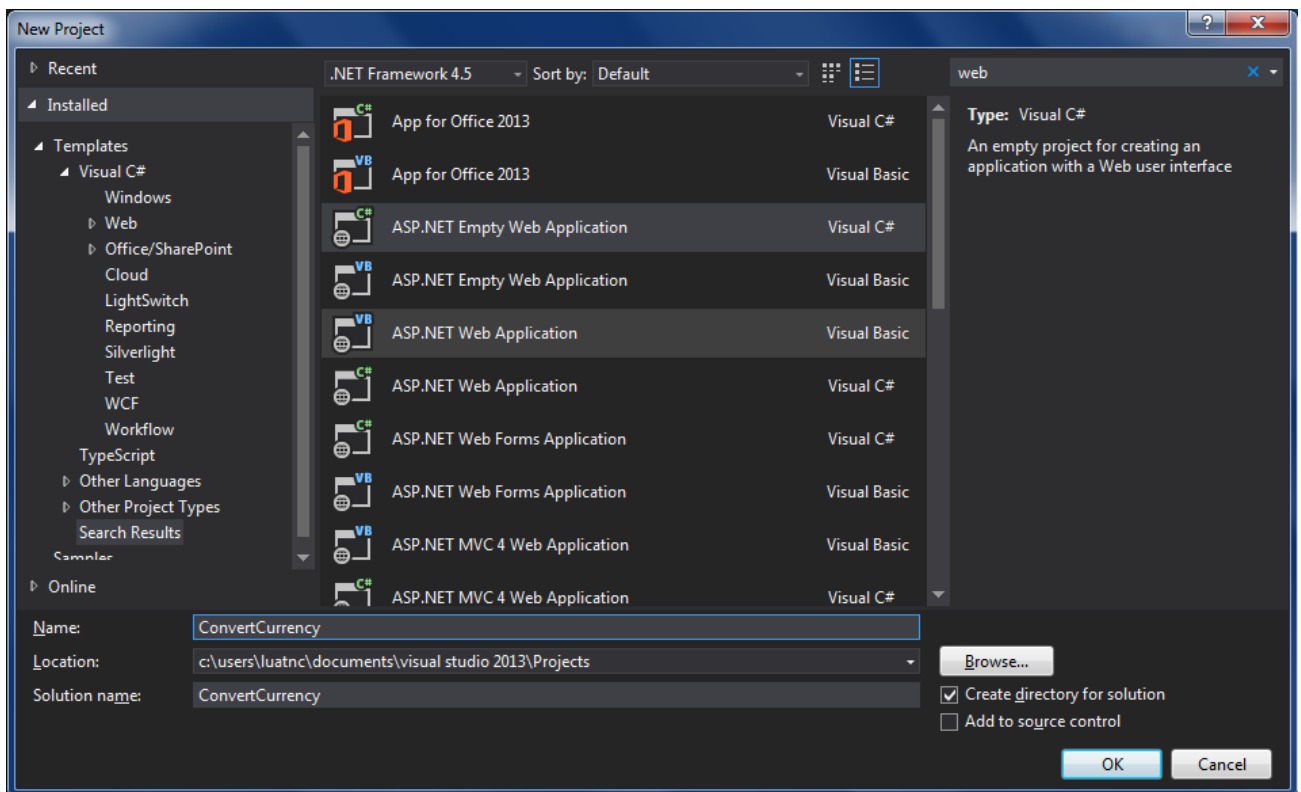
2. Xây dựng Web Service với ASP.NET

Hiện thực hóa bài học trước, ta sẽ cùng xây dựng 1 ứng dụng nhỏ cho Web Service trên nền tảng ASP.NET cho phép chuyển đổi tiền tệ từ Việt Nam đồng sang USD hoặc Euro và ngược lại.

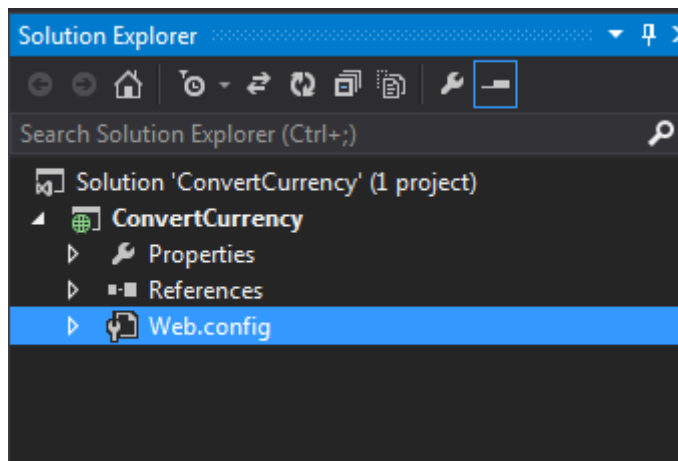
Công cụ sử dụng bao gồm: Visual Studio và ngôn ngữ lập trình ASP.NET, C#.

a. Tạo mới Web Application Project

Chọn menu File > New > Project trong Visual Studio, chọn mục **ASP.NET Web Application** và đặt tên **ConvertCurrency**, chọn thư mục lưu trữ project và click OK.

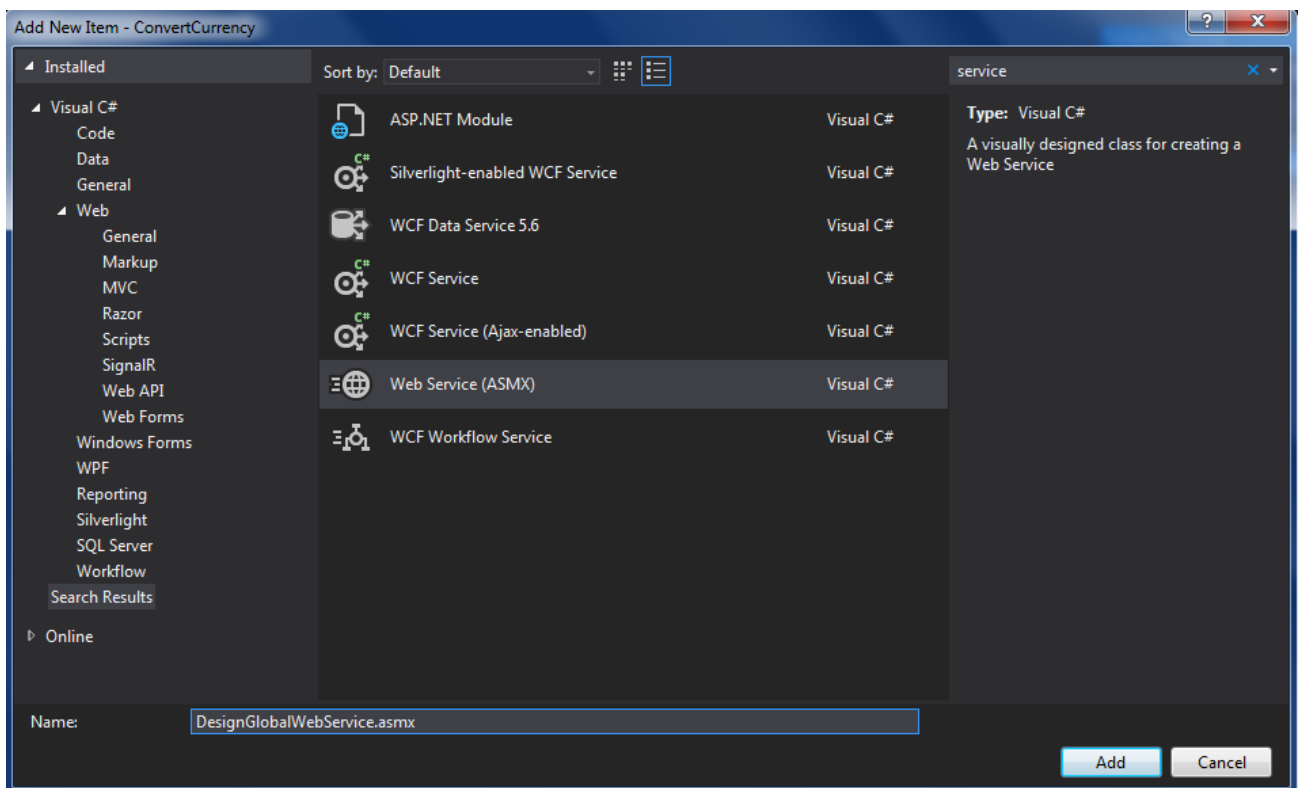


Kết quả, ta sẽ thấy trong cửa sổ *Solution Explorer* như hình sau

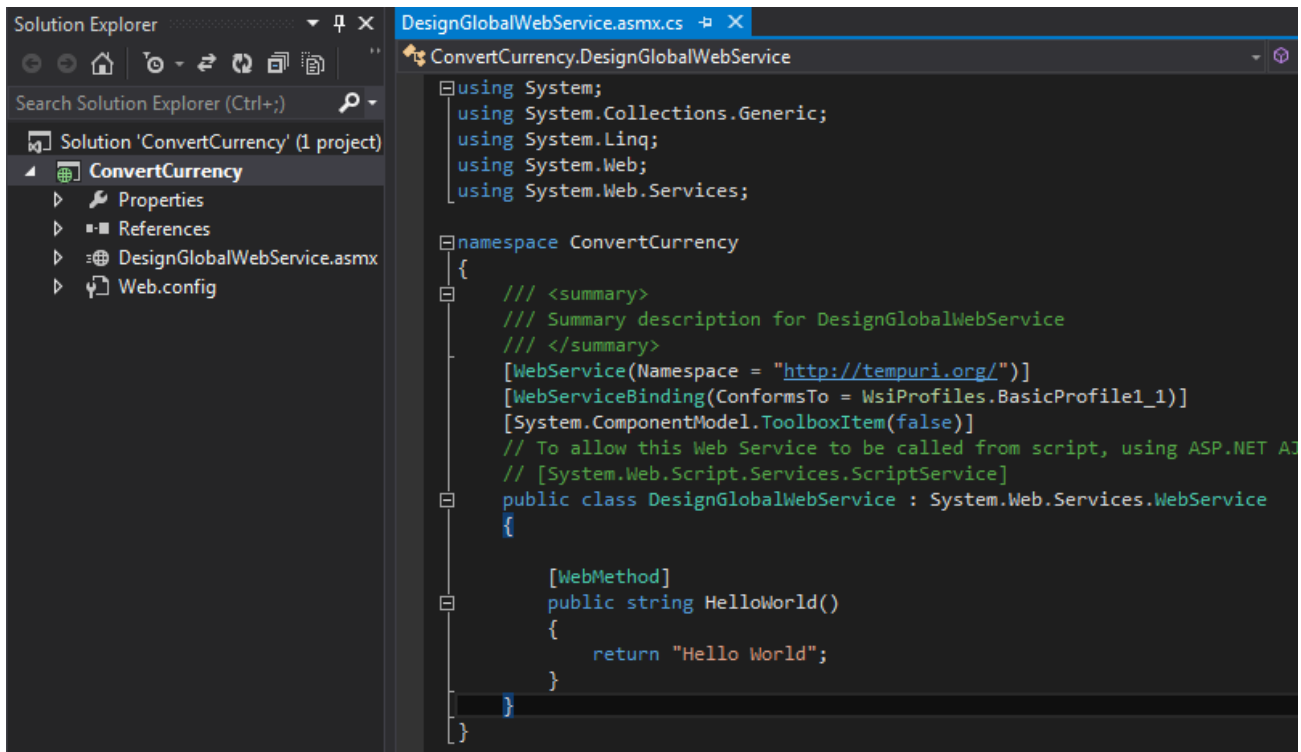


b. Tạo mới Web Service

Click chuột phải vào project và chọn *Add new > New item* và chọn như hình sau



Chọn mục **Web Service (ASMX)** và đặt tên **DesignGlobalWebService.asmx** và click Add, ta sẽ thấy kết quả như hình sau



Nội dung mặc định sẽ được Visual Studio tạo ra như trên, ta tiến hành chỉnh sửa lại Web Service để cung cấp các dịch vụ như mong muốn như sau

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;

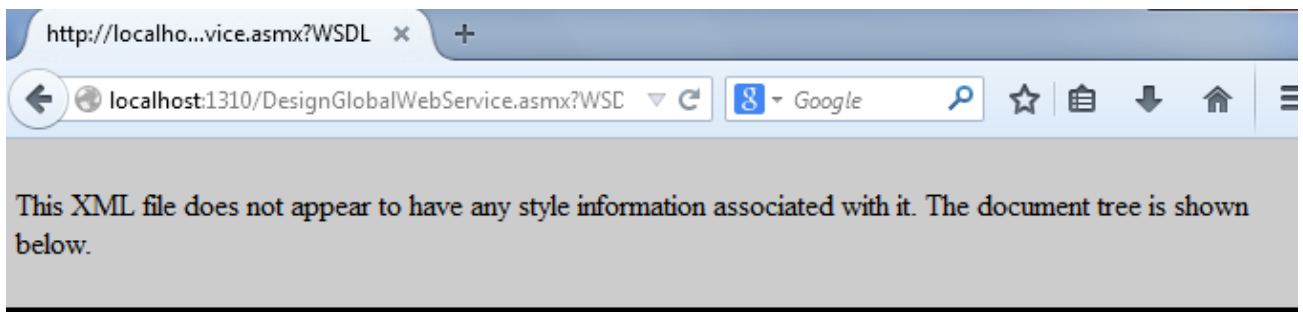
namespace ConvertCurrency
{
    /// <summary>
    /// Web service này dùng chuyển đổi ngoại tệ từ tiền Việt sang USD,
    /// Euro và ngược lại
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [ToolboxItem(false)]
    public class ConvertWS : System.Web.Services.WebService
    {
        private const double USD_RATE = 20000;
        private const double EUR_RATE = 30000;
        [WebMethod]
        public double VND2USD(double dong)
        {
            return dong/USD_RATE;
        }
        [WebMethod]
        public double VND2EUR(double dong)
        {
            return dong / EUR_RATE;
        }
        [WebMethod]
        public double USD2VND(double usd)
        {
            return usd * USD_RATE;
        }
    }
}
```

```
[WebMethod]
public double EUR2VND(double eur)
{
    return eur * EUR_RATE;
}
}
```

Chạy ứng dụng bằng cách nhấn F5, VS sẽ triển khai ứng dụng. Kết quả như sau

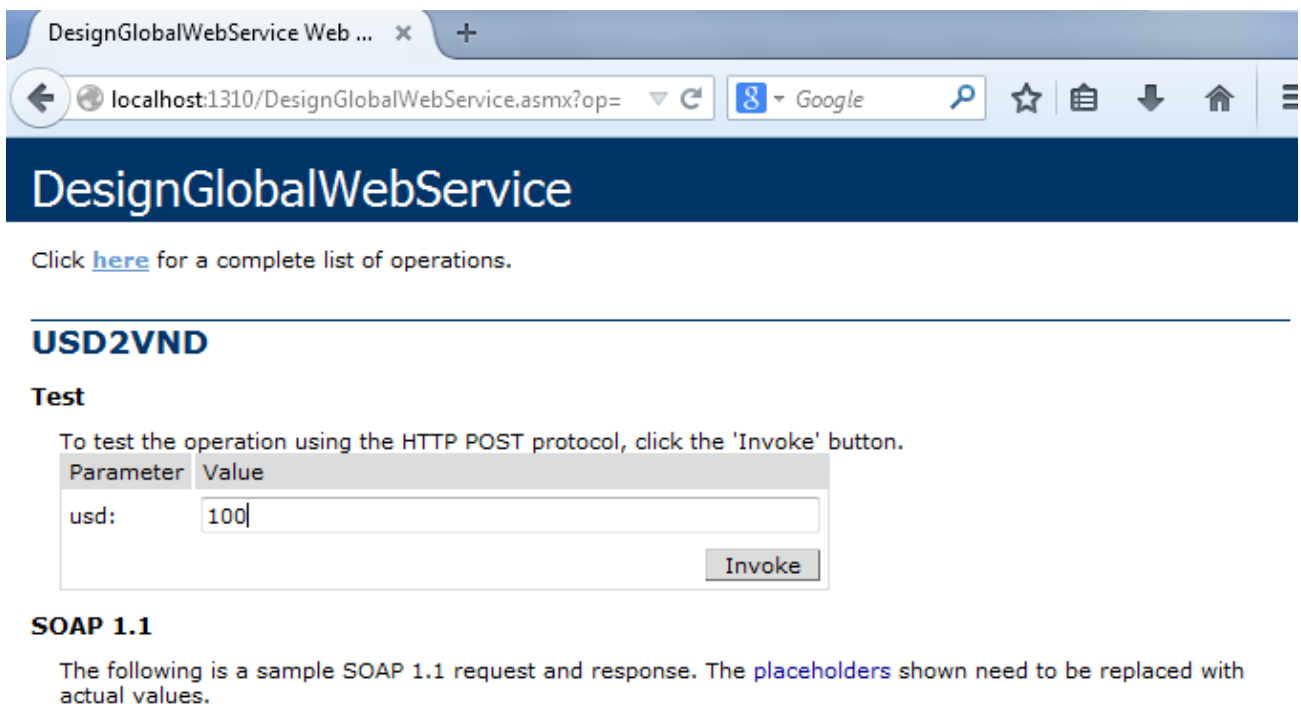


Kiểm tra WSDL của service, ta nhấn link “Service Description” ta có



```
- <wsdl:definitions targetNamespace="http://tempuri.org/">
- <wsdl:types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
- <s:element name="VND2USD">
- <s:complexType>
- <s:sequence>
- <s:element minOccurs="1" maxOccurs="1" name="dong" type="s:double"/>
- </s:sequence>
- </s:complexType>
- </s:element>
- <s:element name="VND2USDResponse">
- <s:complexType>
- <s:sequence>
- <s:element minOccurs="1" maxOccurs="1" name="VND2USDResult" type="s:double"/>
- </s:sequence>
- </s:complexType>
- </s:element>
```

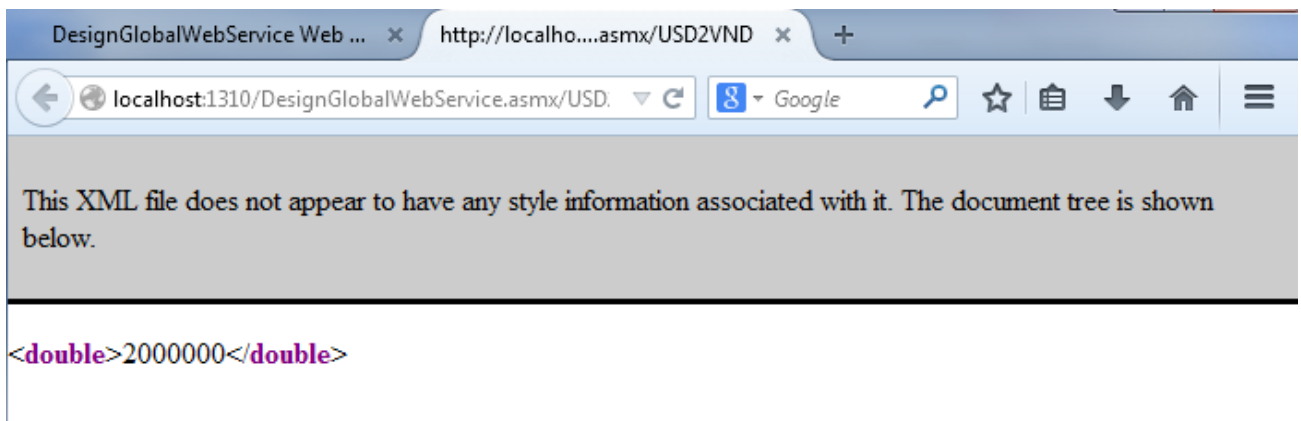
Để thử service, ta có thể chọn bất kỳ link nào trong 4 link EUR2VND, USD2VND VND2EUR, VND2USD. Ở đây ta thử link USD2VND, kết quả như sau



The screenshot shows a web browser window with the address bar displaying `localhost:1310/DesignGlobalWebService.asmx?op=`. The page title is "DesignGlobalWebService". Below the title, there is a link: "Click [here](#) for a complete list of operations." The main section is titled "USD2VND" and contains a "Test" section. The "Test" section instructs the user to click the 'Invoke' button to test the operation using the HTTP POST protocol. Below this instruction is a table with two columns: "Parameter" and "Value". The table has one row with the parameter "usd:" and the value "100". To the right of the table is an "Invoke" button. Below the "Test" section is a section titled "SOAP 1.1" which states: "The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values."

Parameter	Value
usd:	100

Nhập usd có giá trị 100, nhấn Invoke, kết quả nhận được là



The screenshot shows the same web browser window, but the address bar now displays `http://localhost:1310/DesignGlobalWebService.asmx/USD2VND`. The page content shows a message: "This XML file does not appear to have any style information associated with it. The document tree is shown below." Below this message, the XML response is displayed: `<double>2000000</double>`.

Vậy Web Service đã hoạt động và cung cấp trên Internet để các dịch vụ khách có thể truy vấn và sử dụng.

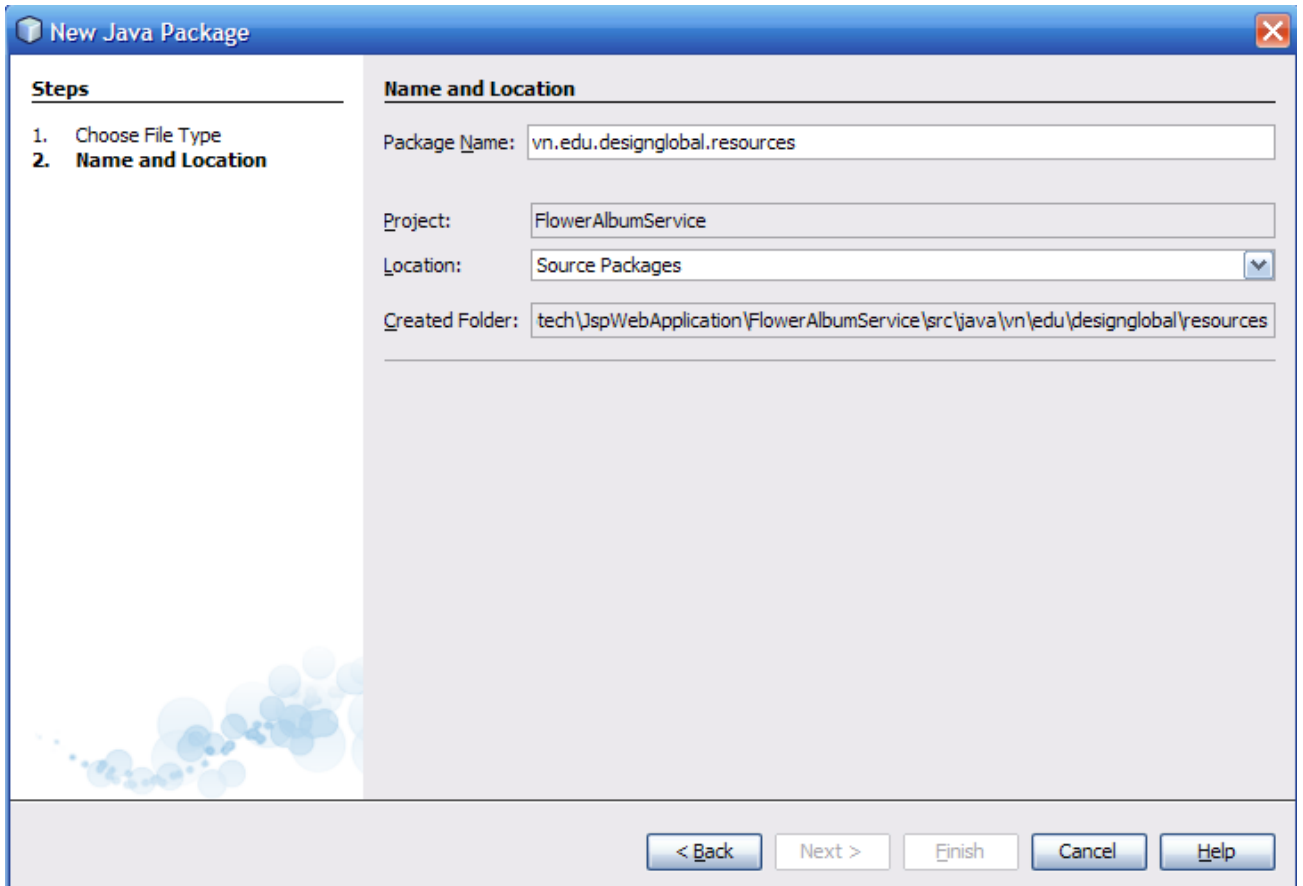
3. Xây dựng Web Service với Java

Công cụ sử dụng bao gồm: Netbeans 7.4 và ngôn ngữ lập trình Java.

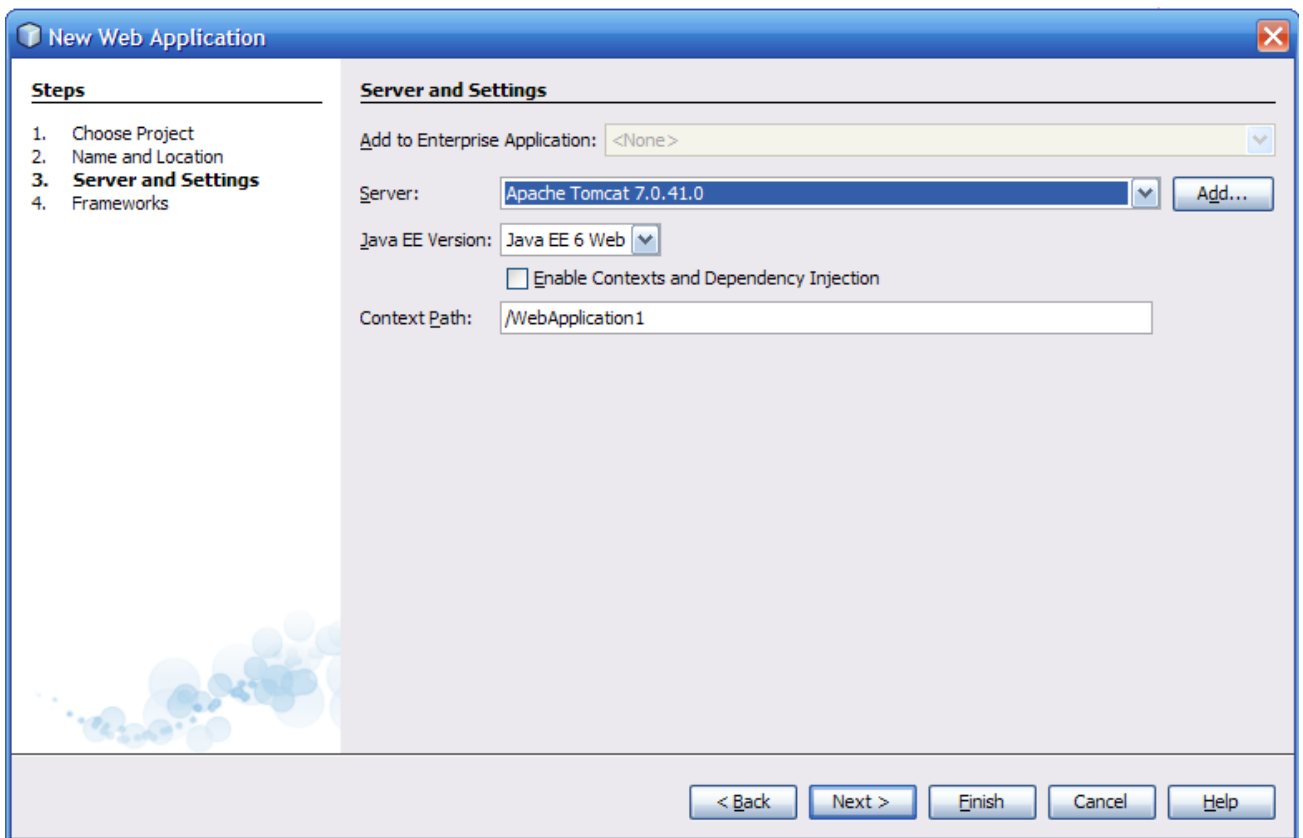
a. Tạo mới JSP Web Application

Mở menu của Netbeans, chọn **File > New Project**, ta chọn mục **Java Web** và chọn **Web Application** ở cửa sổ bên cạnh, chọn **Next**.

Đặt tên FlowerAlbumService cho project, chọn thư mục lưu giữ project. Để các tùy chọn khác mặc định và chọn Next.

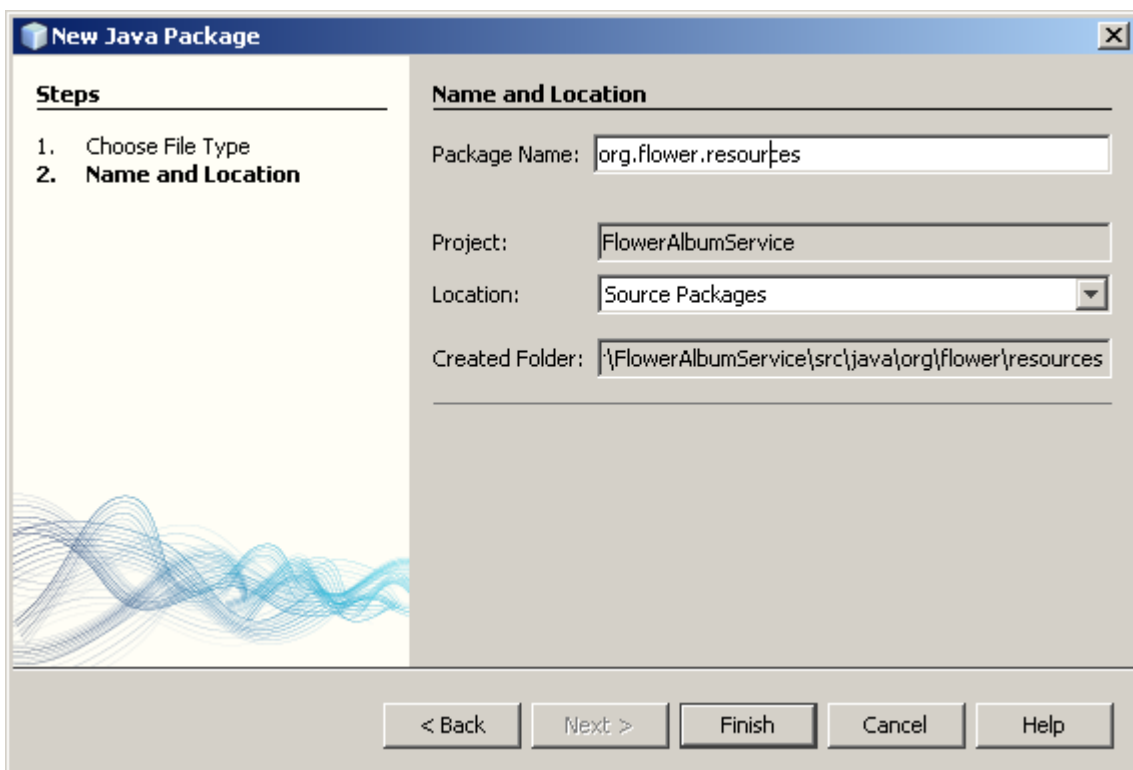


Cửa sổ Server and Settings mở ra, ta chọn **GlassFish** server and **Java EE** version **Java EE 6 Web** hoặc **Java EE 7 Web**. Chọn Finish, FlowerAlbumService project được tạo và trong cửa sổ Project ta thấy như sau



b. Thêm Resource vào project

Chọn chuột phải vào Source Packages và chọn New > Java Package

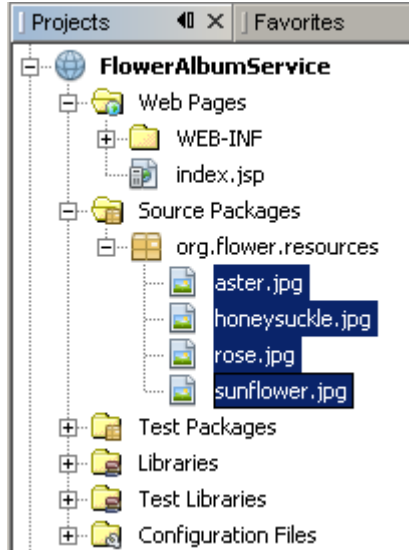


Đặt tên vn.edu.designglobal.resources và chọn Finish.

Ta copy các ảnh sau (tìm trong phần tài nguyên của giáo trình) vào trong mục Resources

- rose.jpg
- sunflower.jpg
- aster.jpg
- honeysuckle.jpg

Ta sẽ thấy như hình sau



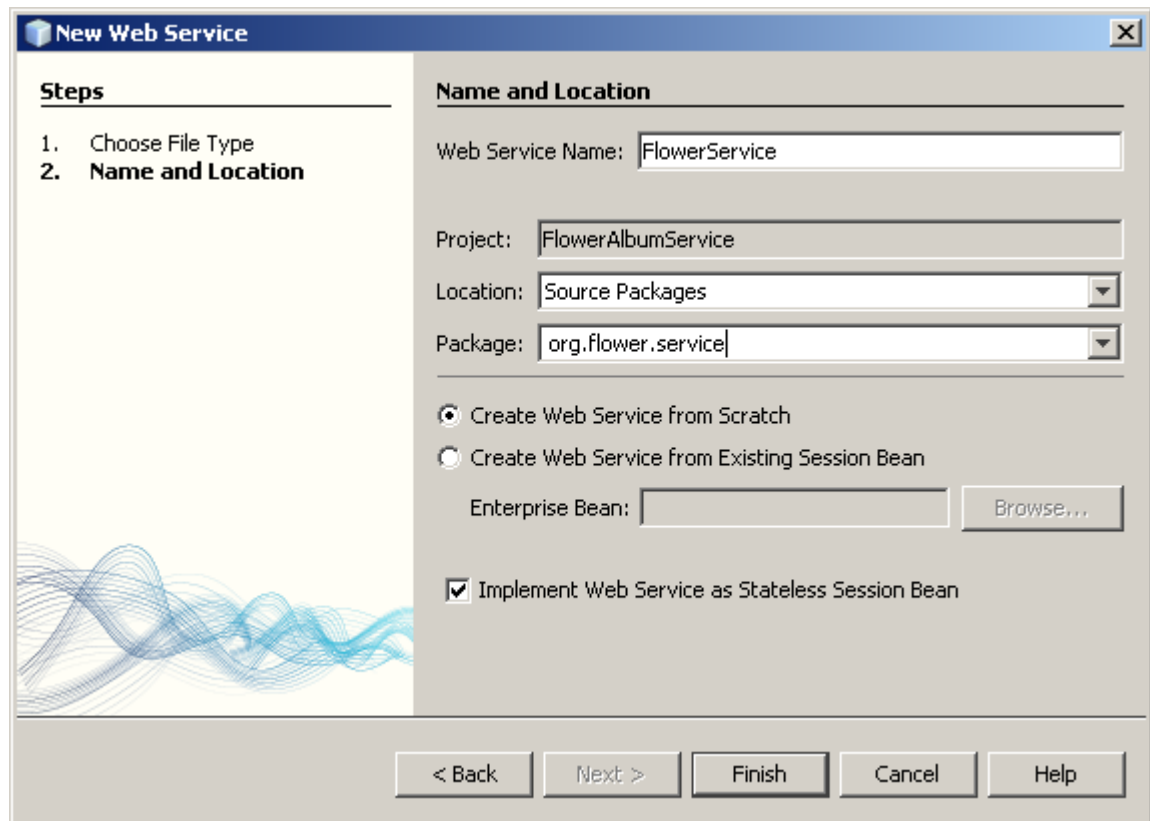
c. Thêm mới Web Service

Trong phần này, ta sẽ tạo Web service kèm với thuộc tính *Stateless session bean*, theo các bước chính sau

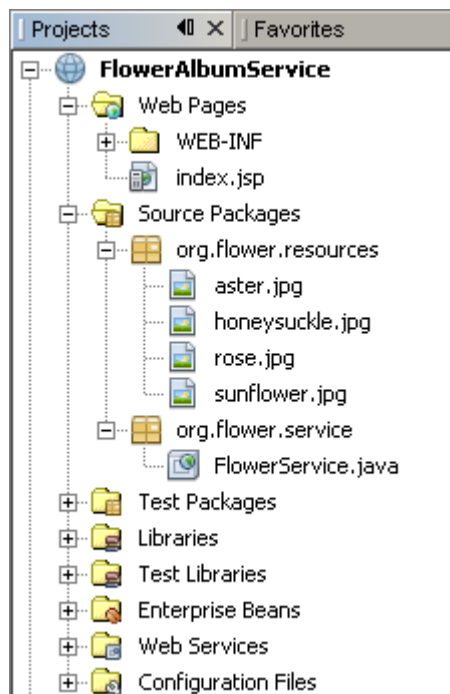
- Tạo hàm cho phép lấy tên và đối tượng Image tương ứng
- Tạo hàm cho phép lấy Thumbnails của tất cả danh mục Image và trả về thông qua đối tượng List

Ta thực hiện trình tự các bước sau:

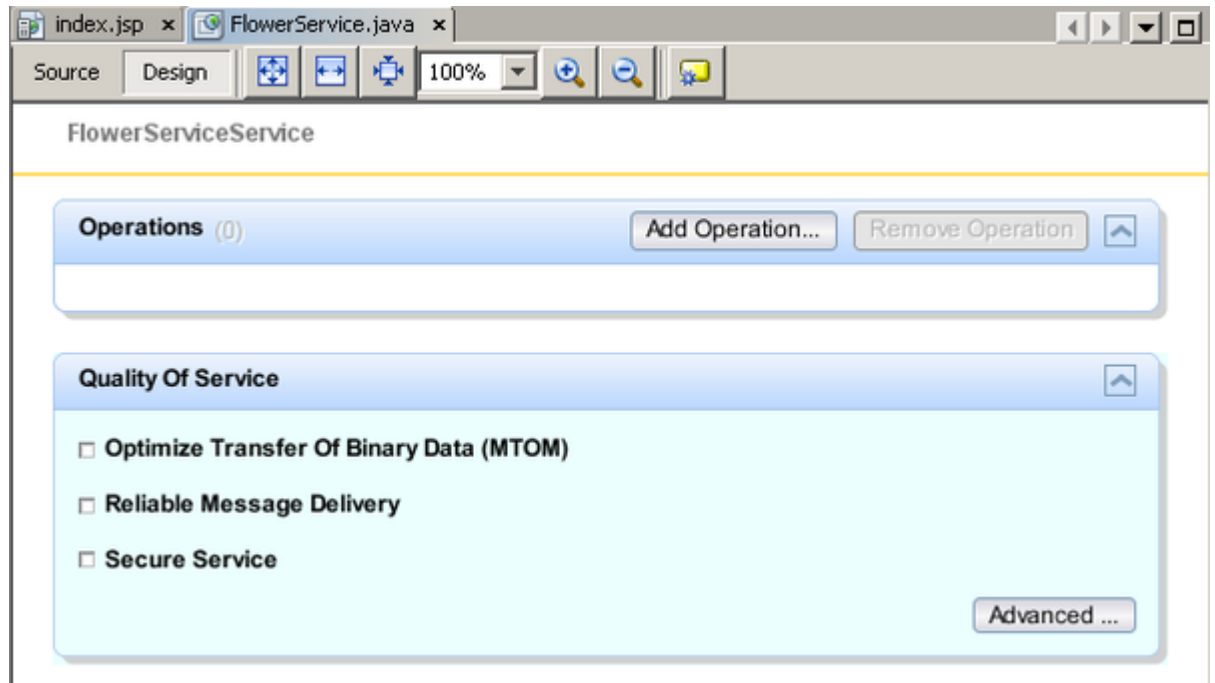
Chọn chuột phải vào project **FlowerAlbumService**, chọn **New > Web Service**. Tại cửa sổ mở ra, ta đặt tên **FlowerService**, chọn package **vn.edu.designglobal.services**. Chọn **Create Web Service from Scratch** và chọn **Implement Service as Stateless Session Bean**. Click chọn Finish để hoàn tất.



Sau khi tạo xong, ta sẽ thấy Web service xuất hiện trong cửa sổ project như hình sau

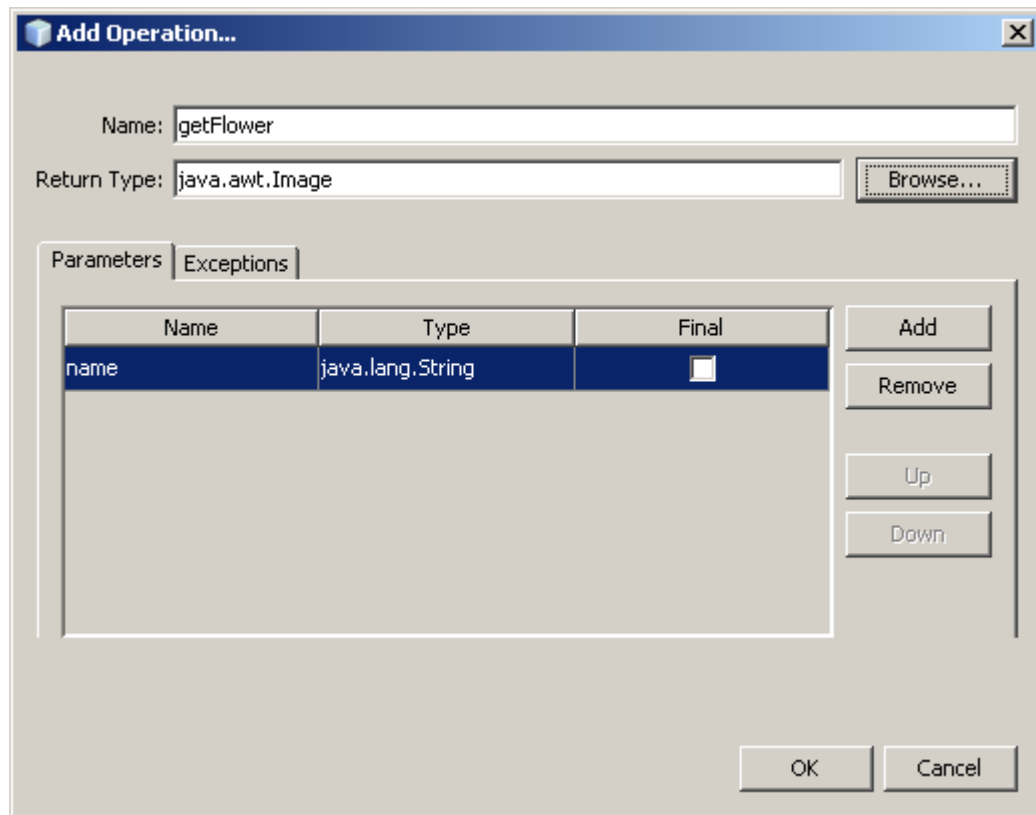


Mở file `FlowerService.java` trong cửa sổ soạn thảo của Netbeans, ta chuyển sang chế độ xem Design. Một cửa sổ xuất hiện cho phép ta thêm các hàm và theo dõi chất lượng (QoS) cho Web Service.

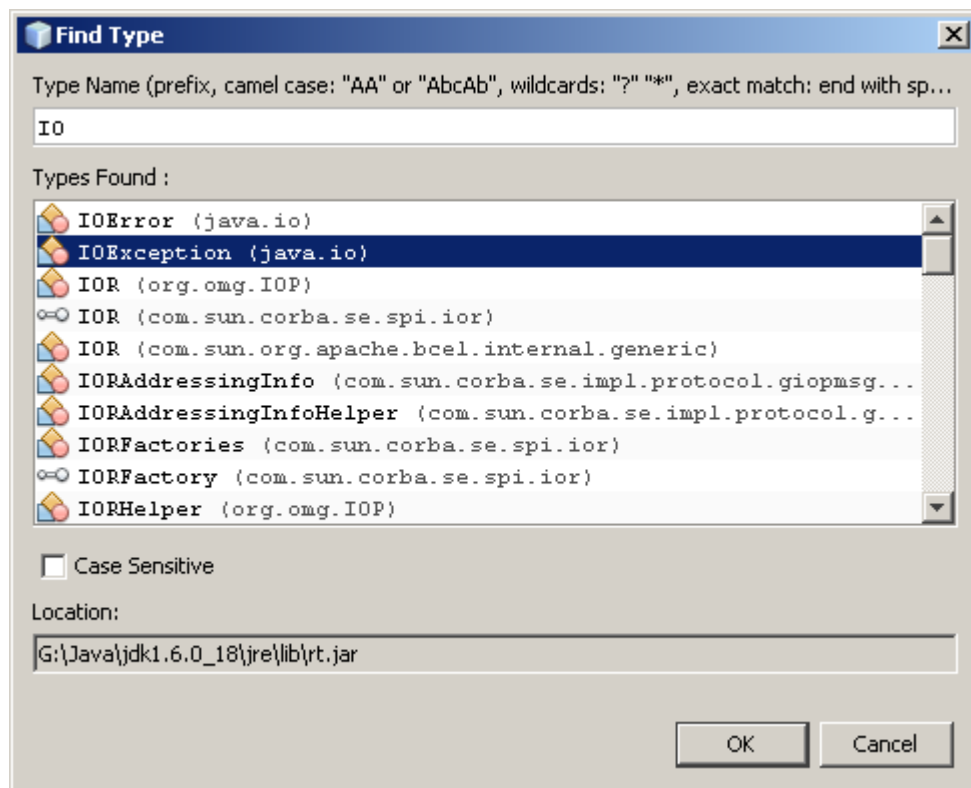


Chọn **Add Operation** để thêm mới các hàm cung cấp bởi dịch vụ.

- Đặt tên **getFlower** cho hàm
- Chọn kiểu dữ liệu trả về **java.awt.Image**.
- Bên thẻ Parameters, chọn **Add**, đặt tên name cho tham số và chọn kiểu dữ liệu **java.lang.String** mặc định. Xem hình sau



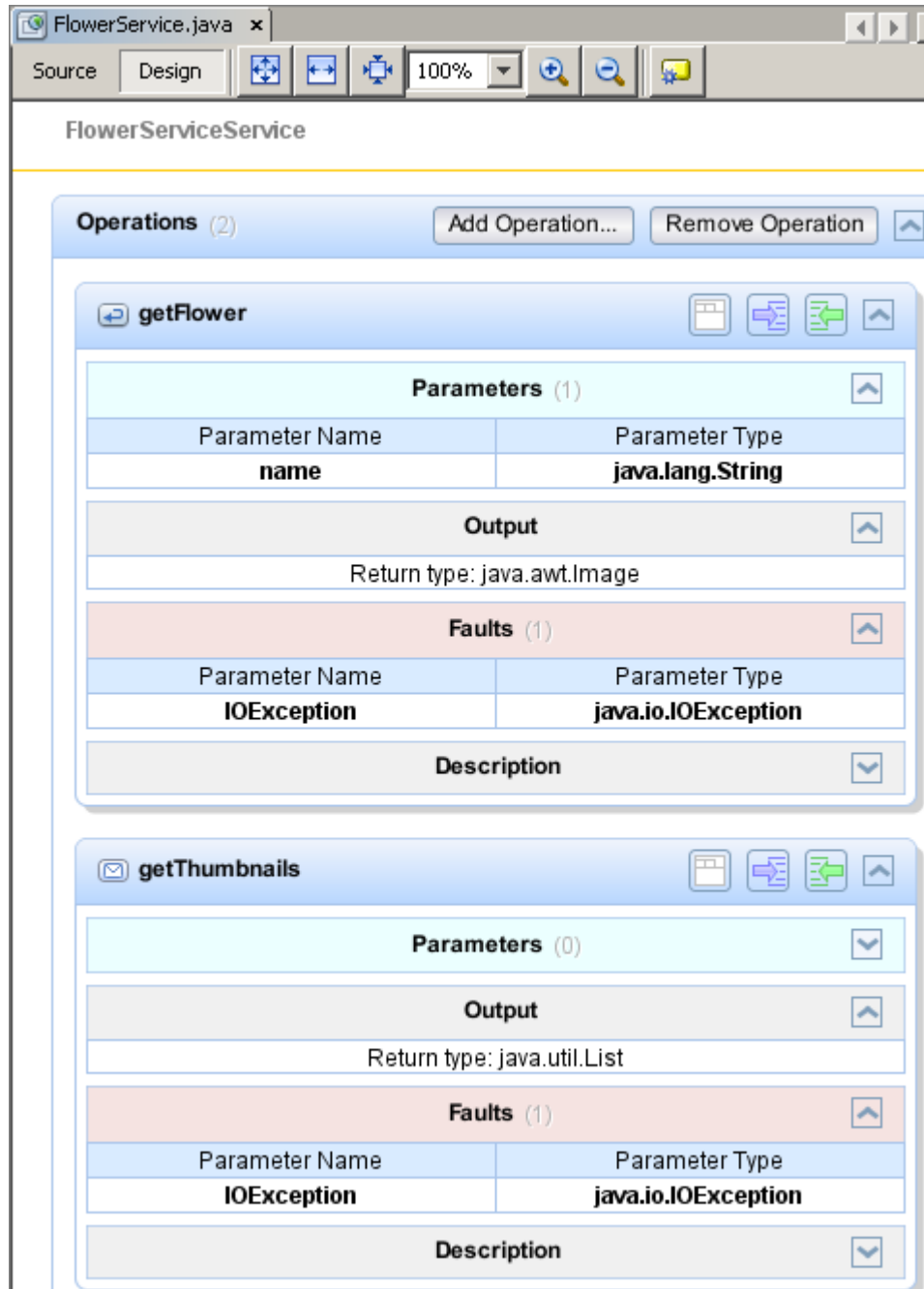
Chọn thẻ **Exceptions**, thêm khai báo ngoại lệ **IOException** cho hàm



Chọn OK và đóng cửa sổ, tương tự, ta sẽ thêm khai báo hàm cho Web service như sau

- **Name:** getThumbnails
- **Return type:** java.util.List
- **Exception:** IOException

Cửa sổ **Design** hiển thị các hàm *getFlower*, *getThumbnails* và khi chuyển sang chế độ xem **List view** ta sẽ thấy như sau



d. Xây dựng hàm cho Web Service

Chuyển sang chế độ *Source View*, ta thêm khai báo sau vào file *FlowerService.java* mã nguồn như sau

- Hàm **getFlower** trả về Image theo tên dịch vụ khách yêu cầu

```
private byte[] getFlowerBytes(String name) throws IOException {  
    URL resource = this.getClass().getResource("/org/flower/resources/"+name+".jpg");  
    return getBytes(resource);  
}
```

Hàm nhận đầu vào là tên ảnh và tìm tới đường dẫn chứa ảnh trong thư mục chứa của Web Service để trả về đối tượng Image tương ứng dưới định dạng mảng byte.

```
private byte[] getBytes(URL resource) throws IOException {  
    InputStream in = resource.openStream();  
    ByteArrayOutputStream bos = new ByteArrayOutputStream();  
    byte[] buf = new byte[1024];  
    for(int read; (read = in.read(buf)) != -1;) {  
        bos.write(buf, 0, read);  
    }  
    return bos.toByteArray();  
}
```

Hàm nhận đầu vào là đường dẫn ảnh vật lý trực tiếp, đọc file theo từng block 1024 byte và lưu vào đối tượng InputStream để trả về dưới định dạng mảng byte.

```
private Image getImage(byte[] bytes, boolean isThumbnail) throws IOException {  
    ByteArrayInputStream bis = new ByteArrayInputStream(bytes);  
    Object source = bis; // File or InputStream  
    ImageInputStream iis = ImageIO.createImageInputStream(source);  
    Iterator readers = ImageIO.getImageReadersByFormatName("jpeg");  
    ImageReader reader = (ImageReader) readers.next();  
    ImageReadParam param = reader.getDefaultReadParam();  
    if (isThumbnail) {  
        param.setSourceSubsampling(4, 4, 0, 0);  
    }  
    reader.setInput(iis, true);  
    return reader.read(0, param);  
}
```

Hàm sẽ tạo đối tượng ***ImageInputStream*** từ mảng byte đầu vào. Và sử dụng đối tượng ***Iterator*** để đăng ký danh sách các đối tượng xử lý ***ImageReader*** giải mã định dạng ảnh ***jpeg***. Nếu hàm yêu cầu tạo ra định dạng ***Thumbnails*** (***param.setSourceSubsampling(4, 4, 0, 0)***) của ảnh thông qua biến ***isThumbnails***. Cuối cùng, ***ImageReader*** sẽ xử lý đối tượng ***ImageInputStream*** để cho ra đối tượng ***Image*** được biểu diễn bởi mảng byte đầu vào.

Thực thi phương thức ***getFlower*** của Web Service như sau

```
@WebMethod(operationName = "getFlower")
public Image getFlower(@WebParam(name = "name") String name) throws IOException {
    byte[] bytes = getFlowerBytes(name);
    return getImage(bytes, false);
}
```

Từ đầu vào là tên ảnh, hệ thống sẽ tìm trong thư mục lưu trữ ảnh với tên tương ứng. Khi tìm thấy, đường dẫn ảnh sẽ được trả lại và hàm ***getFlowersBytes*** sẽ đọc và trả về nội dung ảnh trong mảng byte. Cuối cùng hàm ***getImage*** sẽ chuyển đổi mảng byte về đối tượng ***Image*** trả về cho dịch vụ khách yêu cầu.

- Hàm ***getThumbnails*** trả về danh sách thumbnails ảnh có trên server
Ta thêm khai báo hằng số chứa mảng danh sách tên ảnh cung cấp bởi server

```
private static final String[] FLOWERS = {"aster", "honeysuckle", "rose", "sunflower"};
```

Hàm ***allFlowers*** sẽ đọc danh sách ***Image*** ở trên từ server và trả về trong danh sách mảng byte.

```
private List allFlowers() throws IOException {
    List flowers = new ArrayList();
    for (String flower:FLOWERS) {
        URL resource = this.getClass().getResource("/org/flower/resources/"+flower+".jpg");
        flowers.add(getBytes(resource));
    }
    return flowers;
}
```

Hàm ***getThumbnails*** xử lý như sau

```
@WebMethod(operationName = "getThumbnails")
public List<Image> getThumbnails() throws IOException {
    List<byte[]> flowers = allFlowers();
    List<Image> flowerList = new ArrayList<Image>(flowers.size());
    for (byte[] flower : flowers) {
```

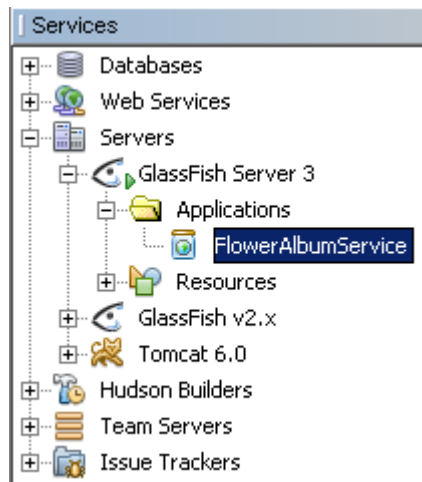


```
flowerList.add(getImage(flower, true));  
}  
return flowerList;  
}
```

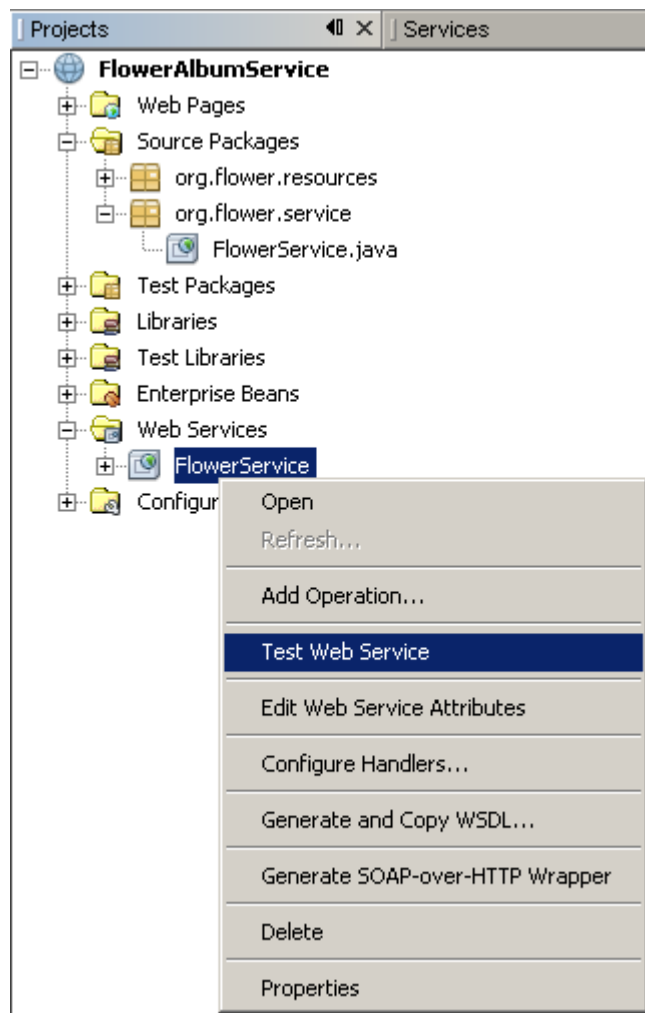
Từ danh sách mảng byte xử lý bởi hàm *allFlowers*, ta sẽ duyệt qua danh sách và chuyển đổi từng mảng byte thành đối tượng *Image* như ở phần trước, các đối tượng Image sẽ được lưu vào danh sách để trả về cho dịch vụ khách yêu cầu.

e. Publish và kiểm tra tính sẵn sàng phục vụ của Web Service

Chọn chuột phải vào *FlowerAlbumService* và chọn *Deploy*. Khi này, Netbeans sẽ khởi động Glashfish server và deploy ra file war của project tới Glashfish. Ta sẽ thấy kết quả như hình sau



Ta mở mục *Web Services* trong khung cửa sổ Project, chọn phải chuột vào *FlowerService* và chọn *Test Web Service*



Trên trình duyệt, ta thấy như sau và nhập “rose” vào ô tham số getFlower.

FlowerServiceService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))


To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract byte[] org.flower.service.FlowerService.getFlower(java.lang.String) throws org.flower.service.IOException_Exception

public abstract java.util.List org.flower.service.FlowerService.getThumbnails() throws org.flower.service.IOException_Exception

Ấn nút getFlower, ta sẽ thấy trên trình duyệt hiển thị như sau.

 **Method invocation trace** ✕ ⊕

Method returned

[B : "[B@2943e5"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:getFlower xmlns:ns2="http://service.flower.org/">
      <name>rose</name>
    </ns2:getFlower>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getFlowerResponse xmlns:ns2="http://service.flower.org/">
      <return>iVBORwOKGgoAAAANSUhEUgAAAOAAAAKACAIAAACDr150AACAAE
    </ns2:getFlowerResponse>
  </S:Body>
</S:Envelope>
```

Vậy Web Service đã sẵn sàng phục vụ.

Tóm tắt bài học

Có 4 giai đoạn chính để xây dựng một dịch vụ Web là xây dựng, triển khai, tiến hành và quản lý.

Có 3 cách tiếp cận chủ yếu để xây dựng nên một dịch vụ Web, có thể từ một ứng dụng đã có (bottom-up); từ một định nghĩa dịch vụ, WSDL để phát sinh một ứng dụng mới (top-down) hoặc có thể từ một nhóm các dịch vụ Web hiện có, kết hợp lại với nhau để tạo nên các chức năng mới hoặc mở rộng thêm chức năng.

Quy trình xây dựng một dịch vụ Web bao gồm các bước sau:

- Định nghĩa và xây dựng các chức năng, các dịch vụ mà dịch vụ sẽ cung cấp (sử dụng ngôn ngữ Java chẳng hạn).

- Tạo WSDL cho dịch vụ
- Xây dựng SOAP server
- Đăng ký WSDL với UDDI registry để cho phép các client có thể tìm thấy và truy xuất.
- Client nhận file WSDL và từ đó xây dựng SOAP client để có thể kết nối với SOAP server
- Xây dựng ứng dụng phía client (chẳng hạn sử dụng Java) và sau đó gọi thực hiện dịch vụ thông qua việc kết nối tới SOAP server.

Lựa chọn một ngôn ngữ, xây dựng các tiến trình nghiệp vụ và chúng ta bắt đầu tạo nên một dịch vụ Web như ý muốn. Sau đó là cung cấp dịch vụ Web này trên Internet.

Bài tập

Tạo 1 cơ sở dữ liệu tên MyAccount gồm 1 bảng tên UserAccount chứa các thông tin sau: UserName_(char 15), Password (char 100), BirthDate (DateTime), Address (varchar 50), Email (varchar 50) .

Tạo 1 Webservice thực hiện các chức năng sau :

- AddNewAccount (...) : phương thức này dùng để thêm mới 1 Account và Password được mã hóa theo thuật toán (MD5 hoặc SHA1, tùy chọn), trước khi thêm cần kiểm tra Account có tồn tại hay không thông qua phương thức CheckAccount.
- DeleteAccount (...): dùng để xóa 1 Account
- UpdateAccount(...) : cập nhật thông tin 1 Account.
- GetAccount() : lấy tất cả thông tin của các Account.
- CheckAccount(...) : Kiểm tra 1 Account có tồn tại hay không

CHƯƠNG IV: KHAI THÁC WEB SERVICE

Mục tiêu:

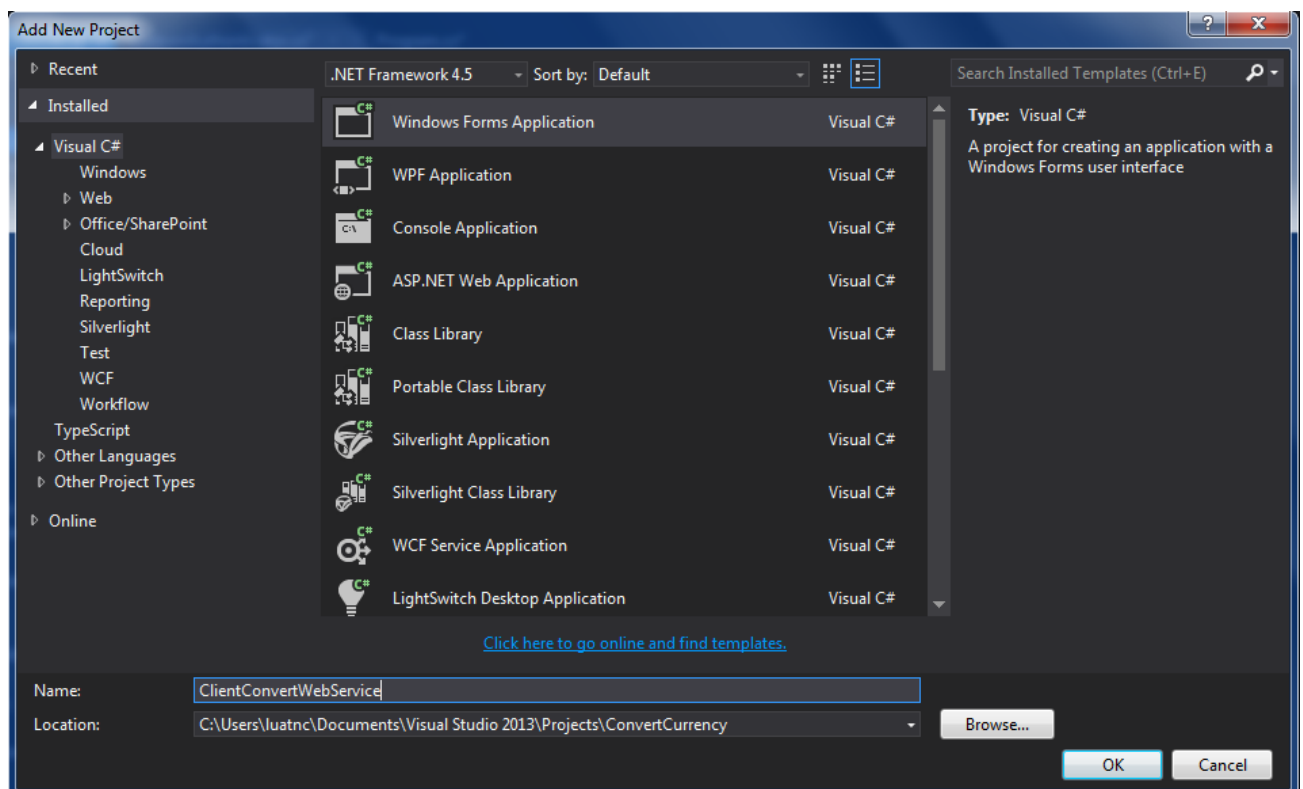
Sau khi kết thúc chương này, bạn có thể:

- Hiểu về cách khai thác các dịch vụ Web đã được cung cấp để xây dựng ứng dụng
- Nắm vững cách sử dụng công cụ hỗ trợ, ngôn ngữ lập trình để khai thác trên các môi trường khác nhau như C#, Java và các dạng thức ứng dụng khác nhau như Application Form, Web Form

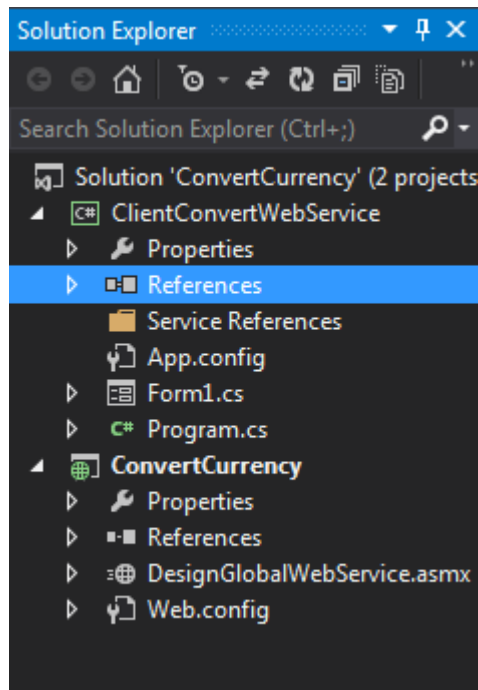
1. Ứng dụng Window Form kết nối tới Web Service

a. Tạo mới ứng dụng Windows Forms Application

Tạo ứng dụng Windows Application bằng cách nhấn phải chuột lên Solution trong *Solution Explorer*, chọn *Add > New Project*



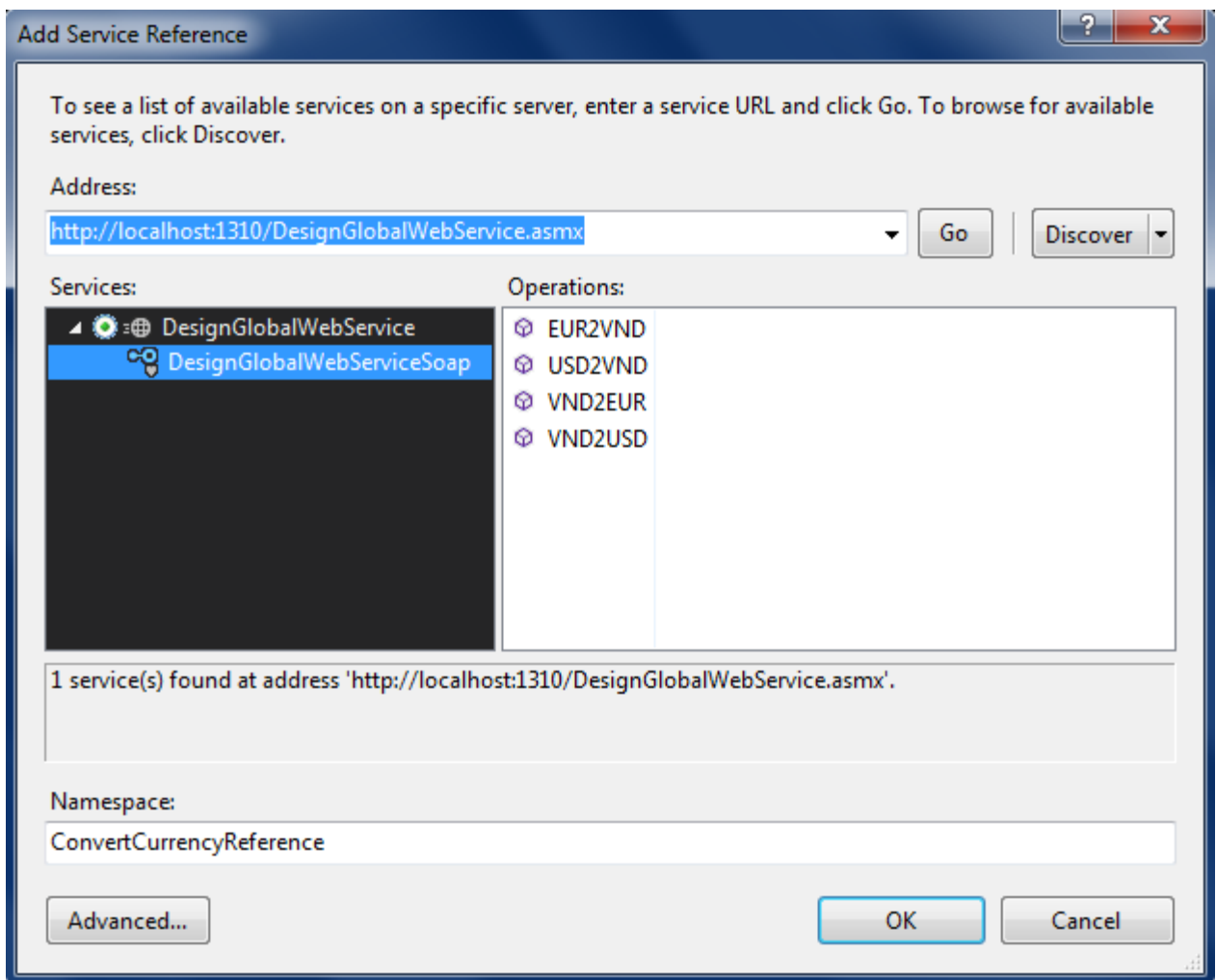
Chọn Template “*Windows Form Application*”, đặt tên là *ConsumeConvertWS*, nhấn OK, ta được



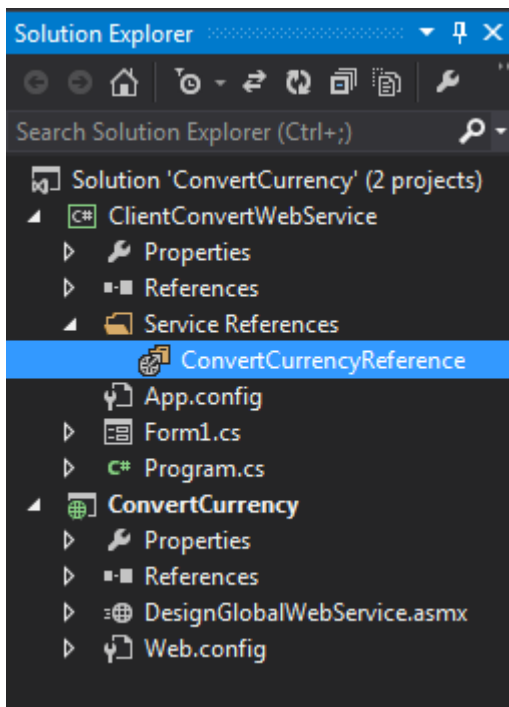
Nhấn chuột phải lên project vừa tạo, chọn ***“Set as Start Up Project”*** để đảm bảo project vừa tạo là project sẽ được thực thi khi nhấn F5.

b. Thêm tham chiếu tới Web Service vào project

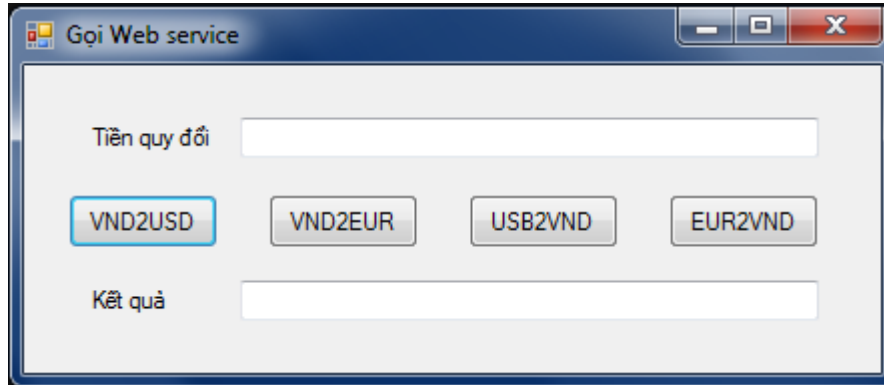
Chọn chuột phải vào project, chọn ***“Add Web Reference...”***. Gõ vào URL đến WSDL vào ô URL, nhấn nút Go, kết quả như sau



Đặt tên cho là *ConvertCurrencyReference*, ấn OK, ta sẽ thấy kết quả như sau



c. Tạo giao diện ứng dụng đơn giản



Đặt tên cho TextBox là txtMoney, Label hiển thị kết quả là txtKetQua.

Thêm đoạn mã thực thi chương trình như sau

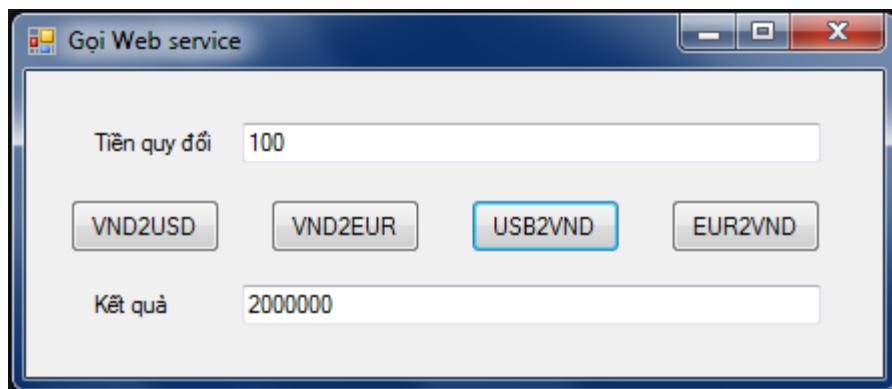
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace ConsumeConvertWS
{
    public partial class Form1 : Form
    {
        private ConsumeConvertWS.convertWS.ConvertWS ws;
        public Form1()
        {
            InitializeComponent();
            ws = new ConsumeConvertWS.convertWS.ConvertWS();
        }
        private void VND2USD_Click(object sender, EventArgs e)
        {
            double mon = Double.Parse(txtMoney.Text);
            double usd = ws.VND2USD(mon);
            lblKetQua.Text = usd.ToString();
        }
    }
}
```

```
private void VND2EUR_Click(object sender, EventArgs e)
{
    double mon = Double.Parse(txtMoney.Text);
    double usd = ws.VND2EUR(mon);
    lblKetQua.Text = usd.ToString();
}
private void USD2VND_Click(object sender, EventArgs e)
{
    double mon = Double.Parse(txtMoney.Text);
    double usd = ws.USD2VND(mon);
    lblKetQua.Text = usd.ToString();
}
private void EUR2VND_Click(object sender, EventArgs e)
{
    double mon = Double.Parse(txtMoney.Text);
    double usd = ws.EUR2VND(mon);
    lblKetQua.Text = usd.ToString();
}
}
```

d. Kiểm tra hoạt động ứng dụng kết nối tới Web Service

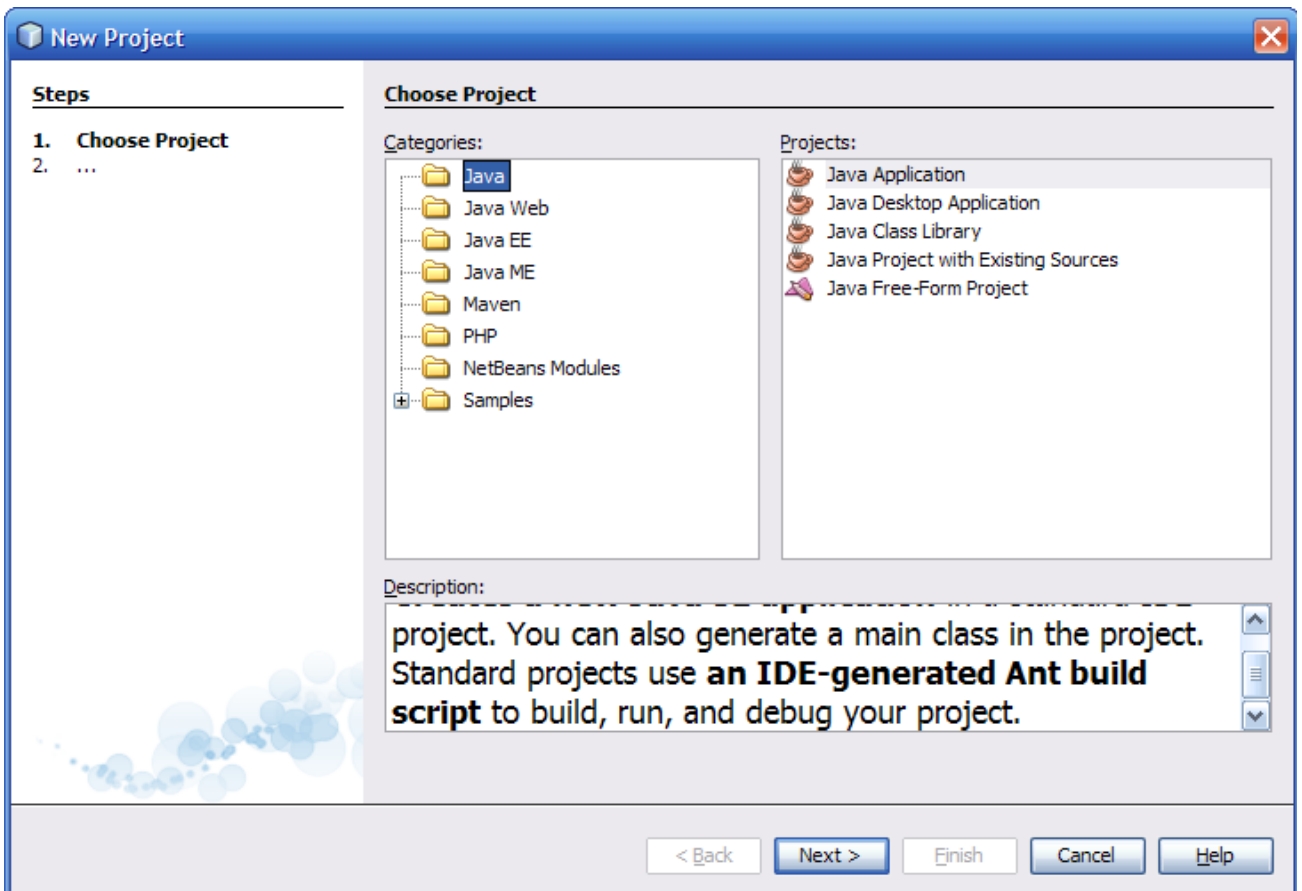
Ta chọn F5 để thực thi ứng dụng, nhập 100 vào ô Tiền quy đổi, click nút USB2VND. Kết quả, ta thấy như sau



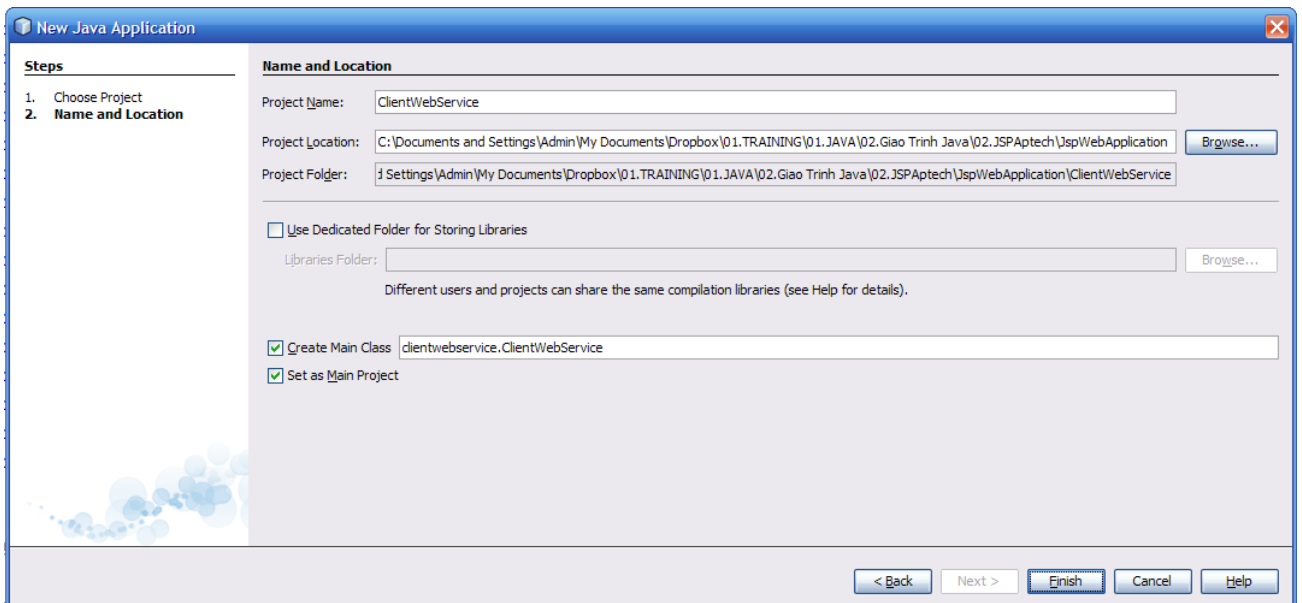
2. Ứng dụng Java Swing kết nối tới Web Service

a. Tạo mới ứng dụng Java Application

Chọn từ menu Netbeans, chọn **File > New Project**, chọn mục **Java** và **Java Application** bên cạnh.

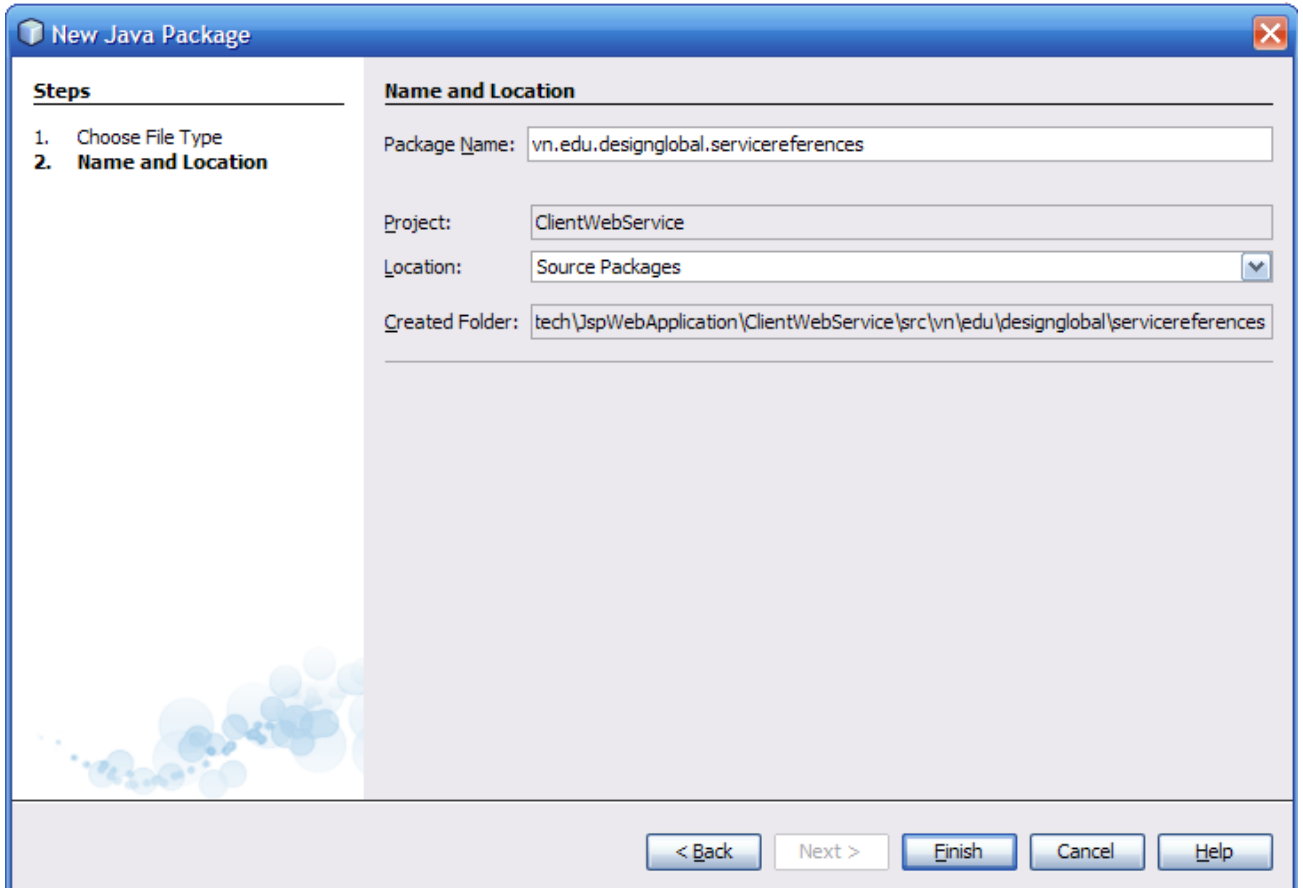


Cửa sổ New Java Application mở ra, ta đặt tên ClientWebService cho ứng dụng và thiết lập *Set as Main Project*.

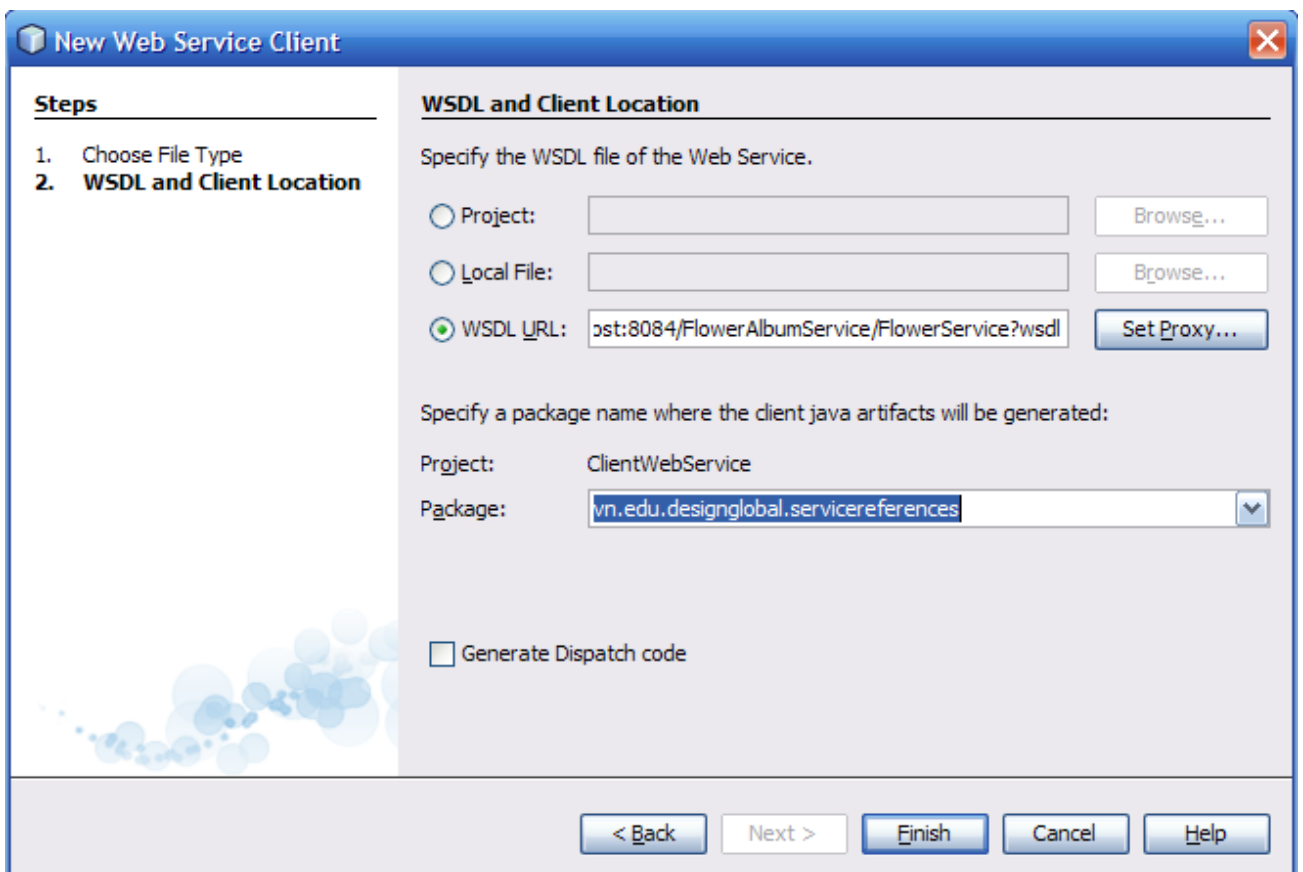


b. Thêm tham chiếu tới Web Service

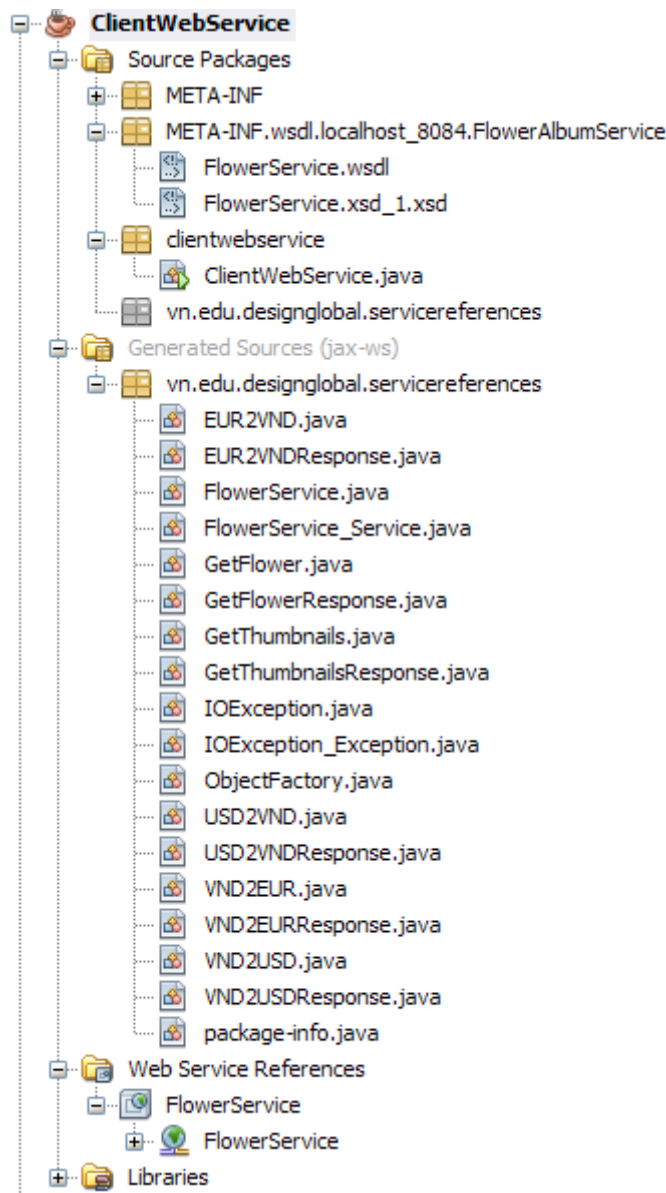
Ta tạo mới package cho project, chọn chuột phải vào project, chọn **New > Java Package**, cửa sổ mở ra như hình sau. Đặt tên ***vn.edu.designglobal.servicereferences*** cho package.



Ta thêm tham chiếu tới Web Service cung cấp chức năng chuyển đổi tiền tệ ở trên. Chọn chuột phải vào project, chọn **New > Web Service Client** ; lựa chọn mục **WSDL URL** và nhập đường dẫn publish của Web Service ; chọn package để lưu giữ tham chiếu của Web Service như hình sau



Sau khi thêm thành công ta sẽ thấy như hình sau



c. Thiết kế giao diện ứng dụng

Thiết kế giao diện ứng dụng như hình sau và F5 để kiểm tra hoạt động ứng dụng, ta sẽ thấy kết quả như sau



Đoạn mã nguồn thực thi chương trình như bên dưới

```
public class ClientWebService extends JFrame implements ActionListener {
    private static final long serialVersionUID = 1L;
    private JTextField tfMoney, tfDisplay;
    private JButton btnVND2USD, btnVND2EUR, btnUSD2VND, btnEUR2VND;

    public ClientWebService() {
        setTitle("Gọi Web service");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(450, 150);
        setResizable(false);
        JLabel lblTitle = new JLabel("CHUYỂN ĐỔI TIỀN TỆ", JLabel.CENTER);
        lblTitle.setFont(new Font("Arial", Font.BOLD, 18));
        this.add(lblTitle, BorderLayout.NORTH);
        createGUI();
    }

    void createGUI() {
        JPanel pCen = new JPanel(new GridLayout(2, 1));
        JPanel pBot = new JPanel();
        this.add(pBot, BorderLayout.SOUTH);
        this.add(pCen, BorderLayout.CENTER);
        JPanel p1 = new JPanel();
        JLabel l1, l2;
        p1.add(l1 = new JLabel("Tiền qui đổi:", JLabel.RIGHT));
        p1.add(tfMoney = new JTextField(25));
        pCen.add(p1);
        JPanel p2 = new JPanel();
        p2.add(l2 = new JLabel("Kết quả:", JLabel.RIGHT));
        p2.add(tfDisplay = new JTextField(25));
        tfDisplay.setEditable(false);
        pCen.add(p2);
        l2.setPreferredSize(l1.getPreferredSize());
        pBot.add(btnVND2USD = new JButton("VND2USD"));
        pBot.add(btnVND2EUR = new JButton("VND2EUR"));
        pBot.add(btnUSD2VND = new JButton("USD2VND"));
        pBot.add(btnEUR2VND = new JButton("EUR2VND"));
        btnVND2USD.addActionListener(this);
    }
}
```

```
        btnVND2EUR.addActionListener(this);
        btnUSD2VND.addActionListener(this);
        btnEUR2VND.addActionListener(this);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        Object o = e.getSource();
        DesignGlobalWebService stub = new DesignGlobalWebService();
        DesignGlobalWebServiceSoap ws = stub.getDesignGlobalWebServiceSoap();

        double dong = Double.parseDouble(tfMoney.getText());
        if (o.equals(btnVND2USD)) {
            try {
                double result = ws.vnd2USD(dong);
                tfDisplay.setText(result + "");
            } catch (Exception e1) {
                e1.printStackTrace();
            }
        } else if (o.equals(btnVND2EUR)) {
            try {
                double result = ws.vnd2EUR(dong);
                tfDisplay.setText(result + "");
            } catch (Exception e1) {
                e1.printStackTrace();
            }
        } else if (o.equals(btnUSD2VND)) {
            try {
                double result = ws.usd2VND(dong);
                tfDisplay.setText(result + "");
            } catch (Exception e1) {
                e1.printStackTrace();
            }
        } else if (o.equals(btnEUR2VND)) {
            try {
                double result = ws.eur2VND(dong);
                tfDisplay.setText(result + "");
            }
        }
    }
}
```



```

    } catch (Exception e1) {
        e1.printStackTrace();
    }
}
}

public static void main(String[] args) {
    new ClientWebService().setVisible(true);
}
}

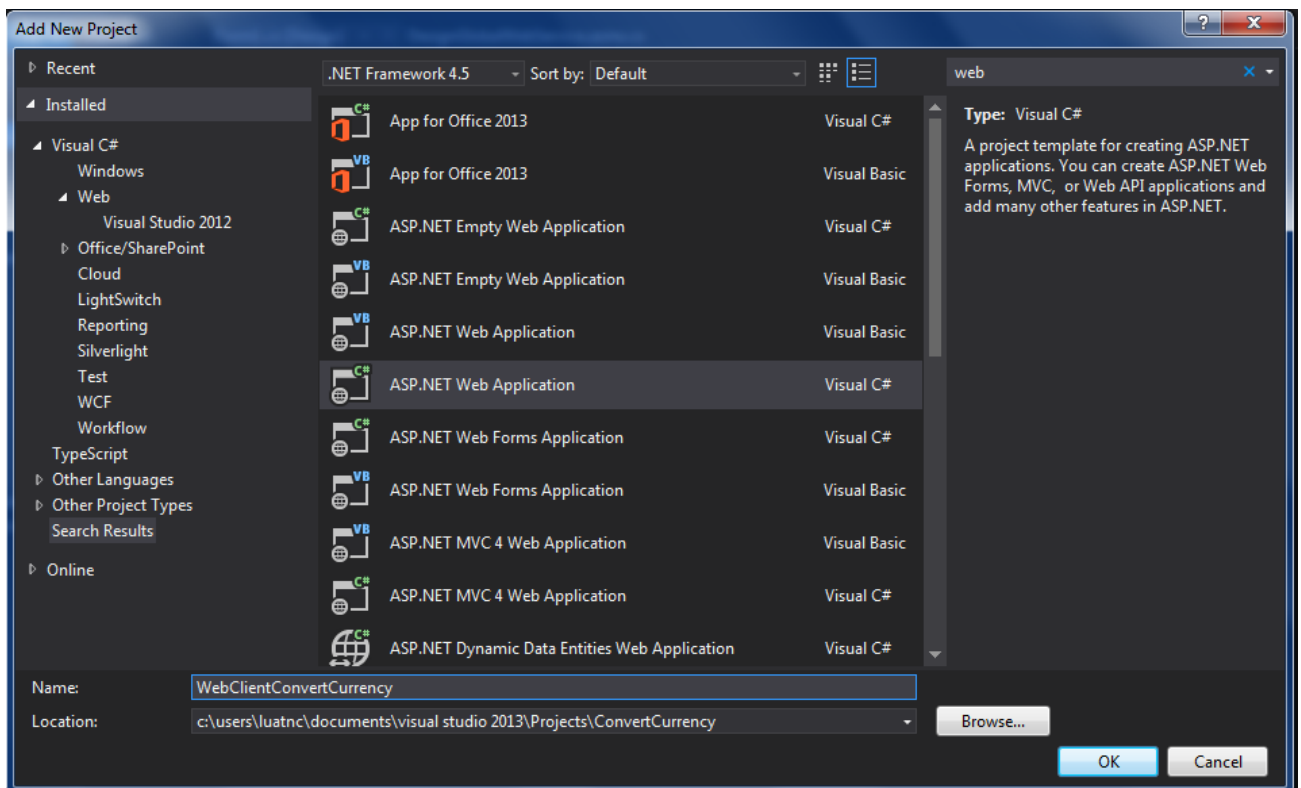
```

Vậy ứng dụng hoạt động như thiết kế mong muốn.

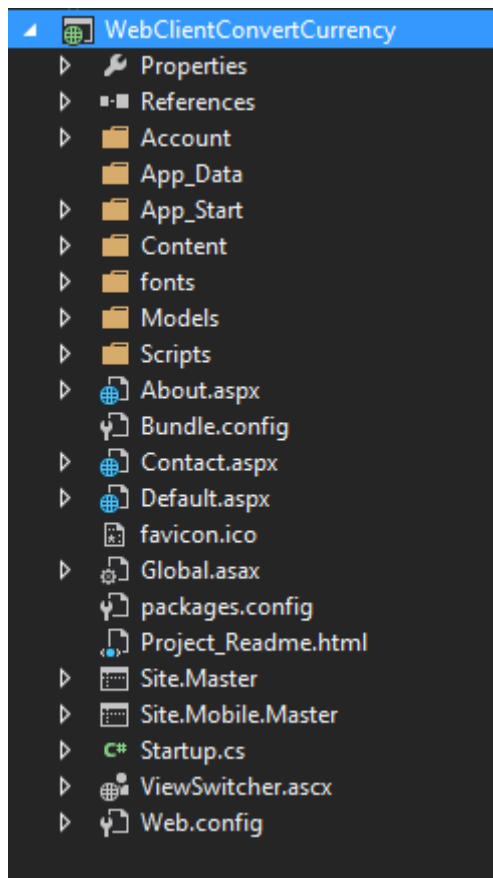
3. Ứng dụng Web ASP.NET kết nối tới Web Service

a. Tạo mới ứng dụng ASP.NET Web Application

Chọn chuột phải solution trong cửa sổ Solution Explorer, chọn mục ASP.NET Web Application, đặt tên WebClientConvertCurrency và chọn thư mục lưu giữ project như hình sau

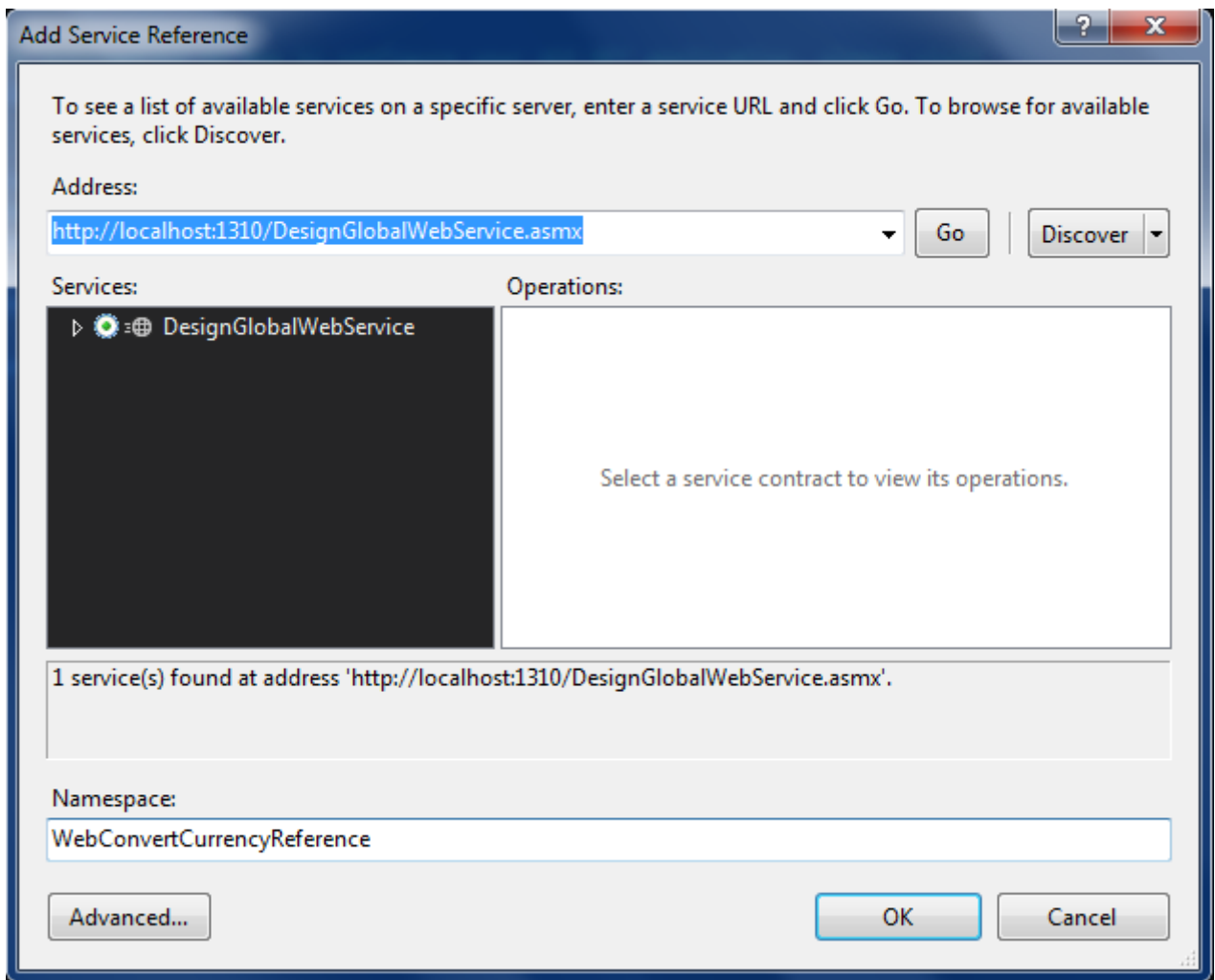


Visual Studio sẽ tạo project với cấu trúc như hình sau

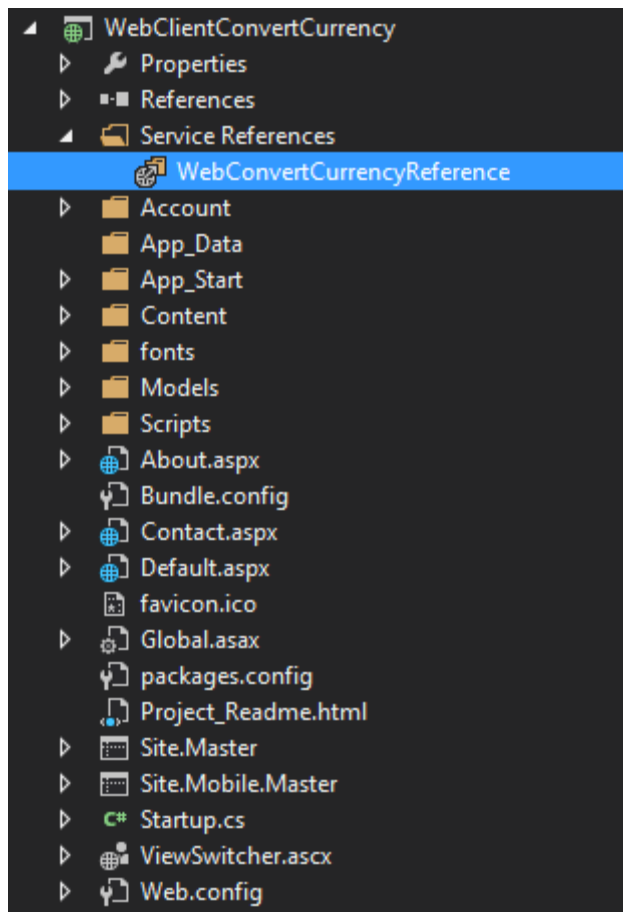


b. Thêm tham chiếu tới Web Service

Chọn chuột phải vào project, chọn Add reference để mở cửa sổ thêm tham chiếu tới Web Service như hình sau



Nhập địa chỉ publish của Web Service cần tham chiếu, chọn **Go**, ta sẽ thấy như hình trên và đặt tên **WebConvertCurrencyReference**. Ấn nút OK, ta sẽ thấy như hình sau



c. Thiết kế ứng dụng Web Application

Thiết kế ứng dụng như hình sau, và chạy thực thi ứng dụng kiểm tra với dữ liệu như hình bên. (mã nguồn tham khảo trong phần Tài nguyên của giáo trình)

Tiền quy đổi	<input type="text" value="100"/>
Kết quả	<input type="text" value="2000000"/>
<div><button>VND2USD</button><button>VND2EUR</button><button>USD2VND</button><button>EUR2VND</button></div>	

Vậy ứng dụng Web hoạt động như mong muốn.

Tóm tắt bài học

Trong chương này, ta đã học về cách khai thác các dịch vụ Web được cung cấp trên các môi trường khác nhau như .NET Web Service – IIS Server và Java Web Service – Glashfish Server (xây dựng trong chương III).

Ta cũng đã học cách xây dựng các ứng dụng trên cơ sở khai thác các dịch vụ Web Service như:

- .Net Application Form
- Java Application Form
- ASP.NET Web Application

Bài tập

Tạo 1 ứng dụng ASP.Net, sử dụng Webservice (bài tập chương III) gồm các trang mô tả như sau :

- Trang Login.aspx : kiểm tra Account, trang này link với trang CreateAccount.aspx. Nếu login thành công thì chuyển sang trang MaintainAccount.aspx.
- Trang CreateAccount.aspx : dùng để thêm 1 Account. Trên trang này có 1 button dùng để kiểm tra 1 UserName có tồn tại ? mà không cần Postback về Server. Trang này link với trang Login.aspx
- Trang MaintainAccount.aspx : trang này gồm 1 GridView cho phép xem , xóa , sửa thông tin Account.

CHƯƠNG V: BẢO MẬT TRONG WEB SERVICE

Mục tiêu:

Sau khi kết thúc chương này, bạn có thể:

- Hiểu về vấn đề bảo mật trên Internet và Web Service đáp ứng chuẩn an toàn cơ bản
- Hiểu về một số kiểu giả mạo, đánh cắp thông tin và cách phòng chống
- Thực hiện bảo mật Web Service với thư viện WSE 3.0 do Microsoft cung cấp.
- Xây dựng dịch vụ Web bảo mật với Visual Studio - ASP.NET, triển khai và công bố dịch vụ Web.
- Đồng thời xây dựng ứng dụng để khai thác và sử dụng chuẩn kết nối bảo mật vừa xây dựng.

1. Tổng quan về vấn đề bảo mật

Cùng với sự phát triển không ngừng của Internet, hệ thống thông tin ngày một phát triển vượt bậc. Khái niệm business không còn giới hạn ở bên trong mà đã phát triển cả ở trên Internet.

Khái niệm thương mại điện tử ra đời là sự minh chứng cho sự phát triển đó. Tuy nhiên, đi cùng với sự phát triển đó thì vấn đề an toàn của nó cũng là một vấn đề nóng và cấp bách. Tuy không có khái niệm về sự an toàn tuyệt đối nhưng cũng phải cần một cơ chế an toàn thích hợp để khai thác thương mại trên Internet. Các giải pháp hiện nay đang được sử dụng như mã hóa khóa, chữ ký số có thể đảm bảo ở một mức nào đó. Nhưng cũng phải cần không ngừng hoàn thiện và phát triển các giải pháp đó để đảm bảo phù hợp hơn trên môi trường Internet.

An toàn thông tin trên Internet là một vấn đề chung hiện nay. Sự an toàn web service lại càng cần được sự quan tâm hơn nữa, khi các thông tin nhạy cảm như tài khoản cá nhân ở ngân hàng có thể bị đánh cắp. Điều gì xảy ra nếu sử dụng business service mua hàng, chứng khoán, chuyển tiền mà không có một sự bảo đảm an toàn cần thiết. Đây là một chuẩn an toàn chung cần thiết cần được đáp ứng:

- **Identification**: định danh được những ai truy cập tài nguyên hệ thống.
- **Authentication**: chứng thực tư cách truy cập tài nguyên của người muốn sử dụng.
- **Authorization**: cho phép giao dịch khi đã xác nhận định danh người truy cập.

- **Integrity**: toàn vẹn thông tin trên đường truyền.
- **Confidentiality**: độ an toàn, không ai có thể đọc thông tin trên đường đi.
- **Auditing**: kiểm tra, tất cả các giao dịch đều được lưu lại để kiểm tra.
- **Non-repudiation**: độ mềm dẻo, cho phép chứng thực tính hợp pháp hóa của thông tin đến từ một phía thứ ba ngoài 2 phía là người gửi và người nhận.

Những yêu cầu trên giúp cho hệ thống an toàn hơn, tránh được phần nào những truy cập không hợp lệ.

HTTP (HyperText Transfer Protocol) là giao thức thường sử dụng nhất cho việc trao đổi thông tin trên Internet, tuy nhiên không an toàn, bởi vì tất cả thông tin được gửi dưới dạng văn bản trong mạng ngang hàng không an toàn. HTTP thuộc về nhóm của những nghi thức như SMTP, telnet, và FTP, được thiết kế trong giai đoạn đầu của Internet khi mà vấn đề an toàn chưa được quan tâm đến nhiều. Một phát triển của HTTP là HTTPS, nó là một chuẩn an toàn cho HTTP. HTTPS cho phép chứng thực client và server qua những chứng thực giữa client và server.

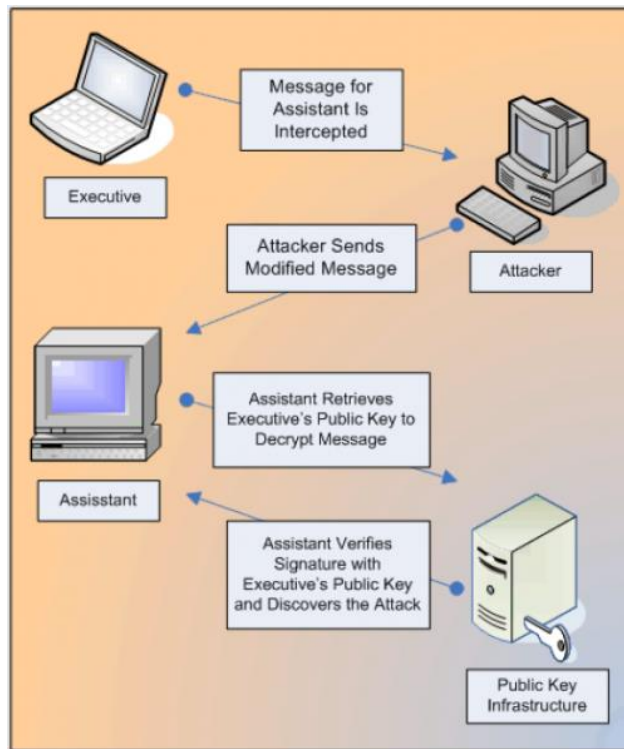
Trước khi có web services security (WS-Security) thì ý nghĩa thông thường của an toàn web service là bảo mật kênh truyền dữ liệu. Nó được thực hiện cho những SOAP/HTTP dựa trên cơ chế truyền thông điệp bằng cách sử dụng giao thức HTTPS. Không giống sự an toàn mức thông điệp, HTTPS cung cấp sự an toàn tới toàn bộ gói dữ liệu HTTP. Bởi vậy, chúng ta không có một tùy chọn nào để áp dụng sự an toàn có chọn lọc chỉ trên những thành phần của một thông điệp.

Mặc dầu HTTPS không bao phủ tất cả các khía cạnh trong chuẩn an toàn chung nhưng nó cũng đã cung cấp một mức bảo chứng đầy đủ với định danh và chứng thực, sự toàn vẹn thông điệp, và độ tin cậy. Tuy nhiên, authentication, auditing, and nonrepudiation chưa được cung cấp. Bên cạnh đó, HTTPS là một giao thức nên khi thông điệp đi qua HTTP server thì lại không an toàn.

2. Một số kiểu giả mạo, đánh cắp thông tin và cách phòng chống:

2.1. Message Replay Attack

Message Replay là một kiểu tấn công trên mạng kiểu bắt và gửi lại gói tin mà client đã gửi cho server.



Trong quá trình này Attacker như một trung gian của client và server, nó bắt gói tin client và sao chép gói tin đó, chỉnh sửa và gửi lại cho server. Và nếu server có gửi lại cho client thì Attacker cũng có thể bắt được.

Ta phải ngăn chặn việc gửi lại gói tin kiểu này bởi vì các gói tin gửi đi sẽ gây ra không nhất quán dữ liệu, các thông tin gửi đi bị sai lệch ảnh hưởng đến client, việc gửi replayed message liên tục và một cách tự động sẽ làm chết server.

Giải pháp cho vấn đề này là sử dụng cache lưu lại tên định danh cho message và server sẽ loại bỏ các message có định danh bị trùng. Như vậy các message có một tên định danh duy nhất được gửi đi và chắc chắn rằng các message không bị giả mạo trong quá trình truyền.

Web service security đã cung cấp việc sử dụng UsernameToken trong đó có thể username và password.

```

<wsse:UsernameToken wsu:Id="Example-1">
  <wsse:Username>... </wsse:Username>
  <wsse:Password Type="...">... </wsse:Password>
  <wsse:Nonce EncodingType="...">... </wsse:Nonce>
  <wsu:Created>... </wsu:Created>
</wsse:UsernameToken>

```

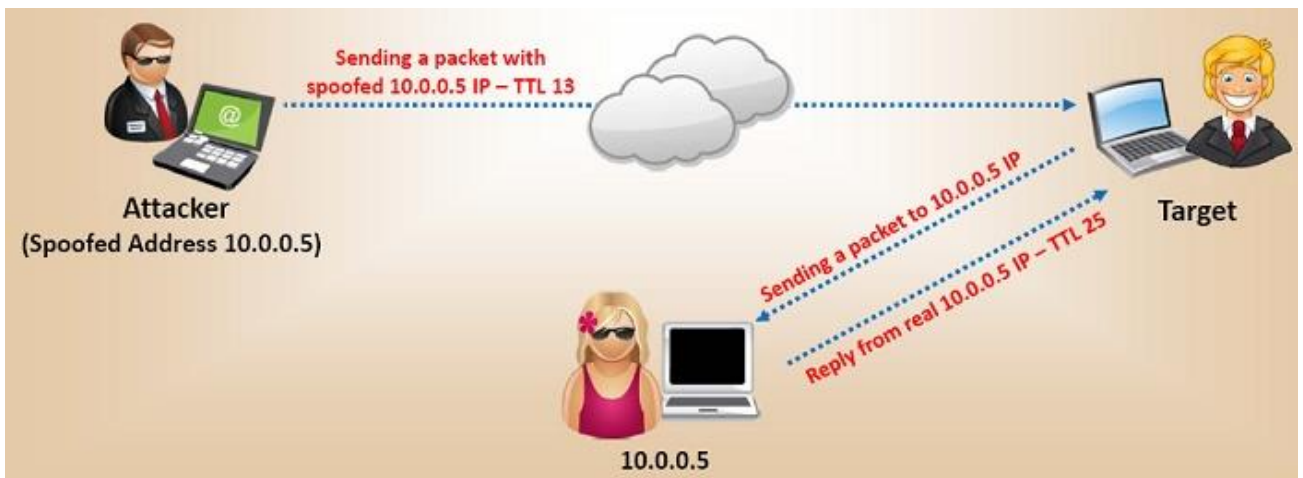
Các giá trị Nonce được tạo ra ngẫu nhiên cùng với Created là thời gian mà message được gửi đi. Các giá trị này đều được Hash để che dấu thông tin.

Giá trị thời gian mà message phải được đồng bộ với thời gian của server. Khi gói tin đến server thì server sẽ giới hạn khoảng thời gian mà message đến, nếu vượt ra khoảng thời gian giới hạn gói tin bị loại bỏ.

2.2. Web Spoofing

Web spoofing là một kiểu lừa đảo trên web phổ biến hiện nay. Có 4 kiểu web spoofing là IP spoofing, ARP spoofing, Web spoofing, DNS spoofing.

a. Ip spoofing



Chúng mình thực sự tin cậy với một máy khác nhằm hợp pháp việc truy cập máy đó và có thể lấy được username, password (có máy giả mạo giả mạo server hoặc client).

Máy khác trở thành đồng phạm vì đã để cho máy giả mạo sử dụng IP giả mạo trùng với IP của máy đó. Kiểu tấn công này thường xảy ra ở các máy client khi độ bảo mật an toàn ở các máy đó không được cao.

Để tránh kiểu tấn công này thì server và client đều có định danh xác thực lẫn nhau.

b. Web spoofing

Người giả mạo tạo ra bản sao của một trang web và đánh lừa người dùng click vào.

Cách thực hiện :đặt trang web giả mạo liên kết với trang web thông dụng. Khi sử dụng mail kích hoạt web mail,attacker gửi mail đến giới thiệu link đến web giả. Kẻ giả mạo còn đánh lừa máy tìm kiếm web để nó trở vào web giả.

Khi user request đến server và sever trả về url của web đó. Người giả mạo có thể hướng người dùng truy cập đến trang web của họ để dễ dàng đánh cắp thông tin.

Đây là kiểu tấn công thường xuyên và phổ biến hiện nay khi lừa một cách trực tiếp đến người dùng cuối.Cách này thông thường kẻ giả mạo đánh lừa người dùng cuối chứ khó có thể giả mạo client thực sự của web service.

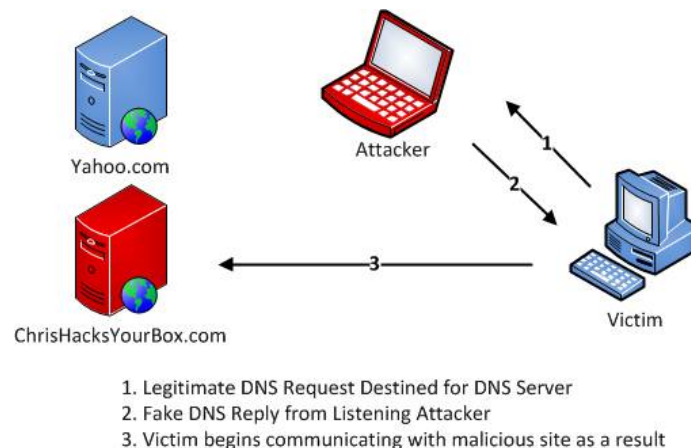
c. DNS spoofing

Cách hoạt động của DNS server :

Giao thức Domain Naming System (DNS) như được định nghĩa trong RFC 1034/1035 có thể được xem như là một trong những giao thức quan trọng nhất được sử dụng trong Internet. Nói ngắn gọn để dễ hiểu, bất cứ khi nào bạn đánh một địa chỉ web chẳng hạn như <http://www.google.com> vào trình duyệt, yêu cầu DNS sẽ được đưa đến máy chủ DNS để tìm ra địa chỉ IP tương xứng với tên miền mà bạn vừa nhập. Các router và các thiết bị kết nối Internet sẽ không hiểu [google.com](http://www.google.com) là gì, chúng chỉ hiểu các địa chỉ chẳng hạn như 74.125.95.103.

Máy chủ DSN làm việc bằng cách lưu một cơ sở dữ liệu các entry (được gọi là bản ghi tài nguyên) địa chỉ IP để bản đồ hóa tên DNS, truyền thông các bản ghi tài nguyên đó đến máy khách và đến máy chủ DNS khác. Kiến trúc máy chủ DNS trong toàn doanh nghiệp và Internet là một thứ khá phức tạp. Như một vấn đề của thực tế, bạn có thể hình dung chúng như các quyển sổ chuyên dụng cho kiến trúc DNS.

Người giả mạo cố tình cung cấp sai thông tin DNS để lừa nạn nhân truy cập vào 1 địa chỉ mà attacker chỉ định nhằm đánh cắp các thông tin như TK ngân hàng, ...



Giải pháp cho vấn đề này là cung cấp một DNS server tin cậy và khi sử dụng dịch vụ DNS phải có chứng thực.

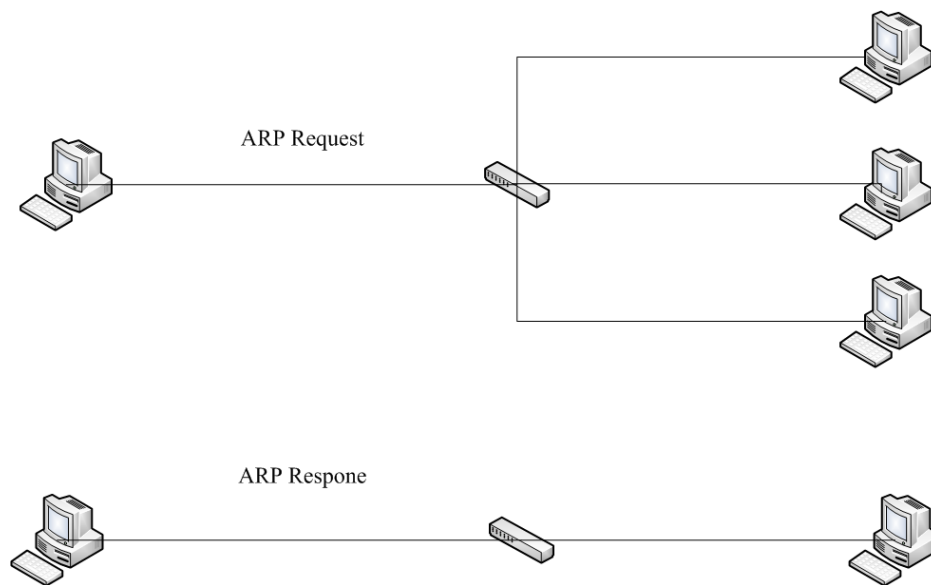
d. ARP spoofing

Arp là giao thức hoạt động ở tầng 2 cho phép ánh xạ địa chỉ MAC.

Giao thức ARP được thiết kế để phục vụ cho nhu cầu thông dịch các địa chỉ giữa các lớp thứ hai và thứ ba trong mô hình OSI. Lớp thứ hai (lớp data-link) sử dụng địa chỉ MAC để các thiết bị phần cứng có thể truyền thông với nhau một cách trực tiếp. Lớp thứ ba (lớp mạng), sử dụng địa chỉ IP để tạo các mạng có khả năng mở rộng trên toàn cầu. Lớp data-link xử lý trực tiếp với các thiết bị được kết nối với nhau, còn lớp mạng xử lý các thiết bị được kết nối trực tiếp và không trực tiếp. Mỗi lớp có cơ chế phân định địa chỉ riêng và chúng phải

làm việc với nhau để tạo nên một mạng truyền thông. Với lý do đó, ARP được tạo với RFC 826, “một giao thức phân định địa chỉ Ethernet - Ethernet Address Resolution Protocol”.

Quá trình truyền ARP bao gồm request và response.



Khi gói Arp request được gửi đi kẻ giả mạo bắt được thông tin đó và gửi trả về cho người gửi và bảng ARP của máy đó sẽ cập nhật địa chỉ của giả mạo.

Cách phòng chống vấn đề này là đảm bảo mức an toàn ở mạng nội bộ.

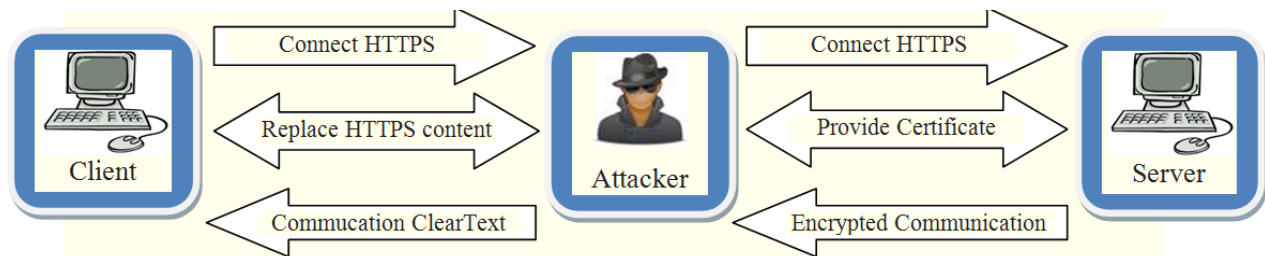
Vẫn còn một số kiểu lừa đảo qua mail nhưng kiểu giả mạo lừa đảo đó thì phải cần sự cảnh giác của client.

e. SSL spoofing: SSL và HTTPS

Phá hủy cơ chế HTTPS:

HTTPS là một kênh thông tin an toàn. Khi đã thiết lập được một kết nối từ máy chủ đến máy khách thì rất khó để có thể phá vỡ được kết nối đó. Tuy nhiên mọi việc đều có thể xảy ra khi máy khách vẫn chưa thật sự kết nối được với máy chủ.

Kết nối HTTPS an toàn tuy nhiên ta có thể phá vỡ cấu trúc đó trước khi kết nối được thiết lập bằng phương pháp “man in the middle”. Chặn request từ client đến server để trở thành kẻ thứ ba trong phiên kết nối.



Quá trình thực hiện như sau:

Lưu lượng giữa máy khách và máy chủ đầu tiên sẽ bị chặn.

Khi bắt gặp một HTTPS URL, sslstrip sẽ thay thế nó bằng một liên kết HTTP và sẽ ánh xạ những thay đổi của nó.

Máy tấn công sẽ cung cấp các chứng chỉ cho máy chủ web và giả mạo máy khách.

Lưu lượng được nhận trở lại từ website an toàn và được cung cấp trở lại cho máy khách.

Quá trình làm việc khá tốt, máy chủ có liên quan vẫn nhận lưu lượng SSL mà không hề biết về sự khác biệt này. Nếu không để ý thì rất khó phát hiện điều dị thường khi kết nối.

3. Bảo mật trong web service

Ngày nay công nghệ web services đã và đang được triển khai và ứng dụng trong rất nhiều lĩnh vực khác nhau bao gồm cả những lĩnh vực nhạy cảm, đòi hỏi tính an toàn cao như tài chính, ngân hàng,... Do đó web service cần cung cấp một mức an toàn đủ để hỗ trợ những công việc như thế. Bên cạnh mặt được của công nghệ web services mang lại thì việc đảm bảo an toàn, tin cậy, toàn vẹn thông tin trao đổi trên web service cũng là một điều rất quan trọng trong quá trình xây dựng web services, bằng việc sử dụng ws security và các thành phần của nó giúp cho thông tin trao đổi trên web services trở nên an toàn hơn.

Trước hết chúng ta xem xét những nhân tố rủi ro ảnh hưởng đến mức an toàn của những ứng dụng dựa trên web service. Chúng ta sẽ sử dụng một kịch bản rút tiền ngân hàng qua mạng để xem xét vấn đề.

Đây là một ứng dụng client/ server đơn giản mô tả một người rút tiền (client) kết nối tới trung tâm dữ liệu của ngân hàng để sử dụng một ứng dụng web service để thực hiện yêu cầu của mình. Nếu không có sự an toàn nào đã được áp dụng, thì có ba nhân tố mạo hiểm chính:

- Những giao dịch không hợp pháp (Unauthorized transactions) : một người nào đó không có quyền nhưng vẫn yêu cầu rút tiền. Giao dịch này không hợp pháp. Chúng ta cấm vấn đề này bằng cách sử dụng cơ chế chứng thực của WS - Security. Một ví dụ của sự chứng thực bao gồm phải có một kết hợp user ID/ password trong SOAP message.

- Những thông báo không mã hóa (Readable messages in clear text-no encryption): số hiệu tài khoản và số dư tài khoản trong gói SOAP rất dễ bị đọc lén trên mạng. Việc lộ thông tin này là do thông tin tài khoản và số dư được gửi qua mạng dưới định dạng văn bản. Để giải quyết vấn đề này, thông tin này phải được mã hóa ở mức kênh chuyển thông điệp hoặc ở mức thông điệp (WS - Security).

- Những thông điệp bị thay đổi hoặc mất mát (SOAP message susceptible to modification-no integrity). Trong quá trình chuyển thông tin từ người rút tiền đến trung tâm

dữ liệu, nó có thể bị chặn. Và những thông tin này có thể bị thay đổi, ví dụ như số tài khoản là 1234 thì bị thay đổi thành số 9876. Vấn đề này dẫn đến thiếu sự toàn vẹn.

Những ví dụ trên, chúng ta đã mô tả sự an toàn liên quan tới những yêu cầu của sự chứng thực, tính bí mật, và sự toàn vẹn thông tin.

Trước khi có WS-Security, sự an toàn kênh chuyển thông điệp rất thường được sử dụng. Sự an toàn kênh chuyển thông điệp ở chỗ là nó mã hóa toàn bộ thông điệp, dẫn đến sử dụng CPU cao hơn. Tuy nhiên với WS-Security, nó cung cấp những cách tối ưu hóa những thao tác an toàn, mà yêu cầu ít thời gian sử dụng CPU hơn. Dựa vào mức an toàn cần thiết mà một hoặc nhiều hơn những cơ chế an toàn này có thể được áp dụng cho một ứng dụng.

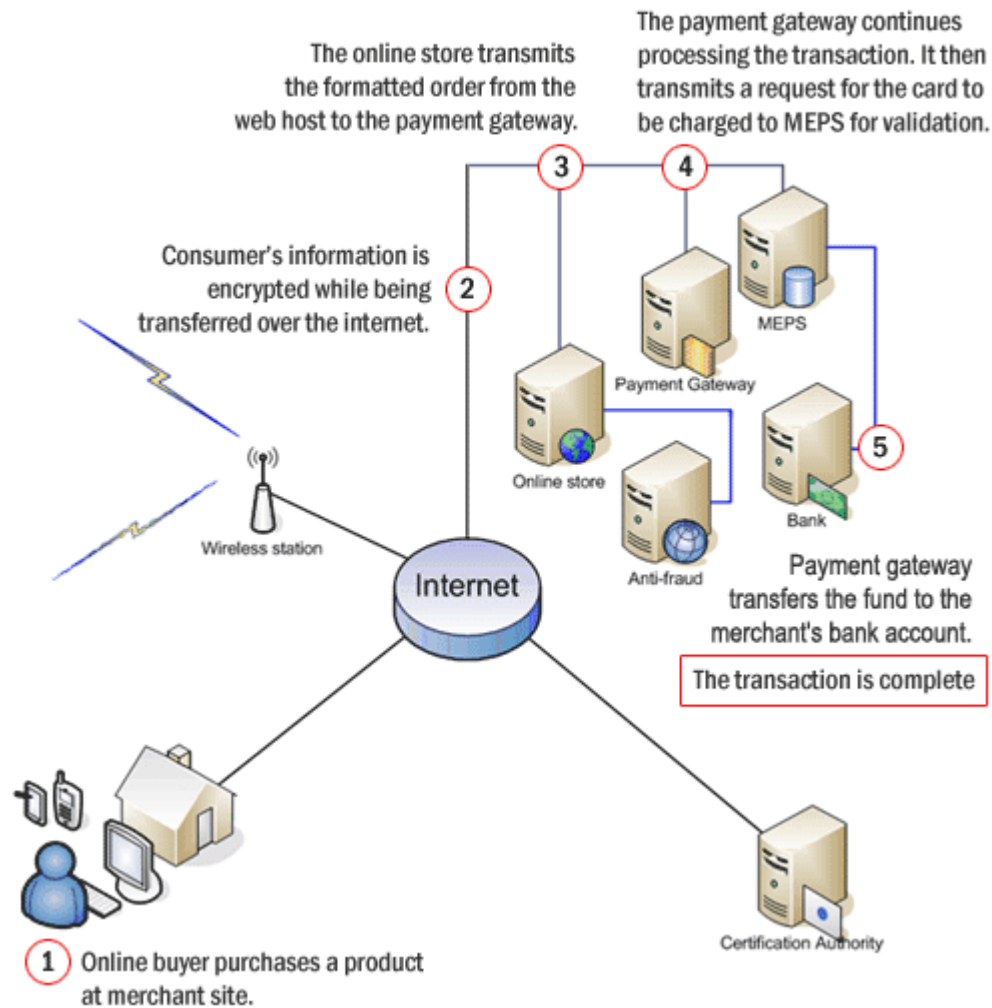
Tuy nhiên việc chọn cơ chế an toàn cho web service phải đòi hỏi sao cho người dùng không cảm thấy quá phức tạp tạo một sự gò bó, do đó việc chọn cơ chế an toàn nào trong ws security thì phụ thuộc nhiều vào loại service và những tính năng mà service này cung cấp, ví dụ như service về giao dịch tài chính ngân hàng phải có cơ chế an toàn hiệu quả hơn so với service chọn lọc và phân loại tin tức hay service cho biết tỷ giá trao đổi giữa các loại ngoại tệ,... Bên cạnh đó còn một điểm cần quan tâm đó là sự an toàn không chỉ phụ thuộc vào những giải thuật, những tiêu chuẩn, và những cơ chế mà ws security mang lại, mà nó còn tùy vào thái độ của các công ty có hiểu rõ tầm quan trọng của an toàn thông tin khi triển khai các ứng dụng, giao dịch trên mạng hay không cũng rất cần thiết.

Có hai hình thức bảo mật, đó là bảo mật trên kênh truyền và bảo mật ở mức thông điệp. Hiện nay hầu hết các dịch vụ đều kết hợp cả hai hình thức để tối ưu cho việc bảo mật.

Bảo mật ở mức kênh truyền : trên kênh truyền phải bảo đảm được thông điệp an toàn và toàn vẹn. Ở mức này thì ta thường dùng kênh truyền an toàn như HTTP + SSL để tạo kết nối an toàn đến client. An toàn ở mức này đòi hỏi cơ sở hạ tầng mạng phải tốt.

Bảo mật ở mức thông điệp : để tăng mức độ an toàn cho thông điệp, ta sử dụng thêm WS security cung cấp mức an toàn cho thông điệp. Các dữ liệu được mã hóa và được sử dụng chữ kí số để tránh bị đánh cắp thông tin. Cả hai bên server và client đều sử dụng các key để có thể chứng thực lẫn nhau.

Online Payment Processing - Standard Routine



Hình ảnh trên là một quá trình sử dụng web service payment trực tuyến

Chúng bao gồm các bước :

- Người mua đặt mua hàng và gửi thông tin giao dịch và thông tin người mua đến người cung cấp dịch vụ bán hàng.
- Hệ thống sẽ chuyển các thông tin đó sang gateway rồi từ gateway tiếp tục chuyển giao dịch đến cho ngân hàng.
- Tất cả các thông tin gửi và nhận đều được mã hóa và thông qua HTTPS.

a. Xác thực client và server sử dụng:

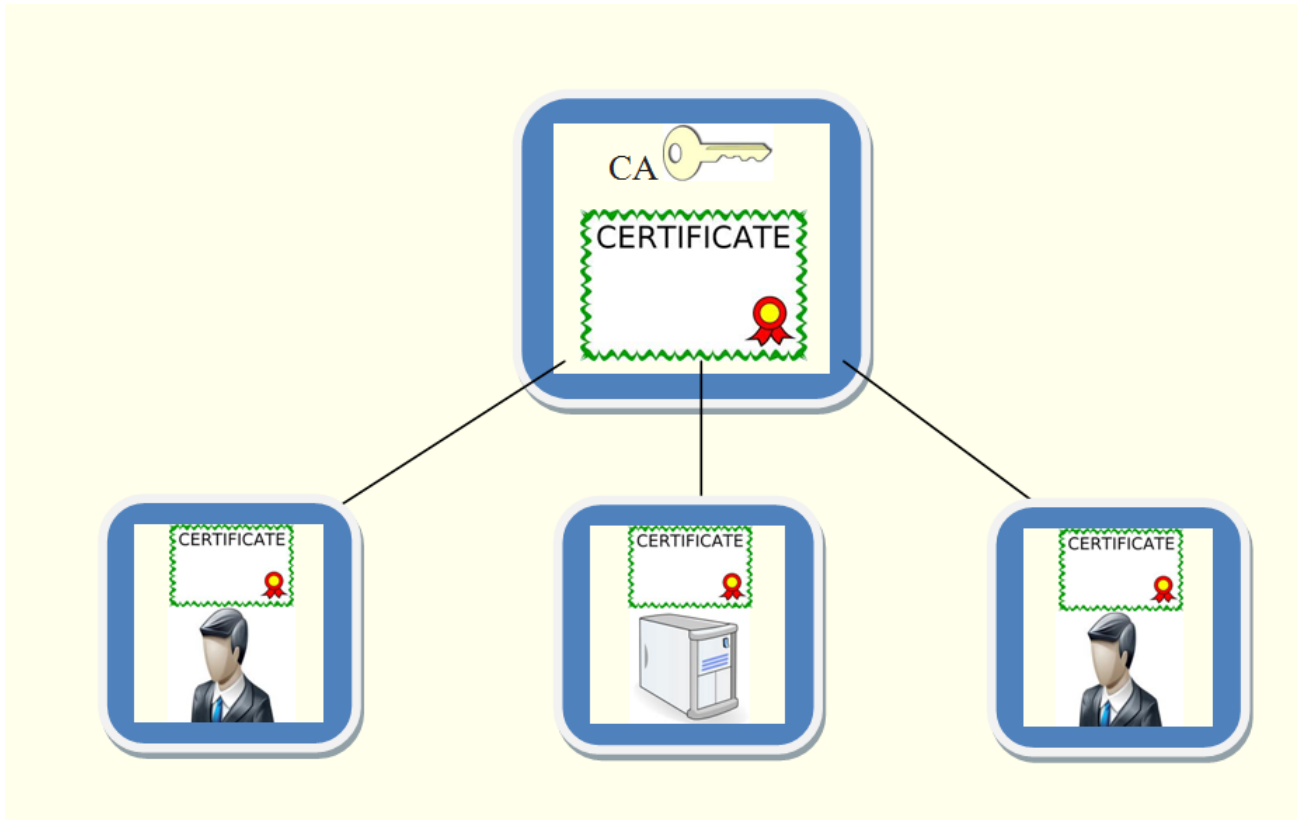
Việc chuyển tiền thanh toán qua mạng đều thực hiện việc chuyễn những thông tin rất nhạy cảm và quan trọng nên thực hiện bảo mật an toàn cho nó là điều rất cần thiết.Trong đó có xác thực cả client và server.

Client ở đây là nơi đăng ký thông tin và sử dụng dịch vụ web như các trang bán hàng qua mạng.

Server là nơi cung cấp dịch vụ và cung cấp cho client mã đăng ký dịch vụ.

Cả client và server đều phải xác thực lẫn nhau để tránh việc có người giả mạo đứng giữa giả một trong hai phía.

Như vậy cả client và server đều phải có một certificate cho riêng mình được một root CA chứng thực chữ kí số.



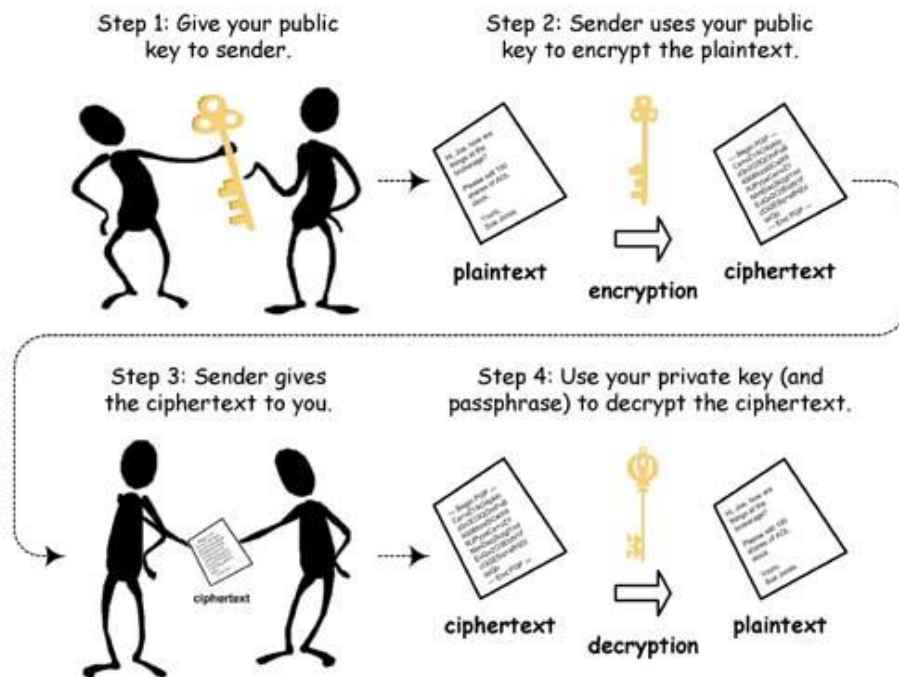
Chữ ký điện tử được sử dụng trong các giao dịch điện tử. Xuất phát từ thực tế, chữ ký điện tử cũng cần đảm bảo các chức năng: xác định được người chủ của một dữ liệu nào đó: văn bản, ảnh, video,... dữ liệu đó có bị thay đổi hay không.

Người gửi sẽ kí bằng cách mã hóa dữ liệu đi kèm với private key của người gửi và người nhận chỉ có thể dùng public key của người gửi để giải mã, mọi public key khác đều không thể giải mã như vậy có thể chứng minh thông tin gửi là của người đó chứ không thể là người khác.

Root CA có vai trò cấp certificate cho cặp private và public key đảm bảo các cặp key đó là chính xác.

Mã hóa dữ liệu truyền sử dụng công nghệ Public Key Cryptography và TripleDes.

Mã hóa sử dụng cặp khóa public key và private key, trong đó public key dùng để mã hóa thông điệp và xác thực chữ kí, còn private key dùng để giải mã thông điệp và tạo chữ kí.



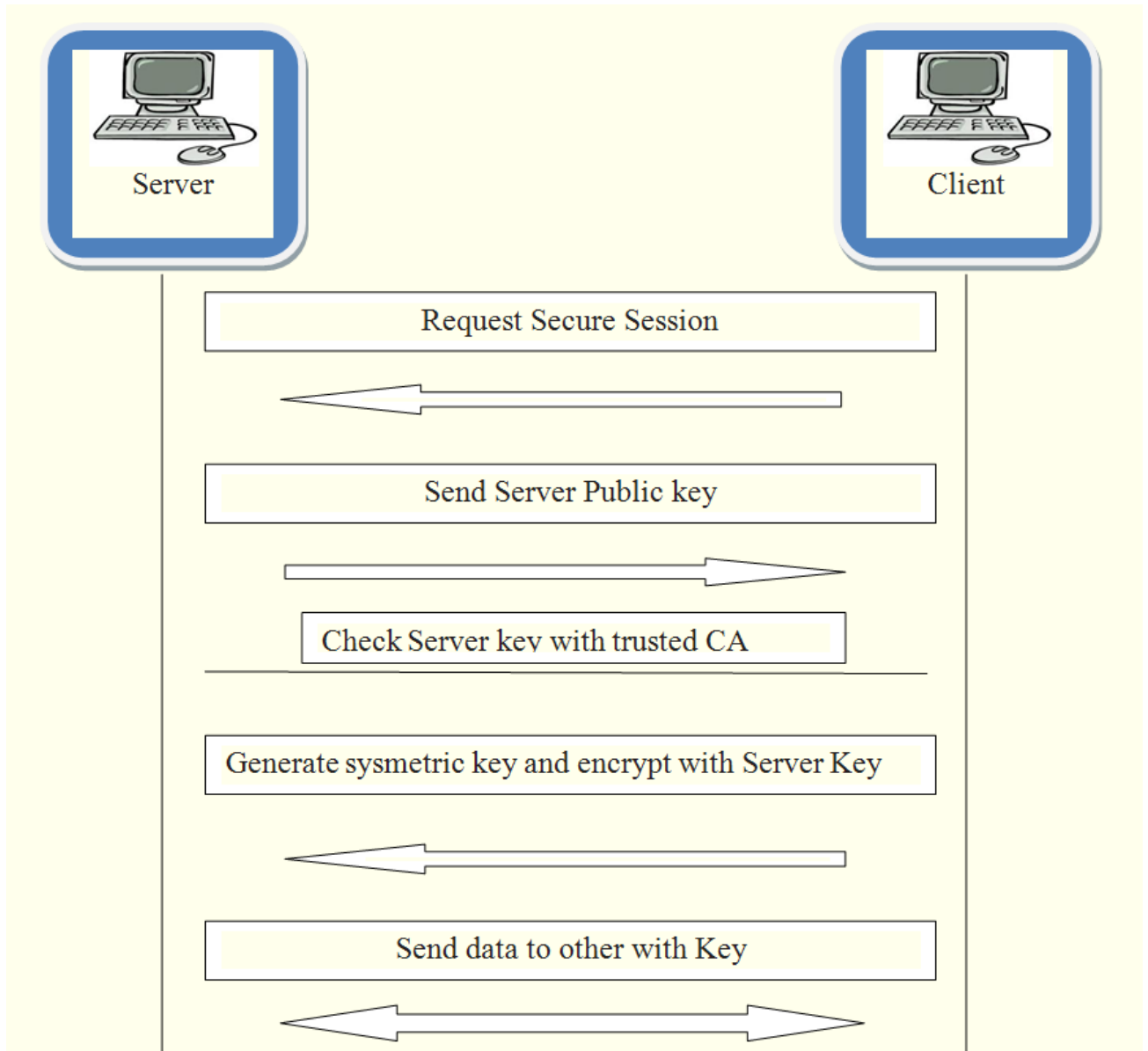
Một trong số những giải thuật mã hóa thông dụng nhất hiện nay là RSA.

RSA là giải thuật mã hóa được đánh giá là tốt nhất và được sử dụng rộng rãi nhất hiện nay do Rivest, Shamir và Adleman phát triển tại MIT vào năm 1978 [RIVE78].

b. Sử dụng protocol HTTPS

Trước khi có HTTPS thì HTTP(*HyperText Transfer Protocol*) là giao thức được sử dụng để truyền thông tin trên Internet dưới dạng văn bản text và ngang hàng không an toàn. Cùng với các giao thức SMTP, telnet, FTP trong những ngày đầu phát triển Internet khi vấn đề bảo mật chưa được quan tâm đúng mức. HTTPS ra đời và sự phát triển của HTTPS đảm bảo dữ liệu an toàn hơn trên đường truyền cho phép client và server chứng thực nhau.

Dưới đây là quá trình bắt tay của Client và server khi sử dụng HTTPS.



Hypertext Transfer Protocol Secure (HTTPS) là một sự kết hợp của Hypertext Transfer Protocol (HTTP) với giao thức SSL / TLS để cung cấp một kênh truyền mà dữ liệu được mã hóa và bảo mật an toàn với máy chủ. Các kết nối HTTPS thường được sử dụng cho các giao dịch thanh toán trên mạng toàn cầu và cho các giao dịch nhạy cảm trong hệ thống thông tin doanh nghiệp.

Sử dụng HTTPS đảm bảo dữ liệu được truyền đi bảo mật, an toàn trên một mạng không an toàn và tránh được middle-attack miễn là máy chủ đã được xác thực và đáng tin cậy.

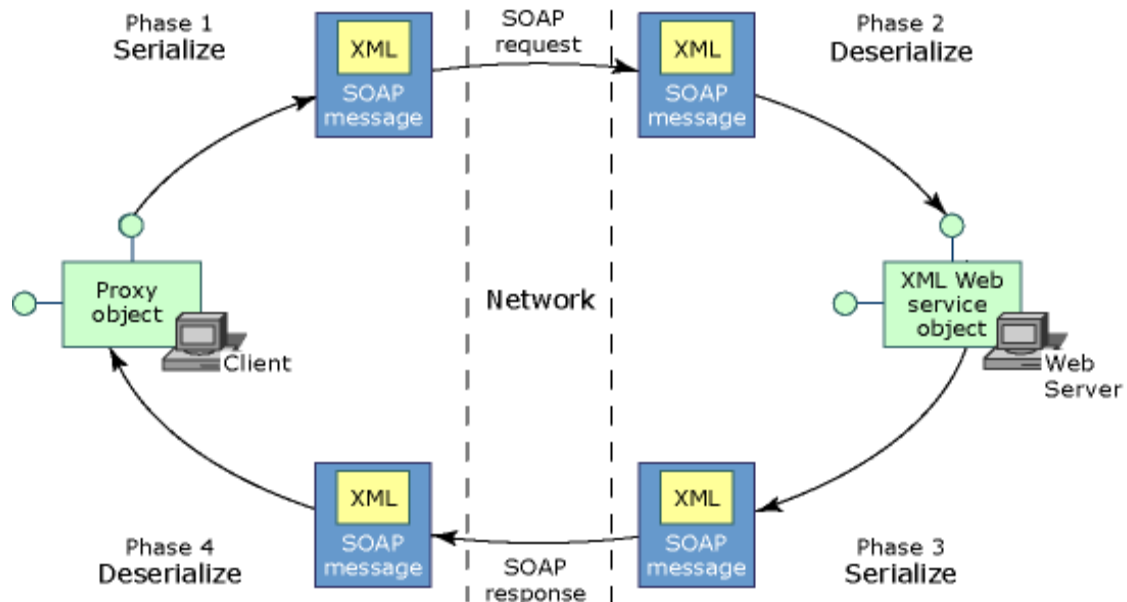
Sử dụng HTTPS tuy không thể nói là đảm bảo tuyệt đối nhưng cũng đã đáp ứng được việc chứng thực và an toàn dữ liệu. Tuy nhiên, authentication, auditing, and nonrepudiation chưa được cung cấp.

c. Soap Header extension

Như các phần đã trình bày ở trên việc sử dụng dịch vụ được thực hiện bởi việc gửi các thông tin yêu cầu qua Soap message.

Ta sẽ mở rộng phần soap message và đưa vào đó phần mở rộng của soap tăng thêm tính an toàn cho thông tin.

Phần mở rộng của Soap cho phép mở rộng cho soap message bằng cách thêm và chỉnh sửa thông tin bên trong Soap message. Ta có thể hiện thực mã hóa hoặc sử dụng các giải thuật nén XML Web service.



Chu trình của XML web service.

Quá trình Serialize : Sau khi các thông tin cần chuyển đã được đưa vào soap message và chuẩn bị gửi. Trong quá trình này ta có thể lấy các thông tin ra và mã hóa.

Quá trình Deserialize : quá trình chuẩn bị nhận soap message. Nếu ta đã mã hóa thông tin thì trong quá trình này ta có thể lấy thông tin ra và giải mã trả về kết quả.

Với Soap extension thì ta có thể mã hóa và giải mã thông tin ở một nơi khác nơi mà ta gọi yêu cầu cho service.

Trước khi gửi thông tin từ soap message đến server ta override lại Soap message nếu cần thiết, thêm vào đó các thông tin đã được mã hóa và gửi đi.

4. Thực hiện bảo mật trong web service

Mục tiêu:

- Tạo Web service Hello và bảo mật thông qua User name và Password, gồm 3 phần
 - Service SecureWebService.asmx
 - Ứng dụng client kết nối tới service

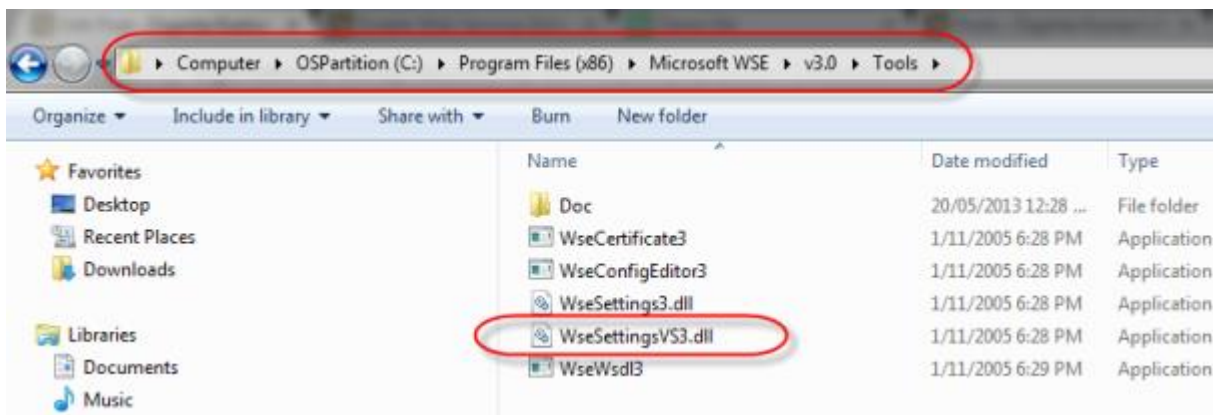
- Đóng gói thư viện bảo mật Web Service
- Service sẽ trả về thông tin **"Hello Authenticated user <name>"** nếu User name và Password so khớp và trả về thông tin thông báo **"Please format the request as a SOAP request and try again."** nếu trái lại.

4.1. Cài đặt WSE 3.0

Download gói cài từ Microsoft tại link sau <http://www.microsoft.com/en-us/download/details.aspx?id=6545>

a. Tích hợp WSE 3.0 vào Visual Studio 2008, 2010, 2012, 2013

Sau khi cài đặt thành công WSE 3.0, ta cần khởi động lại máy và không bật Visual Studio trong quá trình cấu hình. Ta sẽ thấy thư mục WSE như sau



Vào thư mục sau **%ALLUSERSPROFILE%\Application Data\Microsoft\MSEnvShared\AddIns**

Trong thư mục ta sẽ tìm thấy file **WSESettingsVS3.AddIn**, trước khi chỉnh sửa ta sẽ cần backup lại file này. Ta sẽ chỉnh sửa nội dung như hình sau

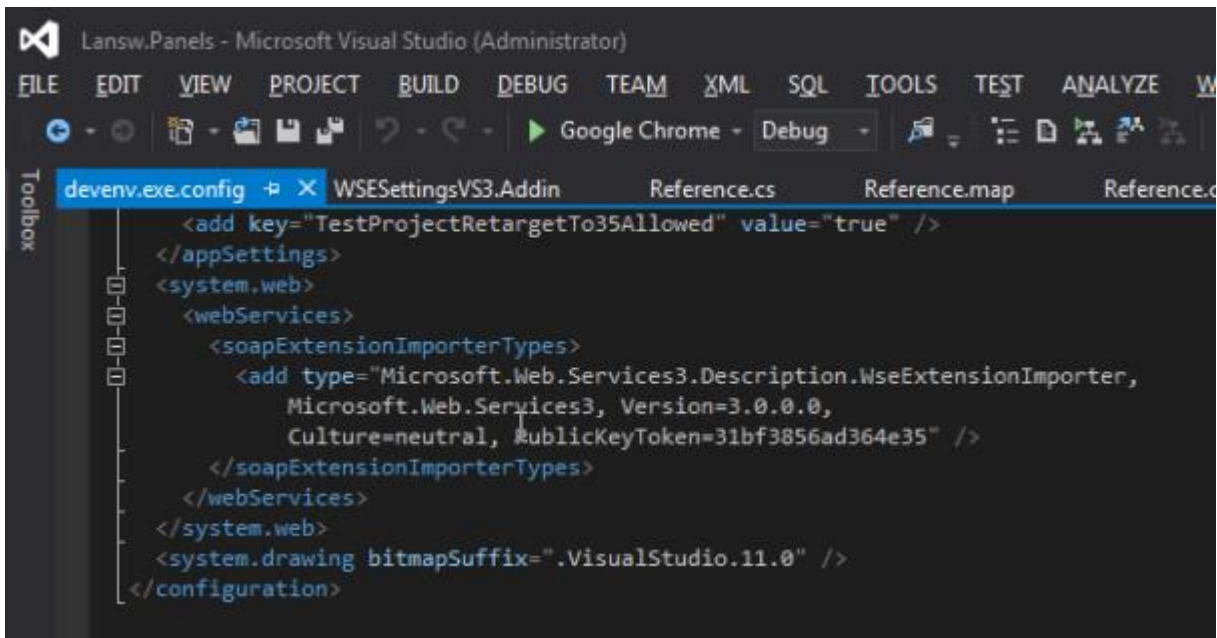
```

WSESettingsVS3.Addin Reference.cs Reference.map Reference.cs UtilMethods.cs
<?xml version="1.0" encoding="utf-16" standalone="no"?>
<Extensibility xmlns="http://schemas.microsoft.com/AutomationExtensibility">
  <HostApplication>
    <Name>Microsoft Visual Studio Macros</Name>
    <Version>8.0</Version>
  </HostApplication>
  <HostApplication>
    <Name>Microsoft Visual Studio</Name>
    <Version>8.0</Version>
  </HostApplication>
  <!--VS2008-->
  <HostApplication>
    <Name>Microsoft Visual Studio Macros</Name>
    <Version>9.0</Version>
  </HostApplication>
  <HostApplication>
    <Name>Microsoft Visual Studio</Name>
    <Version>9.0</Version>
  </HostApplication>
  <!--VS2010-->
  <HostApplication>
    <Name>Microsoft Visual Studio Macros</Name>
    <Version>10.0</Version>
  </HostApplication>
  <HostApplication>
    <Name>Microsoft Visual Studio</Name>
    <Version>10.0</Version>
  </HostApplication>
  <!--VS2012-->
  <HostApplication>
    <Name>Microsoft Visual Studio Macros</Name>
    <Version>11.0</Version>
  </HostApplication>
  <HostApplication>
    <Name>Microsoft Visual Studio</Name>
    <Version>11.0</Version>
  </HostApplication>
  <Addin>
    <FriendlyName>WSE Settings 3.0...</FriendlyName>
    <Description>WSE Settings Tool.</Description>
    <Assembly>C:\Program Files (x86)\Microsoft WSE\v3.0\Tools\WseSettingsVS3.dll</Assembly>
    <FullClassName>WseSettings.Connect</FullClassName>
  </Addin>
</Extensibility>
  
```

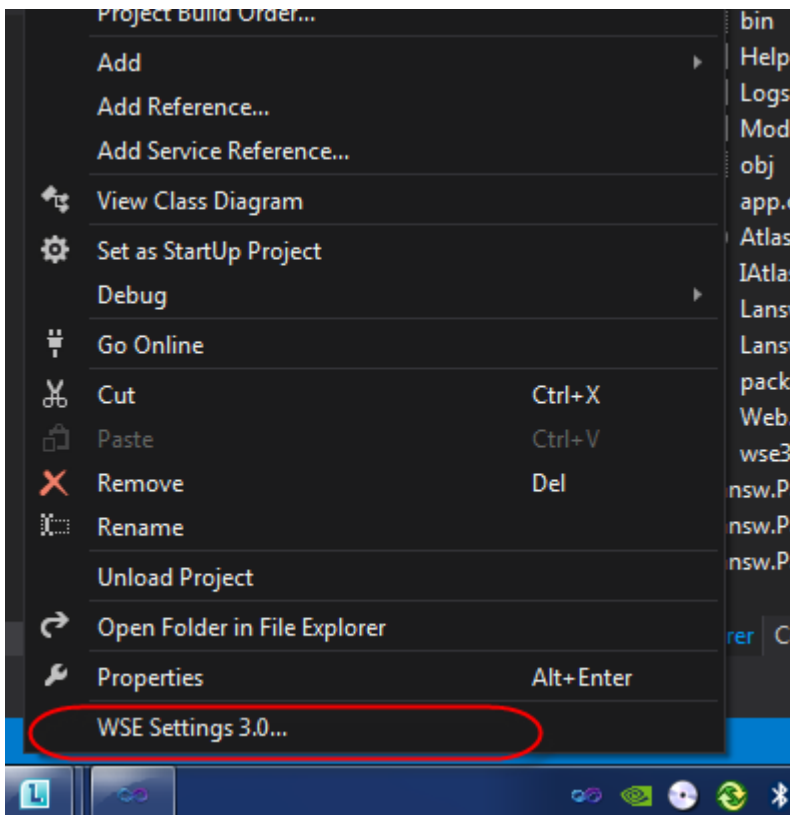
Ta sẽ thêm vào các khối bôi đỏ tương ứng cho Visual Studio 2008, 2010, 2012, 2013.

Tới thư mục *C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE* (cho VS2012) hoặc *C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE* (cho VS2008) hoặc *C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE* (cho VS 2010) hoặc *C:\Program Files\Microsoft Visual Studio 12.0\Common7\IDE* (cho VS 2013).

Mở file *devenv.exe.config* và thêm những dòng sau (nhớ backup lại file trước khi chỉnh sửa)



Tới đây ta đã tích hợp thành công WSE 3.0 vào công cụ Visual Studio

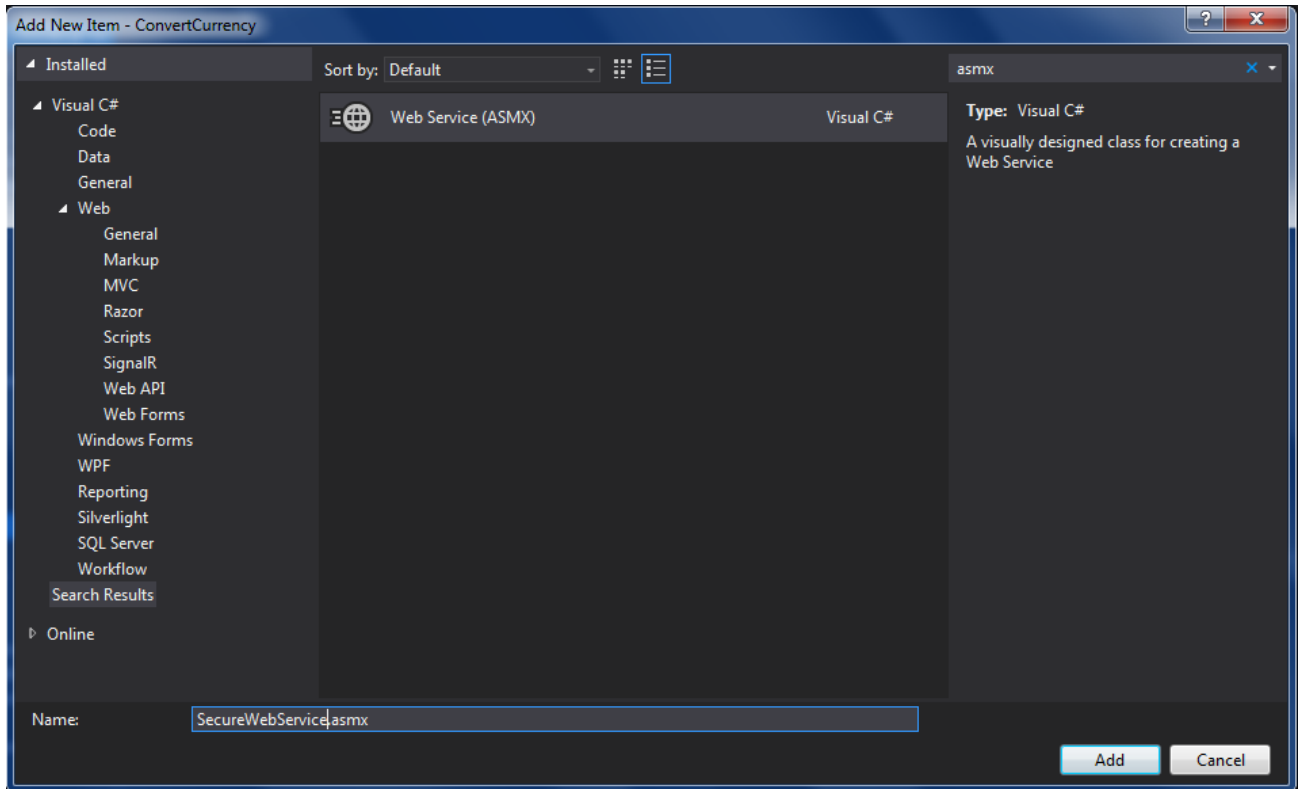


Chú ý: nếu hệ thống không cho phép chỉnh sửa trực tiếp các file trên, ta sẽ tạo ra file mới và ghi đè lên file cần chỉnh sửa.

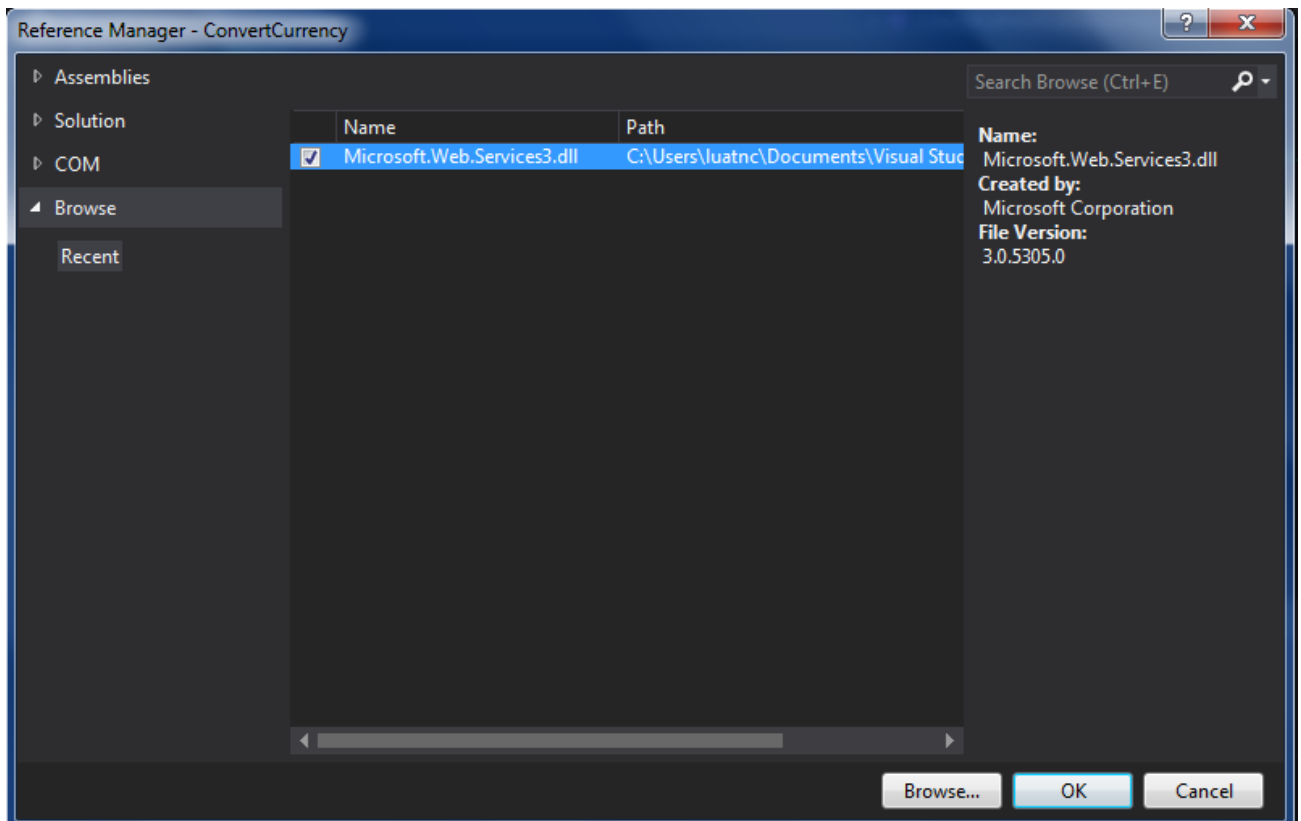
4.2. Xây dựng ứng dụng bảo mật Web service thông qua bảo mật thông điệp SOAP với WS-Security

a. Tạo mới Web Service với WSE 3.0

Tạo ứng dụng bảo mật Web service đặt tên SecureWebService.asmx như sau



Thêm tham chiếu tới thư viện WSE 3.0 đã cài đặt, chọn **Add Reference** và chọn tới file **Microsoft.Web.Services3.dll** trong thư mục đã cài đặt WSE.



Ta thêm khai báo sử dụng thư viện tới file *SecureWebService.asmx* như sau

```
using Microsoft.Web.Services3;  
using Microsoft.Web.Services3.Security.Tokens;
```

b. Thực thi Secure Web Service

```
[WebMethod]  
public string Hello(string name)  
{  
    //Get the current soap context  
    SoapContext ctxt = RequestSoapContext.Current;  
    if (ctxt == null)  
    {  
        //This request is using a different protocol other than SOAP.  
        return "Please format the request as a SOAP request and try again.";  
    }  
  
    //Iterate through all Security tokens  
    foreach(SecurityToken tok in ctxt.Security.Tokens)  
    {  
        if (tok is UsernameToken)  
        {  
            UsernameToken user = (UsernameToken)tok;  
            return "Hello Authenticated user " + user.Username;  
        }  
    }  
    return "Hello Liar";  
}
```

Trong đoạn mã trên, ta lặp qua các thông tin bảo mật Header vì mỗi thông điệp SOAP có thể chứa nhiều khai báo Header. Sau đó, tìm kiếm chắc chắn rằng có khai báo UsernameToken trong đó.

Ta sẽ F5 để chạy ứng dụng Web Service để cho phép nó sẵn sàng phục vụ



The following operations are supported. For a formal definition, please review the [Service Description](#).

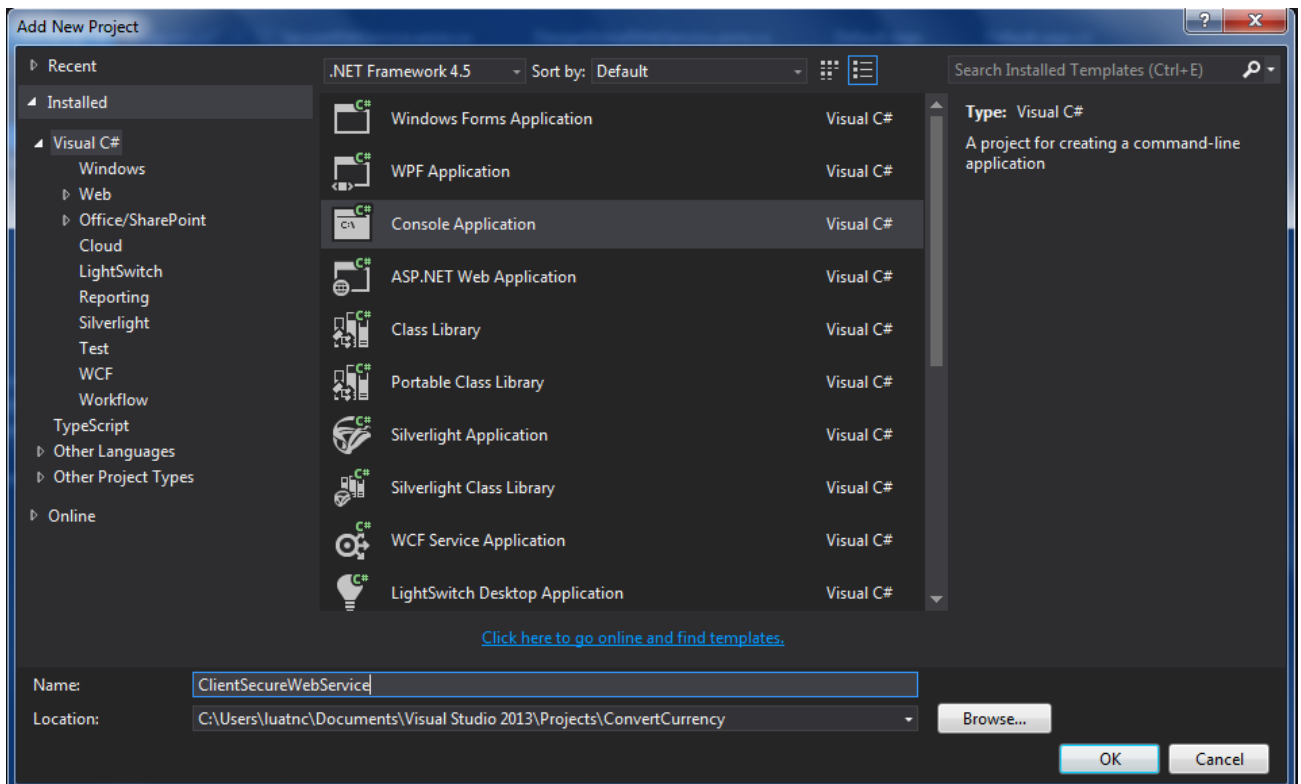
- [CheckAuthentication](#)

This web service is using <http://tempuri.org/> as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

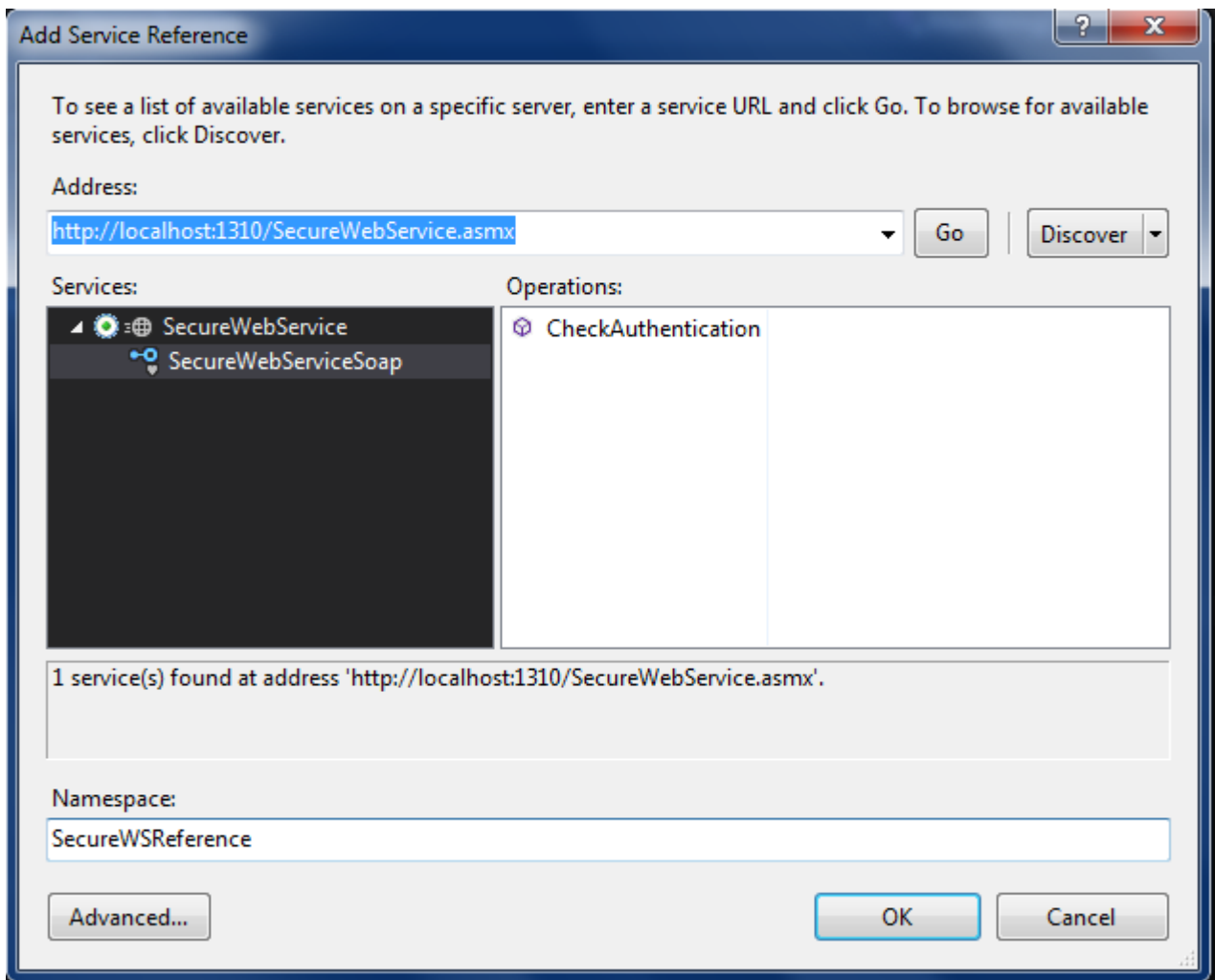
c. Tạo ứng dụng client kết nối tới SecureWebService

Tạo project Console Application đặt tên ClientSecureWebService như hình sau



Thêm tham chiếu tới thư viện WSE 3.0 đã cài đặt, chọn **Add Reference** và chọn tới file **Microsoft.Web.Services3.dll** trong thư mục đã cài đặt WSE.

Thêm tham chiếu tới Secure Web service vừa tạo, bằng cách cung cấp URL như hình sau và đặt tên **SecureWSReference**



Ta thêm khai báo sử dụng thư viện tới file *program.cs* như sau

```
using Microsoft.Web.Services3;
using Microsoft.Web.Services3.Security.Tokens;
```

Phần thực thi service như sau

```
[STAThread]
static void Main(string[] args)
{
    Console.WriteLine("Enter Name: ");
    string name = Console.ReadLine();
    Console.WriteLine("Enter Password: ");
    string password = Console.ReadLine();

    SecureWSReference.SecureWebServiceSoapClient proxy = new
SecureWSReference.SecureWebServiceSoapClient();
```

```

proxy.RequestSoapContext.Security.Tokens.Add(new UsernameToken(name,
                                                                    password, PasswordOption.SendHashed));

proxy.RequestSoapContext.Security.Timestamp.TtlInSeconds = 300;

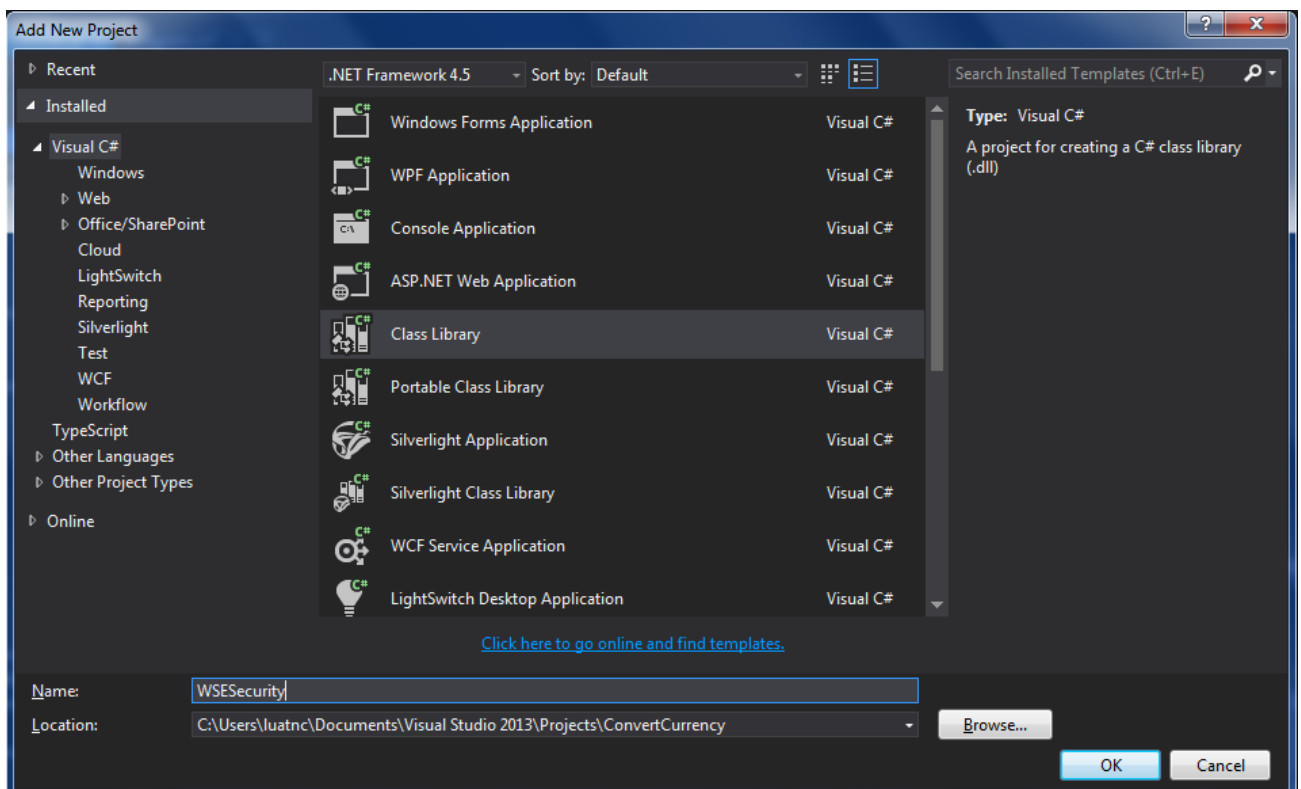
Console.WriteLine(proxy.Hello(name));
Console.WriteLine("Hit enter to end.");
Console.ReadLine();
}

```

Trong đoạn mã trên, thành phần UsernameToken được đóng gói vào trong SOAP Header và gửi đi và Password được băm nhờ giải thuật SHA1 trước khi gửi. Nhờ đó, việc bảo mật thông tin trong thành phần SOAP Header dựa vào WS-Security là rất đảm bảo. Bên cạnh đó, client hoàn toàn có thể cấu hình thời gian Timeout trong trao đổi thông điệp để tăng cường khả năng chống tấn công từ bên ngoài

Ở trên, ta thấy WS-Security đã dùng cơ chế UsernameToken để bảo mật thông tin trong SOAP Header. Tuy nhiên, với những ứng dụng yêu cầu phải có khả năng xác thực phức tạp hơn, như kiểm thông tin tài khoản có được lưu trữ trong Cơ sở dữ liệu, ... Do đó, ta cần đóng gói thành thư viện riêng để đảm nhận xử lý các bussiness logic phức tạp tùy thuộc vào yêu cầu của hệ thống.

Tạo mới project Class Library như sau



Chọn chức năng hỗ trợ WSE bằng cách chuột phải vào project và chọn **WSE 3.0 Setting**.

Đổi tên *Class1.cs* thành *CustomAuthenticator.cs* và nhập vào nội dung như sau

```
[SecurityPermissionAttribute(SecurityAction.Demand,
    Flags = SecurityPermissionFlag.UnmanagedCode)]
public class CustomAuthenticator : UsernameTokenManager
{
    //Returns the password or password equivalent for a user name
    protected override string AuthenticateToken(UsernameToken token)
    {
        if (token == null)
            throw new ArgumentNullException();
        if (token.Username == "hoand")
            return "mypassword";
        else
            return null;
    }
}
```

Ở đây, ta có lớp **CustomAuthenticator** kế thừa từ lớp quản lý **UsernameTokenManager** và ghi đè phương thức **AuthenticateToken** xử lý xác thực.

Vậy ta đã có thư viện đảm nhận việc xác thực thông tin UsernameToken nằm trong SOAP Header gửi tới. Thư viện này sẽ đảm nhận việc xử lý bussiness logic phức tạp và được sử dụng bởi Server và tránh việc lộ các thông tin quan trọng đối với bất cứ thành phần nào khác.

Để sử dụng lại thư viện này ở các project khác, ta cần build ra file WSESecurity.dll và sẽ được tham chiếu tới bởi các Web service project. Thêm nữa là ta cần tích hợp nó vào luồng xử lý WSE pipeline của các Web service bằng cách khai báo thêm (dưới thẻ **<configuration>**) vào file web.config như sau

```
<microsoft.web.services3>
<security>
<securityTokenManager qname="wsse:UsernameToken"
    type="WSESecurity.CustomAuthenticator, WSESecurity"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
        oasis-200401-wss-wssecurity-secext-1.0.xsd"/>
</security>
```

</microsoft.web.services3>

Thư viện **Web Service Enhancement 3.0** cung cấp một phương thức đơn giản nhưng hết sức tuyệt vời để quản lý bảo mật trao đổi thông điệp trong Web Service. Module xử lý xác thực hoàn toàn có thể cập nhật lại bussiness logic mà không cần phải biên dịch lại Web Service thông qua thư viện **WSESecurity** xử lý riêng như ta đã thực hiện ở trên.

5. Tổng kết

Để có được sự an toàn trong việc truyền tải dữ liệu cần phải thực hiện càng nhiều biện pháp bảo mật càng tốt. Tuy nhiên việc kết hợp các biện pháp bảo mật cũng phải tránh rườm rà phức tạp cho người sử dụng và vẫn đảm bảo được tốc độ truyền tải dữ liệu.

Tóm tắt bài học

An toàn thông tin trên Internet là một vấn đề chung hiện nay. Sự an toàn web service lại càng cần được sự quan tâm hơn nữa, khi các thông tin nhạy cảm như tài khoản cá nhân ở ngân hàng có thể bị đánh cắp.

Yêu cầu cho một chuẩn an toàn chung cần thiết cần được đáp ứng gồm:

- Identification: định danh được những ai truy cập tài nguyên hệ thống.
- Authentication: chứng thực tư cách truy cập tài nguyên của người muốn sử dụng.
- Authorization: cho phép giao dịch khi đã xác nhận định danh người truy cập.
- Integrity: toàn vẹn thông tin trên đường truyền.
- Confidentiality: độ an toàn, không ai có thể đọc thông tin trên đường đi.
- Auditing: kiểm tra, tất cả các giao dịch đều được lưu lại để kiểm tra.
- Non-repudiation: độ mềm dẻo, cho phép chứng thực tính hợp pháp hóa của thông tin đến từ một phía thứ ba ngoài 2 phía là người gửi và người nhận.

Hiểu về một số kiểu giả mạo, đánh cắp thông tin và cách phòng chống.

Bảo mật Web Service có hai hình thức bảo mật chính, đó là bảo mật trên kênh truyền và bảo mật ở mức thông điệp. Hiện nay hầu hết các dịch vụ đều kết hợp cả hai hình thức để tối ưu cho việc bảo mật.

Thực hiện bảo mật Web Service với thư viện WSE 3.0 do Microsoft cung cấp. Xây dựng dịch vụ Web bảo mật với Visual Studio - ASP.NET, triển khai và công bố dịch vụ Web. Đồng thời xây dựng ứng dụng để khai thác và sử dụng chuẩn kết nối bảo mật vừa xây dựng.

Bài tập

Tiếp tục thực hiện xây dựng ứng dụng Web kết nối tới Web Service bảo mật với cơ chế WSE 3.0.

PHỤ LỤC I: Tài nguyên và tài liệu tham khảo

- Java Software Environment:
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>
- Netbeans IDE : <https://netbeans.org/downloads/>
- Visual Studio: <http://www.visualstudio.com/downloads/download-visual-studio-vs>
- WSE 3.0: <http://www.microsoft.com/en-us/download/details.aspx?id=6545>
- Link down mã nguồn toàn bộ các ví dụ trong giáo trình:
C#, ASP.NET: ConvertCurrency - http://truli.vn/?attachment_id=583
Java Web Service: FlowerAlbum - http://truli.vn/?attachment_id=582
Java Client Web Service: http://truli.vn/?attachment_id=581