WEB SERVICES

Systems Integration and Architecture 2

Presented by: Mr. Ramil V. Huele

WHAT ARE WEB SERVICES?

- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system.
- Web services are self-contained, modular, distributed, dynamic applications that can be described, published, located, or invoked over the network to create products, processes, and supply chains.
- These applications can be local, distributed, or web-based. Web services are built on top of open standards such as TCP/IP, HTTP, Java, HTML, and XML.
- Web services are XML-based information exchange systems that use the Internet for direct application-to-application interaction. These systems can include programs, objects, messages, or documents.
- A web service is a collection of open protocols and standards used for exchanging data between applications or systems

WHAT ARE WEB SERVICES?

- Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer
- To summarize, a complete web service is, therefore, any service that:
  - Is available over the Internet or private (intranet) networks
  - Uses a standardized XML messaging system
  - Is not tied to any one operating system or programming language
  - Is self-describing via a common XML grammar
  - Is discoverable via a simple find mechanism

BENEFITS OF USING WEB SERVICES

- **Exposing the Existing Function on the network**

  A web service is a unit of managed code that can be remotely invoked using HTTP. That is, it can be activated using HTTP requests.

  Web services allow you to expose the functionality of your existing code over the network. Once it is exposed on the network, other applications can use the functionality of your program.

- **Interoperability**

  Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services.

  For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

BENEFITS OF USING WEB SERVICES

- **Standardized Protocol**

  Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack.

  This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.

- **Low Cost Communication**

  Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services. This solution is much less costly compared to proprietary solutions like EDI/B2B.

  Besides SOAP over HTTP, web services can also be implemented on other reliable transport mechanisms like FTP.

WEB SERVICES SPECIAL BEHAVIORAL CHARACTERISTICS

- **XML-Based**

  Web services use XML at data representation and data transportation layers. Using XML eliminates any networking, operating system, or platform binding. Web services based applications are highly interoperable at their core level.

- **Loosely Coupled**

  A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service.

  A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated.

  Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

WEB SERVICES SPECIAL BEHAVIORAL CHARACTERISTICS

- **Coarse-Grained**

  Object-oriented technologies such as Java expose their services through individual methods. An individual method is too fine an operation to provide any useful capability at a corporate level.

  Building a Java program from scratch requires the creation of several fine-grained methods that are then composed into a coarse-grained service that is consumed by either a client or another service.

  Businesses and the interfaces that they expose should be coarse-grained. Web services technology provides a natural way of defining coarse-grained services that access the right amount of business logic.

- **Ability to be Synchronous or Asynchronous**

  Synchronicity refers to the binding of the client to the execution of the service. In synchronous invocations, the client blocks and waits for the service to complete its operation before continuing. Asynchronous operations allow a client to invoke a service and then execute other functions.

  Asynchronous clients retrieve their result at a later point in time, while synchronous clients receive their result when the service has completed. Asynchronous capability is a key factor in enabling loosely coupled systems.

WEB SERVICES SPECIAL BEHAVIORAL CHARACTERISTICS

- **Supports Remote Procedure Calls (RPCs)**

  Web services allow clients to invoke procedures, functions, and methods on remote objects using an XML-based protocol. Remote procedures expose input and output parameters that a web service must support.

  Component development through Enterprise JavaBeans (EJBs) and .NET Components has increasingly become a part of architectures and enterprise deployments over the past couple of years. Both technologies are distributed and accessible through a variety of RPC mechanisms.

  A web service supports RPC by providing services of its own, equivalent to those of a traditional component, or by translating incoming invocations into an invocation of an EJB or a .NET component.

- **Supports Document Exchange**

  One of the key advantages of XML is its generic way of representing not only data, but also complex documents. These documents can be as simple as representing a current address, or they can be as complex as representing an entire book or Request for Quotation (RFQ).

  Web services support the transparent exchange of documents to facilitate business integration.

WEB SERVICE ARCHITECTURE

- **There are two ways to view the web service architecture −**

  The first is to examine the individual roles of each web service actor.

  The second is to examine the emerging web service protocol stack.

- **Web Service Roles**

  There are three major roles within the web service architecture −

  A. *Service Provider*

  This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

  B. *Service Requestor*

  This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

  C. *Service Registry*

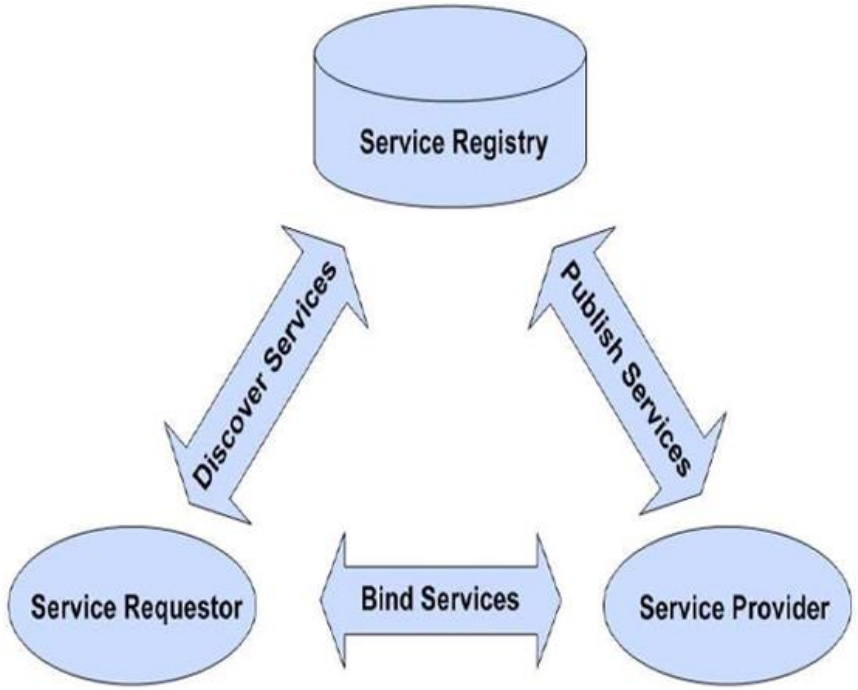
  This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

SERVICE-ORIENTED ARCHITECTURE

- Consider a scenario of finding a plumber to fix a plumbing problem.
  - Find a plumber in Yellow Pages (Discovery)
  - Plumber should have advertised in the Yellow Pages (Publishing)
  - You call plumber to schedule appointment (Binding)

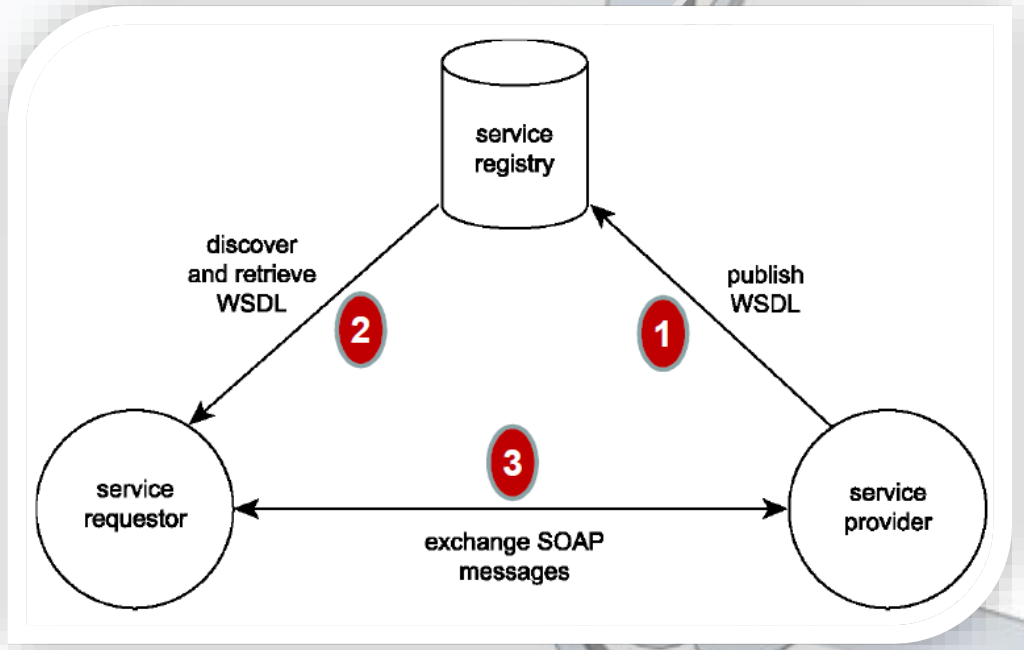
- Three basic participants
  - Service Provider
  - Service Registry
  - Service Requestor

- Three basic operations
  - Publishing services
  - Discovering services
  - Binding services



SERVICE-ORIENTED ARCHITECTURE

- **1. Service Provider**
  - **– Develops an application and converts into a service**
  - **– Creates a WSDL document describing the capabilities of the service and how to access the service**
  - **– Publishes the WSDL document in a service registry (UDDI)**
- **2. Service Requestor**
  - **– Needs a service for specific task/purpose**
  - **– Searches for services in the service registry meeting needs**
  - **– Selects a service that is satisfactory**
- **3. Service Requestor's application invokes the provider's service**
  - **• Upon acceptance, application and service can exchange data using SOAP**



WEB SERVICE ARCHITECTURE

- **Web Service Protocol Stack**

A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers.

A. Service Transport

This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP).

B. XML Messaging

This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP

WEB SERVICE ARCHITECTURE

C. Service Description

This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

D. Service Discovery

This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

As web services evolve, additional layers may be added and additional technologies may be added to each layer.

WEB SERVICE ARCHITECTURE

- **Few Words about Service Transport**

  The bottom of the web service protocol stack is service transport. This layer is responsible for actually transporting XML messages between two computers

  A. *Hyper Text Transfer Protocol (HTTP)*

  Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic.

  This allows XMLRPC or SOAP messages to masquerade as HTTP messages. This is good if you want to integrate remote applications, but it does raise a number of security concerns rise a number of security concerns.

WEB SERVICE ARCHITECTURE

- **Few Words about Service Transport**

  B. *Blocks Extensible Exchange Protocol (BEEP)*

  This is a promising alternative to HTTP. BEEP is a new Internet Engineering Task Force (IETF) framework for building new protocols. BEEP is layered directly on TCP and includes a number of built-in features, including an initial handshake protocol, authentication, security, and error handling.

  Using BEEP, one can create new protocols for a variety of applications, including instant messaging, file transfer, content syndication, and network management.

  SOAP is not tied to any specific transport protocol. In fact, you can use SOAP via HTTP, SMTP, or FTP. One promising idea is therefore to use SOAP over BEEP.

USE OF WEB SERVICES FOR APPLICATION INTEGRATION

- **Web service supports application-to-application communication**
  - **Supports “loosely coupled” integration**
  - **Minimizes the amount of effort required to build integrated applications.**
- **With Web services,**
  - **the applications or programmers can find cooperative programs to accomplish a specific task,**
  - **allowing programmers to rapidly assemble applications by merely tying together application modules.**
- **Web services are designed to enable application modules (objects) to communicate with other application modules.**
- **Once connected, service applications provide transactional or computational services.**

USE OF WEB SERVICES FOR APPLICATION INTEGRATION

- **Support from leading application frameworks**

  Let's quickly review how the leading application frameworks—Microsoft .NET and J2EE—provide support to enable function/method-oriented integration.

  Microsoft .NET

  Microsoft .NET is the Microsoft XML Web services platform. It provides built-in support for building and consuming standards-based Web services. In Microsoft's .NET framework, client applications can invoke a Web service by implementing a Web service listener.

  Using Microsoft .NET, function/method-oriented integration can occur as follows:

  The server application (Web service provider) can implement the Web service in any of the .NET languages, such as C#, VB.NET, or Managed C++, which can be compiled into Microsoft Intermediate Language (MSIL) and executed in a virtual machine called Common Language Runtime (CLR).

USE OF WEB SERVICES FOR APPLICATION INTEGRATION

- **Support from leading application frameworks**

Microsoft .NET

The Web service can be deployed on a .NET platform.

The client application (Web service user) can implement the Web service listener using MSXML or ASP.NET and invoke the Web service using a method call.

J2EE

J2EE is a set of specifications, each of which dictates how various J2EE functions must operate. It provides Java API for XML-based RPC (JAX-RPC) to support function/method-oriented Web services integration.

JAX-RPC uses XML to make remote procedure calls (RPC) and exposes an API for marshalling and unmarshalling arguments, transmitting and receiving procedure calls. As of now, the JAX-RPC reference implementation relies on SOAP 1.1 and HTTP 1.1.

USE OF WEB SERVICES FOR APPLICATION INTEGRATION

- **Support from leading application frameworks**

J2EE

Using JAX-RPC, function/method-oriented integration can occur as follows:

Define and implement a JAX-RPC-based Web service. The implementation of Web service can be in stand-alone Java application or Enterprise Java Bean (EJB). JAX-RPC API can be used to create SOAP-based wrappers that confirm to the WSDL interface for existing Java classes or EJBs.

Deploy the Web service on a server-side JAX-RPC runtime system. The deployment is governed by the implementation of the Web service; for example, if the implementation is provided by EJB, the deployment will be in an EJB container.

Invoke the Web service from the client application on the port described by the WSDL document. To the client application, the invocation of the Web service would appear to be a local method call.

USE OF WEB SERVICES FOR APPLICATION INTEGRATION

- **Business requirements, application architecture, business processes, and enterprise data integration policy determines the use of functions where web services provide standard methods to publish and subscribe to apps over a network.**

- **Where client apps can locate the services published by servers using UDDI. WSDL provides information related to the interface of the service, and XML and SOAP are used for formatting data.**

THANK YOU.

ramil.huele@cvsu.edu.ph