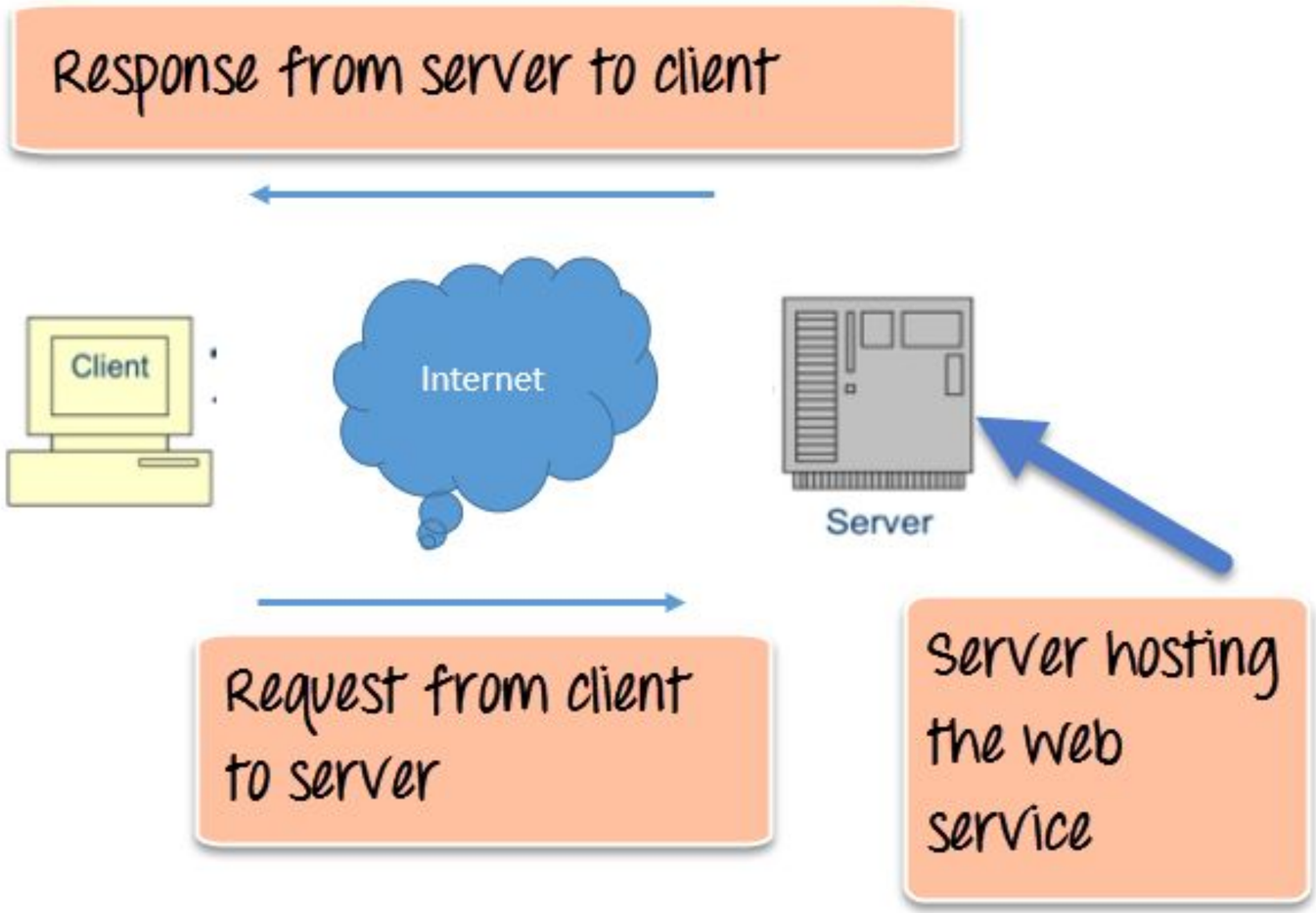
Web Services

Malintha Adikari

Web Services

- Web services are open standard (XML, SOAP, HTTP, etc.) based web applications that interact with other web applications for the purpose of exchanging data.
- Web services can convert existing applications into web applications.
- 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



Web Services

- Main component of a web service is the data which is transferred between the client and the server in XML format.
- XML (Extensible markup language) is the intermediate language that is understood by many programming languages.
- Web services talk in XML when applications talk to each other,.
- XML provides a common platform for application developed on various programming languages to talk to each other.

Message Exchange Protocol

- Web services use SOAP (Simple Object Access Protocol) for sending the XML data between applications.
- The data is sent over normal HTTP.
- The data which is sent from the web service to the application is called a SOAP message.
- The SOAP message is nothing but an XML document.
- Since the document is written in XML, the client application calling the web service can be written in any programming language.

Sample SOAP Message

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-enve
lope"
soap:encodingStyle="http://www.w3.org/2001/12/so
ap-encoding">

<soap:Body
xmlns:m="http://www.example.org/stock">
 <m:GetStockPrice>
   <m:StockName>IBM</m:StockName>
 </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Web Services Components

- **SOAP** - Simple Object Access Protocol
- **WSDL** - Web services description language
- **UDDI** - Universal Description, Discovery and Integration

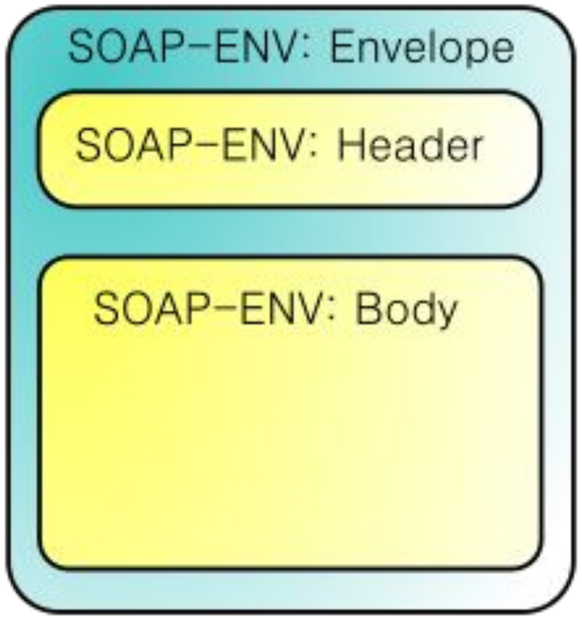
SOAP

- ➔ SOAP provides the Messaging Protocol layer of a web services protocol stack for web services
- ➔ It is an XML-based protocol consisting of three parts:
  - ◆ an envelope, which defines the message structure and how to process it
  - ◆ a set of encoding rules for expressing instances of application-defined datatypes
  - ◆ a convention for representing procedure calls and responses

SOAP

A SOAP message is an ordinary XML document containing the following elements:

Element	Description	Required
Envelope	Identifies the XML document as a SOAP message.	Yes
Header	Contains header information.	No
Body	Contains call, and response information.	Yes
Fault	Provides information about errors that occurred while processing the message.	No



WSDL

- **Web Services Description Language** (**WSDL**) is an XML-based interface definition language used for describing the functionality offered by a web service.
- Provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns.
- Its purpose is roughly similar to that of a method signature in a programming language.

Refer: https://msdn.microsoft.com/en-us/library/ms996486.aspx

Sample WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<description xmlns="http://www.w3.org/ns/wsdl"
        targetNamespace="http://www.tmsws.com/wsdl20sample">
<documentation>
   This is a sample WSDL 2.0 document.
</documentation>
<!-- Abstract type -->
  <types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns="http://www.tmsws.com/wsdl20sample"
          targetNamespace="http://www.example.com/wsdl20sample">
      <xs:element name="request"> ... </xs:element>
      <xs:element name="response"> ... </xs:element>
    </xs:schema>
  </types>
<!-- Abstract interfaces -->
  <interface name="Interface1">
    <fault name="Error1" element="tns:response"/>
    <operation name="Get"
pattern="http://www.w3.org/ns/wsdl/in-out">
      <input messageLabel="In" element="tns:request"/>
      <output messageLabel="Out" element="tns:response"/>
    </operation>
  </interface>
<!-- Concrete Binding Over HTTP -->
  <binding name="HttpBinding" interface="tns:Interface1"
        type="http://www.w3.org/ns/wsdl/http">
    <operation ref="tns:Get" whttp:method="GET"/>
  </binding>

<!-- Concrete Binding with SOAP-->
  <binding name="SoapBinding" interface="tns:Interface1"
        type="http://www.w3.org/ns/wsdl/soap"
        wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"

wsoap:mepDefault="http://www.w3.org/2003/05/soap/mep/request-respons
e">
    <operation ref="tns:Get" />
  </binding>

<!-- Web Service offering endpoints for both bindings-->
  <service name="Service1" interface="tns:Interface1">
    <endpoint name="HttpEndpoint"
          binding="tns:HttpBinding"
          address="http://www.example.com/rest/"/>
    <endpoint name="SoapEndpoint"
          binding="tns:SoapBinding"
          address="http://www.example.com/soap/"/>
  </service>
</description>
```

UDDI

- ➔ Web services discovery provides access to software systems over the Internet using standard protocols.
- ➔ Web Service Discovery is the process of finding suitable web services for a given task.
- ➔ Publishing a Web service involves creating a software artifact and making it accessible to potential consumers.
- ➔ Web Service Providers augment a Web service endpoint with an interface description using the Web Services Description Language (WSDL) so that a consumer can use the service.
- ➔ Optionally, a provider can explicitly register a service with a Web Services Registry such as Universal Description Discovery and Integration (UDDI)
- ➔ The service users or consumers can search Web Services manually or automatically.

Why Web Services

- **Exposing Business Functionality on the network** - Nowadays all applications are on the internet which makes the purpose of Web services more useful. The web service can be anywhere on the internet and provide the necessary functionality as required.
- **Interoperability amongst applications** - Web services allow various applications to talk to each other and share data and services among themselves.
- **A Standardized Protocol which everybody understands** - Web services use standardized industry protocol for the communication.
- **Reduction in cost of communication** - Web services use SOAP over HTTP protocol, can use your existing low-cost internet for implementing web services.

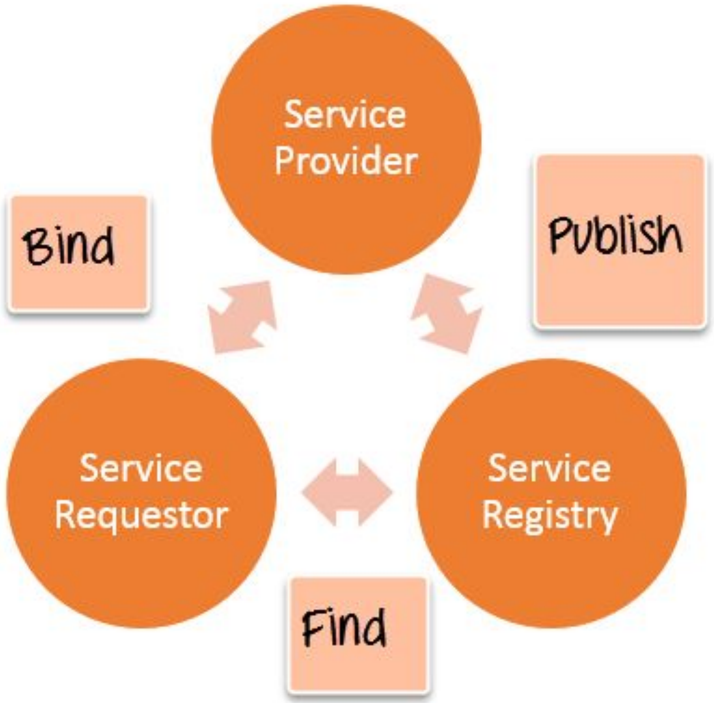
Web Service Architecture

Architecture consists of three distinct roles

1. **Provider** - The provider creates the web service and makes it available to client application who want to use it.
2. **Requestor** - A requestor is the client application that needs to contact a web service. The client application can be a .Net, Java, or any other language based application which looks for some sort of functionality via a web service.
3. **Broker** - The broker is the application which provides access to the UDDI which enables the client application to locate the web service.

Web Service Architecture

1. **Publish** - A provider informs the broker (service registry) about the existence of the web service by using the broker's publish interface to make the service accessible to clients
2. **Find** - The requestor consults the broker to locate a published web service
3. **Bind** - With the information it gained from the broker(service registry) about the web service, the requestor is able to bind, or invoke, the web service.



Web Service Characteristics

- **XML-Based** - Web Services uses XML to represent the data at the representation and data transportation layers. Using XML eliminates any networking, operating system, or platform sort of dependency since XML is the common language understood by all.

- **Loosely Coupled** – Loosely coupled means that the client and the web service are not bound to each other, which means that even if the web service changes over time, it should not change the way the client calls the web service.

- **Synchronous or Asynchronous functionality**- Synchronicity refers to the binding of the client to the execution of the service. In synchronous operations, the client will actually wait for the web service to complete an operation.

Web Service Characteristics

- **Ability to support Remote Procedure Calls (RPCs)** - Web services enable 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.

- **Supports Document Exchange** - One of the key benefits 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.

SOA Principles

- Service-oriented architecture (SOA) is an architectural pattern in which application components provide services to other components via a communications protocol.
- Principles of service-orientation are independent of any product, vendor or technology.
- SOA makes it easier for software components over various networks to work with each other.
- Web services which are built as per the SOA architecture tend to make web service more independent.
- The web services themselves can exchange data with each other and because of the underlying principles on which they are created, they don't need any sort of human interaction and also don't need any code modifications.
- It ensures that the web services on a network can interact with each other seamlessly.

SOA Principles

- **Standardized Service Contract** - Services adhere to a service-description. A service must have some sort of description which describes what the service is about. This makes it easier for client applications to understand what the service does.

- **Loose Coupling** – Less dependency on each other. This is one of the main characteristics of web services which just states that there should be as less dependency as possible between the web services and the client invoking the web service. So if the service functionality changes at any point in time, it should not break the client application or stop it from working.

- **Service Abstraction** - Services hide the logic they encapsulate from the outside world. The service should not expose how it executes its functionality; it should just tell the client application on what it does and not on how it does it.

SOA Principles

- **Service Reusability** - Logic is divided into services with the intent of maximizing reuse. Once the code for a web service is written it should have the ability work with various application types.

- **Service Autonomy** - Services should have control over the logic they encapsulate. The service knows everything on what functionality it offers and hence should also have complete control over the code it contains.

- **Service Statelessness** - Services should be stateless. Means that services should not withhold information from one state to the other.

SOA Principles

- **Service Discoverability** - Services can be discovered. UDDI performs a registry which can hold information about the web service.

- **Service Composability** - Services break big problems into little problems. One should never embed all functionality of an application into one single service but instead, break the service down into modules each with a separate business functionality.

- **Service Interoperability** - Services should use standards that allow diverse subscribers to use the service. In web services, standards as XML and communication over HTTP is used to ensure it conforms to this principle.

Thanks!

Any questions?