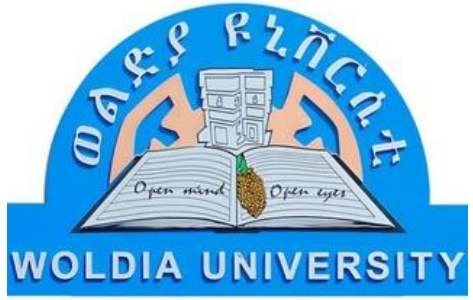


Woldia University
School of Computing

Department of Software Engineering

Course Title: **Web Service,**

Course Code: **SEng5127**

Chapter 1: Web Services Overview

By : Demeke G.

AY:2018 E.C

Outline

- Over view of web service
- Web Service Architecture
- Web Services vs. Other Technologies
- Benefits of Web Services
- Interoperable Applications with SOA
  - Designing an SOA Integration Architecture
  - Implementing SOA with Web Services
- Java Standard APIs for Web Services
  - JAX-WS Overview
  - SOAP-Based Services with JAX-WS
  - RESTful Services with JAX-RS

What is web service?

- A software Component that enables two different applications to communicate and exchange data over a network.
- A web service is a set of open protocols and standards that allow data to be exchanged between different applications or systems.
- A complete web service is, 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
  - ✓Loosely Coupled

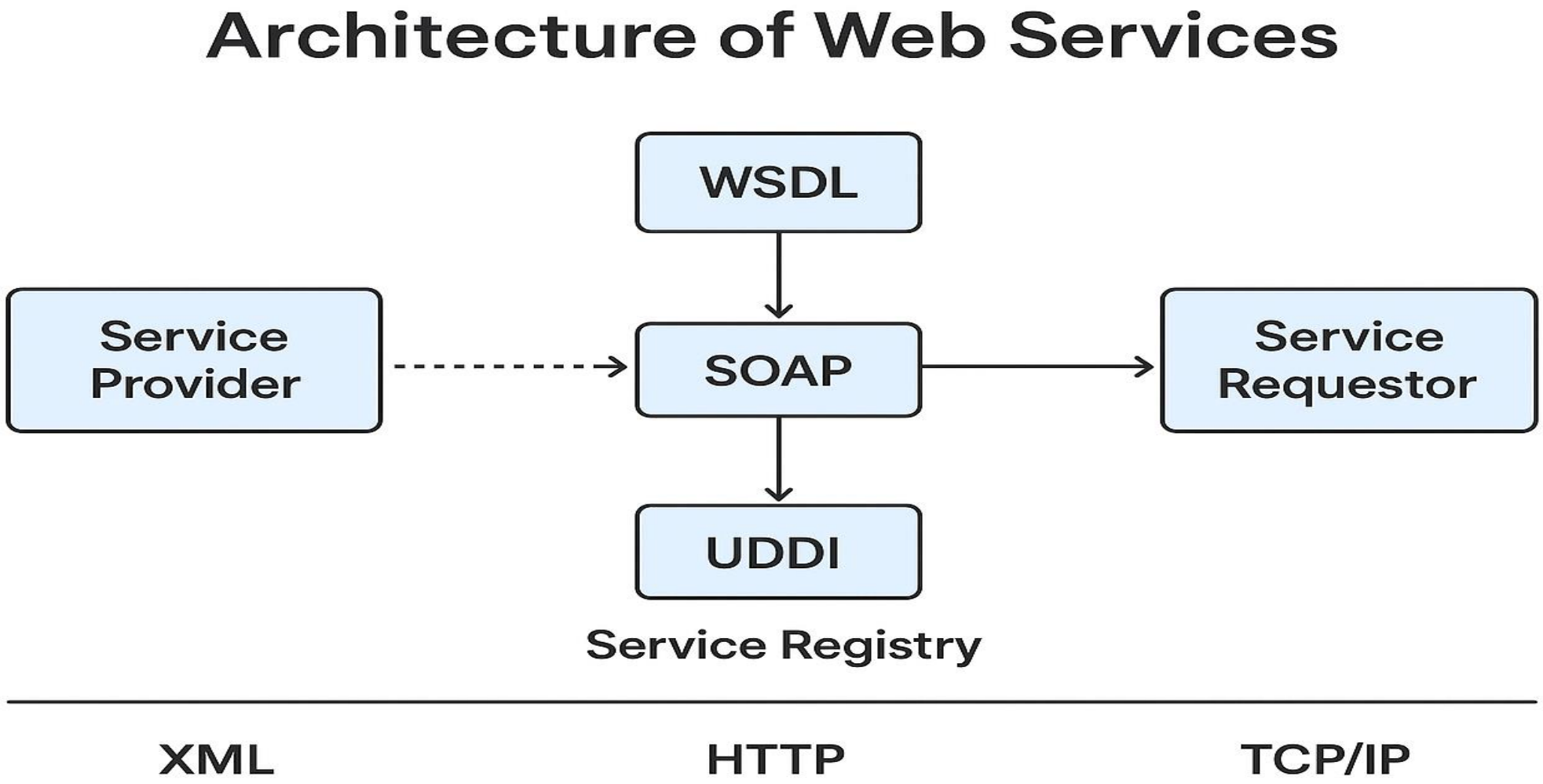
Key Characteristics of Web Service

- **Interoperability**: WS allow applications written in different programming languages to communicate, fostering cross-technology implementations.
- **Standardized Protocols:** They rely on open protocols and standards like Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) to define and interact with the services.
- **Network-Based:** Communication happens over a network, typically the internet, using web-related standards such as HTTP/HTTPS.
- **Machine-to-Machine Interaction:** allow different software applications to interact and exchange data without human intervention.
- **Self-Contained and Modular:** They are often self-contained modules that can be used independently or combined with other services to perform complex tasks.

Architecture of Web Services

- The **structural design** that defines how different components of a web service interact and communicate over the internet.
- It provides a framework that enables interoperability, standardized communication, and integration between different applications running on various platforms and written in different programming languages.
- Web services consist of **Service Provider**, **Service Requester**, and **Service Registry** to enable interaction.
  - **Service Provider :** provider of the web service.
  - **Service Requestor:** any consumer of the web service.
  - **Service Registry**: this is a logically centralized directory of services. (published new or find existing services

Architecture of Web Service



WSDL (Web Services Description Language)

- WSDL is an XML-based language used to describe web services.
- WSDL was developed jointly by Microsoft and IBM.
- It specifies what the service does, where it resides, and how to invoke it.
- WSDL is primarily used with SOAP-based web services.
- It acts as a contract between the service provider and the consumer, defining the technical interface in a machine-readable format.
- Key Components of WSDL:
  - **Types** – Defines the data types used by the web service.
  - **Message** – Defines the messages (requests/responses) exchanged between client and server.
  - **PortType** – Lists the operations (functions) provided by the service.
  - **Binding** – Specifies the protocol (e.g., SOAP over HTTP) and message format.
  - **Service** – Defines the endpoint (URL) where the service can be accessed.

WSDL Example

```
<definitions name="StudentService"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>...</wsdl:types>
  <wsdl:message name="GetStudentRequest">...</wsdl:message>
  <wsdl:portType name="StudentPortType">
    <wsdl:operation name="getStudentDetails"/>
  </wsdl:portType>
  <wsdl:binding name="StudentBinding" type="StudentPortType"/>
  <wsdl:service name="StudentService"/>
</definitions>
```

UDDI (Universal Description, Discovery, and Integration)

- UDDI is a platform-independent registry for web services.
- It allows organizations to publish and discover web services over the Internet.
- UDDI is a specification for a distributed registry of web services.
- UDDI can communicate via SOAP, CORBA, and Java RMI Protocol.
- UDDI uses WSDL to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.
- Key Components:
  - **Business Entity** – Information about the organization providing the service.
  - **Business Service** – The actual web service offered.
  - **Binding Template** – How and where the service can be accessed (e.g., endpoint URL).
  - **tModel** – Technical model describing specifications, standards, or protocols used.

UDDI example

```
<businessEntity businessKey="uuid:1234">
  <name>ABC University</name>
  <description>Provides student information services</description>
  <businessService serviceKey="uuid:5678">
    <name>StudentInfoService</name>
    <bindingTemplates>
      <bindingTemplate bindingKey="uuid:9999">
        <accessPoint useType="wsdlDeployment">
          http://example.com/studentService?wsdl
        </accessPoint>
      </bindingTemplate>
    </bindingTemplates>
  </businessService>
</businessEntity>
```

Web Services vs. Other Technologies

❖Platform Independence

Web services operate across various platforms unlike COM/DCOM which are Windows-specific technologies.

❖ Language Neutral Data Exchange

Using XML or JSON, WS enable seamless data exchange between heterogeneous systems.

❖ Comparison with Other Technologies

Unlike tightly coupled CORBA or RMI, WS support loose coupling and interoperability for distributed systems.

❖ SOAP vs REST APIs

SOAP offers robust features like built-in security, while REST is lightweight and ideal for web applications.

Web Service vs Web API vs REST API vs SOAP API

A comparison of different web communication approache

Web Service

A general term for machine-to-machine communication over a network

SOAP API

A web service following the SOAP protocol

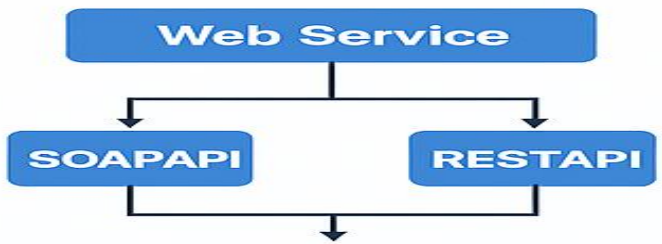
REST API

A web service following REST principles

Web API

An interface to build web communication

	Web Service	SOAP API	REST API	Web API
Architecture style	Service-oriented	Protocol-based	Depends style	Depends on implem
Protocols	HTTP, SOAP,	SOAP	HTTP/HTPS	HTTP/HTTPS
Data format	XML or JàN	XML only	JSON (most common)	HTTP motsly
Communication	Can be SOAP or REST	Strict SOAP envelope	H(? methods (GET: POST. PUT. DELETE)	HTTP methels
Use case	SOAP or REST systems	Enterprise systems (e.g. banking, ERP	Web/moblle apps	Web/mobile apps



Real-world use cases

- SOAP API are commonly used in banking. telecommunication, and ERP systems
- **REST API** are used in web and mobile applications, social netvorks,

Benefits of Web Services

- **Interoperability :** Web services enable seamless communication across different platforms using standard protocols like HTTP, XML, and JSON.
- **Reusability:** Developed web services can be reused across multiple applications, reducing development time and costs.
- **Scalability and Flexibility:** Web services support scalable distributed architectures and promote loose coupling for easier maintenance and updates.
- **Accessibility Over the Internet:** Services are available through standard web protocols, making them globally accessible.
- **Cost-Effective Integration**: Reduces the need for custom connectors between different systems.
- **Supports B2B and Enterprise Integration:** Facilitates business-to-business data exchange and enterprise collaboration.

Interoperable Applications with SOA

- SOA is an architectural approach that allows applications built on different platforms and technologies to interact and share data through reusable, standardized services.
- Web services are the most common technology used to implement SOA, providing interoperability across **heterogeneous systems**.

Core principles of SOA design

- **Loose Coupling:** Services should be designed with minimal dependencies on each other's implementation details. This ensures services can be updated or replaced independently without disrupting the entire system.
- **Reusability:** Services should be designed to be consumed by different applications and business processes to reduce development costs and increase speed to market.

Core principles of SOA design ...

- **Statelessness**: By keeping every request independent, your architecture becomes more scalable and resilient to failures.
- **Standardized contracts:** Services must communicate through a common, agreed-upon communication protocol and message format, often defined by a contract like a WSDL for SOAP or an OpenAPI specification for REST.
- **Abstraction:** Services should hide their internal logic from consumers,.
- **Composability:** Simple services can be combined into more complex, composite services to automate entire business processes.
- **Discoverability**: Services should be documented and discoverable through a central registry or repository, so that potential consumers can easily find and understand them.
- **Autonomy**: Services should have full control over the logic and resources they encapsulate.

Implementing SOA with Web Services

- Web services are the primary technology for realizing SOA principles.
- Implementation Steps:
  - **Define Services:** Identify the business operations to expose.
  - **Describe Services:** Use **WSDL** to define service interfaces.
  - **Publish Services:** Register services using **UDDI** or a private registry.
  - **Discover Services:** Allow clients to search for available services.
  - **Bind and Invoke:** Clients call services using **SOAP** or **REST** over **HTTP**.
- **Technologies Used:**
  - **SOAP (Simple Object Access Protocol)** – Message format for data exchange.
  - **WSDL (Web Services Description Language)** – Describes available operations.
  - **UDDI (Universal Description, Discovery, and Integration)** – Registry for finding services.
  - **XML/JSON** – Standard data formats for interoperability.

SOA vs. Microservices

Feature	SOA	Microservices
Scope	Enterprise-level, integrating multiple applications.	Application-level, focusing on splitting a single app into services.
Service Granularity	Coarse-grained, larger services encompassing broad functionalities.	Fine-grained, small services focused on a single business function.
Data Management	Centralized, often sharing a single database.	Decentralized, with each service managing its own data.
Communication	Often relies on an ESB for centralized message handling.	Uses lightweight communication mechanisms, like RESTful APIs, directly between services.
Deployment	Often involves deploying the entire application as a single unit, which can be complex.	Each service can be deployed and scaled independently.
Best For	Large, complex enterprises integrating many legacy systems.	Fast-moving organizations that prioritize rapid innovation, agility, and continuous delivery.

Java Standard APIs for Web Services

- Java provides standardized APIs for developing both SOAP-based and RESTful web services.
- The main APIs are:
  - JAX-WS (Java API for XML Web Services) – for SOAP-based services.
  - JAX-RS (Java API for RESTful Web Services) – for RESTful services.
- **JAX-WS Overview (Replacing JAX-RPC)**
- JAX-WS replaced the older JAX-RPC API to provide a more flexible and annotation-driven model for developing SOAP web services.
- Based on SOAP protocol and WSDL for service description.
- Use XML messages for structured communication.

RESTful Web Services

- RESTful Web Services are a type web service architecture that allows systems to communicate over the network using standard HTTP methods.
- REST is widely used in modern web and mobile applications because it is lightweight, scalable, and easy to implement.
- REST allows clients (like browsers or mobile apps) to access and manipulate web resources using a stateless communication model.
- Each resource is identified by a URI (Uniform Resource Identifier), and standard HTTP methods are used to perform operations on these resources.
- They are stateless, lightweight, and platform-independent, making them ideal for building scalable web APIs.
- HTTP Methods: GET → Retrieve data, POST → Create data ,PUT → Update data, DELETE → Remove data
- The client handles the UI, and the server manages data and logic.
- Each request is independent; the server does not store client state.

Java API for RESTful Web Services (JAX-RS)

- **JAX-RS** simplifies the creation of RESTful web services using **HTTP methods** and **annotations**.
- Annotations simplify the mapping of Java classes and methods to web resources and HTTP operations.
  - @Path: Defines the URI path for a resource class or method.
  - @GET, @POST, @PUT, @DELETE: Map methods to corresponding HTTP request methods.
  - @Produces: Specifies the MIME types (e.g., application/json, text/html) that a resource method can produce in its response.
  - @Consumes: Specifies the MIME types that a resource method can accept in its request body.
  - @PathParam, @QueryParam, @FormParam, @HeaderParam, @CookieParam: Used to extract values from various parts of an HTTP request.
- Exchanges data in JSON or XML formats.
- Lightweight and ideal for mobile and web applications.
- Simpler and faster than SOAP.

End of the chapter