LECTURE 03

03

Web Services Fundamentals

The building blocks that make SOA implementable in practice: XML, XML Schema, WSDL service contracts, and the UDDI registry standard.

Agenda

1. What is a web service, and why XML?
2. XML fundamentals and XML Schema (XSD)
3. WSDL: describing a service contract
4. Anatomy of a WSDL document
5. UDDI and service registries
6. Putting it together: publish-find-bind with real standards

DEFINITION

Web Service

“A software function accessible over a network using standard, platform-independent protocols, described by a machine-readable interface that any compliant client can consume.”

- The 'web' in web service refers to the use of standard web/internet protocols (HTTP, XML), not necessarily a browser
- Web services are the primary technology used to implement SOA in practice

Why XML Became the Foundation of Web Services

- **Platform-neutral**
  - Plain text, readable by any system regardless of operating system or language
- **Self-describing**
  - Tag names describe the meaning of the data they contain
- **Extensible**
  - New elements can be added without breaking systems that don't understand them
- **Widely tooled**
  - Parsers, validators, and editors exist for virtually every programming language
- XML solved the core SOA need: a common data format every platform could read and write

XML Basics — A Simple Document

```
<?xml version="1.0" encoding="UTF-8"?>
<customer>
  <id>C1042</id>
  <name>Sara Ahmed</name>
  <email>sara.ahmed@example.com</email>
  <address>
    <city>Amman</city>
    <country>Jordan</country>
  </address>
</customer>
```

Elements are opened and closed with matching tags; nesting expresses structure and relationships between data.

Core XML Concepts

- **Element**
  - A tagged piece of data, e.g. <name>Sara Ahmed</name>
- **Attribute**
  - Extra information inside a tag, e.g. <customer id="C1042">
- **Well-formed document**
  - Follows XML syntax rules: one root element, properly nested and closed tags
- **Valid document**
  - Well-formed AND conforms to a schema that defines its expected structure
- Namespaces (xmlns) prevent tag-name collisions when combining XML vocabularies from different sources

DEFINITION

XML Schema (XSD)

“A W3C standard for defining the structure, content, and data types that a valid XML document of a given type must conform to.”

- XSD is itself written in XML, making it tool-friendly and language-neutral
- A service contract uses XSD to define exactly what a valid request or response message must look like

XML Schema — Defining the Customer Element

```
<xs:element name="customer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="xs:string"/>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

This schema states that a valid <customer> element must contain, in order, an id, a name, and an email.

Why Web Services Need More Than Just XML

- XML defines what data looks like — but a service also needs to describe:
  - What operations are available (e.g., GetCustomer, CreateOrder)
  - What messages each operation accepts and returns
  - How to physically reach the service (protocol, address)
- **This is exactly what WSDL was designed to describe**

DEFINITION

WSDL (Web Services Description Language)

“An XML-based, W3C standard language used to describe the functionality, message formats, and network location of a web service, so it can be automatically discovered and invoked.”

- WSDL is the formal, machine-readable contract at the heart of SOAP-based SOA
- Development tools can generate client code automatically from a WSDL file

The Structure of a WSDL Document

types	Data types used in messages (usually via embedded XSD)
message	Defines the input/output data for each operation
portType (interface)	The abstract set of operations offered
binding	Maps the portType to a concrete protocol (e.g., SOAP over HTTP)
service / port	The actual network address (URL) where the service runs

Abstract to Concrete

WSDL moves from the abstract (what data, what operations) down to the concrete (what protocol, what address). This separation is what allows the same abstract interface to be bound to different protocols if needed.

WSDL — Defining Messages

```
<message name="GetCustomerRequest">
  <part name="customerId" type="xs:string"/>
</message>

<message name="GetCustomerResponse">
  <part name="customer" type="tns:customer"/>
</message>
```

Each <message> defines the shape of data flowing in or out of an operation.

WSDL — Defining the Port Type (Interface)

```
<portType name="CustomerServicePortType">
  <operation name="GetCustomer">
    <input message="tns:GetCustomerRequest"/>
    <output message="tns:GetCustomerResponse"/>
  </operation>
</portType>
```

The portType groups related operations into a coherent interface — conceptually similar to an interface in an object-oriented language.

WSDL — Binding and Service Location

```
<binding name="CustomerServiceBinding" type="tns:CustomerServicePortType">
  <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetCustomer">
    <soap:operation soapAction="getCustomer"/>
  </operation>
</binding>

<service name="CustomerService">
  <port name="CustomerServicePort" binding="tns:CustomerServiceBinding">
    <soap:address location="https://api.example.com/CustomerService"/>
  </port>
</service>
```

The binding ties the abstract interface to SOAP over HTTP; the service/port gives the actual URL clients call.

Reading a WSDL: A Consumer's Perspective

- A developer (or their IDE) opens a WSDL file and can immediately learn:
  - What operations exist — e.g., GetCustomer, CreateOrder
  - What input parameters each operation needs
  - What output format to expect back
  - What network address and protocol to use
- Tools like wsimport (Java) or 'Add Service Reference' (.NET) auto-generate client-side proxy code directly from the WSDL
- This automatic code generation was one of SOAP/WSDL's biggest practical advantages in enterprise development

WSDL 1.1 vs. WSDL 2.0

- **WSDL 1.1 (2001)**
  - The version overwhelmingly used in industry; term 'portType' for interfaces
- **WSDL 2.0 (2007, W3C Recommendation)**
  - Renamed portType to 'interface', added native support for REST-style HTTP bindings
- In practice, most enterprise SOAP tooling — and most exam and textbook material — still centers on WSDL 1.1

DEFINITION

UDDI (Universal Description, Discovery, and Integration)

“An XML-based standard for building registries of web services, allowing providers to publish service descriptions and consumers to search for and locate them.”

- UDDI was designed to be the 'yellow pages' of web services in the publish-find-bind model
- Public, global UDDI registries largely failed to gain adoption — but the concept lives on in private, enterprise service registries and modern API catalogs

The Three Data Structures of UDDI

- **White Pages**
  - Basic contact and business identification information about the provider
- **Yellow Pages**
  - Categorization of services by industry, product, or geographic region
- **Green Pages**
  - Technical details about how to interact with the service — including a pointer to its WSDL
- This structure mirrors a real-world phone-book analogy, which is where the naming comes from

Why Public UDDI Registries Failed

- **Trust and quality concerns**
  - No guarantee that a listed public service was reliable, secure, or maintained
- **Limited real business need**
  - Most organizations wanted to integrate with known partners, not discover random public services
- **Complexity**
  - The API and data model were seen as heavier than the actual industry need
- Microsoft, IBM, and SAP shut down the public UDDI Business Registry in 2005–2006
- Lesson: private, governed service registries within an enterprise remain valuable, even though the public vision did not succeed

Publish–Find–Bind With Real Standards

1

1. Publish

Provider writes a WSDL file and registers it (historically in UDDI; today in an API catalog).

2

2. Find

Consumer searches the registry/catalog and retrieves the WSDL document.

3

3. Bind

Consumer's tooling generates a client proxy from the WSDL and invokes the service via SOAP over HTTP.

Web Service Standards Stack — Summary

Standard	Layer	Role
XML	Data format	Represents structured data platform-independently
XSD	Data typing	Validates the structure and types within XML messages
WSDL	Interface description	Describes operations, messages, and endpoint location
SOAP	Messaging protocol	Wraps and transports messages between endpoints
UDDI	Discovery	Registry for publishing and finding service descriptions

Namespaces: Avoiding Naming Collisions

- Two organizations might both define an element called <order> with completely different meanings
- **XML namespaces solve this using a URI as a unique identifier:**
  - xmlns:tns="http://example.com/orders"
- Prefixes (like tns:, xs:, soap:) are just local aliases that map to a full namespace URI
- WSDL files rely heavily on namespaces to unambiguously connect types, messages, portTypes, and bindings

Design-Time vs. Runtime Artifacts

- **Design-time artifacts**
  - WSDL and XSD files — used by developers and tools to build client and server code before deployment
- **Runtime artifacts**
  - The actual SOAP messages exchanged over the network while the system is running
- Understanding this distinction matters: a WSDL rarely changes once published, but the messages flowing through it happen continuously

Contract-First vs. Code-First Development

- **Contract-first (WSDL-first)**
  - The team designs the WSDL contract first, then generates or writes server code to implement it
  - Advantage: contract is stable and platform-neutral from the start
- **Code-first**
  - The team writes the service implementation first, and a WSDL is auto-generated from the code
  - Advantage: faster initial development; risk: contract can leak implementation details
- Most mature SOA governance programs strongly prefer contract-first design

Versioning a Service Contract

- **Services evolve — but existing consumers must not break**
- **Backward-compatible changes**
  - Adding new optional elements or new operations
- **Breaking changes**
  - Removing/renaming fields, changing data types, changing required parameters
- **Common versioning strategies:**
  - Namespace versioning, endpoint versioning (e.g., /v1/, /v2/), or explicit version elements in the message

Tooling in Practice

- **Java: wsimport / JAX-WS generate client and server stubs from WSDL**
- **.NET: 'Add Service Reference' or svcutil.exe generate proxy classes**
- **SoapUI: widely used tool to test SOAP services directly from a WSDL**
- These tools remove most of the manual XML-writing burden — but understanding the underlying WSDL/XML is essential for debugging and designing good contracts

Common Mistakes When Working with WSDL/XSD

- **Overly permissive types**
  - Using generic 'string' for everything instead of precise types and restrictions
- **Exposing internal naming**
  - Copying internal database column names directly into the XSD
- **Ignoring namespace design**
  - Sloppy or missing namespaces causing collisions as the service inventory grows
- **Skipping documentation**
  - A WSDL without <documentation> elements is hard for other teams to understand and reuse

Key Takeaways

1. XML provides the platform-neutral, self-describing data format underlying most web-service standards
2. XML Schema (XSD) defines the precise structure and types a valid XML message must follow
3. WSDL is the formal, machine-readable contract describing a service's operations, messages, and location
4. UDDI provided a registry model for publish-find-bind, though its public vision did not achieve lasting adoption
5. Contract-first design and careful versioning are essential SOA governance practices

DISCUSSION / REVIEW QUESTION

A new developer suggests skipping the WSDL entirely and letting consumers 'just read the code' to understand a service's operations. What problems would this cause in a real enterprise SOA environment?

Think about:

- Consider consumers built in a completely different programming language
- Think about automatic client-code generation tools
- Consider what happens when the internal implementation changes

References & Further Reading

- W3C — Web Services Description Language (WSDL) 1.1 Specification
- W3C — XML Schema Part 0: Primer
- OASIS — UDDI Version 3.0.2 Specification
- Erl, T. — Web Service Contract Design and Versioning for SOA

NEXT LECTURE

Lecture 4 — SOAP-Based Services: Message Structure, Protocols, and Standards