Basics regarding Web-services

Definition…..

A Web service is a software application, accessible on the Web (or an enterprise's intranet) through a URL, that is accessed by clients using XML-based protocols, such as Simple Object Access Protocol (SOAP) sent over accepted Internet protocols, such as HTTP.

In a typical Web services scenario, a business application sends a request to a service at a given URL using the SOAP protocol over HTTP.

The service receives the request, processes it, and returns a response.

Simple example of a Web service

A stock quote service ----
The request asks for the current price of a specified stock, and the response gives the stock price.

This is one of the simplest forms of a Web service in that the request is filled almost immediately, with the request and response being parts of the same method call.

Another example ------
A service that maps out an efficient route for the delivery of goods.

In this case, a business sends a request containing the delivery destinations. The service processes it, to determine the most cost-effective delivery route. The time it takes to return the response depends on the complexity of the routing, but the

response will probably be sent as an operation that is separate from the request.

Web services and consumers of Web services are typically businesses, making Web services predominantly business-to-business (B-to-B) transactions.

An enterprise can be the provider of Web services and also the consumer of other Web services.

For example, a wholesale distributor of spare parts could be in the consumer role when it uses a Web service to check on the availability of gear boxes and in the provider role when it supplies prospective customers with different vendors' prices for gear boxes.

Clients access a Web service application through its interfaces and bindings, which are defined using XML standards,
such as a Web Services Definition Language (WSDL) file.

A Web service enables a *service-oriented architecture*, which is an architectural style that promotes software reusability by creating reusable services.

The most important reasons for the increased use of Web services —

Web services promote interoperability across different platforms, systems, and languages.(Web services are independent of a particular programming language)

They reduce operational costs by enabling organizations to extend and reuse their existing system functionality.

To enable this architecture , you need -----

A mechanism that enables clients to access a service and registry.

• A mechanism to enable different services to register their existence with a registry.

A way for clients to look up the registry of available services.

• A mechanism for services to expose well-defined interfaces and a way for clients to access those interfaces.

Web Service Standards

Web services are registered and announced using the following services and protocols. Many of these and other standards are being worked out by the **UDDI** project, consisting of a group of industry leaders.

Universal Description, Discovery, and Integration (UDDI) is a protocol for describing available Web services components. This standard allows businesses to register with an Internet directory that will help them advertise their services, so companies can find one another and conduct transactions over the Web. This registration and lookup task is done using **XML** and **HTTP(S)**-based mechanisms.

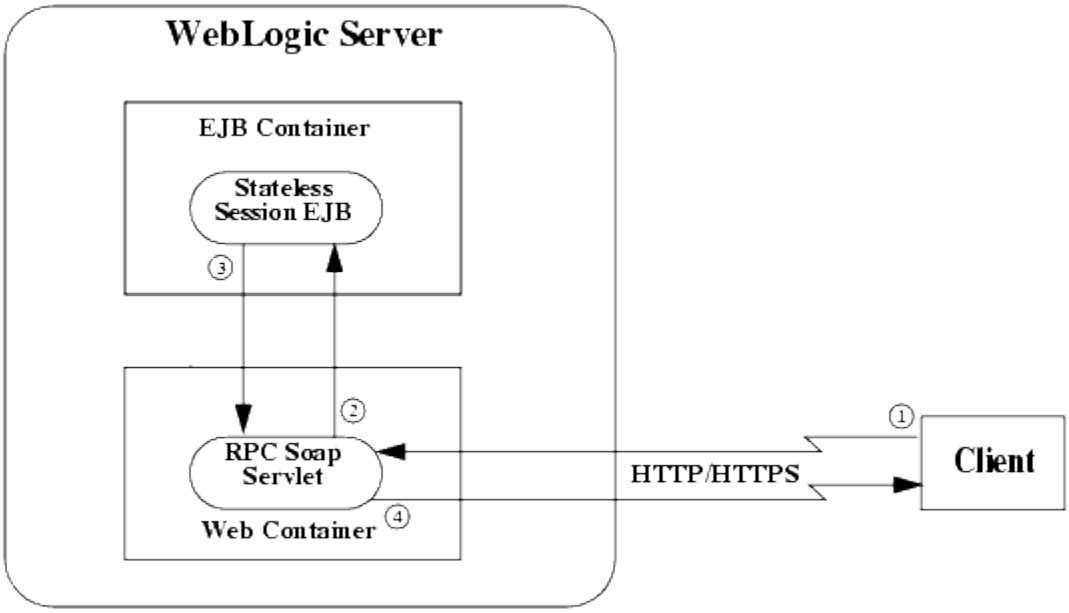
Simple Object Access Protocol (SOAP) is a protocol for initiating conversations with a **UDDI** Service. **SOAP** makes object access simple by allowing applications to invoke object methods or functions, residing on remote servers. A SOAP application creates a request block in XML, supplying

the data needed by the remote method as well as the location of the remote object itself.

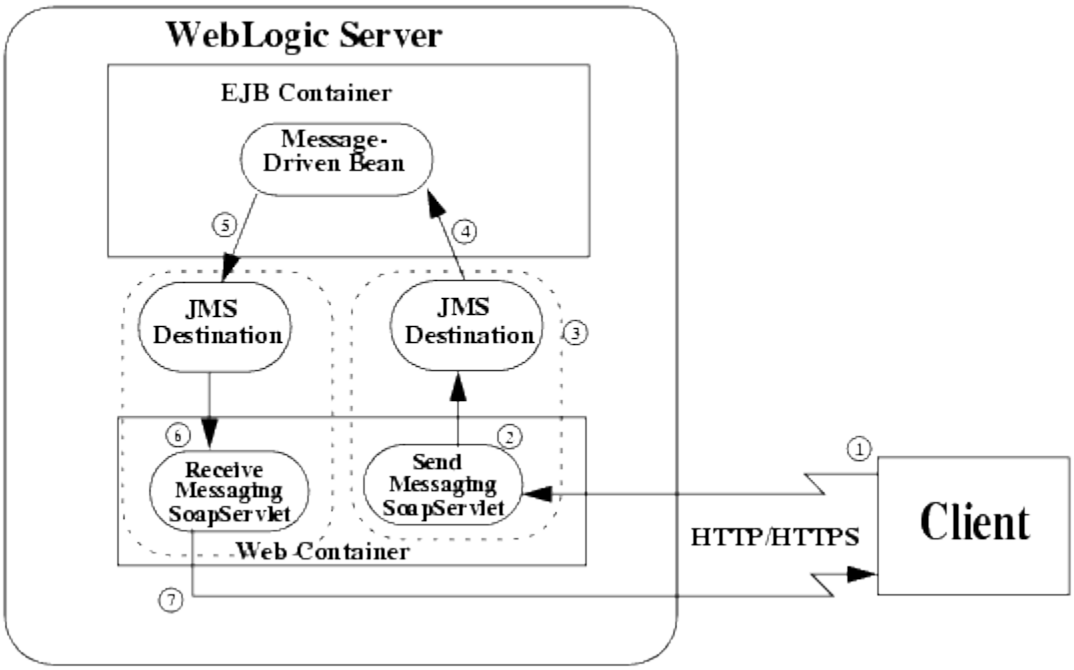
Web Service Description Language (WSDL), the proposed standard for how a Web service is described, is an XML-based service IDL (Interface Definitition Language) that defines the service interface and its implementation characteristics. WSDL is referenced by UDDI entries and describes the SOAP messages that define a particular Web service.

ebXML (e-business XML) defines core components, business processes, registry and repository, messaging services, trading partner agreements, and security.

Implementing Web Services – RPC style implementation.



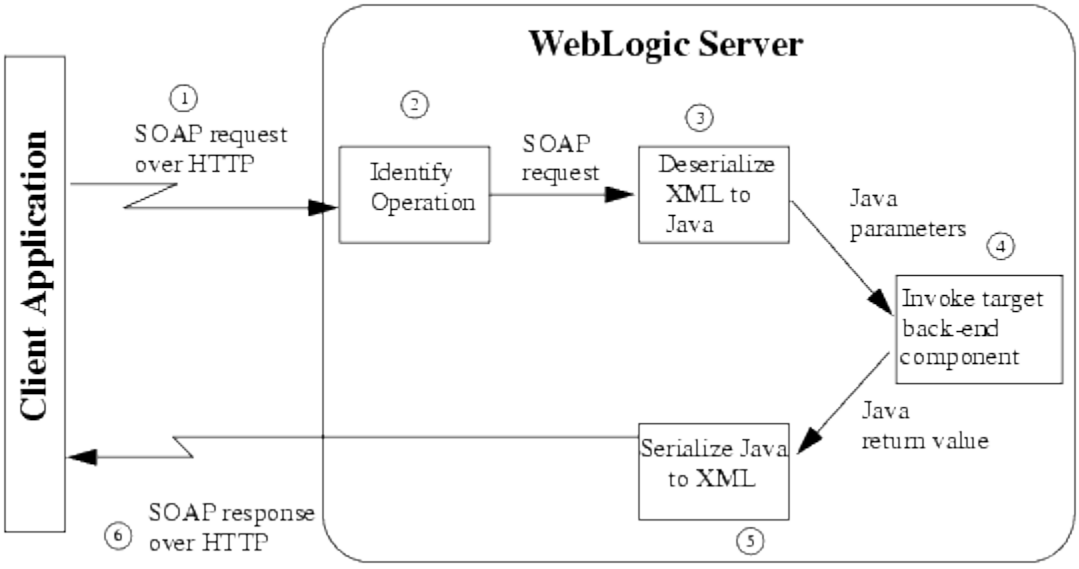
Non- RPC style – loosely coupled architecture.



Here comes a brief step-by-step on how a **Web service** is implemented.

- A service provider creates a **Web service**
- The service provider uses **WSDL** to describe the service to a **UDDI registry**
- The service provider registers the service in a **UDDI registry and/or ebXML registry/repository**.
- Another service or consumer locates and requests the registered service by querying **UDDI and/or ebXML registries**.
- The requesting service or user writes an application to bind the registered service using **SOAP** in the case of **UDDI** and/or **ebXML**
- Data and messages are exchanged as **XML** over **HTTP**

A typical web-service scenario

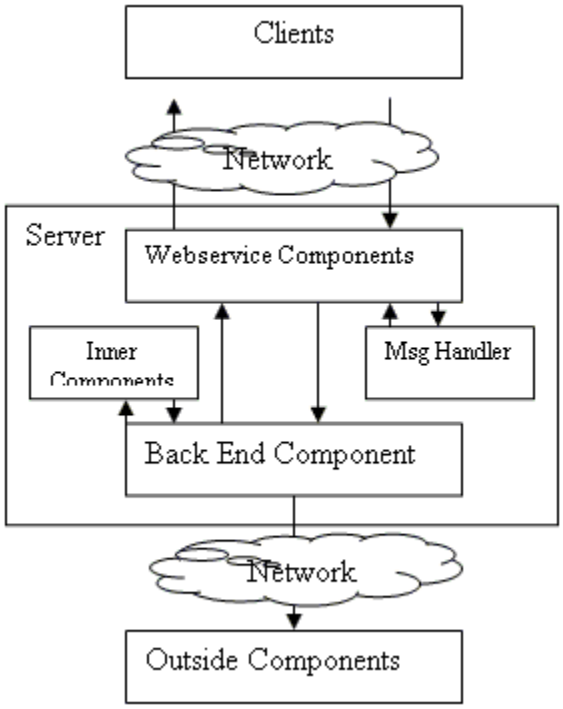


1. The client application sends a SOAP message request to application server over the network.
2. Based on the URI in the request, the server identifies the Web service being invoked.
3. The Web service reads the SOAP message request and identifies the operation that it needs to run.
4. The operation corresponds to a method of a backend component, to be invoked in a later step.
5. A conversion form XML to Java happens for the request parameters from the SOAP message at Web service layer for the invoked operation. A deserializer class is used for this purpose. The deserializer class is either one provided by the application server for built-in data types or a user-created one for non-built-in data types.
6. Appropriate back-end component's method with required Java parameters is invoked.
7. Upon completion back-end component sends back the response to the Web service which converts it back to XML

from Java with appropriate serializer class, and packages it into a SOAP message response.

8. The Web service sends the SOAP message response back to the client application that invoked the Web service.

Layers involved in the communication



Back-end Components

These are objects that are responsible for the execution of business methods.

Stateless Session EJB: Web service operations implemented with this approach are interface driven, which means that the business

methods of the underlying stateless session EJB determine how the Web service operation works. A stateless session EJB back-end has following characteristics:

- The behavior of the Web service can be expressed as an interface.
- The Web service is process-oriented rather than data-oriented.
- The Web service can benefit from EJB facilities, such as persistence, security, transactions, and concurrency.

Java Class: Web service operations implemented with Java classes are similar to those implemented with an EJB method. Creating a Java class, however, is often simpler and faster than creating an EJB. A Java class as a back-end component avoids the overheads of EJB facilities such as persistence, security, transactions, and concurrency.

There are few limitations and restrictions for java classes to be exposed as Web services:

- Should provide thread-safe operations
- Should not start multiple threads within the class.
- Should provide default no-argument constructor.
- Should provide public methods

A JMS message consumer or producer. Web service operations could be written for clients required to send/receive the data to/from a JMS destination/queue.

Important : a single Web service operation cannot both send and receive data; one needs to create two Web service operations for a client application to be able to both send and receive data.

The sending Web service operation is related to the receiving one because the original message-driven bean that processed the message on the server puts the response on the JMS destination corresponding to the receiving Web service operation.

Case Study : Adventure Builder

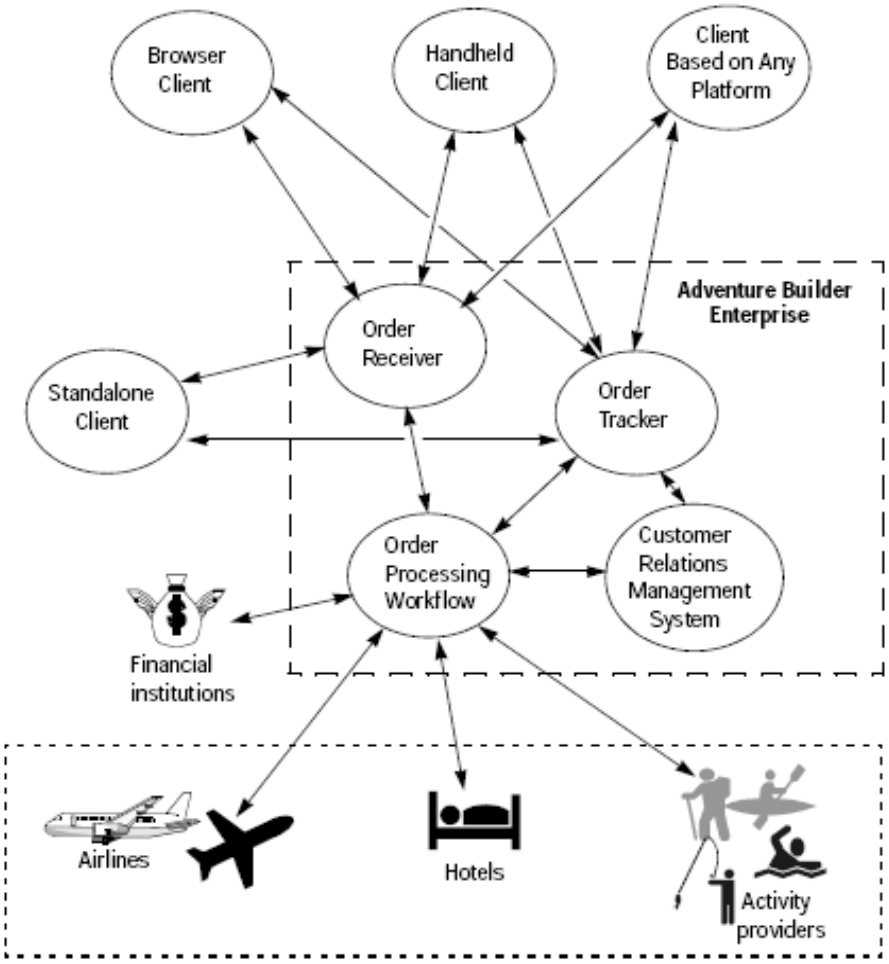


Figure 1.1 Adventure Builder Enterprise Modules

• Interact with clients who want to book adventures. There might be toll-free telephone numbers available to clients.
• Receive orders from clients and process the orders. Orders pass through a
workflow, such as a workflow engine, during their various stages for completion.
• Keep track of client preferences, plus update clients regarding the status of an
order. Customer relations management (CRM) is one such means for tracking
clients.
• Verify and obtain approval for client payment.
• Interact with business partners—airlines, hotels, adventure or activity providers—
so that the enterprise can provide a complete adventure package to the client

The Role of XML and the Java Platform and J2EE

Web services depend on the ability of parties to communicate with each other even if they are using different information systems and different data formats.

XML, a markup language that makes data portable, is a key technology in addressing this need.

Web services also depend on the ability of enterprises using different computing platforms to communicate with each other. This requirement makes the Java platform, which makes code portable, the natural choice for developing Web services.

In addition to data portability and code portability, Web services need to be scalable, secure, and efficient, especially as they grow.

The Java 2 Platform, Enterprise Edition (J2EE), is specifically designed to fill these needs viz . security, distributed transaction management, and connection pool management, all of which are essential for industrial strength Web services. And because components are reusable, development time is substantially reduced.

Overview of the Java APIs for XML

The Java APIs for XML let you write your Web applications entirely in the Java programming language.

They fall into two broad categories: those that deal directly with processing XML documents and those that deal with procedures.

Document-oriented

Java API for XML Processing (JAXP)—processes XML documents using various parsers

Procedure-oriented

Java API for XML Messaging (JAXM)—sends SOAP messages over the Internet in a standard way

Java API for XML Registries (JAXR)—provides a standard way to access business registries and share information

Java API for XML-based RPC (JAX-RPC)—sends SOAP method calls to remote parties over the Internet and receives the results

The most important feature of the Java APIs for XML is that

They all support industry standards

Ensure interoperability.

Various network interoperability standards groups, such as the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS), have been defining standard ways of doing things so that businesses who follow these standards can make their data and applications work together.

JAXP
The Java API for XML Processing (JAXP) makes it easy to process XML data using applications written in the Java programming language.

JAXP specifies the parser standards SAX (Simple API for XML Parsing) and DOM (Document Object Model) .

Using SAX : you can choose to parse your data as a stream of events .

Using DOM you can build an object representation of it.

The latest versions of JAXP also supports the XSLT (XML Stylesheet Language Transformations) standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML.

JAXP also provides namespace support, allowing you to work with XML Schemas that might otherwise have naming conflicts.

JAX-RPC

The Java API for XML-based RPC (JAX-RPC) makes it possible to write an application in the Java programming language that uses SOAP to make a remote procedure call (RPC).

JAX-RPC can also be used to send request-response messages and, in some cases, one-way messages.

JAX-RPC makes it possible for an application to define its own XML schema and to use that schema to send XML documents and XML fragments.

The result of this combination of JAX-RPC and XML Schema is a powerful computing tool.

JAX-RPC makes using a Web service easier, and it also makes developing a Web service easier, on the J2EE platform.

An RPC-based Web service is basically a collection of procedures that can be called by a remote client over the Internet.

The service itself is a server application deployed on a server-side container that implements the procedures that are available for clients to call.

For example, a typical RPC-based Web service is a stock quote service that takes a SOAP request for the price of a specified stock and returns the price via SOAP.

A Web service needs to make itself available to potential clients, which it can do, for instance, by describing itself using the Web Services Description Language (WSDL). A consumer (Web client) can then do a lookup of the WSDL document to access the service.

Interoperability across clients and servers that have been described using WSDL is key to JAX-RPC.

A consumer using the Java programming language can use JAX-RPC to send its request to a service that may or may not have been defined and deployed on a Java platform.

Also a client using another programming language can send its request to a service that has been defined and deployed on a Java platform. This interoperability is a primary strength of JAX-RPC.

Although JAX-RPC implements a remote procedure call as a request-response SOAP message, a user of JAX-RPC is shielded from this level of detail. So, underneath the covers, JAX-RPC is based on SOAP messaging.

JAX-RPC is the main client and server Web services API, largely because of its simplicity.
The JAX-RPC API is simple to use and requires no set up.

Also, JAX-RPC focuses on point-to-point SOAP messaging.

JAXM

The Java API for XML Messaging (JAXM) provides a standard way to send XML documents over the Internet from the Java platform.

Typically, a business uses a messaging provider service, which does the behind-the-scenes work required to transport and route messages.

When a messaging provider is used, all JAXM messages go through it, so when a business sends a message, the message first goes to the sender's messaging provider, then to the recipient's

messaging provider, and finally to the intended recipient. It is also possible to route a message to go to intermediate recipients before it goes to the ultimate destination.

Because messages go through it, a messaging provider can take care of housekeeping details like assigning message identifiers, storing messages, and keeping track of whether a message has been delivered before.

A messaging provider can also try resending a message that did not reach its destination on the first attempt at delivery.

The beauty of a messaging provider is that the client using JAXM technology (JAXM client) is totally unaware of what the provider is doing in the background.

The JAXM client simply makes Java method calls, and the messaging provider in conjunction with the messaging infrastructure makes everything happen behind the scenes.

JAXR

The Java API for XML Registries (JAXR) provides a convenient way to access standard business registries over the Internet.

Business registries are often described as electronic yellow pages because they contain listings of businesses and the products or services the businesses offer.

JAXR gives developers writing applications in the Java programming language a uniform way to use business registries that are based on open standards (such as ebXML) or UDDI

Businesses can register themselves with a registry or discover other businesses with which they might want to do business.

They can submit material to be shared and search for material that others have submitted.

Standards groups have developed DTDs for particular kinds of XML documents, and two businesses might, for example, agree to use the DTD for their industry's standard purchase order form.

Because the DTD is stored in a standard business registry, both parties can use JAXR to access it.