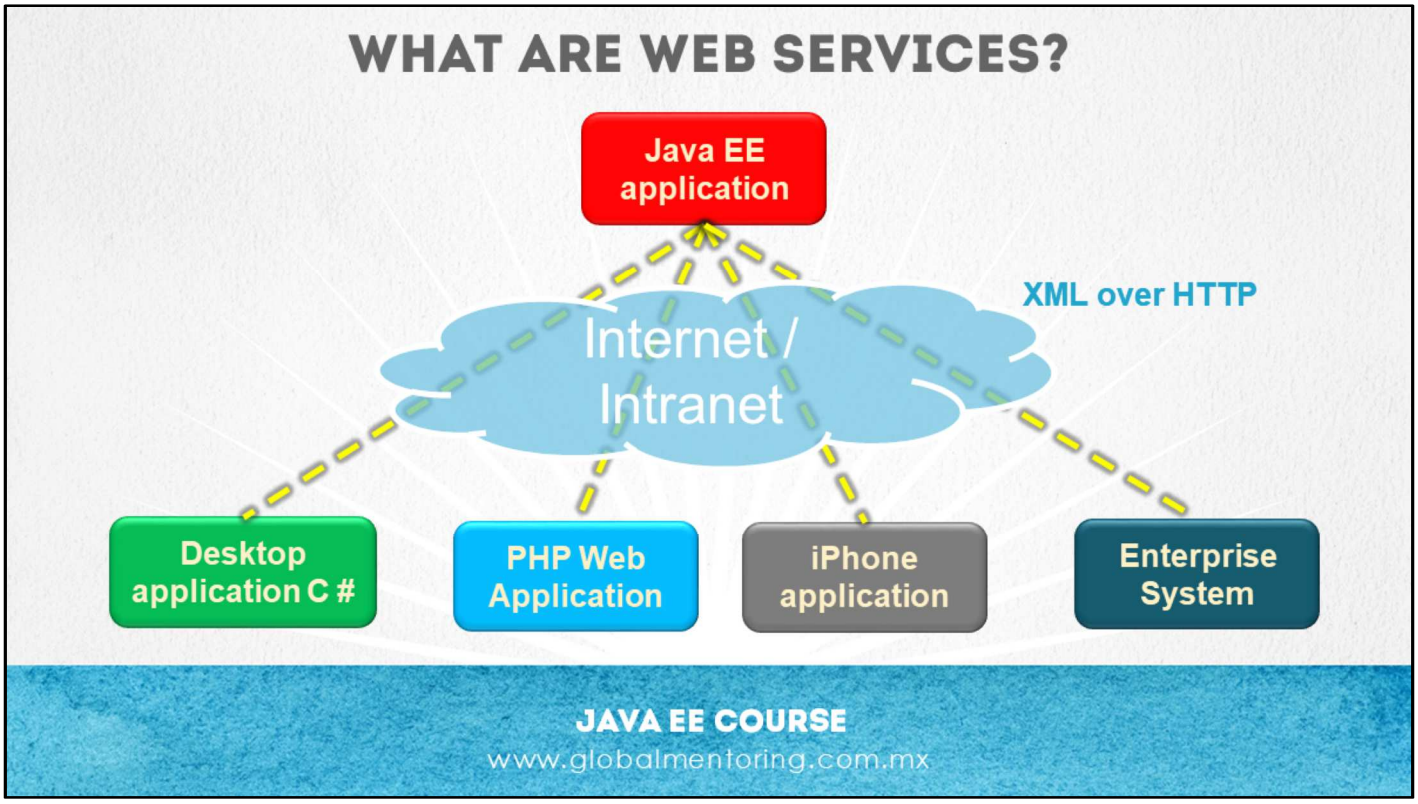
Hello, Ubaldo Acosta greets you again. I hope you're ready to start with this lesson ..

We are going to study the topic of Web Services using JAX-WS technology of Java EE.

Are you ready? Let's go!

© All rights reserved Global Mentoring



The subject of Web Services has been used for several years, over the years they have learned techniques and new ways to establish a more efficient communication between systems created in different platforms or technologies.

The Web Services are a technology oriented to the intercommunication of systems. Some of its characteristics are:

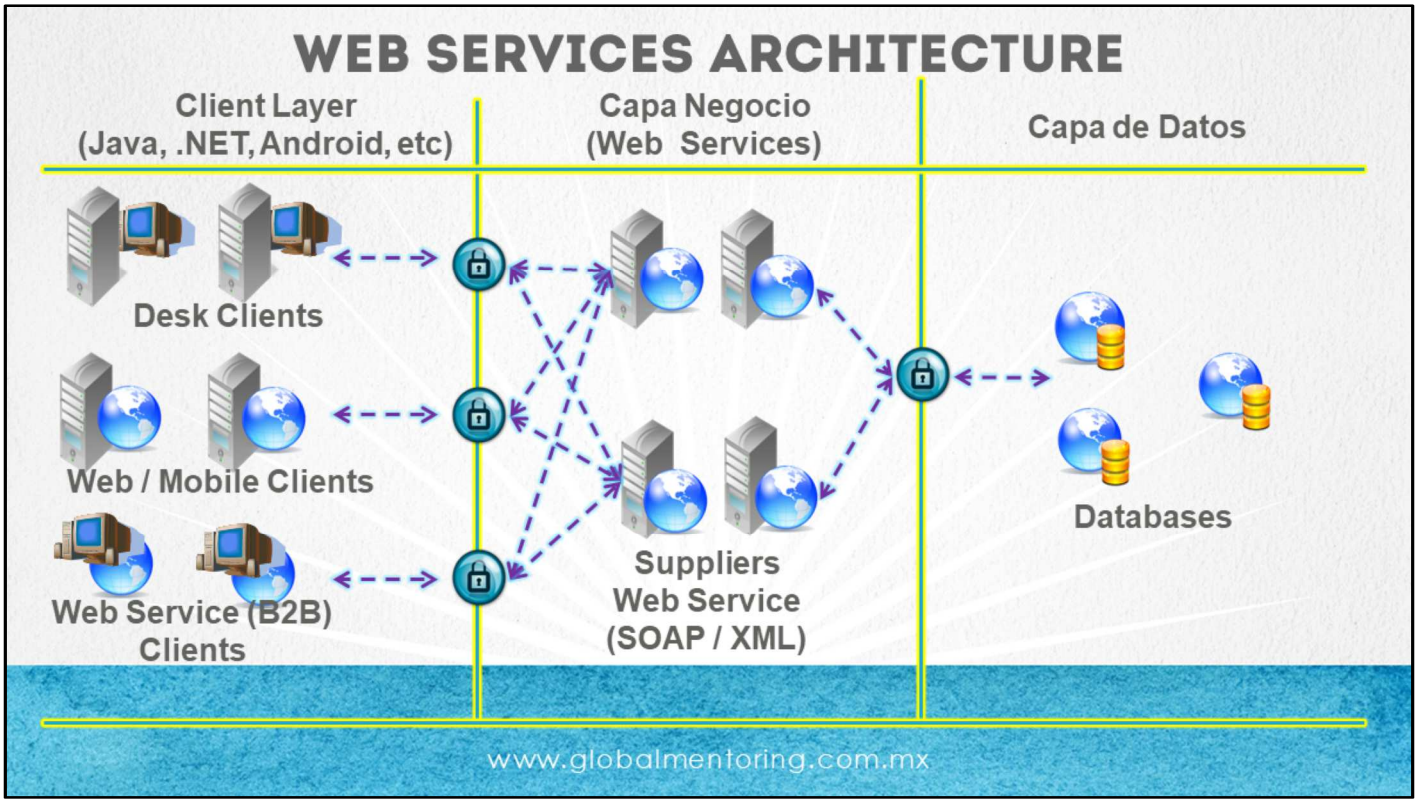
- **Interoperability and Portability**: Systems that use Web Services allow the exchange of information between different platforms and programming languages using the Web (Intranet / Internet) as the basis for communication. One of the ways is to use the protocol in SOAP allowing communication via XML, achieving platform independence. All this supported by the HTTP protocol.

- **Reusability**: It allows to reuse much of the business logic coming from legacy systems or from our current business systems.

- **Availability**: The objective of the web services is that they are available at any time and in any place, for any system and / or person that needs to use them. In addition, they do not require human intervention even in very complex transactions.

As we can see in the figure, in many occasions we need to intercommunicate information systems, which may have been developed in the same technology or not. Java EE has two main APIs for the development of Web Services: Java API for XML Web Services (JAX-WS) and Java API for RESTful Web Services (JAX-RS), which we will study.

© All rights reserved Global Mentoring



In a Java Enterprise application, Web Services can be viewed as an alternative to expose the EJBs without using RMI or using Java.

The EJBs can be exposed as Web Services, with this we reuse your business logic. The above allows the Client to be written in any language such as Objective-C, .Net, PHP, Android, etc., without any dependency on the Java language.

If you are new to Web Services, RESTful Web Services are easier to learn. We will study this type of Web Services later.

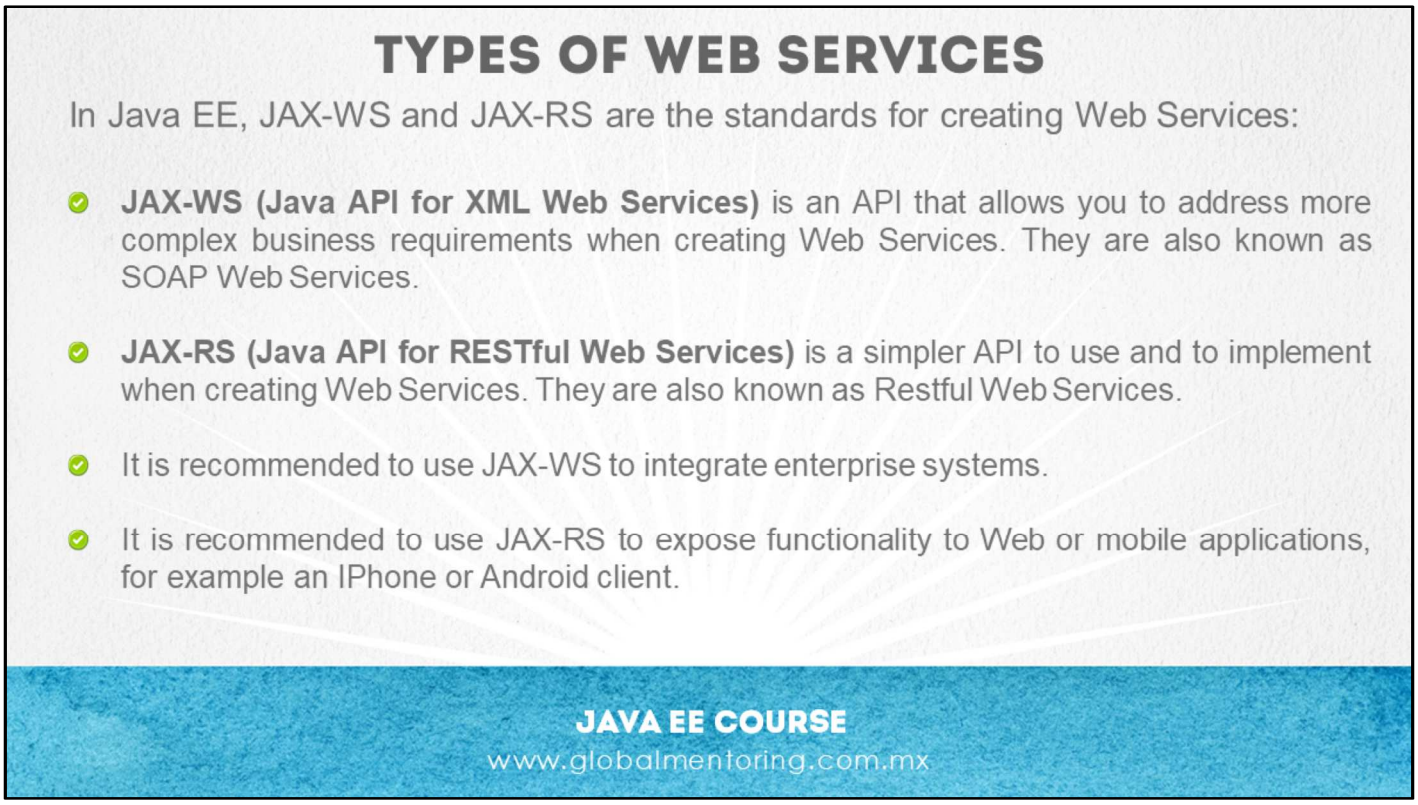
In this first part we will study the SOAP Web Services, which require a basic knowledge of topics such as XML, XSD (XML schemas) and JAXB. If you do not have this knowledge, we will briefly show what these technologies consist of, how we will apply them, and a reference for further study.

It should be noted that because Java EE has greatly simplified the process of creating Web Services, many of these technologies will be only behind the scenes, and each time we work in a new Web Service will gradually acquire more experience in each of them. these topics and / or technologies.

The Web Services are a fundamental part for the development of SOA (Service-Oriented Architecture) architectures with the aim of integrating applications created on the same or different platforms. Also, this is the basis of microservices architectures, so it is very important that we learn the concepts since they will be reused in other types of business architectures such as microservices.

In the following exercises we will see how to expose Stateless SessionBeans EJBs as Web Services.

© All rights reserved Global Mentoring



In Java EE, JAX-WS and JAX-RS are the two standards for creating Web Services:

- **JAX-WS**: Java API for XML Web Services allows you to define Web Services also known as SOAP Web Services. JAX-WS replaces the use of JAX-RPC. The creation of Web Services using JAX-RPC (Remote Procedure Call) was the standard in version 1.4, however in this course we will not study this topic. For more information on creating this type of Web Services: http://docs.oracle.com/javaee/1.4/tutorial/doc/JAXRPC.html

- **JAX-RS**: Java API for RESTful Web Services allows you to create Web Services according to the terminology Representational State Transfer (REST) under the HTTP protocol. JAX-RS is not the standard, but has gained great popularity in recent years.

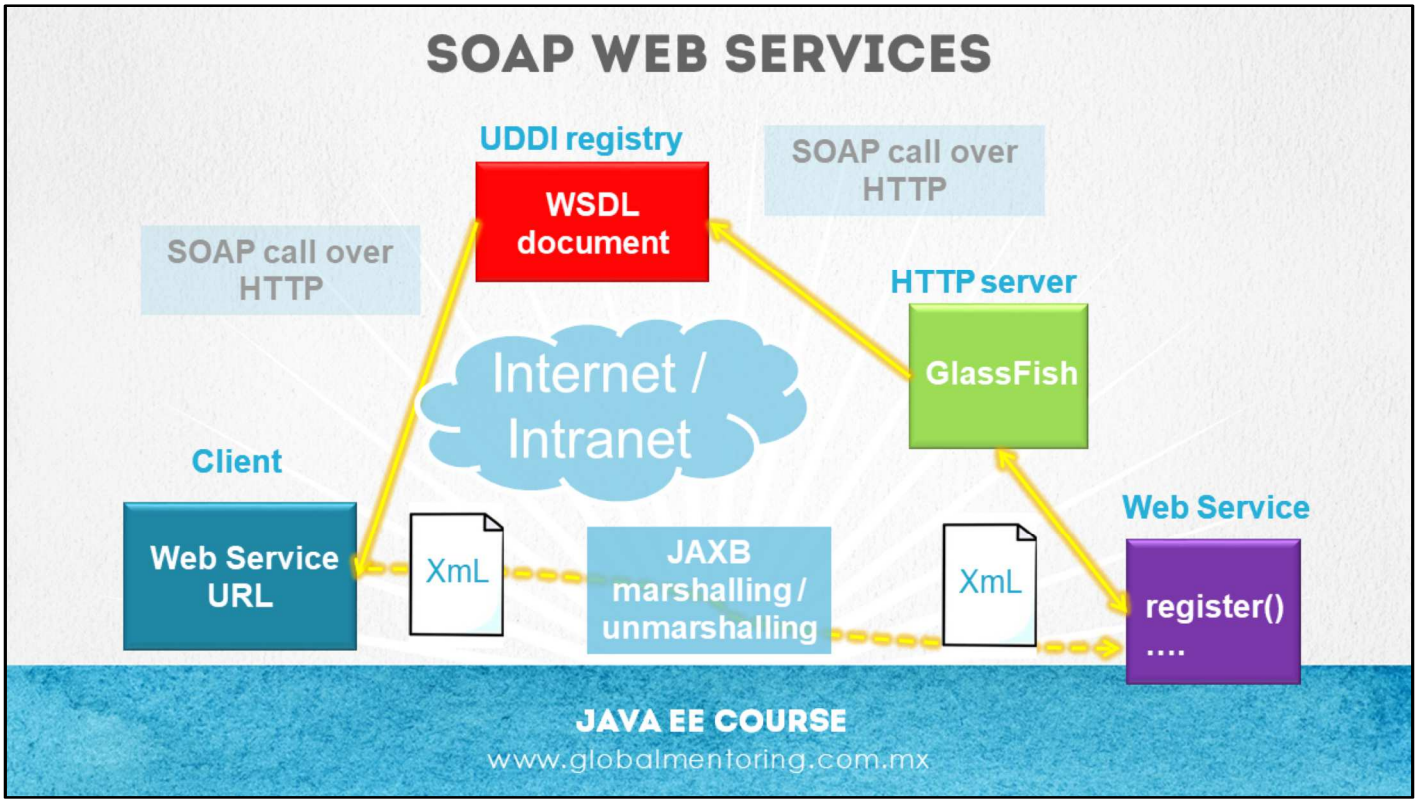
When using Java EE, we must decide if we use the JAX-WS or JAX-RS API. To make this decision we can take into account the following points:

- JAX-WS: This API allows addressing more complex requirements and takes more years in the market. It provides support for the protocols that are standard in the software industry when creating Web Services. These standards provide a very robust way to add security, transactionality, interoperability, between various features between the client and the server when using Web Services.

- JAX-RS: This API allows you to create Web Services in a simpler way, only that it is restricted by the REST style, which uses the HTTP protocol and its supported methods and state codes to establish the basic operations of a Web Service (GET, POST, PUT, DELETE, etc).

These types of Web Services are the ones we will study.

© All rights reserved Global Mentoring



In simple terms, a Web Service is a standard platform that provides interoperability between different applications. For most programmers this means sending and receiving XML messages which are transmitted via HTTP / HTTPS.

Both the XML language and the HTTP protocol are a standard and are widely accepted for sending and receiving information, so this information can be processed by multiple clients with different technologies.

As shown in the figure, a Web Services is published in a type of directory known as UDDI, which means Universal Description, Discovery and Integration. UDDI is a standard and aims to publish and allow access to Web Services through SOAP messages.

SOAP (Simple Object Access Protocol) is a standard protocol which defines the way in which data of XML type is exchanged. This protocol will be analyzed in the following sheet.

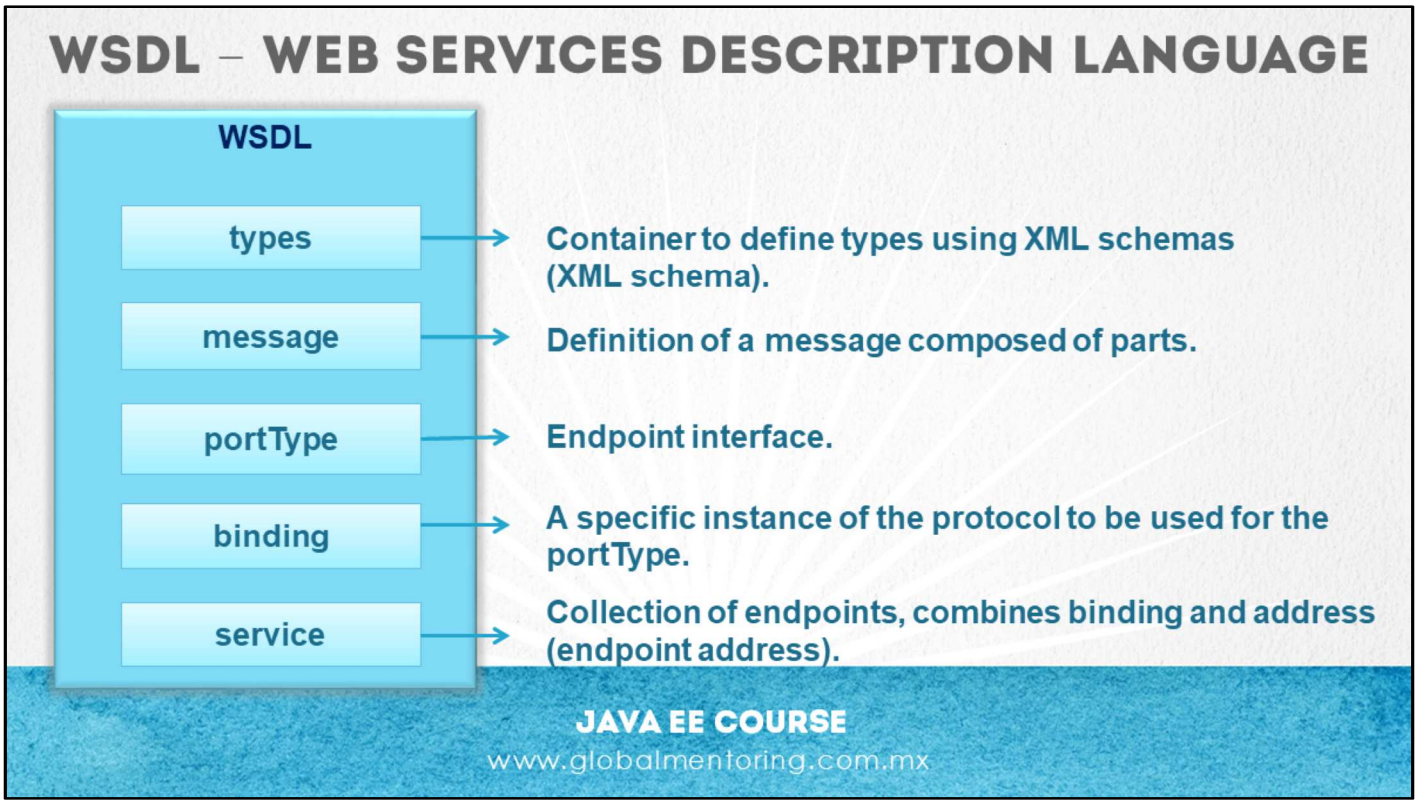
WSDL (Web Services Description Language) allows us to describe Web Services. WSDL describes the public interface of our Web Services and XML is used for its description. We will analyze this descriptor later.

Once we know the location of the Web Services through the WSDL (URI) of the same, we can send the SOAP request and thus execute the exposed method of the Web Service. In this procedure the request and response XML is usually validated using XSD (schema).

Once the XML message has arrived at the Java server, it must be converted into Java objects. For this it is common to use JAXB technology, which will allow us to convert Java objects into XML documents (marshaling) and vice versa (unmarshalling). This technology will be discussed later.

As we can see there are several related technologies to understand in detail the creation of SOAP Web Services, however the good news is that the JAX-WS standard will allow us to hide and automate many of the steps necessary to create a Web Service, as well as the creation of the Web Services Client.

© All rights reserved Global Mentoring



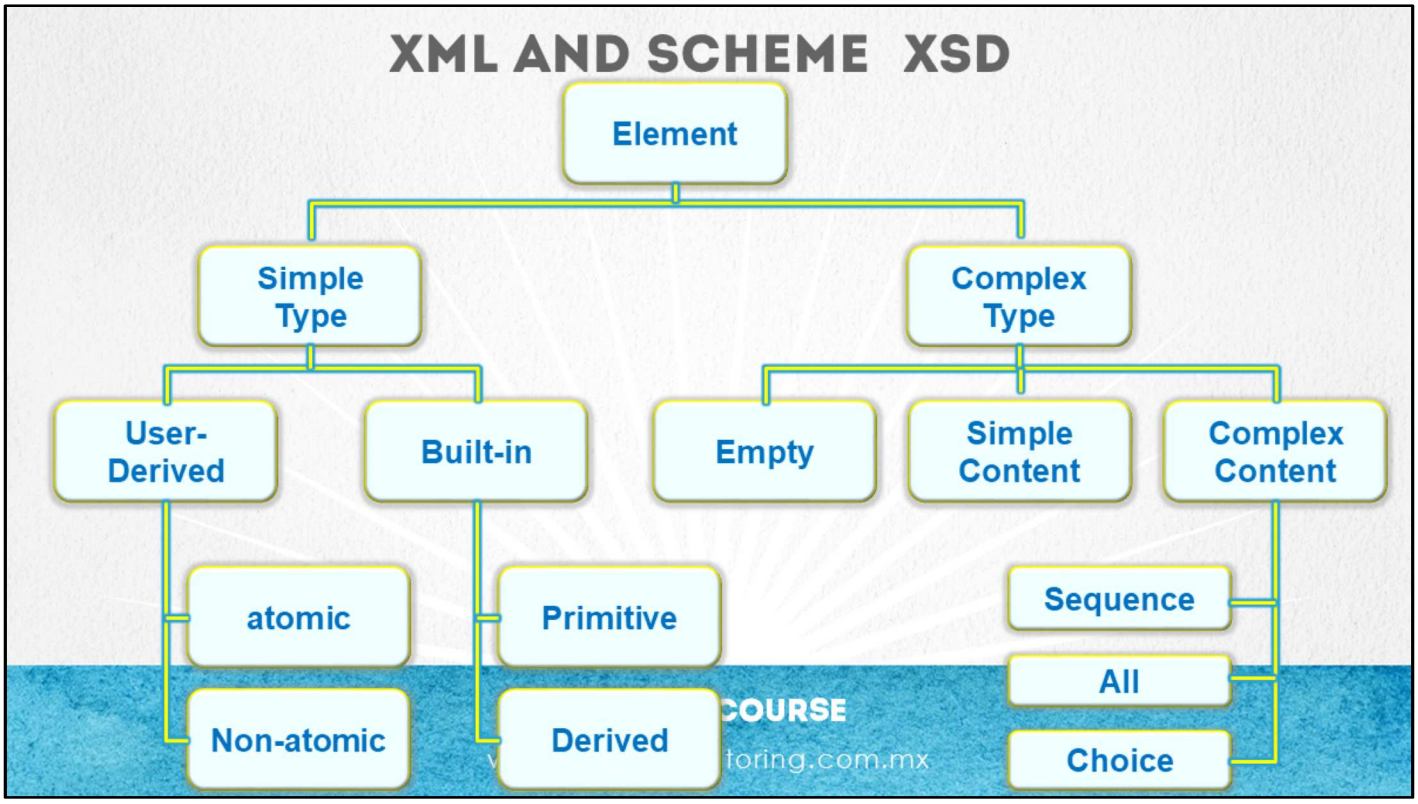
WSDL (Web Services Description Language) is a language based on XML, which provides the description of a Web service. In fact it provides a fairly detailed description of the Web Service and with which it is possible to generate both the Client's code and the associated Web Service. The generated code will automatically handle the conversion of Java code to XML (marshaling) and vice versa (unmarshaling).

In the figure we can see the general structure of a WSDL document.

- **Types**: In the Types section, the types to be used in the XML message are defined. These types are defined using XML schemas (xsd).
- **Message**: This section defines the types of messages that the Web Service supports. Messages can be composed of one or more parts, which can be input (input) or output (output).
- **PortType**: This section defines a group of operations (web service interface). Each operation must have a unique name and contains a combination of input (input) and output (output) elements, which refer to the Message elements.
- **Binding**: This section relates the portType (interface) with the protocol to be used (eg SOAP).
- **Service**: This section relates the Binding section as services and defines an endpoint.

Although this information seems complicated at first, as the first Web Services are carried out, we will become familiar with these terms. However, if we want to handle more advanced topics it will be necessary to be familiar with this terminology and concepts.

© All rights reserved Global Mentoring



There are several ways to validate an XML document. This is basic when transmitting a message through a Web Services. The validation will allow us to automate the messages between the Client and the Server without the need for human intervention.

To validate an XML document it is possible to do it by means of two techniques. The first known as DTD (Document Type Definition) and the second as XSD (Schema XML). The validation of files by DTD is less and less used, because the language used is not XML, and therefore has several limitations when validating XML documents with more advanced restrictions.

On the other hand, validation by XML Schema is increasingly used to validate the structure and restrictions of an XML document. Since it allows to add much more precision when validating elements and complex attributes of an XML document, and because it is created with the same XML language, it favors enough to create robust and flexible validations.

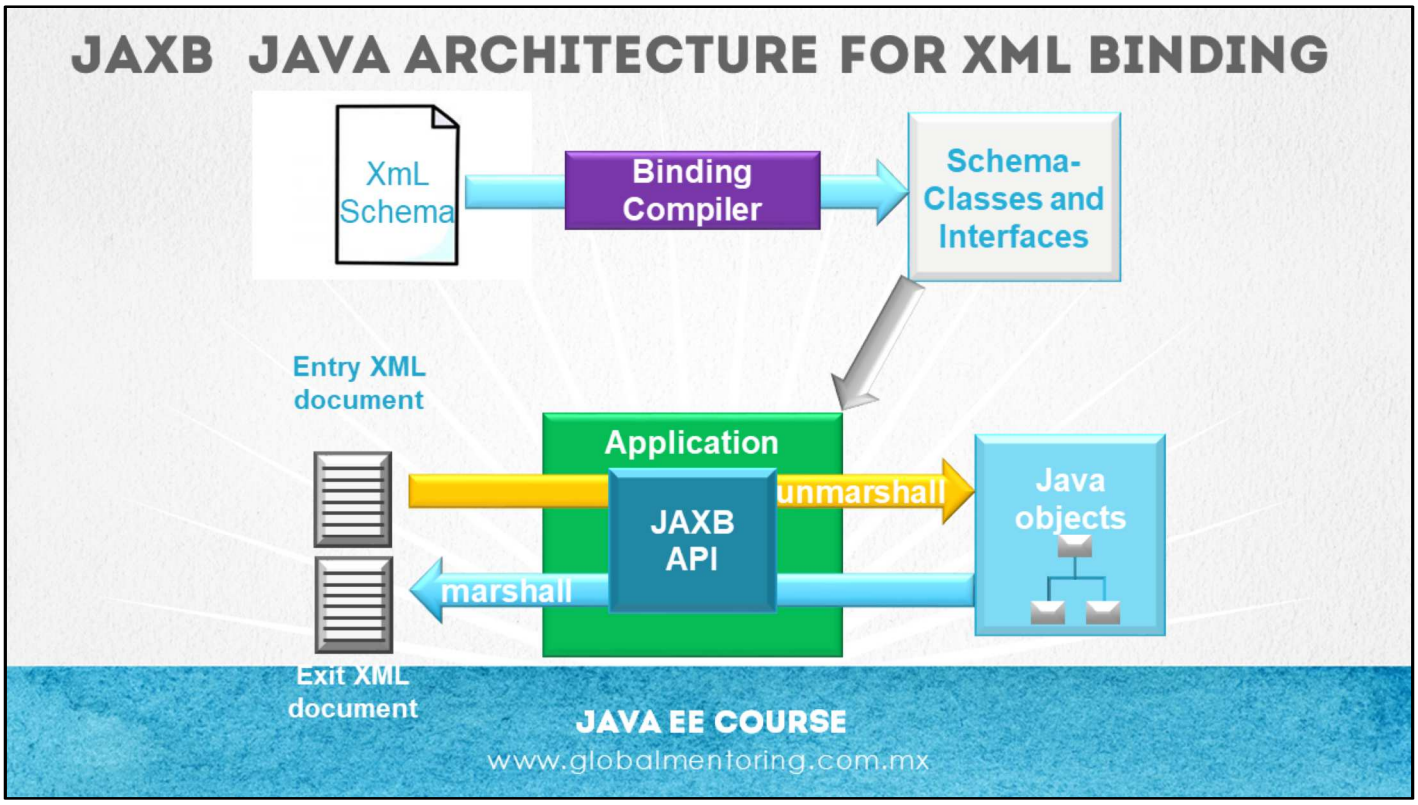
In the figure we can observe the considerations to take into account for the validation of an XML document, and you can see that they are as detailed as needed, for example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="content" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

An example of an XML validated by the previous scheme can be the following example:

```
<note>
  <to>John</para>
  <from>Katty</de>
  <title>Reminder</titulo>
  <content>See you next Monday!</content>
</note>
```

© All rights reserved Global Mentoring

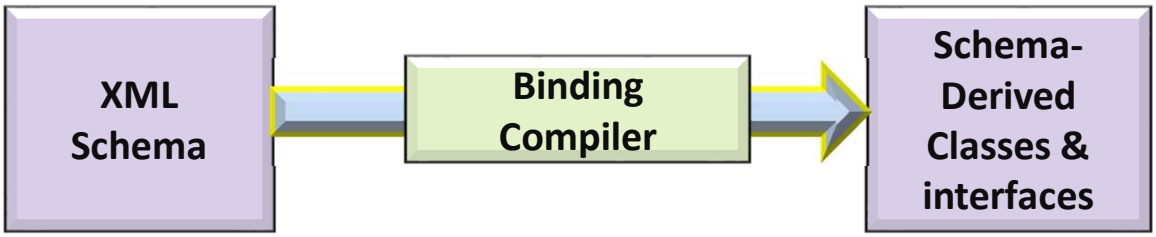


When working with Web Services, it is necessary at some point to convert the XML messages into Java objects and vice versa. There are several APIs to perform this task, however the standard in the Java EE version is the JAXB API (Java Architecture for XML Binding).

As we can see in the figure, the JAXB technology provides two main characteristics. The ability to convert a Java object into an XML document (marshal) and vice versa (unmarshalling).

JAXB allows in a very simple way to access and process XML documents without needing to know in detail XML or other processing APIs.

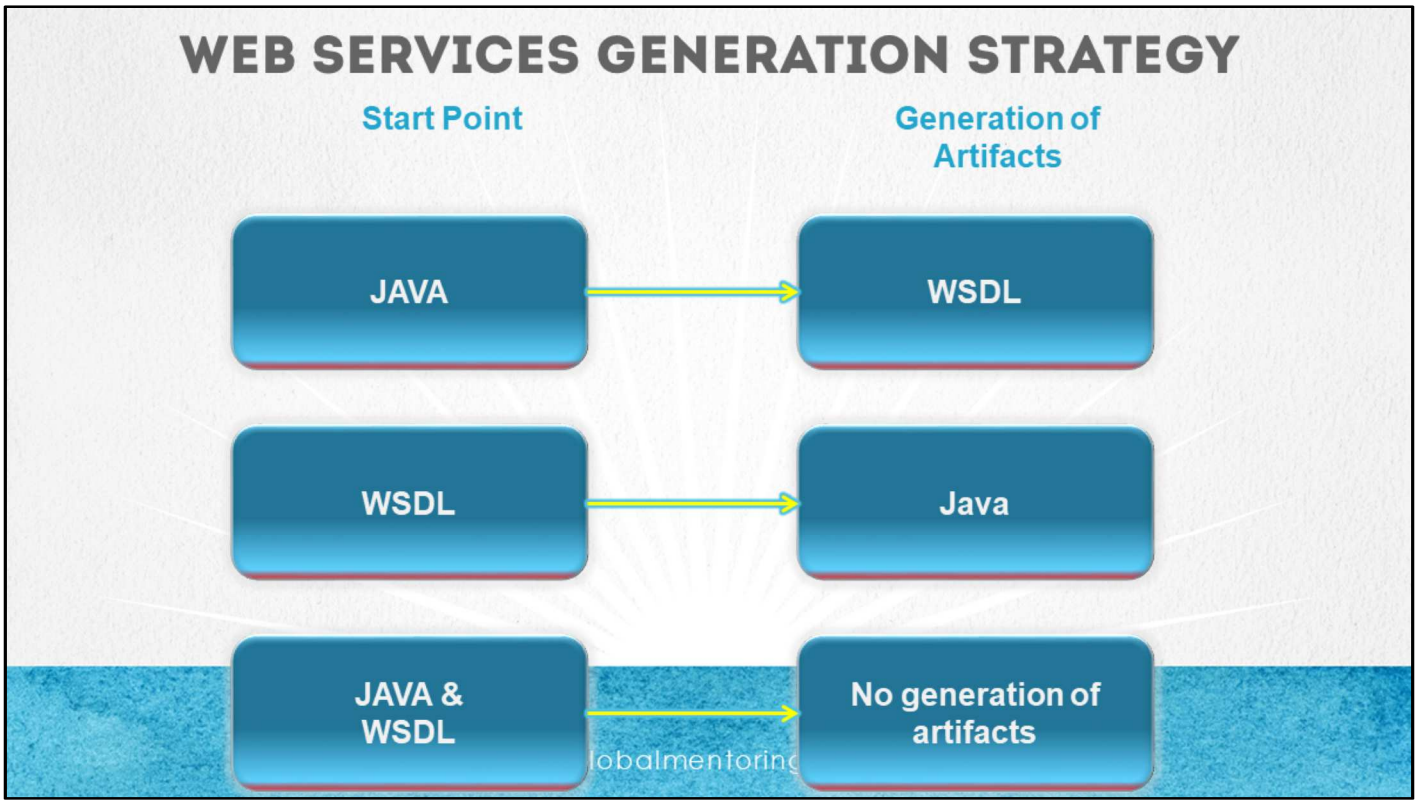
JAXB associates the schema (XSD) of the XML document to be processed (binding), and subsequently generates the Java classes that represent the XML document (unmarshalling).



After the code is generated, the data of the associated XML document can be accessed and displayed, simply by using the generated Java classes and objects. There is no need to use an XML parser or even, there is no need to know the original XML document.

The inverse process (marshalling) is also possible, in which the XSD document that represents the structure of the Java objects and their relation is generated from Java code.

© All rights reserved Global Mentoring



When we create a new Web Service, there are two artifacts that must be generated: The WSDL document and the Java classes that implement the Web Service. In addition, the XSD document (XML schema) associated with the XML message to be exchanged by the Web Service must be generated.

With the WSDL document and the XSD schema, a client of a Web Services can be self-generated with certain tools, by the wsimport Java command. This command allows you to generate the necessary Java code to invoke a Web Service from the URL of the WSDL. Similar tools exist in other languages to create the clients of the SOAP Web Services in a very similar way.

As we can see in the figure and focusing on the two main artifacts, WSDL and Java code, we can take different paths. The strategy to be selected can be for each Web Service independently, that is, it is not a single decision.

When we have ready the Java code, we delegate the generation of the Web Service to the JAX-WS API, which will automatically generate both the WSDL document and the XSD file associated with the XML message to be exchanged, all this based on the Java code and the existing relationship in the Web Service method to expose.

JAX-WS can expose methods of Java classes POJO's or EJB's. Selecting one or the other depends on whether we want to take advantage of the benefits of having an EJB, or use a POJO to expose Java methods as a Web Service.

To expose a Web Service from a Java class, we must only annotate the class with @WebService and add the annotation @WebMethod to the method to be exposed as a Web Service. This will automatically generate the WSDL and the associated XSD schema.

This is the strategy that we will use in our exercises, since we will expose some Java methods of our EJBs as Web Services.

© All rights reserved Global Mentoring

© All rights reserved Global Mentoring