International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 7 No. 4 Aug. 2014, pp. 1574-1581

http://www.ijias.issr-journals.org/

# Web services at the service of e-learning platforms

***Badr HSSINA, Abd elkrim MERBOUHA, and Belaid BOUIKHALENE***

Sultan Moulay Slimane University, Faculty of sciences and technology Beni Mellal, Morocco



**ABSTRACT:** In this article, we defined the steps to create a web service based on an example that relates to the field of e-Learning. In addition, it cited the components and the basic functionality of a web service. In determining the concepts on which is based the web services namely SOAP and WSDL (a description in XML Schema web service) and UDDI, these three elements constitute the life cycle of use of a web service architecture in a distributed client / server.

**KEYWORDS:** Web services, e-learning, XML, SOAP, WSDL, UDDI, JAXWS.

## 1 INTRODUCTION

Web services are modular web-based applications that perform specific tasks and follow a specific format. They allow your applications to use remote features, thus simplifying the exchange of data [1]. Web services allow applications to communicate across the network, regardless of their platform execution and their implementation language. They are part of the continuity of initiatives such as CORBA (Common Object Request Broker Architecture, OMG) [2] However, providing a simple answer, based on recognized and accepted standards and technologies now of everyone.

## 2 WEB SERVICES ARCHITECTURE

The concept of Web Services is currently organized around three concepts [11] as follows:

- SOAP (*Simple Object Access Protocol*)

It is an inter-exchange application independent of any platform protocol, based on XML. A call SOAP [3] [6] service is an ASCII stream framed in XML tags [5] and carried in the HTTP protocol.

- WSDL (*Web Services Description Language*)

WSDL [3] [4] [7] gives the description XML Web Services by specifying the methods that can be invoked, their signatures and the access point (URL, port, etc. ...).

So WSDL is an XML file generated by the server and is provided to the customer on demand.

- UDDI (*Universal Description, Discovery and Integration*)

UDDI [8] normalizes a distributed directory of Web Services solution, allowing both the publication and exploration (search) Web Services [3].

UDDI behaves itself as a Web service whose methods are called via SOAP.

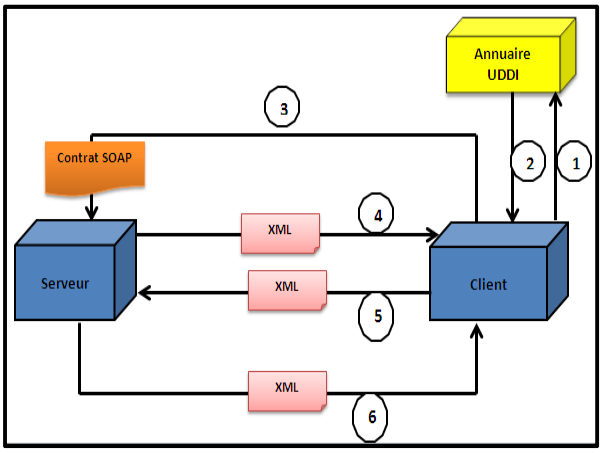



***Figure 1: Life Cycle of using a web service***

Description of Figure 1:

**1**: Looking for a Web Service
**2**: Found here is the server hosting the web service.
**3**: What is the format of calling the service you propose?
**4:** Behold my contract (WSDL).
**5**: I understand how to invoke your service and I will send you an XML document representing my request
**6**: I ran your query and I will return the result

## 3 WEB SERVICES WITH JAX-WS

JAX-WS is the new name of JAX-RPC (Java API for XML-Based RPC), which allows to easily develop web services. JAX-WS provides a set of annotations to map the Java-WSDL mapping. Simply annotate directly the Java classes that will represent the web service. Regarding the client, JAX-WS allows a proxy class to call a remote service and hide the complexity of the protocol. Thus, neither the client nor the server needs to generate or parse SOAP messages. JAX-WS is responsible for the low level processing.

In the example below, a Java class that uses JAX-WS annotations that will allow later to generate the WSDL document. The WSDL document is auto-generated by the application server at deployment time:

```
@WebService()
public class learningService{
@WebMethod
public Cours getCours(Long code) {
...
}
...
}
```

JAX-WS uses JAXB 2.0 API [14] for all that concerns the correspondence between XML and Java objects. A graph of objects comprises the following elements constituting the XML document. JAXB (Java Architecture for XML Binding) facilitates bidirectional mapping by providing a higher level than SAX or DOM [15] and abstraction based on annotations.

For example, if we want to get an XML representation of the Client class, simply annotate avec **@javax.xml.bind.annotation.XmlRootElement**. Other annotations allow you to specify that an attribute is an identifier with **@XmlID** or rename an attribute (e-mail instead of email, for example) with **@XmlAttribute**:

```
@XmlRootElement
public class Apprenant {
@XmlID
private String id;
private String nom;
private String prénom;
@XmlAttribute(name="e-mail")
private String email;
}
```

These annotations are then used to generate the following from the class, and vice versa XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<apprenant>
  <id>3456</id>
  <nom>Alaoui</nom>
  <prénom>Ali</prénom>
  <e-mail> alaoui.ali@gmail.fr</e-mail>
</ apprenant>
```

## 4 HOW TO DEVELOP A WEB SERVICE

In reality, the development of a web service is relatively simple although several technologies are implemented. Such is the power of Java EE [9]. For this, there are several phases of code generation that comes into play and implementing all the technical plumbing. The development and use of a web service consists of four phases:

- Development of the web service itself.
- Generating server-side artifacts (Skeletons).
- Generate client-side artifacts (stubs).
- Start the server by deploying the web service.
- Call the web service to the customer.

An artifact is composed of all the necessary documents to a web service. We can cite for example the WSDL or Java classes that form the XML SOAP message exchanges.

The following diagram shows the architecture for the development of web service with JAX-WS [13].

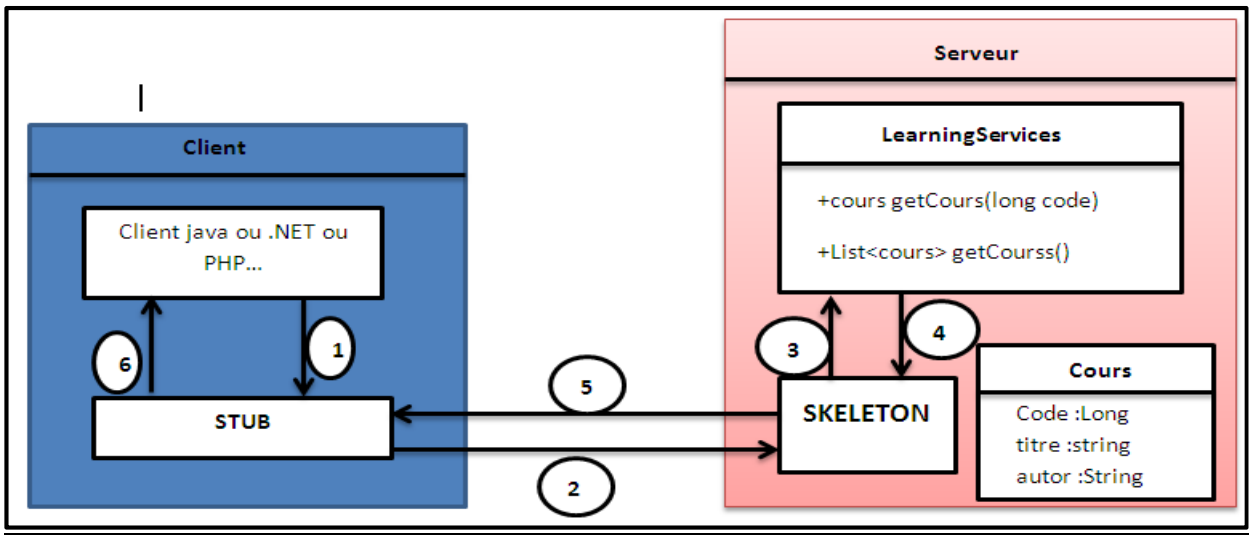



*Figure 2: architecture development web service with JAX-WS*

1: The client tells the stub to call the getCours () method.
2: Stub Skeleton connects and sends a SOAP request.
3: Skeleton uses the web service method.
4: The web service returns the result to the Skeleton.
5: Skeleton sends the result in a SOAP request to the Stub
6: The Stub provided the required result to the client.

### 4.1 DEVELOP A WEB SERVICE

-Tools to install:

- o JDK1.6
- o java Publisher: Eclipse.

-Create a java project using JDK1.6 as java compiler and runtime environment.

-Create learningWS.java class package ws:

```
package ws;
import java.util.Date;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;

@WebService(targetNamespace="http://bk/test",serviceName="learningService")
@SOAPBinding(style=SOAPBinding.Style.DOCUMENT,use=SOAPBinding.Use.LITERAL)
public class learningWS {


@WebMethod(operationName="titre_cours")
    public Compte getCours(@WebParam(name="code")Long code){
            return new Compte(code,”Mathematique”,”Newton”);
                    }
@WebMethod()
    public List<Cours> getCourss(){
            List<Cours> nb=new ArrayL ist<>();
            nb.add(new Cours(1L,”Physique”,”Enstien”));
            nb.add(new Cours(2L,”Chimie”,”Bernoli”));
            return nb;
    }
@WebMethod(operationName="dateDuServeur")
    public Date getServerDate(){
            return(new Date());
    }
}
```

- This class defines a web service named learningService which is defined by

- method named *getCours* which allows you to return a course.
- *getCourss* method to return the list of courses .
- A method to return the server date.

JAX-WS annotations are specific to web services. They allow influencing the structure of a WSDL document by changing some parameters of the service or methods that compose it. In this section, we describe more precisely the behavior of these annotations.

- The main annotation to define a web service @javax.jws.WebService web service. This annotation applies to a class and possibly using several attributes that allow, for example, to specify the location of a web service.

  Here is the list of attributes that can be implemented:

  1- **name** : Specifies the name of the web service. This name is now in the WSDL file in the name attribute of the tag <portType> . The default value is the name of the implementation class here learningWS.

  2- **serviceName** : Sets the service name. This name is found in the WSDL file in the name attribute of the tag <service> . The default value is the name of the implementation class learningWSService suffix "Service" here then.

  3- **wsdlLocation** : sets the URL , either absolute or relative , an existing WSDL file. By default, this file is auto-generated during deployment.

  4- **endpointInterface** : Sets the full name of the interface "endpoint " defining the methods of the web service. This allows the developer to separate the "contract" (interface) of the implementation. The implementation of this interface service is not required. The advantage of this solution is to assign annotations to Java mapping to the WSDL in the interface and not the implementation class.

- The **@WebMethod** annotation is used to declare the methods accessible by the web service (we are talking about operation in the SOAP protocol). Here are the details of the attributes it contains:
  - **operationName**: Sets the value of the name attribute of the tag <operation> in the WSDL file . It represents the name of the operation.
  - **exclude**: mark a method as unavailable by the web service. This parameter is used when we want to hide an inherited method, for example.
  - The method parameters and return value can also be changed. The **@javax.jws.WebParam** controls the generation of the WSDL for the parameters of the method. Thus, the annotation **@WebParam** allows you to specify information about the arguments of the operations defined in the WSDL file.

### 4.2 BUILD SERVER

- Our web service is ready to be deployed in a J2EE application server. In this work, we will create our own server.
- In the src folder, create a ServeurWS.java class whose code is as follows :

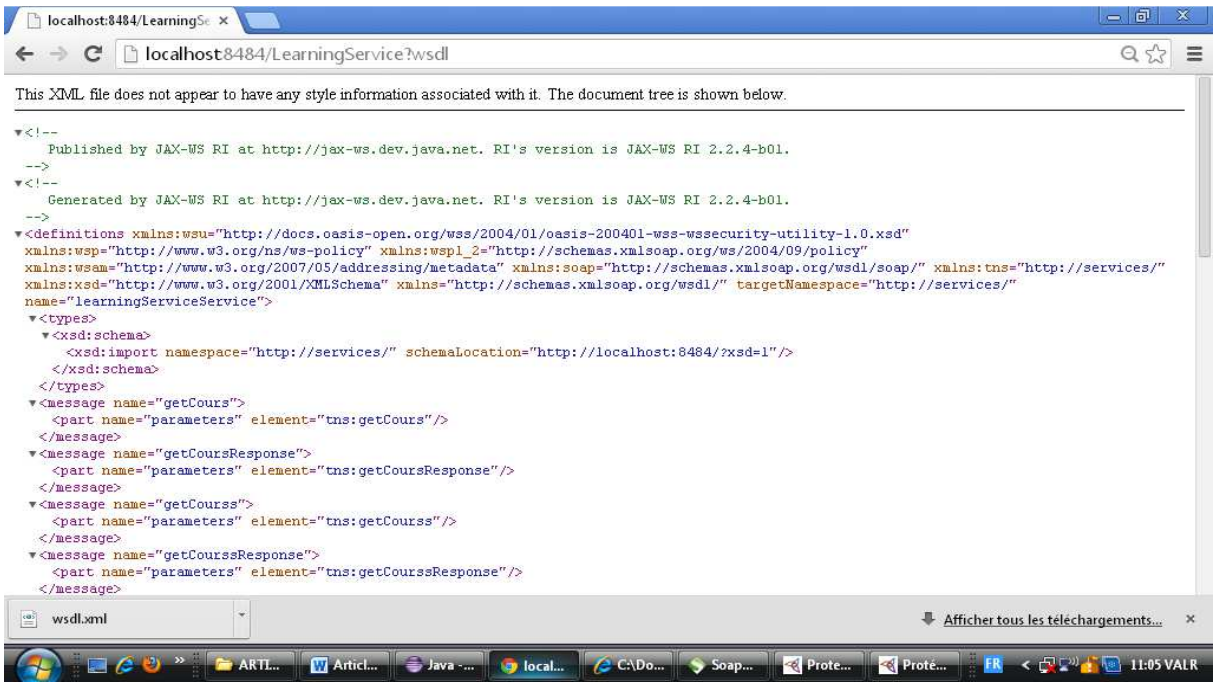
```
package serveur;
import javax.xml.ws.Endpoint;
import services.learningService;
public class ServeurWS {
        public static void main(String[] args) {
                String
url="http://localhost:8484/";
        Endpoint.publish(url,new
learningService());
        System.out.println(url);
        }
}
```

- Start the server by running the LearningService application. The server starts by displaying the following message:

Web Service published http://localhost:8484/

### 4.3 TESTING THE WEB SERVICE WITH A WEB BROWSER

- Launch a browser and enter the url : http://localhost:8484/LearningService?wsdl



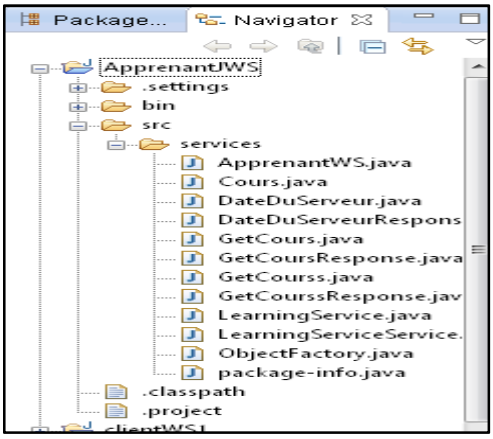

- To view the XML schema for SOAP messages, type the url :http://localhost:8484/LearningService?xsd=1

### 4.4 CREATING A JAVA CLIENT

On the command line, navigate to the src folder of your project. And type the command:

**wsimport –s . LearningService.wsdl**

The following files will be generated:




- Now is the time to easily create a java client for our web service.
- In the src folder create the following ApprenantWS class:

```
package services;
import java.util.List;


public class ApprenantWS {


	public static void main(String[] args) {
		LearningService stub=new LearningServiceService().getLearningServicePort();
		Cours res=stub.getCours(1L);
		System.out.println(res.getCode());
		System.out.println(res.getTitre());
		System.out.println(res.getAutor());
		System.out.println("-----------------------------------");
		List<Cours> nb=stub.getCourss();
		for(Cours c:nb){
			System.out.println(c.getCode());
			System.out.println(c.getTitre());
			System.out.println(c.getAutor());
			System.out.println("----------");
		}

	}
}
```

- **Result of execution :**

```
1
mathematique
Newton
-----------------------------------
1
physique
enstein
----------
2
chimie
bernoli
----------
```

## 5 CONCLUSION

This work presents the steps required to create a web service based on an example that relates to the field of e -Learning. In addition, it cited the components and the basic functionality of a web service. We also describe how educational content can be dynamically created and disseminated in a distributed e-Learning system based on Web services.

Our studies have shown that most existing solutions that can manage a dynamic educational content way, are always in the context of centralized systems and use proprietary formats. Web services come to resolve this obstacle interoperability of e -Learning [12] systems. Web services technologies can contribute to solving these problems.

The few cases the use of web services in e -Learning distributed system [10] are currently at a conceptual level, and are deployed in large or medium called enterprise systems. Our approach is based on simple principles using as much as possible of standard concepts and open source tools.

We have shown an example of implementation of web services for the contribution to the development of an e -Learning system. It remains as a perspective to upgrade this environment we achieved to allow students to practice learning divers.

## REFERENCES


[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, Web Services - Concepts, Architectures and Applications. Springer-Verlag, 2004.

[2] CORBA : des concepts à la pratique: GEIB Jean-Marc, MERLE Philippe Université des sciences et technologies de Lille, France

[3] F. Curbera "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI", IEEE Internet Computing, vol. 6, no. 2, pp. 86 -93 2002

[4] R. Akkiraju et al.. "Web Service Semantics - WSDL-S." 2005. Retrieved January, 2011, from http://www.w3.org/Submission/WSDL-S/.

[5] Extensible Markup Language (XML), W3C Recommendation 16 August 2006, Authors and Contributors: Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan.

[6] Simple Object Access Protocol (SOAP): http://www.w3.org/2000/xp/Group/.

[7] Web Services Description Language (WSDL): http://www.w3.org/2002/ws/desc/.

[8] Universal Description, Discovery and Integration (UDDI): http://www.uddi.org.

[9] Java Technology and Web Services: http://java.sun.com/webservices/.

[10] Vossen G., Westerkamp P. Maintenance and Exchange of Learning Objects in a Web Services Based e-Learning System. *Electronic Journal of e-Learning Volume 2 Issue 2 2004.* 292-304.

[11] E. Newcomer, Understanding Web services: XML, WSDL, SOAP, and UDDI, Addison-Wesley,Boston, Mass., May 2002.

[12] Vossen, G., P. Westerkamp (2003). E-Learning as a Web Service; Technical Report No. 92, Department of Information Systems, University of Münster, January 2003.

[13] D. Austin, A. Barbir and S. Garg, "Web Services Architecture Requirements," W3C working draft, Apr. 2002.

[14] The Java architecture for XML binding JAXB, Mark Volkmann, Partner, object computing, Inc

[15] Jinyu Wang, Scott Brewton, “Making XML Technology Easier to Use”