Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 2, pp. 251-260

Implementing Web Services Using Java Technology

M. Pirnau

Mironela Pirnau
"Titu Maiorescu" University Bucharest
Romania, 22 Dambovnicului Street, District 4, 040051, Bucharest, Romania
E-mail: mironela.pirnau@utm.ro

Abstract:
There are several essential activities that need to take place in any service-oriented environment such as: a Web service has to be created, to have its interfaces and invocation methods defined, its service has to be published to one or more intranet or Internet repositories to locate potential users. A Web service needs to be located in order to invoke potential different users. The technologies offered by Web Services enable us to cast older applications, we can use them and the packages already existing in a certain enterprise. The infrastructure associated with older applications can also be wrapped as a serial set of services. In the following pages I tried to present general characteristics of the Web services up to present, as well as the usage of Java technologies for the implementation of web services. The present paper targets a systematisation of the main web services, as well as highlighting and testing the most important technologies used to develop them.
Keywords: Web Service, Java, XML, SOAP and Semantic Web.

1 Introduction

Web Services are software applications that conform to the Web Service Interoperability Organization. They are becoming more and more popular and reveal a model in which discrete tasks within e-business processes are widely distributed throughout a value net. The label for "web services" has two levels of meaning - one specific and one conceptual. Web services represent standards, which describe services oriented to applications with architecture having a base of components. These are software components, which can be reusable, having included functionality [11, 12, 16]. The Web Services use standard XML languages. They use HTTP for sending messages and they are independent of the platform and of the language. Web Services are freer systems, and the customer may not have knowledge of the Web Service up to the time it invokes, being suited to join requests of applications meeting demands of an Internet-wide [22].

2 Web Services Structure

Web Services systems maintain important decoupling and dynamic bonds of the elements. The entire elements in a system represent and publish an Application Programming Interface (API) message to different components in the network that work together. Services are comprised among applications by employing service discovery for the dynamic bonds of the partners. The Web Services show such architecture full of service orientated approach by orchestrating and finding out available network services, or even precise integration of the incidences that are based on the term of constructing applications.

The Web Services' architecture need three basic performs such as publish, find and bind. The providers publish services toward a certain service interloper [1, 4]. The requesters "find" demands services using an interloper and "bind" to them. The architecture of Web Services describes certain principles for building and also creating dynamic systems with no single involving.

There are many ways to instantiate a Web Service, choosing various implementation techniques for the roles and operations of the Web Services structure. From the structure of Web services we may mention: the service processes that involve more than one Web service, including discovery - belongs to this part of the architecture as it allows us to locate one particular service from among a collection of Web services. The architecture of Web Service invokes many layers and joins technologies [22] as we can see in the figure 1. Web Services can support other network protocols and distributed computing technologies. The implementation, according to the SOA principles, is based on the following protocols: HTTP, SOAP, WSDL, and UDDI [17].

HTTP[3] represents the most used Internet protocol and also the leading method of transmitting the information within the Web. Protocol of demand-answer type between the clients and the servers is another type of method to send data through Web. Simple Object Access Protocol (SOAP) is a format used to transmit XML message [1, 2]. It also represents a certain type of communication protocol among applications within Internet. It is a platform independent protocol no matter the language used and it permits the avoiding firewalls. Web Services Description Language (WSDL) is also a XML language that permits the specification of the accessing manner within the Web services. It allows the description of the web services, too.

Universal Description Discovery and Integration has the following acronym UDDI. It is a system of registers that is based on XML; it is independent of the platform. UDDI is requested by the SOAP messages that demand a web service [10]. It offers access towards the WSDL descriptive document within the web service, too.

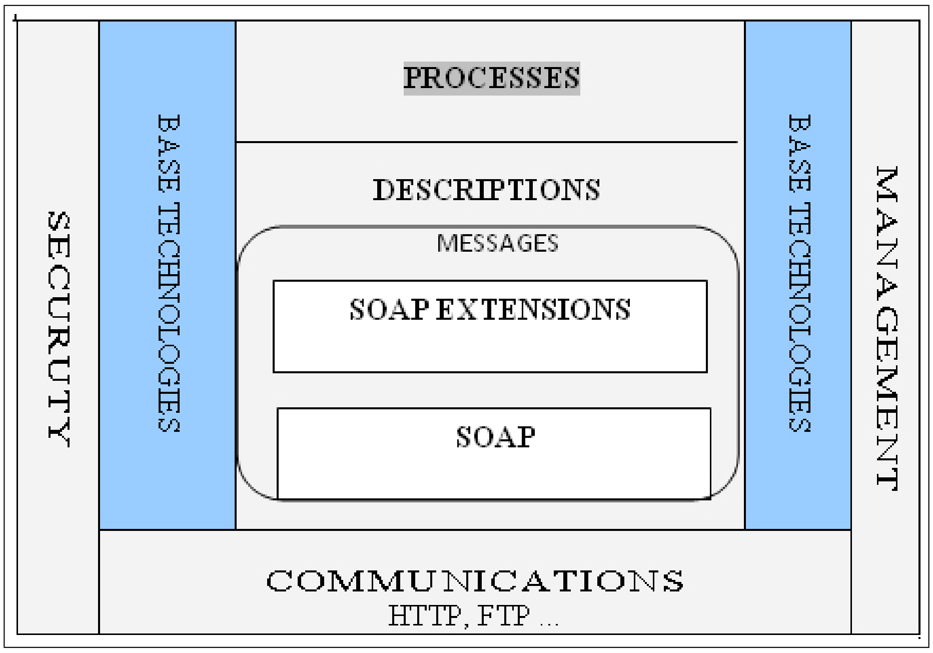


Figure 1: The architecture layers of the Web Services

The SOAP mainly specifies the manner we should format demands towards the server and the manner the server could build its own reply. SOAP represents a popular selection for Web Services. Along the transport, messages might be transmitted within a specific manner between servers and clients. The protocol of choice is represented by the HTTP architecture.

The pile depicted above (figure 2) enables Web Services to leverage the existing Internet infrastructure. It creates a low cost of entry to an omnipresent environment. The flexibility is not compromised by the interoperability requirement provided as alternative and value-add technologies [1]. An action that makes a WSDL document available is the service provider sending a WSDL document directly to a service requestor, named "direct publication". The service provider can publish the WSDL document describing the service to a host local WSDL registry, a private UDDI registry or to the UDDI operator.

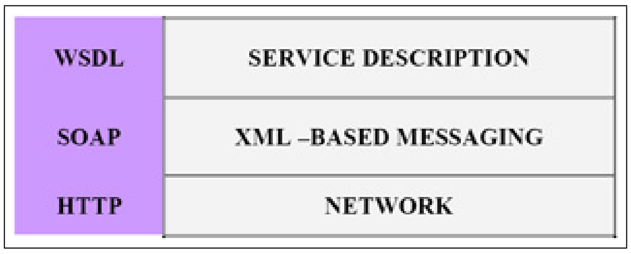


Figure 2: The Interoperable base Web Services stack

3 Java Technologies

Web services represent web-based enterprise applications which use transport protocols that are XML- based, open standards in order to exchange data to their clients [18]. From the Java technologies we mention the following, presented in the table below [20].

Table 1: **Java Technologies**

WSIT	Web Services Interoperability Technology
JAX-WS10	The Java API for XML Web Services, Package javax.xml.ws
JAX-RPC	The Java API for XML Processing
JAXP	Provides a Java interface to DOM, SAX, and XSLT
JAXB	Java Architecture for XML Binding, Package javax.xml.bind
SAAJ	The SOAP with Attachments API for Java, Package javax.xml.soap
JAXR	The Java API for XML Registries
JSTL	JavaServer Pages Standard Tag Library
JSR 181	Java Web Services Metadata, Package javax.jws

As an example of these technologies I shall present the SAAJ [22] technology that stands for SOAP with attachments API for JAVA and that provides a standard set of APIs in order to send XML documents including attachments over the Internet. It is similar to JAX-RCC but it needs additional effort both on the client's and on the server's side [15, 21]. The SAAJ APIs provides a standard means for handling SOAP messages. It can be used to create, to inspect and to alter SOAP messages. SAAJ also contains methods for creating and sending attachments with the SOAP message. The attachments may contain data in the shape of any format, not being limited to XML as in the case of the SOAP messages. The attachments can be used to send inappropriate data in XML format as images for example. The SAAJ APIs offers the AttachmentPart class in order to represent the attachment part of a SOAP message [16]. Any SOAP Message object has automatically a SOAP Part object and requires sub-elements but because the AttachmentPart objects are optional, we must create and add them ourselves. SAAJ contains the API in order to create, populate and access SOAP messages, according to SOAP and SOAP Attachment specifications. SAAJ contains API needed to send non-provider-model messages [21]. We present the package relationship in figure 3 .

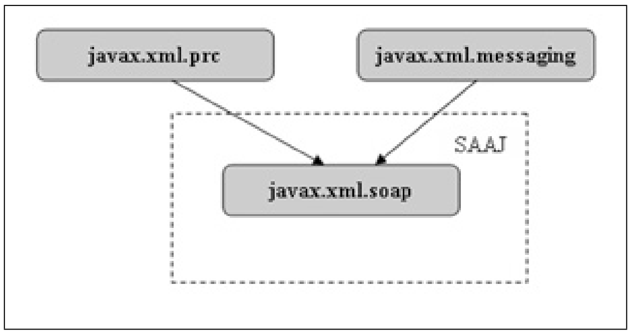


Figure 3: SAAJ - package relationship

SAAJ represents a part from JAXM (Java API for XML Messaging) which was evolved from the JSR expert group [1, 13, 21]. The final form of the specifications show two different aspects: the Core functionality concerned with manipulating of the SOAP messages and a very high level of messaging facility. Regarding SAAJ his an API of a very low standard, which is completely aware of the messages that are exchanged within the Web Service and its customers [16]. An application that uses SAAJ needs to construct SOAP messages little by little and take data out of the reply messages.

The usage of SAAJ demands a great deal of work deposed from the developer's side than JAX-RPC. We present a few cases in which it would be better to be helped by SAAJ and its relative JAXM instead of using JAX-RPC [1, 7].

JAX-RPC can be accessed just when the customer and the service are active at the same specific time and we also assume that there is a free network way to bond them. In the case the service is not free the moment the customer accesses it, there is no possibility of using JAX-RPC [17, 20].

JAXM is a service that provides a confident service able to deliver without need to demand the customer elements to be included in the way the confidence is offered. SAAJ as well as JAXM deliver a wide result disposal for messages based on XML.

The API from the **javax.xml.soap** package permits specifications such as: creating a piece of XML; creating access, adding, modifying parts of the SOAP message; creating a point-to-point contact to a certain endpoint; creating, adding, modifying a certain SOAP fault of information type etc. The table 2 includes the **javax.xml.soap** package and includes the API for building SOAP messages [4, 20]. The package is defined and included in SOAP with attachments API for JavaTM is named SAAJ.

SAAJ a tool for creating and sending messages

To allow sellers to bring up the implementations, the entire SAAJ API is built of interfaces and some classes that are abstract. In order to create a SOAP message, there is need to be used a MessageFactory type message. Part of the **javax.xml.soap** package represents an abstract class and that might be obtained by using the newInstance method. This type of method is represented by the public static MessageFactory newInstance that propagates SOAPException.

The Message Factory implies two methods to create the SOAP messages. The first method is represented by one public SOAPMessage createMessage that throws SOAPException. One public SOAPMessage createMessage(MimeHeaders headers, InputStream) which throw SOAPException. The second of the methods is used to non-serialize messages in a link of an input.

Table 2: **Methods included in the javax.xml.soap package**

Class Summary	
AttachmentPart	Single attachment to a SOAPMessage object
MessageFactory	Factory for creating SOAPMessage objects
MimeHeader	Object that stores a MIME header name and the value
SOAPConnection	Connection that a client might use for transmitting messages to a certain remote party (URL)
SOAPConnectionFactory	Factory for creating SOAPConnection objects
SOAPElementFactory	Creating SOAPElements
SOAPFactory	Creating various objects existing in the SOAP XML stack
SOAPMessage	SOAP messages root class

The default messages that comes back by the MessageFactory createMessage method contains a SOAPPart but also contains no types of attachments. An object of SOAPEnvelope style is included in the SOAPPart. In the middle of the envelope there are certain parts such as the header, made by an object able to implement the SOAPHeader interface and a SOAP message body in the shape of an object of SOAPBody classification. The body and the header, both contain certain XML elements. The Java technologies represent methods for constructing web services [8].

4 Related Work

The Web Semantic implemented and proposed use of the machinery to extend the Web represents a common goal. The multiple demands on the Web create a semantically temptation having a highly expressive formalisms because the Web represents a principled architecture of standards, formalisms and languages [6, 14].

What is specific for Web 2.0 is that it connects humans among them and for Web 3.0 it is specific that it connects humans to information in a very innovative new manner.

Web 3.0 functions as an intelligent assistant, which perceives the concepts of the Web pages and transmits important information of one individual. It not only looks for robotically through keywords. Tim Berners-Lee is the person who founded the World Wide Web. He defines the future online environment as being the Semantic

Web [5, 8, 9]. This is the same Web 3.0 a type of Internet that perceives information and not just transmits it to the client. Web 3.0 will be capable to offer for visitors complete data on any subject. The communities that are based on the same complete data exchange will certain develop. Mobile Web which is supported by mobile intelligent terminals would be able to enable the data access everywhere and anytime.

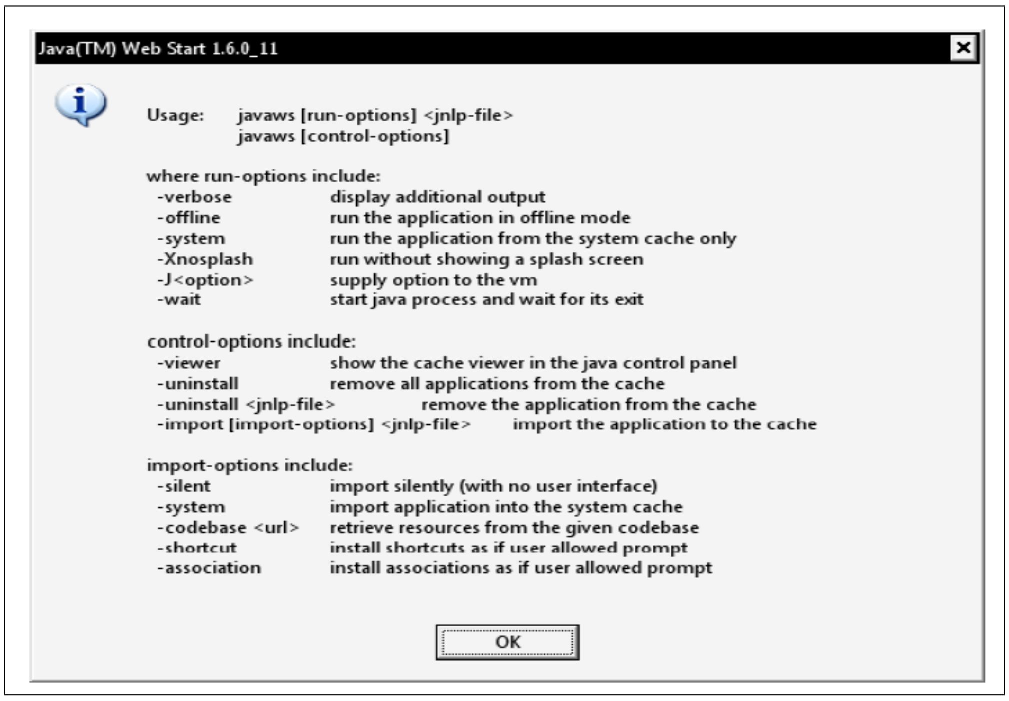
Rich Internet Applications (RIA) would appear more and more. Web 3.0 represents a much more direct, closer, measurable, reliable and honest communication.

The Metro style is made up from: JAX-WS, JAXB, and WSIT and within it is possible to create Web services that are secure, transactional, confident and interoperable, also certain customers [15]. The Metro elements are a certain part of the Project Metro, which part of GlassFish, Java EE (Java Platform, Enterprise Edition), and only part in Java SE (Java Platform, Standard Edition). Both GlassFish and Java EE represent support for JAX-RPC API [15, 17].

5 Web Applications. Java Web Start Technology

JWS (Java Web Start) represents a technology for installing, launching, and updating the java applications directly from the Web and uses the JNLP (Java Network Launching Protocol). For Java applications, that use of the JWS technology launches a range of order of javaws application as the following syntax:

javaws [options] [URL], and the options are those in figure 4.



```
Usage:    javaws [run-options] <jnlp-file>
          javaws [control-options]

where run-options include:
  -verbose            display additional output
  -offline            run the application in offline mode
  -system             run the application from the system cache only
  -Xnosplash          run without showing a splash screen
  -J<option>          supply option to the vm
  -wait               start java process and wait for its exit

control-options include:
  -viewer             show the cache viewer in the java control panel
  -uninstall          remove all applications from the cache
  -uninstall <jnlp-file>        remove the application from the cache
  -import [import-options] <jnlp-file>      import the application to the cache

import-options include:
  -silent             import silently (with no user interface)
  -system             import application into the system cache
  -codebase <url>     retrieve resources from the given codebase
  -shortcut           install shortcuts as if user allowed prompt
  -association        install associations as if user allowed prompt
```

Figure 4: Options for Java Web Start

The role of the programmer: creates a Java application and archives the program along with its all resources; signs the jar archive (if necessary) and creates a configuration folder (jnlp) that describes the application; creates a Web page having a link to the configuration folder and copies on the web server, the **jar** folder, the **jnlp** and **Html** one.

The role of the user: accesses the link to the application from the programmer's Web page; the application will be installed automatically, locally and installs a JRE if necessary; launches in execution the application; for each execution of the application, it verifies if a new version on the programmer's new web page exists.

As a practical application, I have created a project that contains Java classes with administration role of the students. The purpose of the application is to use the Java Start technology and to realize "a following" in using the application for this technology as well as other technologies that I have tested (for example the Servlets technology etc). To realise the source, I also used the classes from the **Swing, AWT, SQL** etc packages. If we open the application in the NetBeans IDE 6.7 [23] environment, the project window looks like the one in figure 5.

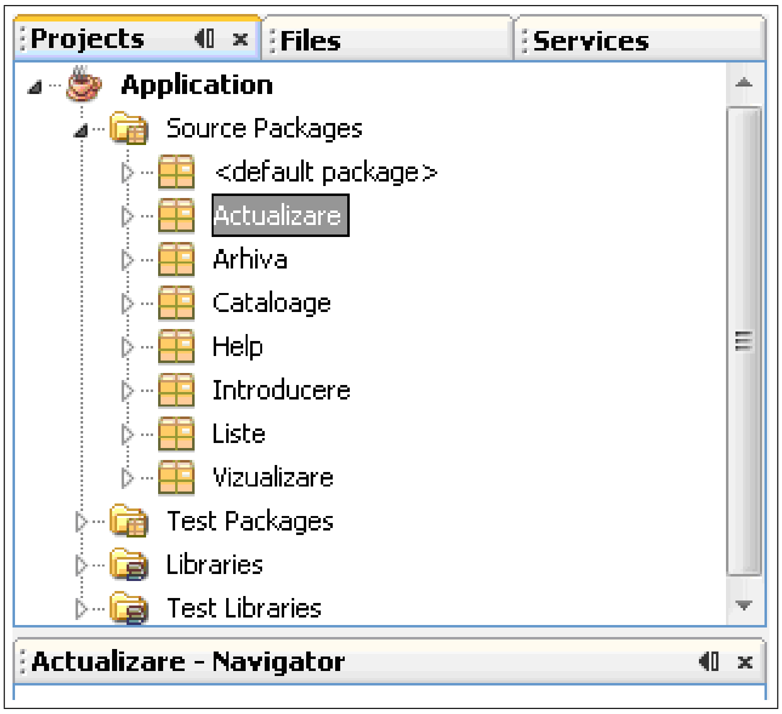


Figure 5: The window of the application opened in Netbeans

Helped by Java Web Start, the users may access one Java application only with click of a mouse on an HTML link towards a JNLP file for the certain applications from inside the web browser [19].

This should be installed onto the customer machine. It is not necessary to install JDK (Java Development Kit). In order to permit our Java application to be ran with Java Web Start, we have to configure the main properties of the manner the IDE could construct the project. The moment Java Web Start is accessed, the IDE creates automatically a JNLP file and an HTML page along with the link to the JNLP file, along with the JAR file. For the beginning, we must configure the project in order to activate Java Web Start and to locally test the application. So we must choose the Properties option from the project's shortcut menu, and in the window from the figure 6 the suitable settings are set.

The application's project must be configured so that the Java Web Start technology may be used at the launching.Certain sources are mixed together and then the main class of the project is launched in execution. The IDE mixes sources and we might see the Java screen starts, and in the warning window we are asked if the application might be executed and also to Select the checkbox and click Run inside the proper warning window.

The *"Application"* application begins, the same as in figure 7.

After activating the application, the **Dist** folder is created and contains for Java Web Start, the following two additional folders:

- launch.jnlp

It is an XML file having special parts and specifications instructing browsers how to run the application. The attributes for the JNLP files might include the JNLP version, application title, the seller's name, a link of the application of the JAR file etc. The configuration folder *jnlp* is in an XML format and offers information regarding the application that will be installed using JWS. Its structure is presented in figure 8.

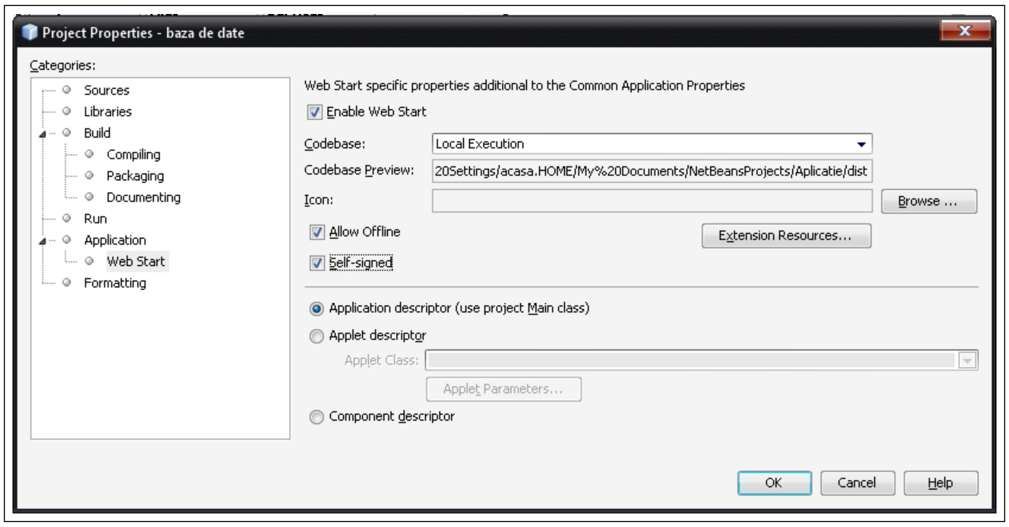
- launch.html

It is generated in an automatic HTML page a link to the JNLP file. The users might click the link in order to begin applications through Java Web Start (figure 9).

How to Run an Application from a Distant Location

The moment we are sure that the application begins in a successfully manner using Java Web Start from local sources, we might upload this to a distant location and start launching it from far there.

In order to develop applications using Java Web Start inside the web, the server of the Web is able to manipulate JNLP files. The web server can be configured in order to recognize JNLP files as applications, such as the MIME type for JNLP. Each Web server has a certain manner for adding MIME types. To develop our application to the


Project Properties - baza de date

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Application
  - Web Start
- Formatting

Web Start specific properties additional to the Common Application Properties

[✓] Enable Web Start

Codebase: Local Execution

Codebase Preview: 20Settings/acasa.HOME/My%20Documents/NetBeansProjects/Aplicatie/dist

Icon:

Browse ...

[✓] Allow Offline

Extension Resources...

[✓] Self-signed

[● Application descriptor (use project Main class)]

○ Applet descriptor

Applet Class:

Applet Parameters...

○ Component descriptor

OK Cancel Help

Figure 6: Project Properties Web Start

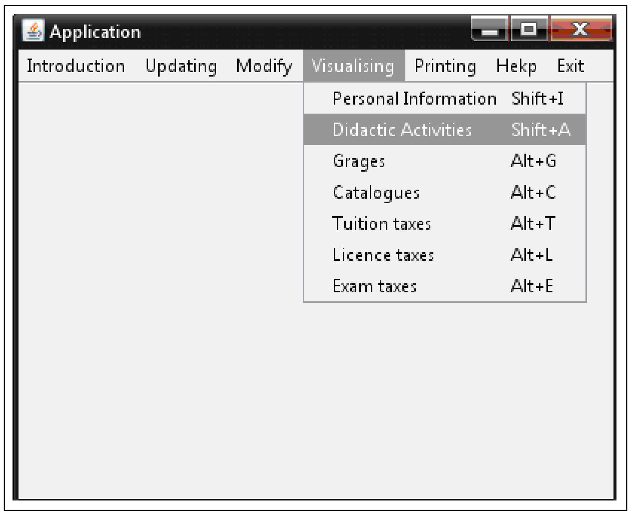
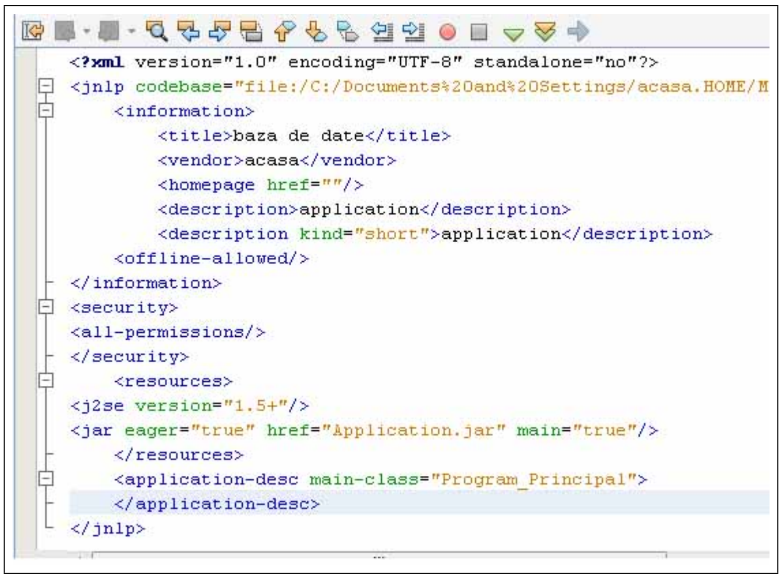
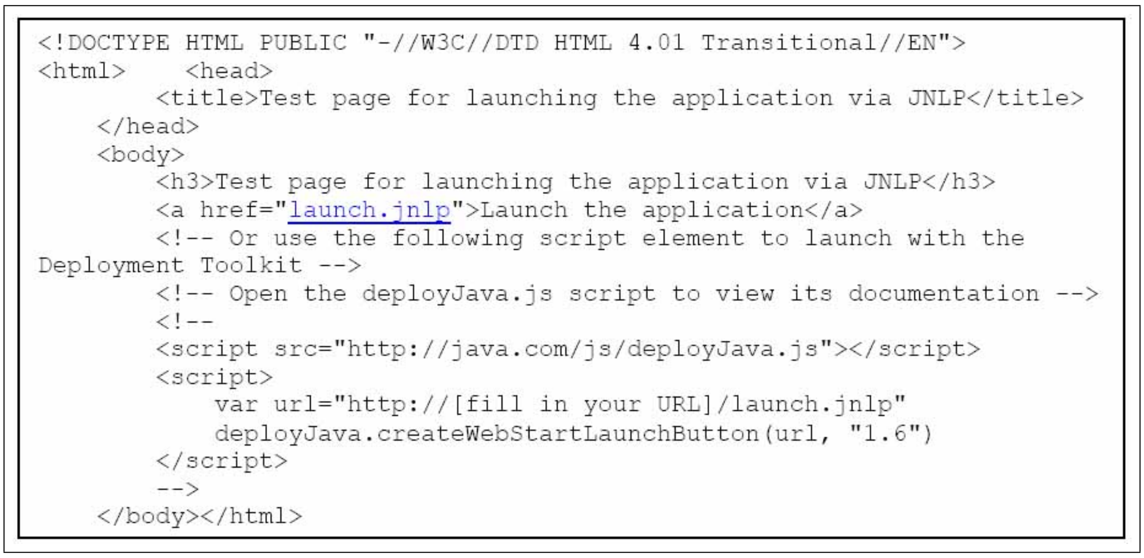


Figure 7: The main menu of the application



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<jnlp codebase="file:/C:/Documents%20and%20Settings/acasa.HOME/M
    <information>
        <title>baza de date</title>
        <vendor>acasa</vendor>
        <homepage href=""/>
        <description>application</description>
        <description kind="short">application</description>
    <offline-allowed/>
</information>
<security>
<all-permissions/>
</security>
    <resources>
<j2se version="1.5+"/>
<jar eager="true" href="Application.jar" main="true"/>
    </resources>
    <application-desc main-class="Program_Principal">
    </application-desc>
</jnlp>
```

Figure 8: The structure of the **launch.jnlp** file

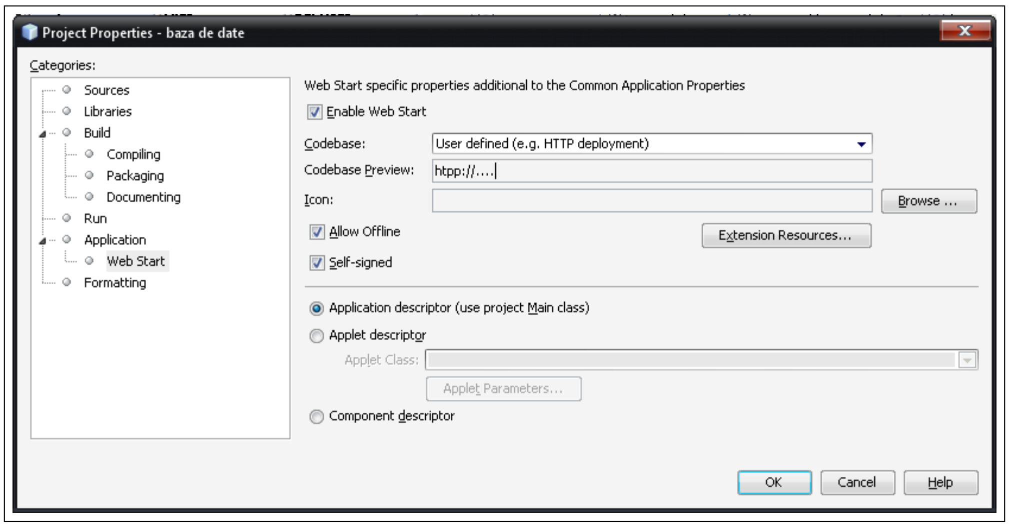


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>    <head>
        <title>Test page for launching the application via JNLP</title>
    </head>
    <body>
        <h3>Test page for launching the application via JNLP</h3>
        <a href="launch.jnlp">Launch the application</a>
        <!-- Or use the following script element to launch with the
Deployment Toolkit -->
        <!-- Open the deployJava.js script to view its documentation -->
        <!--
        <script src="http://java.com/js/deployJava.js"></script>
        <script>
            var url="http://[fill in your URL]/launch.jnlp"
            deployJava.createWebStartLaunchButton(url, "1.6")
        </script>
        -->
    </body></html>
```

Figure 9: The structure of the **launch.html** file

customer, we should be sure that a Web server accesses all the files that contain our application. This may represent typically amounts for copying one or many of the JAR files, with a JNLP file, to the directories of the Web server.

The settings needed to enable the Web site sustain that Java Web Start is quite the same to develop a content based on HTML. In order to launch the application out of the web, we have to use a certain link of the files of applications source in the JNLP file. Regarding this, the project is chosen and in the **"Properties"** window the **CodeBase** section is defined. The URL where the files are loaded has the JNLP extension and also the jar and *Html* extension (figure 10).



Project Properties - baza de date

Categories:
- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Documenting
- Run
- Application
  - Web Start
- Formatting

Web Start specific properties additional to the Common Application Properties

☑ Enable Web Start

Codebase: User defined (e.g. HTTP deployment)

Codebase Preview: htpp://....

Icon:

☑ Allow Offline

☑ Self-signed

● Application descriptor (use project Main class)

○ Applet descriptor

Applet Class:

○ Component descriptor

Figure 10: Settings for running applications on the Web

At this moment we may run our application, in the window of a browser, we type the URL towards the launch.html file and then press the "Launch the application" connection. Our "Application" application begins with Java Web Start. After analyzing the current application, we may observe that the integration of the applications and of the data sources must be realized without significant changes of the applications and of the data. Because the Web allows the easy access to information and services, it became a main element of communication having the condition to be used for applications that run it, adequate technologies such as those implemented on the Java platform.

6 Conclusions

The Web services represent a wide range of technologies used in the application of distributed implementation at the web system level. These technologies develop a program-to-program communication pattern using the XML standard, which assures a neutral shape report to platforms, programming languages and software specifications.

A Web services application is an application that interacts with the world using XML for data definition, WDSL for service definition and SOAP for communication with another software. The fundamental idea of the web services is the integration [16]. This concept represents a set of standard technologies that facilitate the interoperability among heterogenic systems at an organizational level or on the Internet.

For the optimization of the Web aplications, the companies head more and more to the programming environment of Java and of the relevant platforms. Java represents an independent language regarding the platform that is projected for the Web and, at the same time, it is a developing platform of various aplications, no matter the company's specific. Next to the Java platform the XML technology has developed having the goal to exchange data among the aplications.

As future research I intend to run more applications on Java different platforms and to compare both the manner of using these for working in the environment and the security level implemented the platforms.

Bibliography

[1] E. Hewitt, *Java SOA Cookbook*, O'Reilly, 2009.

[2] J. Governor, D Hinchcliffe, D , *Web 2.0 Architectures*, O'Reilly, 2009.

[3] J. Zukowski, *Java 6 Platform Revealed*, SpringerLink Date, 2006.

[4] *** , *SOAP with Attachments API for Java (SAAJ*), Sun Microsystems, Tech. Rep., 2004.

[5] R. Volz, S. Handschuh, S. Staab, L. Stojanovic, and N. Stojanovic, Unveiling the Hidden Bride: Deep Annotation for Mapping and Migrating Legacy Data to the Semantic Web, *J. Web Semantics*, vol. 1, no. 2, pp. 187-206, 2004

[6] I. Dzitac, B. E. Barbat, Artificial Intelligence + Distributed Systems = Agents, *International Journal of Computers, Communications and Control*, 4(1):17-26, 2009.

[7] R. Chinnici, *Java API for XML-Based RPC (JAXRPC)*, Java Community Process, Tech. Rep., 2003.

[8] R. Nagappan, R. Skoczylas, R. Sriganesh, *Developing Java Web Services*, Wiley 2003.

[9] I. Dzitac, S. Dzitac, E. Valeanu, Web Distributed Computing for Landscape Architecture, in Eleftheriadis, N., Styliadis, A., Paliokas, I. (eds), *Proceedings of the LANT07, International Conference Landscape Architecture and New Technologies*, 25-26 May 2007, Drama, Greece, pp. 25-36, 2007.

[10] ***, *Java 2 Platform, Enterprise Edition Specification*, Sun Microsystems, Tech. Rep., 2003;

[11] L. Duta, F.G. Filip, Control and decision making in disassembling used electronic products, *Studies in Informatics and Control*, 17 (1):17-28, 2008.

[12] L.A. Zadeh, D. Tufis, F.G. Filip, I. Dzitac (Eds), *From Natural Language to Soft Computing: New Paradigms in Artificial Intelligence*, Editing House of the Romanain Academy, Bucharest, 2008.

[13] N. Kassem, A. Vijendran, and Rajiv.Mordani, *Java API for XML Messaging (JAXM)*, Sun Microsystems, Tech. Rep., 2003, http://java.sun.com/xml/downloads/jaxm.html.

[14] http://www.nowpublishers.com/.

[15] http://java.sun.com/webservices/technologies/index.jsp.

[16] http://en.wikipedia.org/wiki/Service-oriented_architecture.

[17] http://java.sun.com/developer/technicalArticles/WebServices/soa/.

[18] http://java.sun.com/javase/technologies/desktop/javawebstart/index.jsp.

[19] http://java.sun.com/javase/6/docs.

[20] http://java.sun.com/webservices/docs/1.6/tutorial/doc/.

[21] http://www.javapassion.com/webservices.

[22] http://www.w3.org/TR/ws-arch/.

[23] http://www.netbeans.org.

Mironela Pirnau (1966) has a M.Sc. in Automatics, Faculty of Automatics and Computing, Politechnical University in Bucharest (1992) and PhD in Electronics (2003) from University of Pitesti. Her current interests include Applied Informatics. Since 2008 she has been a Lecturer PhD at the Faculty of Science and Technology of Information, Titu Maiorescu University in Bucharest. She has published books and articles on Informatics. She has recently been interested in researches on the use of Java technologies in developing applications of Informatics.