ijcrr
Vol 04 issue 15
Category: Research
Received on:17/06/12
Revised on:02/07/12
Accepted on:13/07/12

EFFECTIVE WAY TO DEVELOP WEB SERVICES

Girish Tere[1], R. R. Mudholkar[2], Bharat Jadhav[3]

[1]Department of Computer Science, Shivaji University, Kolhapur
[2]Department of Electronics, Shivaji University, Kolhapur
[3]Department of Electronics and Computer Science, Y.C. Institute of Science, Satara

E-mail of Corresponding Author: girish.tere@gmail.com

ABSTRACT

We propose a prototype framework, called web service Application Fabric (WSAF) that is used for developing WSDL-centric approach to Java web Services. To encourage good design and make programming easier is to use an application framework. Framework offers a layer of abstraction on top of complex tools. A good SOA framework should encourage use of XML schema libraries and reuse schema across WSDL documents. A web service is a programmatic interface for application-to-application communication that is invoked by sending and receiving XML. “WSDL-centric” means creating a web service by building its WSDL and using that WSDL document with references to the Java elements that implement it. Instead of using standard WSDL format we are generating web service interface using our own style. These generated files are easy to process. We found that WSAF performance is better than Apache Axis 2 and JAX-WS. WSAF is an open-source web Services application fabric supporting HTTP protocols such as REST, SOAP, XML, and JSON. From the description written in simple XML, WSAF generates the Client APP (.jar), the Java server code template (.war), the WSDL and the documentation of the description in HTML. We compared performance of WSAF with Axis 2 and JAX-WS frameworks.

Keywords: SOA, web services, WSAF, WSDL, XML

INTRODUCTION

A web service is a platform neutral, language neutral and can accessed across the network. A web service has one or more ports. Each port is a binding deployed at a certain network address (endpoint). While most descriptions of web based solutions emphasize their distributed characteristics, their decentralized nature – they have distinct management and control environments and communicate across trust domains – has much more impact on architecture of this framework and the requirements of the underlying protocols. So, we focus our framework first on supporting application-to-application integration between enterprises having disjoint management, infrastructure and trust domains. The focus of this paper and the framework it defines is a model for describing, discovering and exchanging information that is independent of application implementations and the platforms on which applications are developed and deployed. Every web Services platform, like Apache Axis [1], Xfire, JBOSS, JAX-WS [5], Spring or something else, has to provide three core subsystems:

- Invocation
- Serialization
- Deployment

WEB SERVICE APPLICATION FABRIC

Java is a powerful development platform for service-Oriented Architecture (SOA) [20, 24]. Because robust web Services technology is the foundation for implementing SOA, Java now provides the tools modern enterprises require integrating their Java applications into SOA infrastructures. JWS has weaknesses, particularly when it comes to the development approach known as “Start from WSDL and Java” [27]. The JWS [5, 10] standards present a Java-centric approach to web Services. This approach is difficult when we need to work with established SOA standards and map Java application to existing XML Schema documents and WSDLs. One way to encourage good design and make programming easier is to use an application framework. For example, the Apache Struts framework encourages web applications development based on the Model View Controller (MVC) framework. Frameworks [6, 7] offer a layer of abstraction on top of complex toolsets. The layer of abstraction encourages you to program in a certain way. By restricting your programming choices to a subset of proven patterns, the framework makes your job easier and less confusing. Application frameworks can also encourage good design. A good SOA framework, therefore, should encourage the use of XML Schema libraries and promote the reuse of schema across WSDL documents. A good SOA framework should separate compiled schemas and WSDL from the rest of the application classes [25]. A common framework identifies specific functions that need to be addressed in order to achieve decentralized interoperability. It does not determine the particular technologies used to fulfill the functions but rather divides the problem space into sub-problems with specified relationships. This functional decomposition allows differing solutions to sub-problems without overlaps, conflicts or omitted functionality. This is not to say that all applications must offer the same facilities, rather that when a feature is offered it should fit into a common framework and preferably have a standard expression. A web application framework is a type of framework, or foundation, specifically designed to help developers build web applications. These frameworks typically provide core functionality common to most web applications, such as user session management, data persistence, and templating systems. By using an appropriate framework, a developer can often save a significant amount of time building a web site.

IMPLEMENTATION OF WSAF

WSAF is a web service application fabric. WSAF is designed to be able to create web applications based on defined specifications. WSAF is based on a simple way to send requests (using URLs) and handle result (Simple XML format). It generates a set of HTML pages from the specification and some forms to test the application. WSAF generates Java code to invoke the web application and to develop it. It also generates WSDL, unit tests and stubs. WSAF detects if the parameters are conform to the specification. WSAF includes concepts like load balancing, fail over, logging, security, properties and statistics. WSAF accepts protocols: REST, SOAP [4], XML-RPC, JSON [8], JSON-RPC, and Front-end Framework. Designing specification of WSAF is shown in Fig. 1. web services are the most important ingredients in any cloud computing application. A cloud based application can be fabricated using efficient web services. Therefore, there is a need of efficient technique to design and develop good web services to suit the requirements of cloud paradigm.

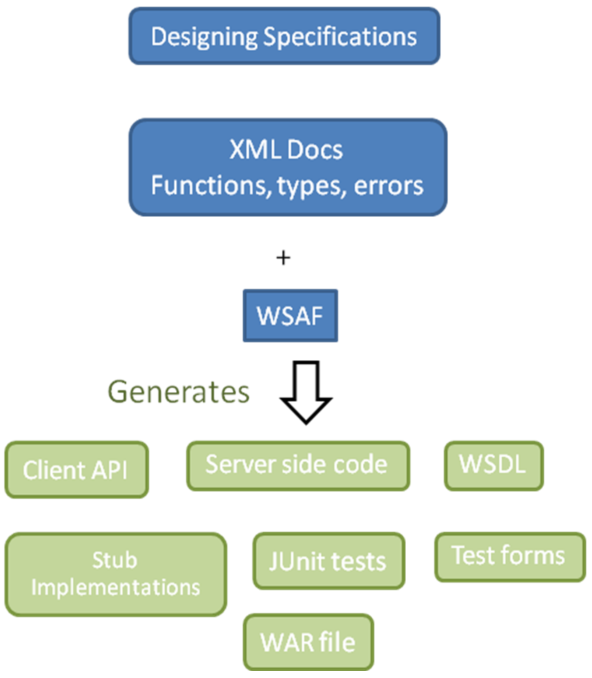


Fig. 1. Designing descriptions of WSAF

Runtime descriptions are shown in Fig. 2. All web applications take individual HTTP requests and build appropriate responses. This can be handled in a variety of ways, somewhat dependent on the server platform. The most popular overall design pattern of web application frameworks is Model-View-Controller (MVC).

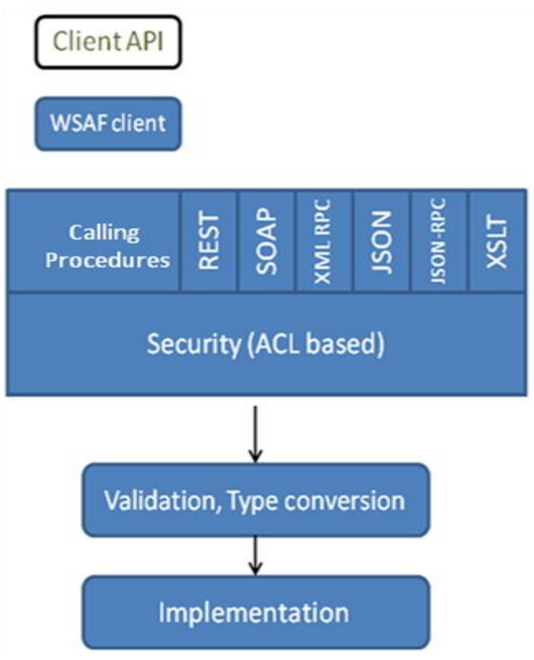


Fig. 2. Runtime description of WSAF

The initial code of the framework [13, 16], or the platform itself, valuates the URL and passes responsibility to the appropriate application controller. The controller then performs any necessary actions with the application's model and then lets the view build the actual response content.

DEVELOPING APPLICATION (APP) USING WSAF

We demonstrate here the development procedure of Factorial web service. This web service returns the factorial of a given integer. Following steps need to be followed for developing this web service:

Write definitions

An application (APP) is always part of one project. An APP can contain many functions (operations). Every function is again part of one APP. We created a directory named MyApps as a working directory. This directory contains all files for the project. In this directory, we created a file named wsaf-project.xml

```
<?xml version="1.0"?>
<!DOCTYPE project PUBLIC "-//WSAF//DTD
WSAF Project 2.3//EN"
"http://wsaf.org/dtd/wsaf-project.dtd">
<project name="MyApps" domain="com.sws">
<api name="factapp">
<impl />
<environments />
</api>
</project>
```

Next we created APP “factapp” in the application. Since an APP should contain at least one function, user will be requested to provide the name and description of a first function.

As a sample, we are presenting herewith contents of a function CalculateFactorial.

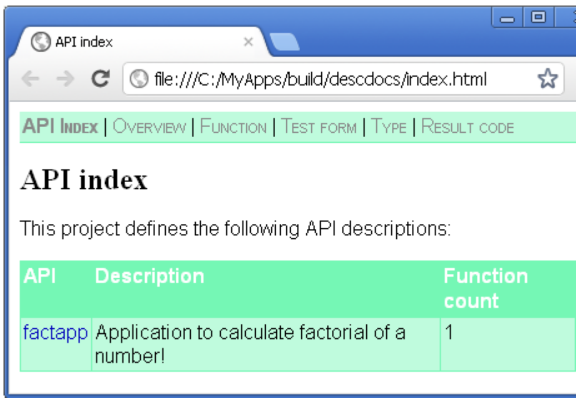
```
<?xml version="1.0" encoding="UTF-8"?>
<function name="CalculateFactorial">

<description>calculates and return
factorial...</description>
```

```
<input>
  <param name="number"
   required="true" type="_int16">
    <description>Input example
    </description>
  </param>
</input>
<output>
  <param name="outputInt64"
   required="true" type="_int64">
   <description>output parameter...
   </description>
  </param>
</output>
</function>
```

The input parameter name is "number" and the output parameter name is "outputInt64".

Generation of various codes

With this definition our system will generate HTML documentation with test forms; server-side code, with Javadoc documentation; client-side code, with Javadoc documentation; an empty-shell web application in the form of a WAR file. Various screen shots of the generated codes in browser are shown in Fig. 3, 4, 5 for factapp API. Example shows the successful calculation of 5!



API index

file:///C:/MyApps/build/descdocs/index.html

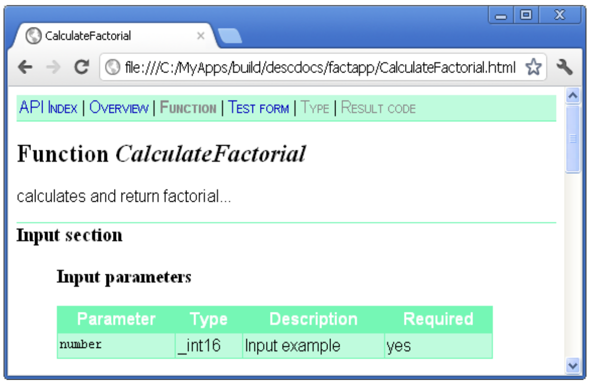
API Index | Overview | Function | Test form | Type | Result code

API index

This project defines the following API descriptions:

API	Description	Function count
factapp	Application to calculate factorial of a number!	1

Fig. 3. APP overview



CalculateFactorial

file:///C:/MyApps/build/descdocs/factapp/CalculateFactorial.html

API Index | Overview | Function | Test form | Type | Result code

Function *CalculateFactorial*

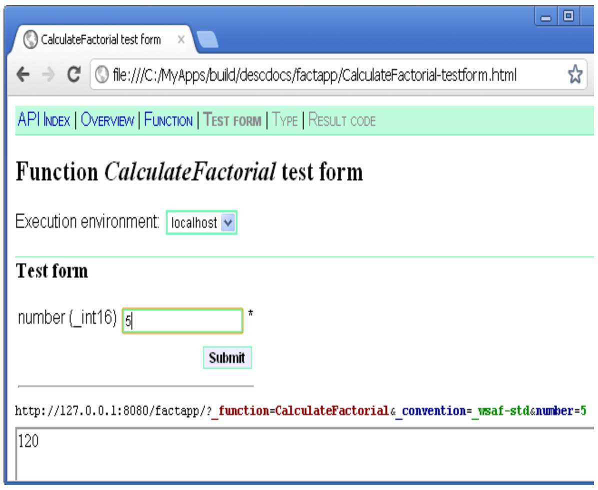
calculates and return factorial...

Input section

Input parameters

Parameter	Type	Description	Required
number	_int16	Input example	yes

Fig. 4. Screen shot of Function CalculateFactorial



CalculateFactorial test form

file:///C:/MyApps/build/descdocs/factapp/CalculateFactorial-testform.html

API Index | Overview | Function | Test form | Type | Result code

Function *CalculateFactorial* test form

Execution environment: localhost

Test form

number (_int16) 5 *

Submit

```
http://127.0.0.1:8080/factapp/?_function=CalculateFactorial&_convention=_wsaf-std&number=5
```

```
120
```

Fig. 5. Output of web service

In same way we developed Fibonacci web services and String services. Fibonacci web service returns i^{th} element in Fibonacci series (1, 1, 2, 3, 5, 8, 13, ...) String services can be used for performing various operations on string like reverse, changing case, concatenation of two strings. These 3 web services (Factorial, Fibonacci and string) are implemented in WSAF, Axis 2 [1] and JAX-WS [10].

EXPERIMENTS PERFORMED

Testing

We tested and compared performance of our framework, WSAF, with some other frameworks like Axis 2 [1] and JAX-WS [10]. Following software was used:

- WSAF
- Axis 2

- NetBeans 6.8 with Glasfish V2+Java EE+JAX-WS 2

Environment for test:

- Windows XP Professional on both server as well as on client
- JDK 1.6.0
- Dell Inspiron with Intel Core 2 Duo CPU @ 2.00 GHz and 4 GB RAM (Used for publishing web services)
- One Compaq Laptop Presario C700 with Intel Core 2 Duo CPU @ 2.00 GHz and 2 GB RAM (Used as a client)
- These two computer connected using cross over cable
- Software started: NetBeans 6.8, Ant Commander, WSAF and Google Chrome browser.

We used WSDL files generated by 3 frameworks.

Results obtained

We measured Round Trip Time (RTT) in msec of different web services using following Java code fragment:

```
long start = System.currentTimeMillis();
<...call to web server ...>
...after receiving response from server...

System.out.println("Time taken: "+
System.currentTimeMillis() - start)) + "ms");
```

We run the web services five times and calculated the average value. These values are shown in Table 1 and graphically plotted in Fig. 6.

Table 1. Comparison of different Application Frameworks

	Round Trip Time of different web services in msec (Average of 5 runs)		
web service	**WSAF**	**Axis 2**	**JAX-WS**
Factorial	22	32	38
Fibonacci	32	44	54
String library (to uppercase)	18	19	27

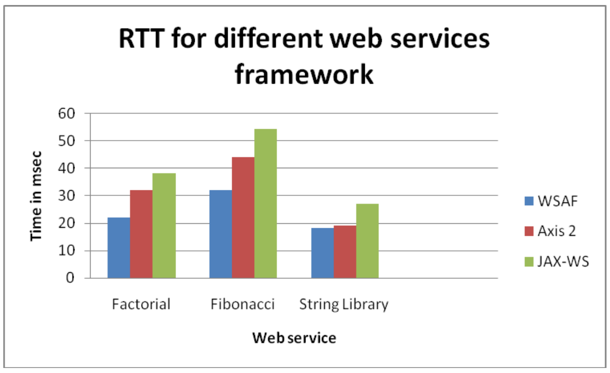


Fig. 6 Round Trip Time for different frameworks

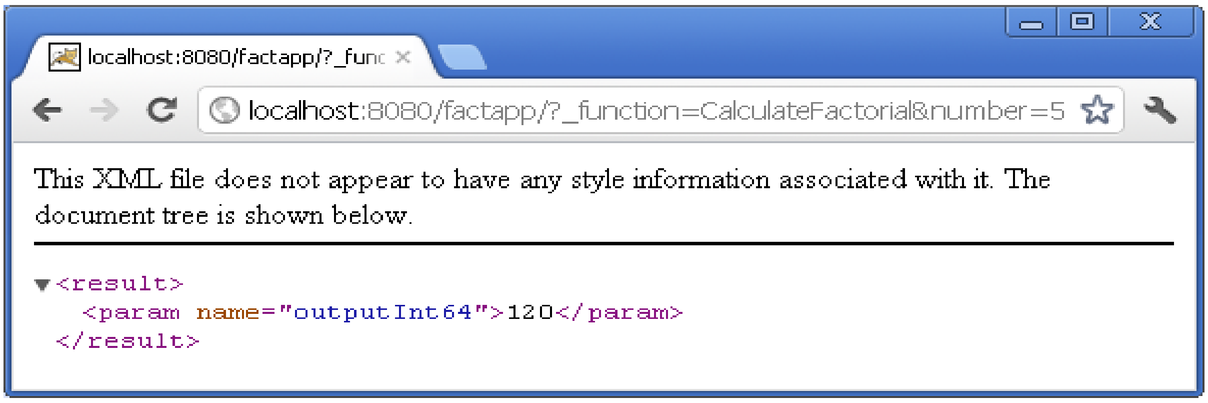
Every web service can be used by its WSDL (web service definition language). WSDL is a service contract between producer and consumer of the service. The WSDL document is useful for both creating and executing clients against a web service. We have tested the deployed factorial web service in a Google Chrome browser. Automatically generated WSDL document is shown below.

http://localhost:8080/factapp/?_function=_WSDL

```
<definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="urn:factapp" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soapbind="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/" name="factapp"targetNamespace="urn:factapp">
<!--
WSDL generated by WSAF 3, on 2012.06.04 18:58:08.406. -->
<types>
<xsd:schema targetNamespace="urn:factapp">
<xsd:element name="CalcFactRequest">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="number" type="xsd:short"
 minOccurs="1">
<xsd:annotation>
<xsd:documentation>Input
example</xsd:documentation>
</xsd:annotation>
</xsd:element>
```

```
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="CalcFactResponse">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="outputInt64" type="xsd:lo
ng" minOccurs="1">
<xsd:annotation>
<xsd:documentation>output parameter
...</xsd:documentation>
</xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</types>
<message name="CalcFactInput">
<part name="parameters" element="tns:CalcFa
ctRequest"/>
</message>
<message name="CalcFactOutput">
<part name="parameters" element="tns:CalcFa
ctResponse"/>
</message>
<portType name="factappPortType">
<operation name="CalcFact">
<documentation>Calculates
factorial</documentation>
<input name="CalcFactInput" message="tns:Ca
lcFactInput"/>
<output name="CalcFactOutput" message="tns:
CalcFactOutput"/>
</operation>
</portType>
<binding name="factappSOAPBinding" type="
tns:factappPortType">
<documentation>Calculates and returns
fact....</documentation>
<soapbind:binding style="document" transport=
"http://schemas.xmlsoap.org/soap/http"/>
<operation name="CalcFact">
<documentation>Calculates
factorial</documentation>
<soapbind:operation soapAction=""/>
<input name="CalcFactInput">
<soapbind:body use="literal"/>
</input>
<output name="CalcFactOutput">
<soapbind:body use="literal"/>
</output>
</operation>
</binding>
<service name="factappService">
<port name="factappPort" binding="tns:factapp
SOAPBinding">
<soapbind:address location="http://127.0.0.1:80
80/factapp/?_convention=_wsaf-soap"/>
</port>
</service>
</definitions>
```

The generated .war file can be copied in \webapps directory in Tomcat installation to deploy the web service. Fig. 7 shows the web service returining the answer of 5! Using Tomat.


localhost:8080/factapp/?_func
localhost:8080/factapp/?_function=CalculateFactorial&number=5

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<result>
   <param name="outputInt64">120</param>
 </result>
```

Fig. 7. Output of web service returning 5! using Tomcat

Analysis of Result

We observed that for all developed and tested web services, web services developed with WSAF requires less time as compared with developed by other two frameworks viz. Axis 2 and JAX-WS. Latency time in case of WSAF is decreased by at least 20% - 30%. WSAF is giving better result because we used SAX XML parser and SOAP document-literal encoding was used.

CONCLUSIONS

To encourage good design and make programming easier one can use an application framework. Framework offers a layer of abstraction on top of complex tools. A good SOA framework should encourage use of XML schema libraries and reuse schema across WSDL documents. We developed a web service Application Framework, WSAF, which is an open-source web Services framework supporting HTTP protocols such as REST, SOAP, XML, JSON, JSON-RPC and more. From the specifications written in simple XML, WSAF generates the Client API (.jar), the Java server code template (.war), the WSDL and the documentation of the specification in HTML. We found that for tested WSDLs performance of our framework is better than Apache CFX and JAX-WS framework. The performance depends more on the hardware, the network, Internet connectivity, the Servlet container where web services are published.

ACKNOWLEDGMENT

Authors wish to thank staff members of Department of Computer Science, Shivaji University for providing all necessary infrastructure as well as we express our gratitude towards Principal, Thakur College of Science and Commerce, Mumbai for her kind support and her excellent library facility.

REFERENCES

1. Apache Axis 2.x., http://ws.apache.org/axis2
2. Apache CXF, http://cxf.apache.org/ , Accessed on 10th Feb 2012
3. Auletta, V.; Blundo, C.; De Cristofaro, E.; Raimato, G.; , "A Lightweight Framework forWeb Services Invocation over Bluetooth," web Services, 2006. ICWS '06. International Conference on , vol., no., pp.331-338, 18-22 Sept. 2006
4. Ben Shil, A.; Ben Ahmed, M.; Additional Functionalities to SOAP, WSDL and UDDI for a Better web Services' Administration, 2nd International Conference on Information and Communication Technologies, 2006. ICTTA '06. Volume : 1, pp: 572 - 577
5. Bobby Bissett, Building JAX-WS 2.0 Services with NetBeans 5.0 and GlassFish, http://jax-ws.java.net/articles/jaxws-netbeans/glassfish.html, Accessed on 10th Feb 2012
6. Cao Hong-Hua; Ying Shi; Cui Hua; Xiao Yang; , "Towards a Framework for Designing, Deploying and Executing Semantic web service-Based Process," Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on , vol., no., pp.1-4, 12-14 Oct. 2008
7. Chaoying Ma; Bacon, L.; Petridis, M.; Windall, G.; , "Towards the Design of a Portal Framework for web Services Integration," Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and web Applications and Services/Advanced International Conference on , vol., no., pp. 163, 19-25 Feb. 2006
8. Crockford D. "The application/json Media Type for JavaScript Object Notation (JSON)." The Internet Engineering Task Force (Network Working Group) RFC-4627, July 2006, http://tools.ietf.org/html/rfc4627
9. Davanum Srinivas, Paul Fremantle, Amila Suriarachchi, and Deepal Jayasinghe, web Services are Not Slow, Published on WSO2 Oxygen Tank, 2007, http://wso2.org/print/588, Accessed on 15th Feb 2012

10. Eckstein, and Robert Rajiv Mordani. “Introducing JAX-WS 2.0 with the Java SE 6 Platform, Part 2,” November 2006.http://java.sun.com/developer/technical Articles/J2SE/jax_ws_2_pt2/Monson-Haefel, Richard. J2EE web Services. Addison-Wesley Professional, ISBN 0130655678, October 2003.
11. Gomez-Perez, A.; Gonzalez-Cabero, R.; Lama, M.; , "ODE SWS: a framework for designing and composing semantic web services," Intelligent Systems, IEEE , vol.19, no.4, pp. 24- 31, Jul-Aug 2004
12. Haidar, A. N.; Abdallah, A. E.; Abstractions of web Services, 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009, pp: 182 - 191
13. Jen-Yao Chung; An industry view on service-oriented architecture and web services, IEEE International Workshop on service-Oriented System Engineering, 2005. SOSE 2005. pp:59
14. Kawahara, Y.; Kawanishi, N.; Ozawa, M.; Morikawa, H.; Asami, T.; , "Designing a Framework for Scalable Coordination of Wireless Sensor Networks, Context Information and web Services," Distributed Computing Systems Workshops, 2007. ICDCSW '07. 27th International Conference on , vol., no., pp.44, 22-29 June 2007
15. Kohsuke Kawaguchi, JAX-WS RI 2.1 benchmark details, http://weblogs.java.net/blog/kohsuke/archive/2007/02/jaxws_ri_21_ben.html, Accessed on 10th Feb 2012
16. Kumar, A.; "Distributed system development using web service and Enterprise Java Beans," Services Computing, 2005 IEEE International Conference on , vol.2, no., pp. xiii vol.2, 11-15 July 2005
17. Li Zhang; , "Requirement engineering for web applications," web Site Evolution, 2008. WSE 2008. 10th International Symposium on , vol., no., pp.1, 3-4 Oct. 2008
18. Rama Pulavarthi, Monitoring SOAP Messages Made Easy With JAX-WS RI 2.0.1, http://weblogs.java.net/blog/ramapulavarthi/archive/2006/08/monitoring_soap.html, Accessed on 10th Feb 2012
19. Rama Pulavarthi, Useful Goodies for web service Developers in JAX-WS 2.1 RI, http://weblogs.java.net/blog/ramapulavarthi/archive/2007/02/useful_goodies.html, Accessed on 10th Feb 2012
20. Sandy Carter, “The New Language of Business: SOA & web 2.0”, IBM Press, 2007
21. Siblini, R.; Mansour, N.; Testing web services, The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005. pp: 135
22. SOAP Version 1.2 Part 0: Primer. W3C Recommendation, June 24 2003, www.w3.org/TR/soap12-part0
23. Tanaka, M.; Ishida, T.; Murakami, Y.; Morimoto, S.; , "service Supervision: Coordinating web Services in Open Environment," web Services, 2009. ICWS 2009. IEEE International Conference on , vol., no., pp.238-245, 6-10 July 2009
24. Thomas Erl, “service-Oriented Architecture: Concepts, Technology, and Design”, Pearson Education, Inc., 2007.
25. Tsai, W.T.; Paul, R.; Yamin Wang; Chun Fan; Dong Wang; Extending WSDL to facilitate web services testing, Proceedings of 7th IEEE International Symposium on High Assurance Systems Engineering, 2002, pp: 171 - 172
26. Walmsley, Priscilla. Definitive XML Schema. Prentice-Hall PTR, ISBN 0321146182, December 2001.
27. web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft, August 3, 2005. www.w3.org/TR/wsdl20/
28. web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. W3C Working Draft, August 3, 2005. www.w3.org/TR/wsdl20-adjuncts