May 08 | Volume 1 |Issue 11 | Free

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING

Introduction to Web Services

XML IN WEB SERVICES

WEB SERVICE FRAMEWORKS

WEB SERVICES COMPONENTS

WEB SERVICE PROTOCOLS

WEB SERVICES BENEFITS

India's Cheapest web Service Provider

RoseIndia

OUR SERVICES

- ERP Soluiotns
- Software Solutions
- Web Development
- Web Designing
- Web Redesigning
- Domain Registration
- Web Promotion
- SEO
- Article Writing
- Blog Writing
- News Writing
- SEO Copywriting
- Technical Documentation
- E-Commerce Solutions
- CRM
- Outsourcing

Extend your reach with our solutions...

http://www.roseindia.net/services

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING

April 2008 Volume I Issue X

"Optimism with determination lets you hit the goal harder"

Published by

RoseIndia

JavaJazzUp Team

Editor-in-Chief

Deepak Kumar

Sr. Editor-Technical

Ravi Kant
Noor-En-Ahmed

Graphics Designer

Santosh Kumar

Register with JavaJazzUp

and grab your monthly issue

"Free"

Editorial

Dears Readers,

We are back here with the May 2008 special issue of Java Jazz-up. This current May edition is designed as '**Web Services**' special issue. From the last issue we decided to release our magazine as special issue on a single technology. Our last issue was designed as 'Ajax' special issue. The issue really proved very popular and valuable among our readers. We hope the issue will be more popular and valuable for all readers.

Web Service is one of the most popular and useful techniques in the landscape of software development. Web Services allows different applications to talk to each other and share data and services among them.

Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services are used to make the application platform and technology independent. In the issue, our readers will learn reating and deploying our own web services with xis2, Apache's Web services framework.

In addition as per the growing popularity of Web Services we have included two relevant tutorials on Web Services published in IBM developerWorks with consent. We are thankful to the team of IBM for granting permission regarding the same.

To make it interesting for readers we have categorized each section with different colors and images that would certainly lure readers while reading technological stuffs. We are providing it in PDF format that you can view and even download it as a whole or a part. This PDF version provides you a different experience.

Please send us your valuable feedback about this issue and participate in the reader's forum with your problems and issues concerned with the topics you want us to include in our next issues.

Editor
Deepak Kumar
Java Jazz up

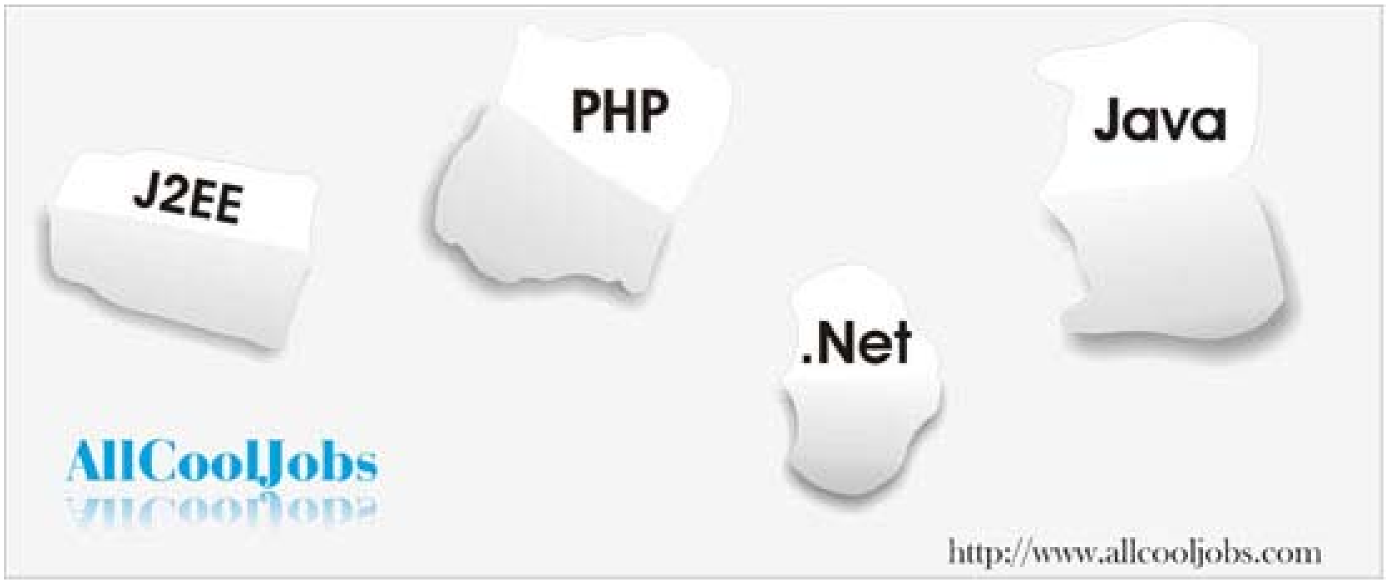
Content

Web Services

Web services are web-based application that translates your application into browser-based application using XML, SOAP, WSDL and UDDI. World Wide Web Consortium (W3C) has defined it as a software system designed to communicate machine to machine over a network without time-consuming custom coding. These applications usually allow organizations to communicate data without interfering in one another's system.

In web service, XML tags the data within the message; SOAP is a protocol specification that transfers the data, the Web Services Description Language (WSDL) defines the description of available service while Universal Description Discovery and Integration (UDDI) provides the list of available services.

Web service does not provide the Graphics User Interface by default but the programmers can add it to provide it to the clients by making it as a specific web service application.



Why to use Web Services

Interoperability

Web services have the tendency of higher Interoperability that means web services allows the companies to communicate each other on the basis of business. Technically sound users will find a way to access the desired information, which was not early so easy. For accessing data in the missing web services, users would need to resort the methods.

Simplicity

The system of web services is so simple that even non-programmers can assemble web service-based integration solutions through using the integration platform tools. Besides these, the portal's application server can easily devour SOAP messages. This made the integration easier, faster, and cheaper.

Reusability

After making the connection and exposing the web service, it becomes easy to connect to other applications.

Cost Effective

Web services are cost effective and can help to manage costs. It integrates easily with other companies and also permits to measure your business. Web application also provides the services like language interpreter, currency converter and in weather reports.

How to use?

Due to inbuilt with a set of tools web services can be used several means but the three most common styles of use are RPC, SOA and REST.

Remote Procedure Calls (RPC)

Remote Procedure Call (RPC) is a protocol, which a program from one computer can use to request any service to another program of different computer system. The program performs the request within the network without assessing the network detail
RPC uses the client/server model in which, the requesting program is known as a client model and the service-providing program is known as the server. RPC is a synchronous operation functions like a regular or local call. It runs successfully when its requesting program is on halt until the results of the remote procedure are returned. In spite of this, lightweight processes or threads, which shares the same address space performs multiple RPCs operations concurrently. The Basic unit of RPC Web Service is the WSDL operation.

Remote Procedure Call compiles into an executable program when a program statements uses it. During compilation, a stub is attached in compilation code, which acts as the representatives of the remote procedure code. When the program runs and issues the procedure call, the stub receives the request and forwards it to a client runtime program in the local computer.

Why to use Web Services

Service-Oriented Architecture

Web services can also be used to implement Service-oriented architecture (SOA). SOA is a collection of services that mutually communicate one another, where the basic unit of communication is a message, rather than an operation. It is also called message-oriented services.

XML schema that is also known as XSD usually defines the service, which SOA communicates. Mostly the bigger software vendors and industry analysts support SOA service and maintain it by registry that functions like a directory listing. Applications can found and access the services.

Representational state transfer

Representational state transfer (REST) is a pattern of software architecture used for getting information content from a Web site by reading a designated Web page containing an XML file, which describes and includes the desired content. Sometimes, REST has been used to specify any simple interface that passes on domain-specific data over HTTP without any extra messaging layer like SOAP or session tracking via HTTP cookies.

Role of XML in Web Services

Because of its innovative characteristics, web services are called the nexgen (next generation) web technology. For developing any application or website, the programmers have to create lots of coding, generates many files which produces difficulties in handling those data if it is done manually. The developers have to categorise those data according to its category like data for users, data for administrators, data for login and data for device etc. These functions consume lots of time and manpower and resulting the losses in the productivity.

eXtensible markup language Solve the such problems by enabling the developers to assign data to be exchanged between PCs, smart devices, applications, and Web sites. An arranged and clustered data according to format and style definitions can be easily organized, programmed, edited, and exchanged between any Web sites, applications, and devices.

XML makes communication between two or more applications, similar like users communicate to applications. XML convene this facility and developers can call web services from JavaScript.

Web Service Components

Web service has three main components:

i) **SOAP**
ii) **WSDL and**
iii) **UDDI**

i) SOAP

Simple Object Access Protocol (SOAP) is a communication protocol that communicates between applications without understanding the configuration and programming of any operating system and programming language. A program from one computer system can communicate with other program on another computer system within and beyond network using SOAP protocol on the Internet.

SOAP can run on any operating system and need not to assess the programming code. It is based on XML and HTTP and runs using the nexus of both on to the Internet. SOAP is simple and extensible and allows the user to woo firewall. It is a mechanism generally used for information exchange over the Internet.

Since the emerging of web protocols that runs on all major operating system platforms, HTTP and XML solve the problem of running within different operating systems and in different programming language. Similar like this, SOAP also describes how to encode an HTTP header and an XML file to make communication between two computers and call any program from one's to others. SOAP will also describe the criteria of returning the response of the program.

Microsoft, DevelopMentor, and Userland Software have jointly developed the SOAP and promoted it as a standard interface to the Internet Engineering Task Force (IETF). It is a little bit analogous to the Internet Inter-ORB Protocol (IIOP). IIOP is a protocol that is a part of the Common Object Request Broker Architecture (CORBA). The program calls through SOAP protocol has a benefit of that it can interact firewall servers, which screen out the request other than those for known applications using designated port mechanism.

ii) WSDL

Web Services Description Language (WSDL) is an XML-based language written in XML and used for describing Web services and the process of accessing it. It is also used for locating web services. WSDL specify the network services, location service and operation services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

The operations and messages are illustrated conceptually, and then leap to a concrete network protocol and message configure to identify an endpoint. The integrated concrete endpoints are tied up with abstract endpoints (services). The WSDL navigates for individuals and other business to access the service

The Web Services Description Language (WSDL) is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services by electronic means. WSDL is the base of the Universal Description, Discovery, and Integration (UDDI) that facilitate businesses to be listed themselves and their services on the Internet.

Web Service Components

WSDL is consequent from SOAP of Microsoft and NASSL (Network Accessible Service Specification Language) of IBM. WSDL has replaced Simple Object Access Protocol (SOAP) and IBM's (NASSL) both for describing business services in the UDDI registry. To check the available function on server, a client program can read the WSDL connected with Internet.

UDDI

Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based directory for storing information about web services and for businesses worldwide to list themselves on the Internet. UDDI uses WSDL to describe interface to web services that communicates through SOAP. It is also used for discovering the business or industry of all size.

Earlier, before the evaluation of UDDI, it was not possible to reach their customers and partners and their products to buy or sale on the Internet. It was written in August 2000 and was introduced in the public as a beta version in November 2000. Microsoft, IBM, and Ariba spearheaded have jointly founded UDDI. UDDI was originally proposed as a core Web service standard and later integrated into the Web Services Interoperability (WS-I) standard as a central pillar of web services infrastructure.

UDDI can define how to enable commerce for the desired business. It is also used for getting new customers, enhancing the navigation of current customers, extending offerings and expanding market reach, solving customer related problems and illustrating the services and business processes programmatically in a single, open, and secure environment.
A UDDI business consists of three components: White pages, Yellow Pages and Green Pages. White pages are contains address, contact, and known identifiers, Yellow pages are known for industrial categorizations based on standard taxonomies and green pages reflect technical information about services exposed by the business.

Web service Frameworks

Web service Frameworks:

There are many frameworks for web services. Here is the list of some frameworks available:

Name	**Plateform**
Apache Axis	Java/C++
Apache Axis2	Java/C
Java Web Services Development Pack	Java
XFire	Java
JSON-RPC-Java	Java
Web Services Interoperability Technology	Java
Web Services Invocation Framework (WSIF)	Java
AlchemySOAP	C++
csoap	C
gSOAP	C/C++
NuSOAP	PHP
Windows Communication Foundation	.Net
ActionWebService	Ruby (on Rails)

Web service Protocols

Web service protocols:

Here is the list of some protocols used for Web Services:

Name	**Acronym For**
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery, and Integration
WSDL	Web Services Description Language
REST	Representational State Transfer
WSCL	Web Services Conversation Language
BEEP	Blocks Extensible Exchange Protocol
BPEL	Business Process Execution Language
XML-RPC	XML Remote Procedure Call

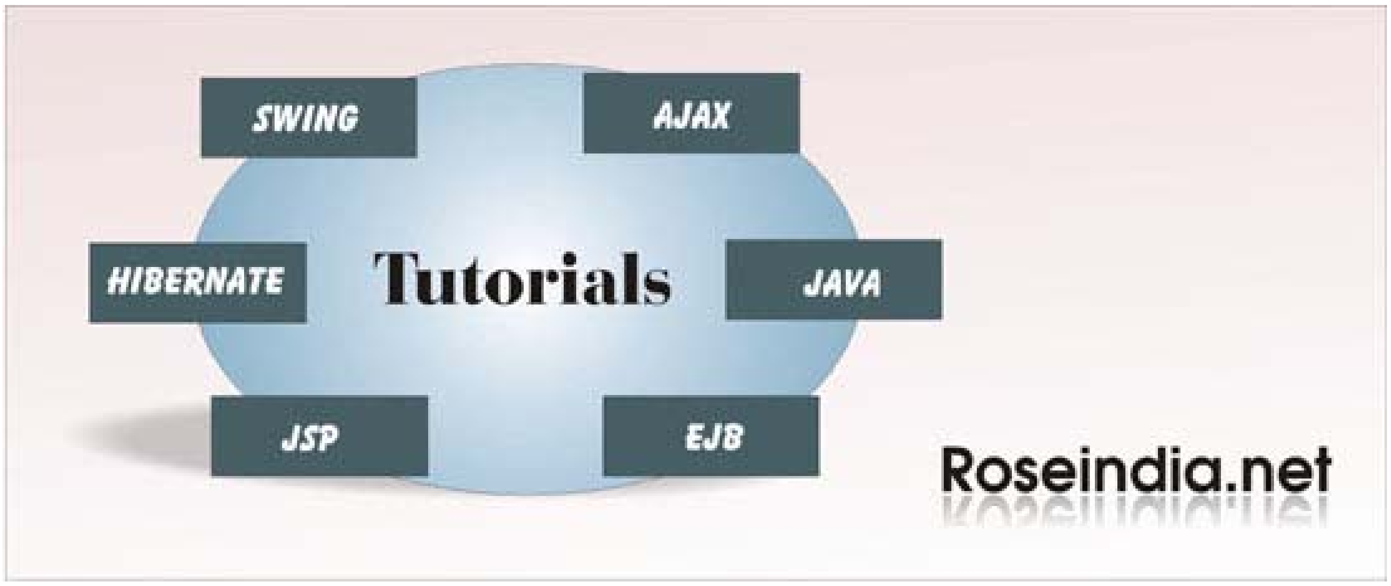
Come to get right Web Hosting
http://www.roseindiahosting.com/

What is axis2?

What is axis2?

Axis2 is Apache's Web services framework with two implementations Apache Axis2/Java and Apache Axis2/C. As discussed earlier Web services are very good means of inter application communication. This communication is possible with the help of XML. SOAP protocol defines the schema for a message to be sent when using web services. SOAP engine is required to create and interpret SOAP messages. AXIS2 is such a SOAP engine.

Apache Axis2 supports both SOAP 1.1 and SOAP 1.2 and also has integrated support for the widely popular REST-style of web services. Apache Axis2 is more efficient, modular, and scalable than the older version Axis1.x. Axis2 provides many new features with enhanced functionality. Some of the main features are Speed, Low memory foot print, AXIOM, Hot Deployment, Asynchronous Web services, MEP Support, Flexibility, Stability, Component-oriented Deployment, Transport Framework, WSDL support, Add-ons, Composition and Extensibility.



Installing Axis2:

Installing Axis2:

- Download “**axis2-1.4.zip**” (Standard Binary Distribution) from to http://ws.apache.org/axis2.
- Unzip it into c:\axis2-1.4 folder.
- Set environment variable AXIS2_HOME to the path name of axis2 directory i.e. “**c:\axis2-1.4**”.

Make sure you have installed JDK(version 1.4 or later). Set the environment variable JAVA_HOME to the pathname of the directory into which you installed the JDK.

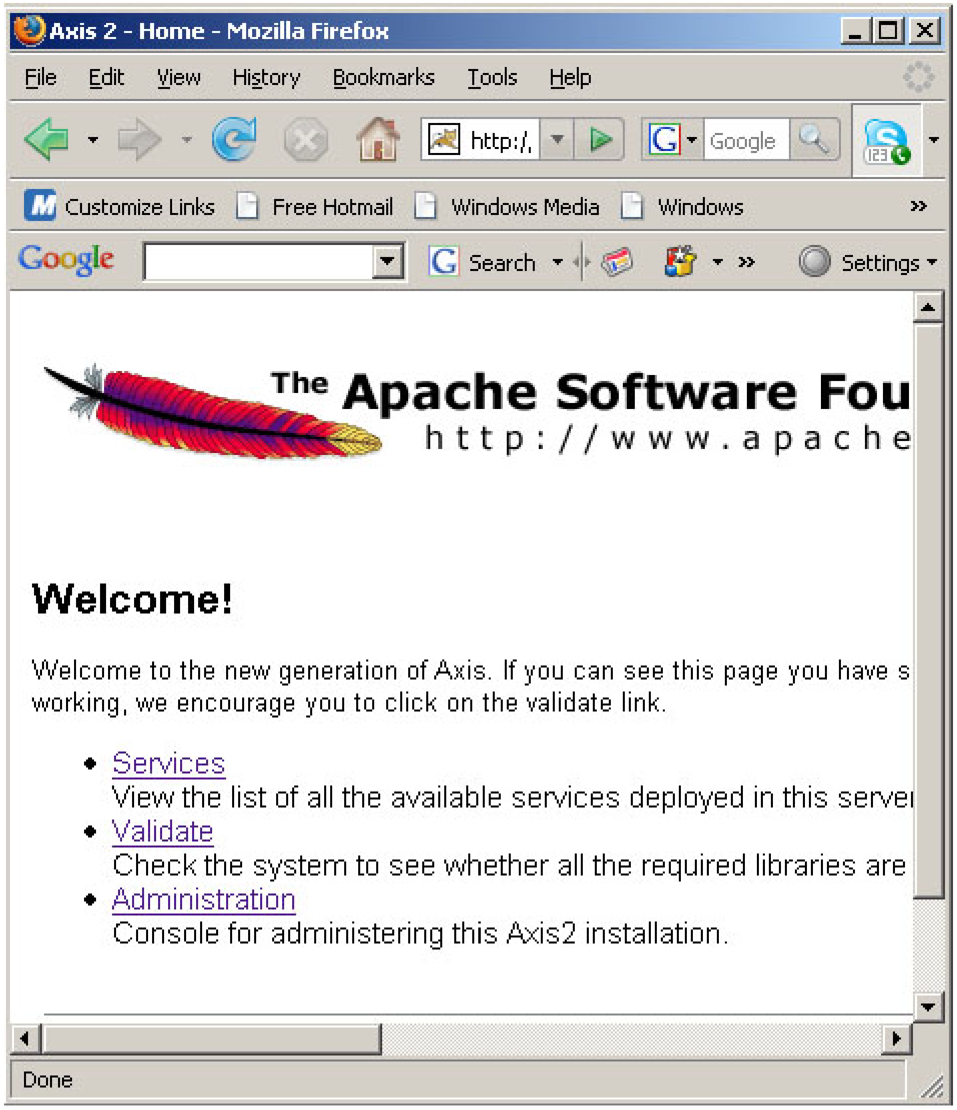
```
<%
   int counter=0;
   ListIterator litr = webUIService.getTopLevel
  Categories ().listIterator();
while(LITr.hasNext())
{
  if((counter%4)==0)
{
  out.print("<tr>");
 }

%>

<td valign="top" align="left" class="dt-in-txt" style="padding-left:10px>
```

Developing Simple Web Service

In this section, we will learn developing web services with Axis2 by an easy example "**HelloWorldService**" and deploying it on the tomcat web server.

Step 1: Build Axis2 Web Application: (axis2.war)

1. Download and install Apache Ant (version 1.6.5 or later) from http://ant.apache.org.
2. Go to "**webapp**" folder of the downloaded axis2. You will get "**build.xml**" file.
3. Open the command prompt and go up to the "**webapp**" directory and run "**ant**" command.
4. The above command will generate "**axis2.war**" in the "**dist**" directory of axis2.
5. Copy the "**axis2.war**" into the "**webapp**" folder of the tomcat web server.
6. Restart the server.
7. Open the browser with url http://localhost:8080/axis2 in address bar. It displays the page as below:


Axis 2 - Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http:/,
Google
Customize Links Free Hotmail Windows Media Windows
Google Search Settings
The Apache Software Fou
http://www.apache
Done

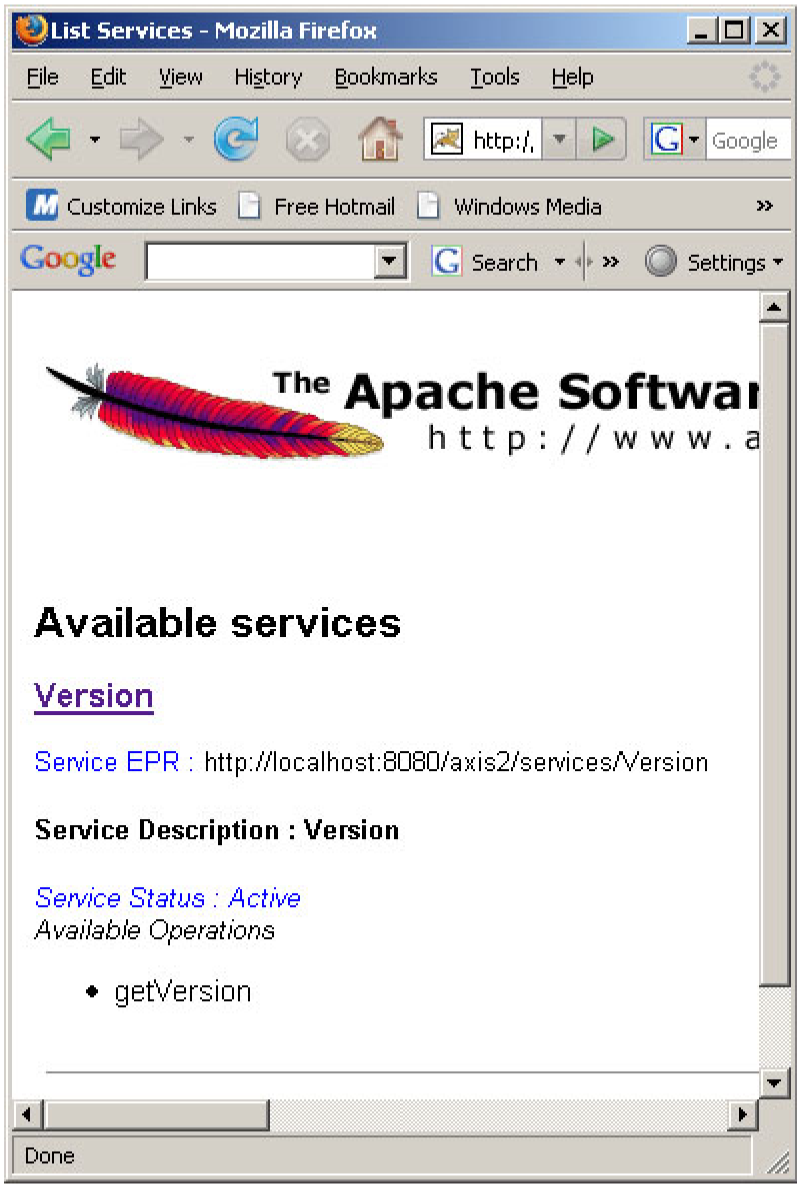
Welcome!

Welcome to the new generation of Axis. If you can see this page you have s
working, we encourage you to click on the validate link.

- Services
  View the list of all the available services deployed in this serve
- Validate
  Check the system to see whether all the required libraries are
- Administration
  Console for administering this Axis2 installation.

8. Click the link “**validate**”. It displays Axis2 happiness page like below. This page ensures that everything is correct. If it is not so then war file is not properly installed. **(Image file axis2happiness.bmp)**

9. Clicking the link “**Services**” displays the page like below where the service named “**Version**” is displayed. This service is available already with the war file. When you create your own web services and deploy it on server, you can see all those web services in this page.



Available services

Version

Service EPR : http://localhost:8080/axis2/services/Version

Service Description : Version

Service Status : Active
Available Operations

- getVersion

Developing Simple Web Service

Step 2: Create a simple Java class: (HelloWorldService.java)

Now we need to create a simple class as a web service. In this web service, we can create different methods to perform different operations. Different clients when consuming this web service can use these operations.

For this, in this example, we have created java file named "**HelloWorldService.java**". This file has "**HelloWorldService**" class and the class has "**sayHello()**" method. The method takes a single String parameter and simply returns the same String with "**Hello**" text added as a prefix.
HelloWorldService.java

```
/**
* The service implementation class
*/
package examples.axis2;
public class HelloWorldService {
    public String sayHello(String name) {
   return "Hello "+name+" !";
    }
}
```

Save this file as **"HelloWorldService.java"** in "**examples\axis2**" folder and compile it.

Step 3: Creating the service descriptor: (services.xml)

Our next step is to inform the Axis engine about this service. For this we need to create "**services.xml**" file containing the description of different web services.

services.xml:

```
<service name="HelloWorldService">
    <description>
        This service is to say hello to the user.
    </description>
   <parameter name="ServiceClass">examples.axis2.HelloWorldService</parameter>
   <operation name="sayHello">
        <messageReceiver  class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />
    </operation>
</service>
```

Save this file as services.xml in "META-INF" folder.

Information of each service is described in a single <**service**> element. In this example service name is defined as "**HelloWorldService**".

You can take multiple service elements for multiple services i.e. include group of services. All these services are enclosed with in <**ServiceGroup**> element.

Developing Simple Web Service

Within service element there is parameter element, which specifies the name of the java class as a web service implementation. In the above example, parameter element with name attribute as "**ServiceClass**" is provided the name of the java class "**examples.axis2.HelloWorldService**" with full path.

Within service element there is another element <**operation**>, which specifies the name of operation i.e. the name of the method created in the web service implementation class. This element has child element <**messageReceiver**> which specifies the message receiver that is to be used for that operation. In the above example, we have set the name of the operation as "**sayHello**".

Step 4: Create the Web Service Archive: (HelloWorldService.aar)

Next step for developing web service with Axis2 is to create **".aar**" (Axis Archive) file. For our example, we have created "**HelloWorldService.aar**".

In the above steps we have created two files:

1. **"HelloWorldService.java"** file in "**examples\axis2**" folder and
2. **"services.xml"** in **"META-INF"** folder.

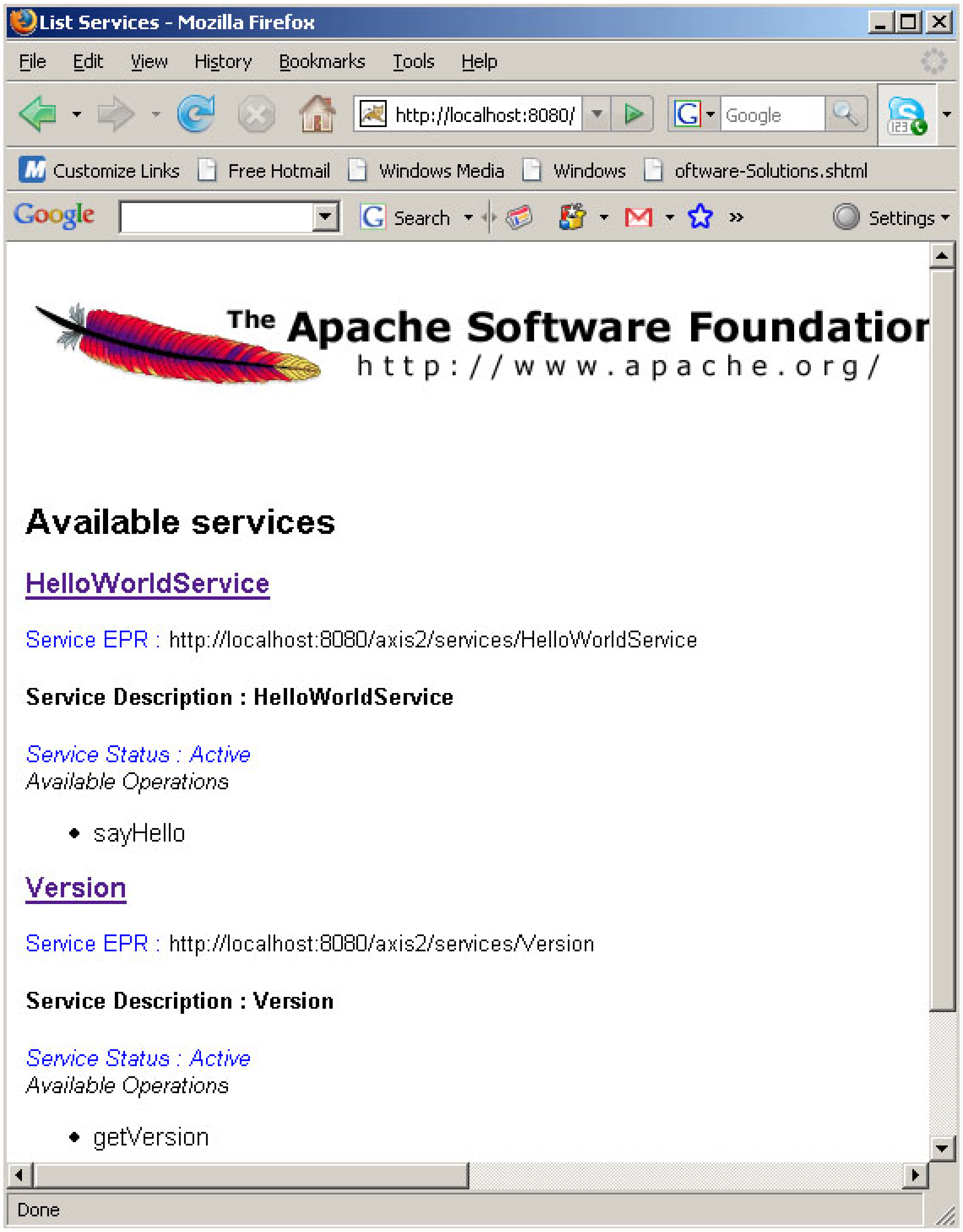
Now create **.aar** file we can use jar tool in command prompt. Navigate to the path in the command prompt where these above two folders have been saved and run the following command:

```
Jar cvf HelloWorldService.aar META-INF sample
```

This will create "**HelloWorldService.aar**" file in the current directory.

Step 5: Deploy the Web Service:

1. Drop the "**HelloWorldService.aar**" file into the "**services**" folder of **"/webapps/axis2/WEB-INF**" of tomcat.
2. Restart tomcat.
3. Open the browser with url **http://localhost:8080/axis2**. Its home page is opened.
4. Click the link "**Services**" on the home page.
5. Check the service you have deployed is available on the page i.e. you should see the page like below.


List Services - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/
Customize Links Free Hotmail Windows Media Windows oftware-Solutions.shtml
Search Settings
The Apache Software Foundatior
http://www.apache.org/
Done

Available services

HelloWorldService

Service EPR : http://localhost:8080/axis2/services/HelloWorldService

Service Description : HelloWorldService

Service Status : Active
Available Operations

- sayHello

Version

Service EPR : http://localhost:8080/axis2/services/Version

Service Description : Version

Service Status : Active
Available Operations

- getVersion

If you see the output as above, it means you have successfully deployed "**HelloWorldService**" web service on Axis2.

From the above steps you learned how to create own web services with Axis2. In the next issue you can learn how to consume web services.

Web Services benefits

Web Services benefits:

Here are the benefits of using Web Services:

1. Exposing the function on to network: A Web service can be remotely invoked using HTTP. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

2. Connecting Different Applications: Web Services allows different applications to talk to each other and share data and services among them. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services are used to make the application platform and technology independent.

3. Ease of integration: One of the major benefits is Web services' ease of integration. You will easily integrate your application with other pieces of software application i.e. with external data sources. You can run it on all kinds of machines from desktop to mainframe and from within your enterprise to external sites i.e. Web services link applications, services, and devices together.

4. Standardized Protocol: Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) use the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.

5. Low Cost of communication: Web Services use SOAP over HTTP protocol for the communication, so you can use your existing low cost internet for implementing Web Services. This solution is much less costly compared to proprietary solutions like EDI/B2B.

6. Support for Other communication means: Beside SOAP over HTTP, Web Services can also be implemented on other reliable transport mechanisms. So, it gives flexibility use the communication means of your requirement and choice. For example Web Services can also be implemented using ftp protocol (Web services over FTP).

7. Loosely Coupled Applications: Web Services are self-describing software modules, which encapsulates discrete functionality. Web Services are accessible via standard Internet communication protocols like XML and SOAP. These Web Services can be developed in any technologies (like c++, Java, .NET, PHP, Perl etc.) and any application or Web Services can access these services. So, the Web Services are loosely coupled application and can be used by applications developed in any technologies. For example, I have heard of people developing Web Services using Java technologies and using the Web Services in VB or .NET applications.

8. Web Services Sharing: These days due to complexness the business, organizations are using different technologies like EAI, EDI, B2B, Portals etc. for distributing computing. Web Services supports all these technologies, thus helping the business to use existing investments in other technologies.

9. Web Services are Self-Describing: Web Services are self-describing applications, which reduces the software development time. This helps the other business partners to quickly develop application and start doing business. This helps business to save time and money by cutting development time.

Web Services benefits

10. Automatic Discovery: Web Services automatic discovery mechanism helps the business to easy find the Service Providers. This also helps your customer to find your services easily. With the help of Web Services your business can also increase revenue by exposing their own Web Services available to others.
11. Business Opportunity: Web Services has opened the door to new business opportunities by making it easy to connect with partners.

Deploy Web services in Apache Geronimo

Walk through the process with Amazon Web services

Level: Intermediate

28 Jun 2005

Deciding on an application server to support your Web services development efforts? Meet the Apache Geronimo application server, one of the latest projects from the Apache Software Foundation. Java™ specialist Kunal Mittal introduces you to Geronimo's Web services capabilities by showing you how to write and develop standard J2EE Web services code in Geronimo. You'll learn how to consume Amazon Web services using Apache Axis as the underlying Simple Object Access Protocol (SOAP) implementation, and you'll see how to use a simple JavaServer Pages (JSP)-based client to access the Web service.

Geronimo supports Web services standards
The new Apache Geronimo project is a powerful open source J2EE application server built on top of J2EE 1.4 standards. It uses all open source implementations and will soon be J2EE compliant. Interestingly, Geronimo is built upon a collection of code gathered from many other open source projects. Geronimo uses Apache Axis and Apache Scout (see Resources) to support the following Web services standards:

- Java Specification request (JSR) 109 (implementing Enterprise Web Services 1.1)
- Java API for XML-based Remote Procedure Call (JAX-RPC)
- SOAP with Attachments API for Java (SAAJ) 1.2
- Java API for XML Registries (JAXR) 1.0

Support for these standards makes Geronimo a viable option when deciding on an application server to support your Web services development efforts. If you've already used the open source projects that make up Geronimo, the transition to Geronimo as your deployment container should come naturally. To make development and deployment of J2EE applications on Geronimo even easier, several Eclipse plug-ins are also available. These plug-ins are part of the Eclipse Web Tools Platform (WTP) (scheduled to have its 1.0 launch in July 2005). See Resources for links to the WTP and related information.

Using Amazon Web Services (AWS) as an example, you'll learn how to consume Web services with Geronimo. You'll be guided through taking the Web Services Description Language (WSDL) for AWS and using Apache Axis to build the consumer code. You'll see a simple Java class that consumes the Web service and learn how this class is invoked from a JSP file to display the results of the Web service. Finally, you'll bundle the code as a J2EE WAR file and deploy it on Geronimo.

Consuming Amazon Web Services
To consume AWS, you need to build the service consumer using Apache Axis as the SOAP implementation. (Axis is the underlying SOAP implementation that Geronimo supports.) If you have done this in the past and have the code, feel free to jump ahead.

Requirements for generating AWS code
Begin by setting up your environment. To generate the AWS consumer code, you need four things:

- Java software development kit (JDK) 1.4.2 or later. (See Resources to link to the Java Web site.)

- Apache Axis 1.1 or later. (See Resources for the Apache Axis Web link.)
- The Web Services Description Language (WSDL) for the Web service. (See Resources for a link to download the WSDL.)
- An AWS subscription ID from Amazon so you can use their Web services. (See Resources for a link to register for the free ID.)

Set up your environment
The next series of steps guides you through setting up your environment by downloading the elements listed above as follows:

- Install the JDK into C:\jdk_142_05. Set the JAVA_HOME to this directory.
- Extract Apache Axis into C:\axis1-2, and define AXIS_HOME as this directory.
- Copy the WSDL file into the AXIS_HOME directory.
- Register for the AWS subscription ID.

Now that you have the basic environment ready to go, you can start generating the code to consume the Web service.

Generate the Java code from the WSDL
First, generate the Java code that will consume the Web service from the WSDL file. Apache Axis comes with a utility called WSDL2Java that performs this task for you. To run this tool, make sure the following Java Archive (JAR) files are in your classpath. The sample setenv.bat script shown in Listing 1 does this for you.

Listing 1. Setenv.bat

```
set AXIS_HOME=c:\axis-1-2
set CLASSPATH=.
set CLASSPATH=%AXIS_HOME%\lib\axis.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\commons-discovery.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\commons-logging.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\jaxrpc.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\saaj.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\log4j-1.2.8.jar;%CLASSPATH%
set CLASSPATH=%AXIS_HOME%\lib\wsdl4j.jar;%CLASSPATH%
```

Now, from the AXIS_HOME directory, run the following command to generate the Java code:

```
java org.apache.axis.wsdl.WSDL2Java AWSECommerceService.wsdl
```

This process takes a few seconds to create a directory called com under AXIS_HOME. Now you can start creating a Web Archive (WAR) file that you will eventually deploy on Geronimo.
Create a directory called C:\amazon-client. You'll use this directory to store the code for the WAR file. Under this directory, create a directory called WEB-INF and a directory called src. Copy the com directory from your AXIS_HOME to C:\amazon-client\WEB-INF\src.
This code is the basic code for consuming a Web service. For now, you don't need to go into the details of this code.

Write code to consume the Web service
Next, write a client to the Web service. In this example, you'll write a simple Java class that calls the appropriate classes that were generated using WSDL2Java. (Tools such as Eclipse and IBM® Rational® Application Developer generate a stubbed version of this code for you automatically. Thus, I won't spend much time on the specifics of how to write this code.)

Deploy Web services in Apache Geronimo

For simplicity, I created a class called AmazonClient in the same package structure that was generated using WSDL2Java (com.amazon.xml.AWSECommerceServer). The code for this class is shown in Listing 2. This class exposes a single method called lookupISBN. As the name suggests, this method calls AWS and returns information about the book represented by the ISBN number passed in.

Listing 2. AmazonClient.java

```
package com.amazon.xml.AWSECommerceServer;

import java.lang.*;
import java.util.*;

public class  AmazonClient {

    public AmazonClient() {    }

    public Items[] lookupISBN(String isbn) throws Exception {

        try {

            System.out.println("Given ISBN is " + isbn);

            AWSECommerceServiceLocator locator =
                    new AWSECommerceServiceLocator();

            AWSECommerceServicePortType type =
                    locator.getAWSECommerceServicePort();

            String itemId[] = {isbn.trim()};
            ItemLookup lookup = new ItemLookup();
            lookup.setAssociateTag("MY ID");   // fill in your own
            lookup.setSubscriptionId("MY ID"); // fill in your own
            ItemLookupRequest lookupReq = new ItemLookupRequest();
            lookupReq.setMerchantId("All");
            lookupReq.setItemId(itemId);
            lookupReq.setResponseGroup(new String[]
           {"Medium", "OfferFull", "Variations", "Images"});
            ItemLookupRequest[] requests = lookup.getRequest();
            requests = new ItemLookupRequest[1];
            requests[0] = lookupReq;
            lookup.setRequest(requests);
            ItemLookupResponse response =
                    type.itemLookup(lookup);
            Items[] items = response.getItems();
            if (items != null && items.length > 0) {
                System.out.println("Number of results "+ items.length);
                return items;
            }
        } catch (javax.xml.rpc.ServiceException se) {
            throw new Exception(se.getMessage());
        }
```

```
      return new Items[0];
   }
}
```

You must register with Amazon to get a subscription ID (see Consuming Amazon Web Services above). After you get your subscription ID, replace it in the code in Listing 2.

Now you're ready to compile this code. After you have run the setenv.bat file, you should be able to compile the code easily by running javac *.java:. You get several class files in this directory. For simplicity of packaging, I copy the entire src directory into a new directory called WEB-INF/classes. Then, from WEB-INF/classes, I delete all the Java source files; from WEB-INF/src, I delete all the class files. If you use an integrated development environment (IDE), such as Eclipse, a lot of this manual work is done for you automatically.

Now you can invoke your Web service and see the results. Use a JSP file to do this.

Write a JSP file to display results from the Web service

Under the amazonclient directory, create a JSP file called searchAmazon.jsp. The code is shown in Listing 3.

In the JSP file shown in Listing 3, you're calling the AmazonClient class that you defined and receiving an array of Item objects. You then iterate over this array to display the values.
You now have all the code you need.

Final steps before deploying the code
The last step is to set up the deployment descriptors so you can deploy the WAR file on Geronimo. Under the WEB-INF directory, create two simple XML deployment descriptors. The first is the standard J2EE WAR deployment descriptor, called web.xml (see Listing 4).

Listing 4. web.xml

```
 <?xml version="1.0"  encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_3.xsd"
   version="2.3">

  <display-name>Amazon Sample</display-name>
  <welcome-file-list>
     <welcome-file>t;searchAmazon.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Geronimo requires a geronimo-jetty.xml descriptor. The code is shown in Listing 5.

Listing 5. geronimo-jetty.xml descriptor

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app
xmlns="http://geronimo.apache.org/xml/ns/web/jetty"
xmlns:naming="http://geronimo.apache.org/xml/ns/naming"
configId="amazonclient"
parentId="amazonClient">
    <context-root>/amazonclient</context-root>
   <context-priority-classloader>true</context-priority-classloader>
</web-app>
```

Now you can save this file as a WAR file called amazon.war. To do so, run the following command from the amazonclient directory:

```
jar cf amazon.war
```

Now you're ready to deploy this code on Geronimo.

Deploy the consumer code on Geronimo
Although you built a standard J2EE WAR file, you could easily deploy this WAR file on any application server (or servlet engine) that you choose. The next sections show you how to deploy this code on Geronimo.

Install Geronimo
Begin by downloading and installing Geronimo. (See Resources to link to the download site.) Download the Geronimo 1.0 M3 Installer. To run the installer, use the following command:

```
java -jar geronimo-1.0-M3-installer.jar
```

Follow the steps to install Geronimo into GERONIMO_HOME, defined as C:\geronimo.
Start Geronimo

To start the server, from the GERONIMO_HOME directory, run the following command:

```
java -jar bin/server.jar
```

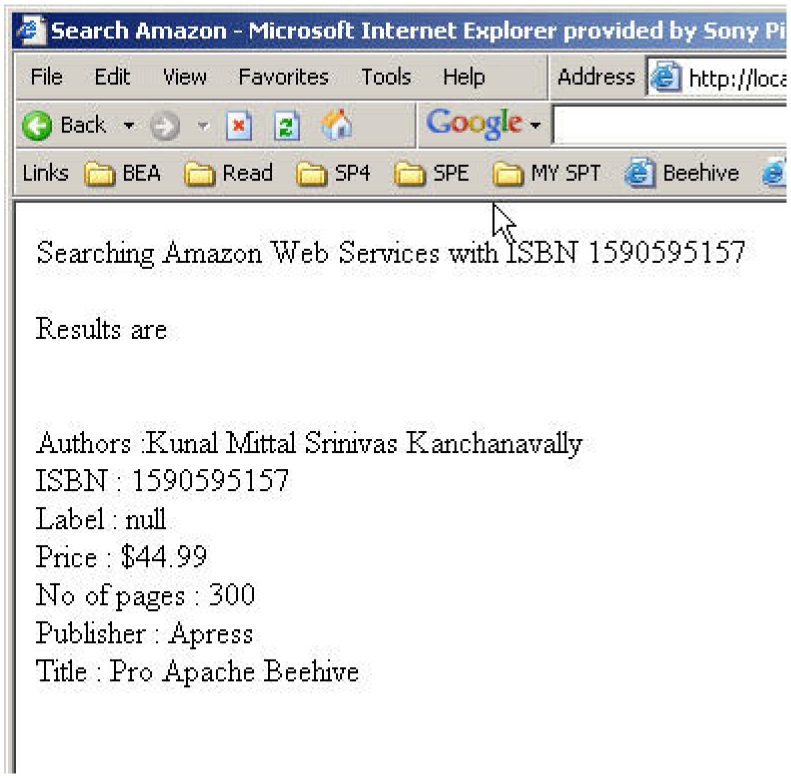
Deploy the Amazon.war file
You should have all the JAR files defined in the setenv batch file in your classpath. Copy the files from AXIS_HOME\lib into GERONIMO_HOME\lib. Many of these JAR files will already exist in GERONIMO_HOME\lib. Do not overwrite the files that exist, even if the files you're copying are later versions of the same file. Click No when prompted to replace existing files.
To deploy the amazon.war file on the application server, copy the WAR file into GERONIMO_HOME. From this directory, run the following command:

```
java -jar bin/deployer.jar deploy amazon.war
```

Run searchAmazon.jsp
The last task left is to run the JSP file you wrote. Open a browser and type http://localhost:8080/amazon/searchAmazon.jsp in the address bar.

Figure 1 shows the result. You can modify the JSP file to include a form field in which you can enter an ISBN number and get a result.

Figure 1: Execution of searchAmazon.jsp


Search Amazon - Microsoft Internet Explorer provided by Sony Pi
File Edit View Favorites Tools Help
Address http://loca
Back
Google
Links BEA Read SP4 SPE MY SPT Beehive

```
Searching Amazon Web Services with ISBN 1590595157

Results are


Authors :Kunal Mittal Srinivas Kanchanavally
ISBN : 1590595157
Label : null
Price : $44.99
No of pages : 300
Publisher : Apress
Title : Pro Apache Beehive
```

You have now successfully deployed your first Web services consumer in Geronimo.

Standard J2EE Web services
The example in this article is simple, but it has shown that Geronimo supports standard J2EE Web services. As an exercise in using Web services, you can try out other options available in AWS. For example, change the ISBN search to be a title or author search. Then, from the Web service results, figure out how to display a URL that links to Amazon.com, and show an image of the book. Another interesting exercise is to use other J2EE technologies along with Apache Axis to consume Web services, and then deploy them in Geronimo. For example, replace the AmazonClient.java Plain Old Java Object (POJO) with a stateless session Enterprise JavaBean (EJB) component, and then deploy it in Geronimo.

You can take any Web service from webMethods or the Google Web services as an example. If you apply the same steps described in this article, you can quickly deploy a consumer to these Web services in Geronimo.

Summary
Now that you've met Geronimo and seen its Web services capabilities, you should be ready to get started using the Geronimo application server in your Web services development efforts.

Create collaborative and dynamic method content using Web 2.0

Level: Intermediate

17 Apr 2008

Leverage Web 2.0 technologies to extend software development process content, which is typically published static as HTML. This article describes how you can develop the ability to collaboratively edit method content and have access to the latest dynamic content within a method context.

Introduction
IT practitioners commonly use software development methodologies, such as the IBM® Rational® Unified Process (RUP®). Methods like this can be applied across a variety of software development disciplines and industry verticals. Software development methods like RUP and IBM RUP Service-Oriented Modeling and Architecture (SOMA) provide static process guidance that's published as HTML. IBM Rational Method Composer is a tool that process engineers can use to customize a process; however, you can publish the new process as just read-only Web pages.

For the method to be truly useful it needs to be augmented with context-specific assets. These are generally content, tooling, and people assets. The content assets include a range of resources, such as documents, presentations, models, social bookmarks, and others. For example, if this method is being applied to aid in business modeling in a telephone company (telco) vertical account, then the method should provide guidance around specific tooling and content that can be leveraged.

Because the method content is frozen after being published, it's not extensible. Therefore, you can leverage Web 2.0 technologies to augment the static content with supplemental wiki pages that enable collaborative editing and dynamic Web feeds. These pages are referred to as extension points in the next section.

So why is the lack of method extensibility a problem? Because method contents:

- Become outdated; for example, guidance artifacts such as templates, assets, or tool mentors become obsolete.
- Lack specific context details without being customized. For example, due to the nonspecific nature of the off-the-shelf method content, it needs to be adapted for the situation in which it's being applied, such as the organization, industry domain, roles, activities, assets, and tools.
- Customizations require republishing.
- Lack the ability to be augmented by the user or practitioner and, thus, can't take advantage of the user's contribution to community content development and enhancement. Consequently, opportunities for feedback and collaboration by the user community is often lost.
- Lack the ability to leverage media-rich content, such as videos, podcasts, and flash demonstrations because they typically become outdated quickly.
- Tend to lack detailed commercial off-the-shelf tool guidance.
- Off-the-shelf, lack guidance for organizational assets or tools.

Create collaborative and dynamic method content using Web 2.0

Another problem is that process engineering departments don't have enough resources to produce all of the method content needed in the field. For example, they usually can't provide content for different versions of the same tool. Hence, the method content remains unchanged, and the organization loses the collective knowledge and understanding of its practitioners who can revise the content based on real-world field knowledge.

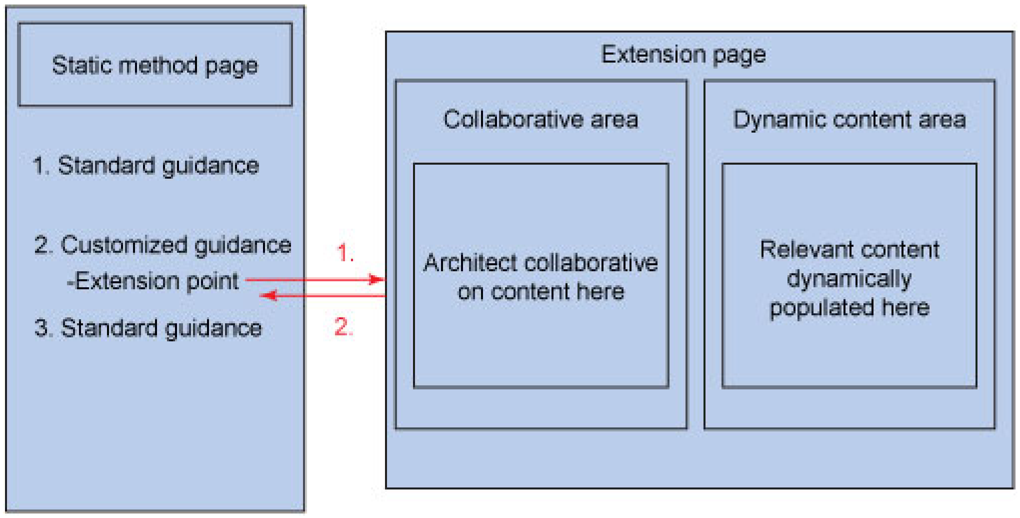
Note: Some of the content is often priority to an organization; an example is the Insurance Application Architecture (IAA) model, which is proprietary to a company like IBM.

Other techniques have been used to solve the same problem. Consider the following: An administrator or process engineer can republish the static method on a regular basis (for example, monthly, weekly, daily). However, a process needs to be established to incorporate practitioner feedback or contributions into the method. Also, the problem is aggravated by the fact that the method is published along with a tool, such as IBM Rational Software Architect and doesn't get updated until Rational Software Architect gets updated. Practitioners build their own pages with up-to-date information, but these pages are scattered and not integrated with method and process contents.

Create extensible method content with Web 2.0

Ideally, you want to be able to extend software development processes and methods with extensions built collaboratively, and dynamically populated at the time practitioners use the method. Figure 1 shows you what this looks like.

Figure 1: Collaborative and dynamic method overview



There are several activities associated with this collaborative and dynamic content. Let's take a closer look at these activities.

Extension point identification

The process engineer identifies extension points in a static method. Typically these extension points are in areas of the method that describe new or improved techniques, or both, and in areas for which content needs to be built with the help of community or will quickly become out of date.

Create collaborative and dynamic method content using Web 2.0

Extension page creation
After an extension point has been identified, the process engineer creates an extension page for this extension point. The purpose of this extension page is to provide up-to-date guidance in addition to the contents of the static method. The extension page contains two areas:

- Collaborative guidance content area
- Dynamic content area

Collaborative guidance area
The collaborative guidance content area provides up-to-date guidance on the method for this extension point. Typically the initial content in this page is populated by another process engineer skilled in the art of this particular extension point. The content in this area can be, for example, the latest information on tools and how to obtain them, and then used to execute this extension point more effectively. This collaborative area is also editable by the user (practitioner or architect) to allow for field-based lessons and input to be captured, thus keeping the best practices and field-based lessons learned about this extension point up to date. An example of a Web 2.0 implementation of this collaborative guidance area is a wiki.

Dynamic content area
The dynamic content area dynamically provides assets and artifacts to the user or practitioner to help him or her execute the task described by this extension point. The kinds of assets and artifacts can include social bookmarks; subject matter experts (including their instant messaging statuses); documents; publications; presentations; media-rich syndicated content, such as podcasts and movies; and education materials, including blogs, online course material, and classes. Because the information in this area is dynamic, the system builds content at the time the practitioner requests the page so that he or she is always guaranteed of the latest information. An example of a Web 2.0 implementation of this dynamic content area is aggregated Web feeds.

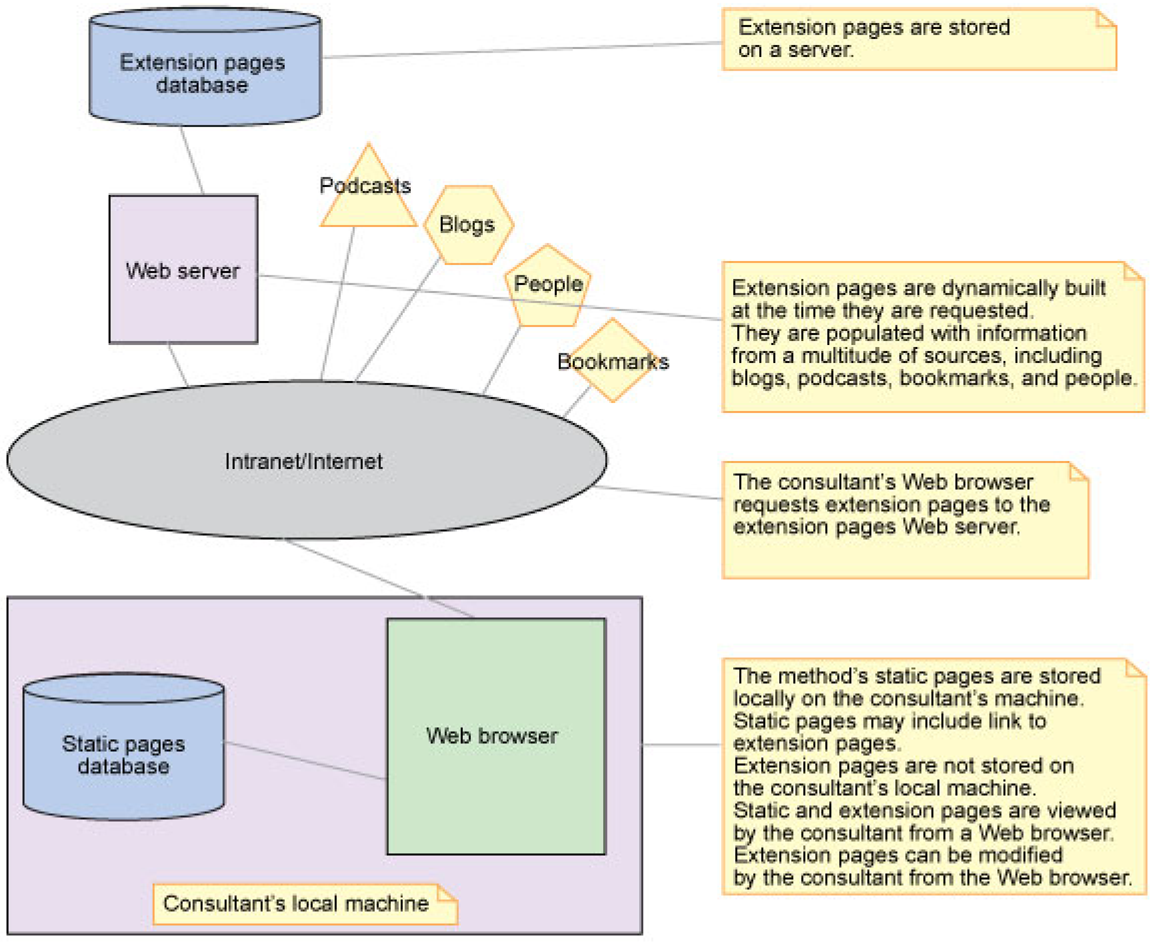
The static method process engineer repeats the creation of extension pages for each extension point identified in the static method.

By adopting this technique for creating a method, the dynamic content is automatically built when a user requests a specific page and includes information from many different sources outside of the core method contents. This dynamic content can be provided by practitioners, not just method authors (process engineers). This dynamic content is accessible from the method, not scattered over the Internet or intranet.

Figure 2 illustrates where method contents are located (topology view). It's best to read the comments from bottom to top, starting from the consultant's local machine.

Create collaborative and dynamic method content using Web 2.0

Figure 2. Collaborative and dynamic method structure


Extension pages database
Extension pages are stored on a server.
Podcasts
Blogs
People
Bookmarks
Web server
Extension pages are dynamically built at the time they are requested. They are populated with information from a multitude of sources, including blogs, podcasts, bookmarks, and people.
Intranet/Internet
The consultant's Web browser requests extension pages to the extension pages Web server.
Static pages database
Web browser
The method's static pages are stored locally on the consultant's machine. Static pages may include link to extension pages. Extension pages are not stored on the consultant's local machine. Static and extension pages are viewed by the consultant from a Web browser. Extension pages can be modified by the consultant from the Web browser.
Consultant's local machine

The advantages of adopting this kind of approach to software method development are:

- Method content isn't restricted to what's being shipped by a method at a specific version.
- Method contents are always up to date.
- Practitioners, not just process engineers, can provide contents for the method.
- Practitioners don't need to download new versions of a frozen method.
- Contents are no longer restricted to what an engineering department can produce given their time, budget, and headcount constraints.
- Media-rich and innovative contents (for example, list of experts on a topic, podcasts, and flash movies) are easily integrated into method contents.
- Methods can contain both frozen (static) core contents and spots where organizations or communities can extend and customize contents.
- Practitioners can easily find experts on a given topic.

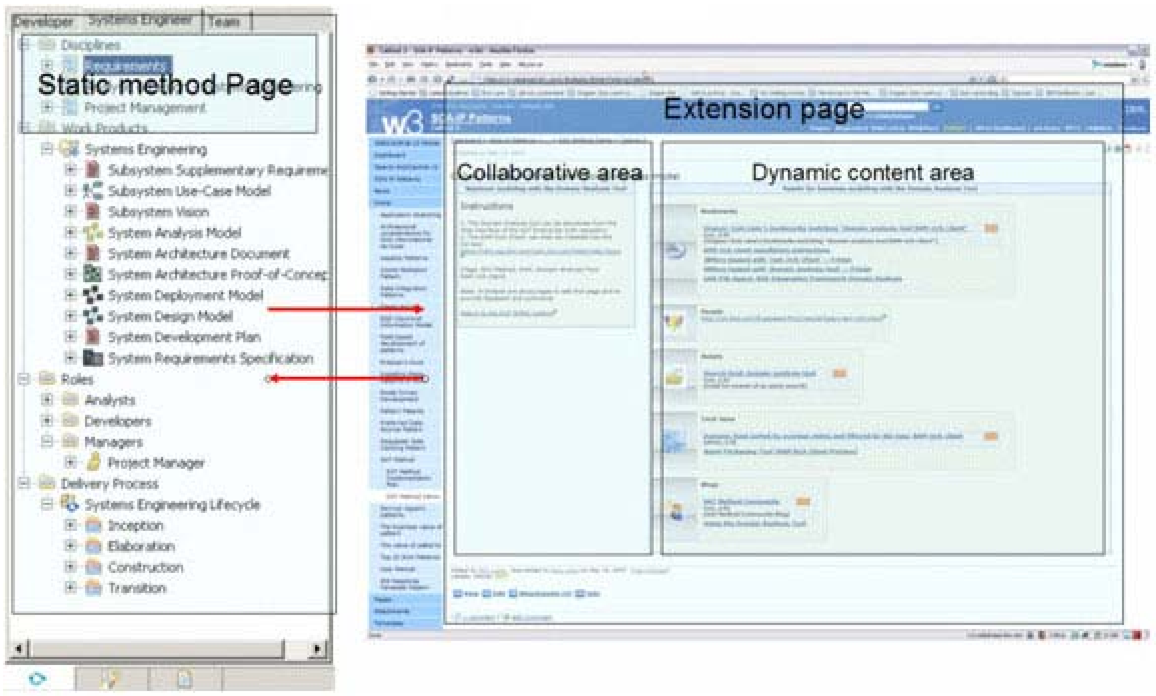
Create collaborative and dynamic method content using Web 2.0

An implementation of this kind of method:

- Identifies the points in the static method that can be extended.
- Provides links to collaboratively built dynamic contents from these extension points.
- For each link, provides a wiki page with two areas:
  - Up-to-date editable textual information
  - Dynamically populated Web feeds on social bookmarks, people, activities, blogs, or assets

Here's how the method works and is implemented for a particular Service-Oriented Architecture (SOA) analysis scenario (see Figure 3):

Figure 3. Collaborative and dynamic method implementation



An example of how it can be used is as follows: A software architect in an SOA engagement is currently involved in the analysis of an SOA system. In SOA, one of the core activities is called service identification. Because SOA projects are by their very nature complex, and because SOA is still in the process of being understood, SOA activities and the tools to support these activities are fully understood only by a small number of practitioners.

It's highly unlikely that a static method includes up-to-date content on the best practices around these activities or, indeed, the tooling that could make these activities more streamlined. To address this shortcoming, extension points are identified in the base method. Where the static method is rendered in HTML, these extension points are hyperlinks from inside the method. In this example, the SOA architect is doing service identification. The static method contains high-level guidance around service identification and a link to collaboratively built dynamic content on specific tools used for service identification. This link brings the architect to a collaborative Web site, such as a wiki dedicated to this extension point.

Wiki implementation
This wiki site contains the following two areas.

Collaborative content area
The first area contains current information on service identification tool offerings to make the activity more consistent and streamlined. Because this is a collaborative area, the SOA architect is encouraged to edit these instructions to keep them up to date and to provide the latest field-based development thinking around these tools.

Dynamic content area
The second area of the wiki Web site provides dynamic content relevant to the architect for this particular extension point in the form of Web feeds. In the case of the service identification extension, the area is populated by dynamic content around service identification. This dynamic information is filtered based on a set of unique tags associated with this particular extension point (for example, tags for a service modeling activity might be: services modeling and service-modeling). Such syndicated dynamic content includes:

- Social bookmarks and subject matter experts of service identification (including their instant messaging status).
- Reusable assets from an asset repository (including documentation patterns and even tooling for services identification).
- Media-rich content (including technical presentations, movies, or podcasts).
- Educational content (including blogs, course material, other reading material, and IBM Redbooks®).
- Activities that the architect needs to follow to complete the extension point around service identification successfully.

This is made possible by being able to embed all of this dynamic content in a wiki with syndicated Web feeds formatted as RSS or Atom. Each of the dynamic items above has its own Web feed, and these feeds are aggregated to provide all of the dynamic content. This is already possible, and by standardizing on Web 2.0 technology and tags, for example, all service identification-related content checked into an asset repository are tags with the service identification and dynamic-method keyword. After an item is checked into the Web feed-enabled asset repository with these keywords, the Web feed provided by the asset repository is automatically updated. As the extension page is refreshed for the service identification extension point, the updated content is now available to the method practitioner (architect). This can also be done with all of the content for the other dynamic feeds.

Advertise with JavaJazzUp

We are the top most providers of technology stuffs to the java community. Our technology portal network is providing standard tutorials, articles, news and reviews on the Java technologies to the industrial technocrats. Our network is getting around 3 million hits per month and its increasing with a great pace.

For a long time we have endeavored to provide quality information to our readers. Furthermore, we have succeeded in the dissemination of the information on technical and scientific facets of IT community providing an added value and returns to the readers.

We have serious folks that depend on our site for real solutions to development problems.

JavaJazzUp Network comprises of :

http://www.roseindia.net
http://www.newstrackindia.com
http://www.javajazzup.com
http://www.allcooljobs.com

Advertisement Options:

Banner	Size	Page Views	Monthly
Top Banner	470*80	5,00,000	USD 2,000
Box Banner	125 * 125	5,00,000	USD 800
Banner	460x60	5,00,000	USD 1,200
Pay Links		Un Limited	USD 1,000
Pop Up Banners		Un Limited	USD 4,000

The http://www.roseindia.net network is the "real deal" for technical Java professionals. Contact me today to discuss your customized sponsorship program. You may also ask about advertising on other Technology Network.

Deepak Kumar
deepak@roseindia.net

India's Cheapest web Service Provider

Web Packages

Package (in INR)

Package Name	Price
Starter Package	Rs 5,100 + Taxes Extra
Business Package	Rs 9,111 + Taxes Extra
Corporate Package	Rs 22,500 + Taxes Extra
Smart Package	Rs 45,500 + Taxes Extra

* Domain Registration free with every package

Packages Specifications

Starter Package

- 5 Web Pages
- 10 Stock Images
- 5 POP 3 Accounts
- 50 MB Hosting Space
- Unlimited Edits

Business Package

- 10 Web Page Design
- Flash Site Animation
- 25 Stock Images
- 10 POP3 Accounts
- 50 MB Hosting Space
- Unlimited Edits

Corporate Package

- 25 Web Page Design
- Flash Site Animation
- Flash Intro Page
- 50 Stock Images
- 25 POP3 Accounts
- 100 MB Hosting Space
- Unlimited Edit

Smart Package

- 200 Web Page Design
- Catalog with 200 items
- Flash Site Animation
- Flash Intro Page
- Unlimited Stock Images
- 50 POP3 Accounts
- 250 MB Hosting Space
- Unlimited Edits

RoseIndia
Technologies Pvt. Ltd.

Logon to: http://www.roseindia.net/services/

Valued JavaJazzup Readers Community

We invite you to post Java-technology oriented stuff. It would be our pleasure to give space to your posts in JavaJazzup.

Contribute to Readers Forum

If theres something youre curious about, were confident that your curiosity, combined with the knowledge of other participants, will be enough to generate a useful and exciting Readers Forum. If theres a topic you feel needs to be discussed at JavaJazzup, its up to you to get it discussed.

Convene a discussion on a specific subject

If you have a topic youd like to talk about . Whether its something you think lots of people will be interested in, or a narrow topic only a few people may care about, your article will attract people interested in talking about it at the Readers Forum. If you like, you can prepare a really a good article to explain what youre interested to tell java technocrates about.

Sharing Expertise on Java Technologies

If youre a great expert on a subject in java, the years you spent developing that expertise and want to share it with others. If theres something youre an expert on that you think other technocrates might like to know about, wed love to set you up in the Readers Forum and let people ask you questions.

Show your innovation

We invite people to demonstrate innovative ideas and projects. These can be online or technology-related innovations that would bring you a great appreciations and recognition among the java technocrates around the globe.

Hands-on technology demonstrations

Some people are Internet experts. Some are barely familiar with the web. If you'd like to show others aroud some familiar sites and tools, that would be great. It would be our pleasure to give you a chance to provide your demonstrations on such issues : How to set up a blog, how to get your images onto Flickr, How to get your videos onto YouTube, demonstrations of P2P software, a tour of MySpace, a tour of Second Life (or let us know if there are other tools or technologies you think people should know about...).

Present a question, problem, or puzzle

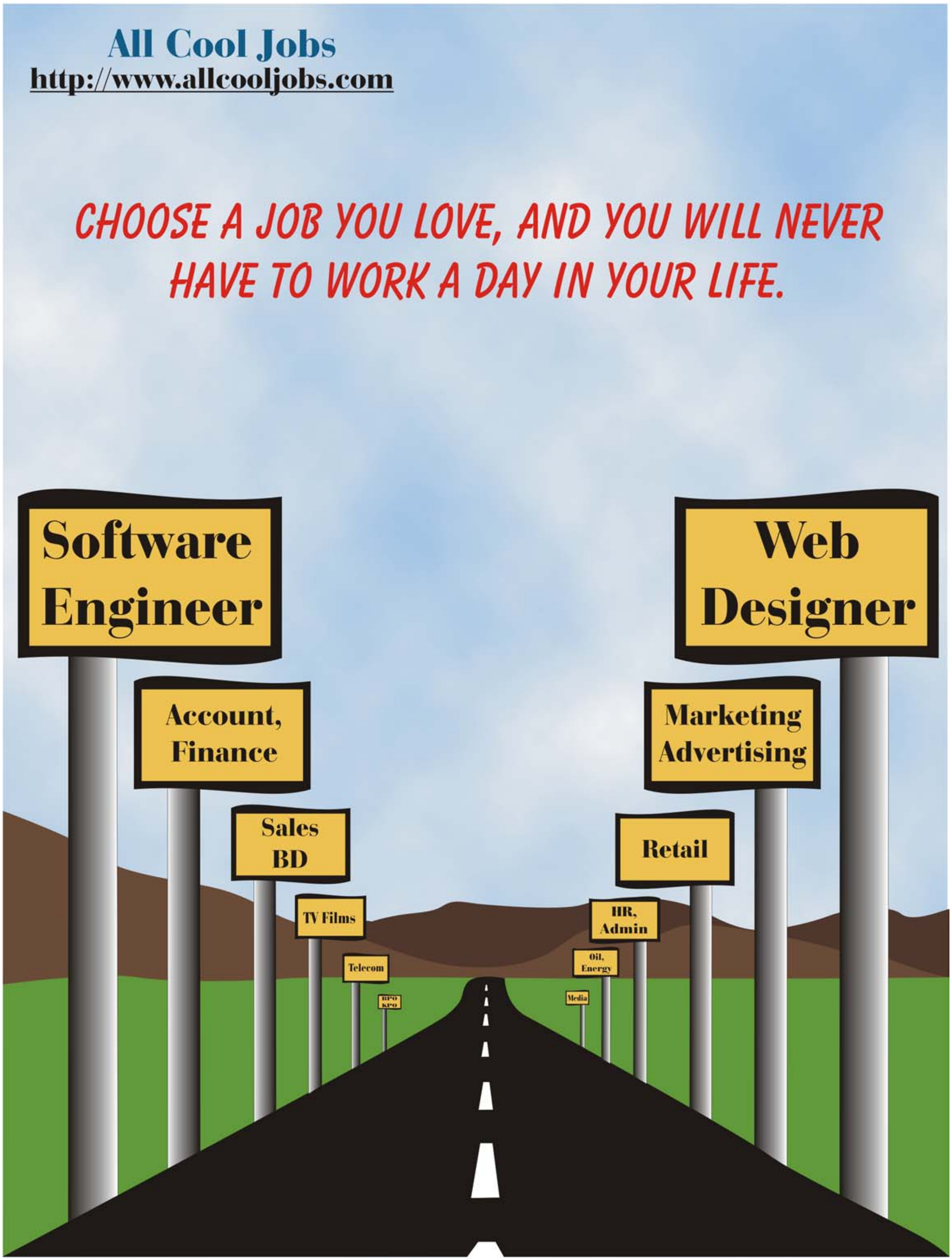
Were inviting people from lots of different worlds. We do not expect everybody at Readers Forum to be an expert in some areas. Your expertise is a real resource you may contribute to the Java Jazzup. We want your curiosity to be a resource, too. You can also present a question, problem, or puzzle that revolves around java technologies along with their solution that you think would get really appreciated by the java readers around the globe.

Post resourceful URLs

If you think you know such URL links which can really help the readers to explore their java skills. Even you can post general URLs that you think would be really appreciated by the readers community.

Anything else

If you have another idea for something youd like to do, talk to us. If you want to do something that we havent thought of, have a crazy idea, wed really love to hear about it. Were open to all sorts of suggestions, especially if they promote readers participation.


All Cool Jobs
http://www.allcooljobs.com
CHOOSE A JOB YOU LOVE, AND YOU WILL NEVER HAVE TO WORK A DAY IN YOUR LIFE.
Software Engineer
Web Designer
Account, Finance
Marketing Advertising
Sales BD
Retail
TV Films
HR, Admin
Telecom
Oil, Energy
BPO KPO
Media

NEWSTRACK india

Broadcasters are storytellers, newspapers are fact providers and we serve the both.

http://www.newstrackindia.com