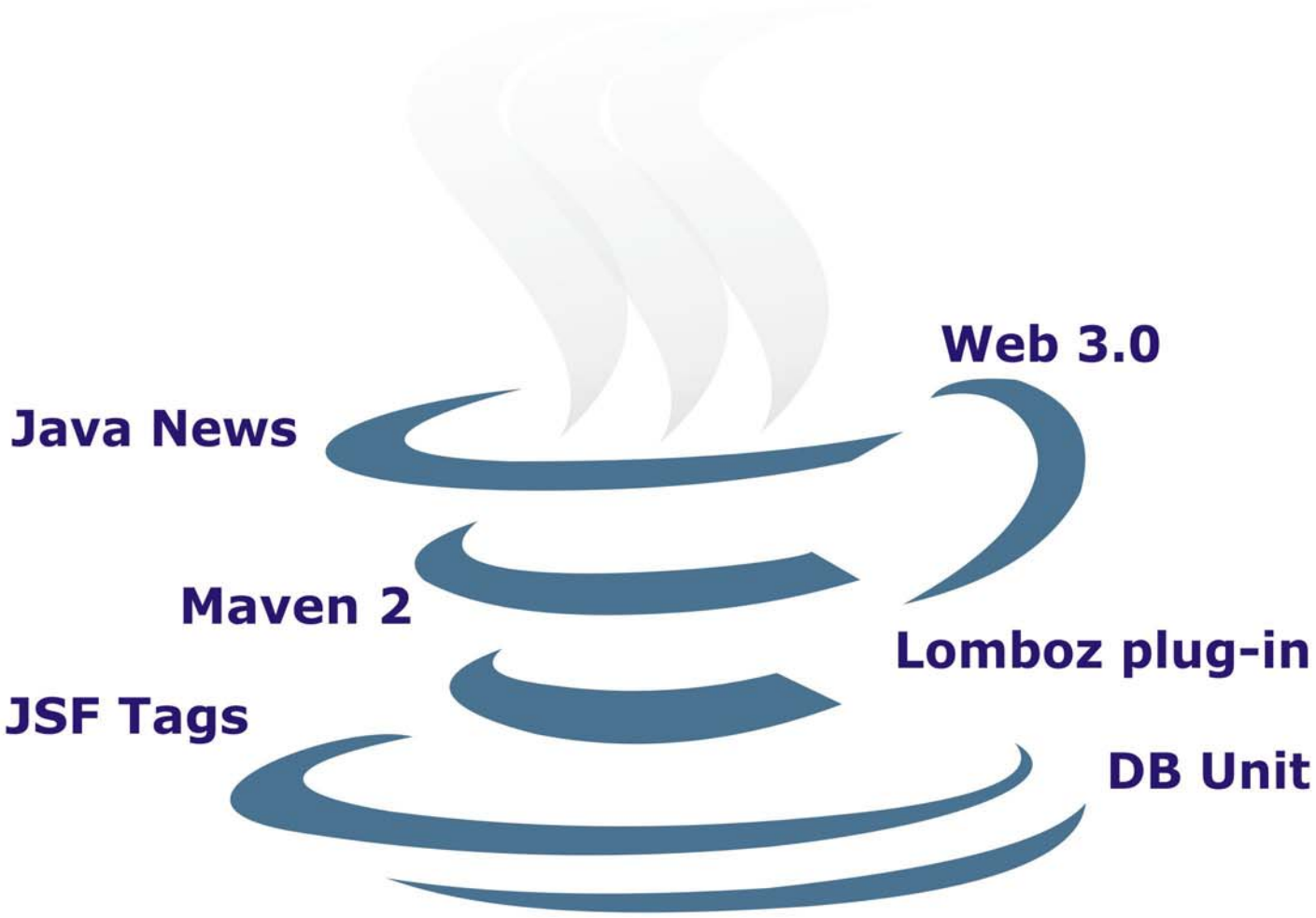
Sept 07 | Volume 1 |Issue 3 | **Free**

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING



Develop java persistence example with Maven2

Java Jazz up
A BETTER WAY TO LEARN PROGRAMMING

September 2007 Volume I Issue III

"The roots of technology may be bitter,
but the fruits are always sweet,
Just put a spark in your deeds and
the sky comes under your feet"

Published by

RoseIndia

Editorial

Dear Readers,

We are here again with the third issue of Java Jazz-up. The current edition highlights the interesting Java articles and tutorials in a well described manner developed by the Java Jazz-up developer's team. This issue reflects our consistent attempts to avail quality technological updates that enforce the reader to appreciate a lot and be a part of its readers community.

The current release of Java Jazz-up tries to provide new features orienting around developing enterprise level applications that involves issues like setting up the Project Repositories within an organization, implementing internationalization concepts, testing databases within an application, how to work with the Tomahawk tags and a lot more...

Set of tutorials discussing technologies like Maven2, Design patterns, JSF framework, web services, spring framework and various complementary open source technologies are provided in such a manner that even a novice learns and implements the concepts in a easy manner.

Java News and update section provides the latest things going around the globe that makes the readers aware of the java-technological advancements. In this section you will learn about the new features introduced in the existing tools, utilities, application servers, IDEs, along with the Java API updates.

To make it interesting for readers we have categorized each section with different colors and images that lure readers while reading technological stuffs. We are providing it in a PDF format that you can view and even download it as a whole and even get its hard copy.

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

Editor
Deepak Kumar
Java Jazz up

Content

05 **Java News |** Java Around the Globe- Sony Ericsson to provide new online testing service for Java ME application...

08 **New Releases |** Spring Web Services 1.0 Released: Enhanced Features - Interface21 has announced the release of Spring Web Services 1.0. With this new release, Java programmers can develop applications for Web services with the...

10 **Java Developers Desk: Internationalization |** Internationalization is one of the key features of Java language, which makes a java application... internationalized

13 **JSF Tags : Tomahawk Tags |** Tomahawk tags are collection of standard components with extended functionality and many more extra set of components with rich set of functionality.

24 **Building Project : Maven2 |** Maven2 is an Open Source build tool that made the revolution in the area of building projects.

35 **Develop java persistence example with Maven2 |** Download Maven2 from maven.apache.org, unzip the archive into your local directory...

40 **Database Testing with DbUnit |** DbUnit is an open source Framework created by Manuel Laflamme. This is a powerful tool for simplifying Unit Testing of the database operations.

44 **Creational Design Pattern |** The singleton design pattern deals with one and only one instance of an object that encapsulates the control of the object from a common place. There are various ways of achieving this pattern.

49 **Developing web Services with Lomboz Plug-in |** Lomboz is an open source and free JEE development environment used for businesses and individual purposes.

55 **Spring Framework: Spring Core with Data Access Framework |** In this article, we will move on to the main process of any enterprise application: Data Persistence.

59 **Web 3.0 |** Web 3.0 is a term, which definition is not confirmed or defined so far as several experts have given several meaning, which do not match to each other, but sometimes it is referred to as a Semantic Web.

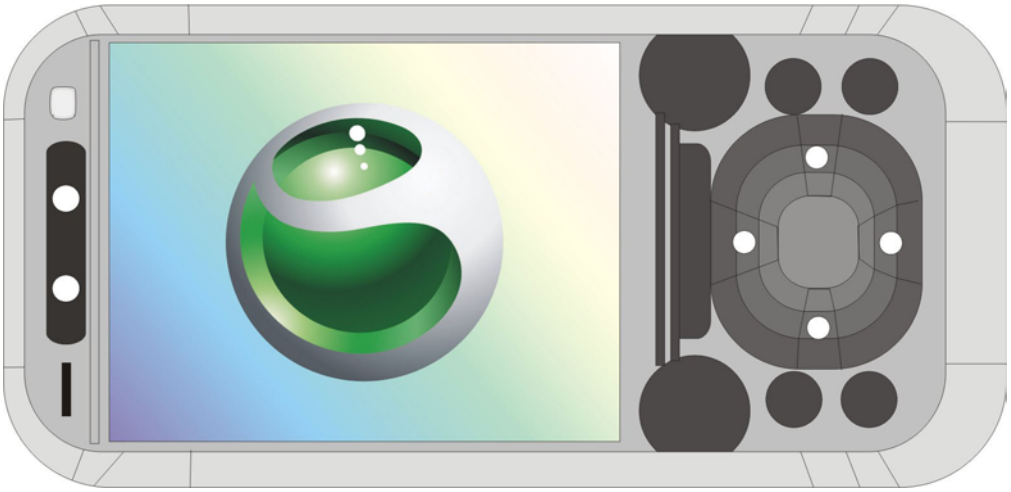
65 **Tips 'n' Tricks |** This Java program lets you download files from one or more URLs and save them in the directory where you want.

76 **Advertise with JavaJazzUp |** We are the top most providers of technology stuffs to the java community.

77 **Valued JavaJazzup Readers Community |** We invite you to post Java-technology oriented stuff.

JAVA AROUND THE GLOBE

Sony Ericsson to provide new online testing service for Java ME application:



"Sony Ericsson Virtual Lab" will provide Java developers an online testing service to test and monitor Java applications on Sony Ericsson phones. This will make the process quicker and cheaper than before. The online service will be available globally 24 × 7 to all members of Sony Ericsson Developer World. This online service enables developers to test remotely the pre-commercial mobile phones, released by the company (like 2007 phones) and will help in developing the perfect application for these new phones. Now, developers will no longer need to invest in device labs, shipping of phones, updating firmware or be physically present on an operator's network.

Sun adds mobile capability to messaging and apps platforms:

Sun has planned to incorporate mobile synchronization capabilities into its communications and applications platform suites. For this purpose, sun has signed a licensing deal with "**Synchronica**" to incorporate "**SyncML**" technology from Synchronica's Mobile Gateway server into the products.

"This relationship extends Sun's reach beyond the wide range of supported desktop clients to a large number of mobile devices. This collaboration also enables our application server developers to build feature-rich and scalable mobile applications that support both connected and disconnected operations."said Sun's vice president of software infrastructure, Karen Tegan Padir.

NetBeans Going GPL

Netbeans is in the process to adopt the GPL v2 with Classpath exception license as a second license option along with CDDL that OpenJDK uses. GPL v2 is an open-source license. This will help NetBeans in getting greater adoption in the Linux community. Not only this, the release of Netbeans 6.0 version is in a process.

Sun Open Java License Unveils:

Now the Open JDK Community Technology Compatibility Kit (TCK) License is available for enhanced portability of the solution. This is for the Java Compatibility Kit (JCK) that concludes whether an

implementation consents with the Java Platform Standard Edition 6 specification or not. The JCK license would be fully in consent of the terms of the GNU General Public License version 2 (GPLv2) which is available at:http://openjdk.java.net/legal/openjdk-tck-license.pdf.

Pioneer Java Spreadsheet SDK is now an Open Source Innovation

Extentech Inc, an innovative developer of Java components and development tools, announced the release of OpenXLS (under the GPL license), which is an open source version of the company's Java Software Development Kit for Spreadsheets, ExtenXLS. OpenXLS is a robust and risk-free way to integrate spreadsheets into Java programs and web applications.

"With the release of OpenXLS, Extentech has leveled the playing field for developers who need to embed spreadsheet functionality in their applications," states John McMahon, CEO of Extentech Inc.

Magical Java Charts in Minutes!

Developers can now design and configure attractive graphical charts on any platform. These are ElegantJCharts, which are a collection of three charting utilities: ElegantJ Chart Designer IDE, ElegantJ Chart Library, and ElegantJ Indicator and Gauges.

Developers can take advantage of ElegantJCharts Designer IDE for the visual configuration of charts along with the flexibility of integrating chart library in their applications in an exclusive way through a comprehensive API.

Java SOA Use Increases, while .NET Declines

Use of Java is increasing at large in SOA deployments as found by Evans Data Corporation. They observed that more companies are planning to deploy JavaV based SOA solutions rather than deploying .NET based SOA solutions. This may be because of the newly available java's feature to virtually tie with the .NET.

Java EE 5.0: New Enhanced Features

The new version of Java EE i.e. (Java EE) 5 is capable of developing robust and scalable enterprise applications. It provides a superb platform for SOA and the development of the next-generation web applications.

Following are few of its features:

- The focus in Java EE 5 is made on the ease of development. With Java EE 5, there is less code to write.
- It supports latest web service APIs that is an ideal platform for Service-Oriented Architectures (SOA).
- It also includes JavaServer Faces (JSF) technology, the JSP Standard Tag Library (JSTL), AJAX etc for building of new generation web applications.
- Java EE 5 supports rich thin-client technologies such as AJAX, technologies that are crucial for building applications for Web 2.0.
- Annotations are used extensively to reduce the need for deployment descriptors.
- Supports EJB 3.0 which makes programming more simpler with the use of Plain Old Java Objects (POJOs).
- It also introduces a new persistence API.

JSR 286, Portlet 2.0 Enters Public Review

JSR 286, the portlet specification 2.0 has entered its public review phase. The public review held till August 27th. It included the features of the new specification like

Java EE 1.4 support, introduction of the portlet filters, inter-portlet communication, and enhancements to the portlet tag library. Several other enhancements were also discussed. In addition, the specification called for coordinating the JSF expert group to better align JSF with portlets.

Magnolia 2.0: First Open Source CMS Supporting Java Content Repository

The launching of Magnolia 2.0, the first major upgrade to the Magnolia Open Source content management project, has been announced and it is available for free download. Now it has become the first CMS to support the Java Content Repository and also gets improved usability and optimizations for J2EE. Currently, Magnolia is being used by companies like Siemens Enterprise and many pharmaceutical industries. It is available for online testing at http://www.magnolia.info/demo.

New Releases

Spring Web Services 1.0 Released:

Enhanced Features Interface21 has announced the release of Spring Web Services 1.0. With this new release, Java programmers can develop applications for Web services with the facilities of creation of contract-first, document-driven Web services, delivering the flexibility, productivity and ease of use. The contract-first approach avoids interoperability issues and simplifies development.

It works with common Web Services technologies like POX (plain old XML), SOAP and WSDL (Web Services Description Language). This release also facilitates the best practices such as the WS-I basic profile and loose coupling between contract and implementation that allows creating flexible Web Services**.**

Apache Struts 2.0.9 Released: More Elegant Framework

Compared to older versions of Struts, this version is more elegant for creating enterprise-ready Java web applications and is the "best available" version of Struts. This new version is simpler to use and includes an important security fix regarding a remote code exploit. It is available as a separate library, source, documentation distributions etc.

GridGain 1.5 Released: Brings Open Source Grid Computing to Java

GridGain 1.5 has been released, it is available for download at:www.gridgain.org/downloads.html. It is an open source grid computing platform for Java with high-performance development. It has a unique set of grid computing features with open source LGPL licensing.

This release has new features with improved functionality like integration with Spring, Weblogic, JBoss, Coherence, Mule, etc. It is considered to be the widest selection of compatible products in its class.

Google Web Toolkit 1.4 RC2 Released: 100 bugs fixed

Finally, Google Web Toolkit (GWT) 1.4 RC2 has been released and is available for download from Google website. It is being presumed to be the best GWT release to date because in this release, near about 100 bugs have been fixed. Moreover it is expected to be the final 1.4 release and GWT 1.5 is now on the way. GWT 1.5 will soon be released supporting Java 5.0. Hopefully, GWT client code will be based on Java 5.0 syntax and will support all its available features.

Groovy MDA 1.0 Released: A Code Generation Library

Groovy MDA 1.0 has been released and it is available for download from its official website i.e. http://groovy-mda.sourceforge.net Groovy MDA library generates software project artifacts employing the Groovy scripting language from the UML models. It is even capable in generating JPA entity beans from a UML model. Even it enables the user to customize the templates of his choice with ease.

Portals Pluto 1.1.4 Released: Implements Java Portlet Specification

Pluto is the Reference Implementation of the Java Portlet Specfication. The current version of this specification is JSR 168. Portlets are designed to run in the context of a portal. They are written with the Portlet API which is similar to the Servlet API.

With the advent of Portals Pluto 1.1.4, it has become easier to integrate Pluto's portal container into a portal. It's a major refactoring of Pluto 1.0.1, which allows this integration. Moreover, it's

the fifth GA release of the 1.1 line of Pluto. It also supports JSP 2.1. Current Pluto 1.1.4 release has binary compatability with Pluto 1.1.3 which its earlier versions lacked.

Apache Geronimo v2.0 - Release delayed due to security issue

The Apache Geronimo project produces a server runtime framework that pulls together the best Open Source alternatives to create runtimes that fulfil the needs of developers and system administrators.

The release of Apache Geronimo v2.0 has been delayed due to the detection of a security bug. The origin of this problem has been taken into consideration and some tests are going on to fix this problem.

JGroups 2.5 Released: Boosting Group Communication

JGroups is a toolkit, which is used for the development of distributed Java applications based on group communication. It has a feature of flexible protocol stack, which meets the developer's application requirements.

The release of JGroups 2.5 has made the things much easier as it includes upgraded features like multiplexer improvements, concurrent stack, new failure detection protocols etc. This release offers major improvements over previous releases most notably concurrent messages delivery, channel multiplexing and full virtual synchronous support. JGroups users may upgrade their applications to 2.5 releases [4] as it is a straightforward process; API is backward compatible to 2.2.7 release, the only change should be made in the channel configuration files.

Swerve Framework Beta 0.9 Released:

Swerve SOA which is a full stack POJO framework has arrived for developing Flex 2 / Java RIA solutions with the ant script build support. Its provides support for:

- Asynchronous processing that may be in-lined, queued, or fully concurrent.
- Text, XML, binary, and AMF0 return types.
- POJO based REST API to quickly implement REST based services.
- SEDA based messaging fabric with integrated CometD.
- Fast XML Marshalling API.

Jython 2.2 Released: Implements Python 2.2 and 2.3

Jython 2.2 has arrived with the implementation of the internal architecture of Python 2.2. It also contains some features from Python 2.3. Jython that is integrated with the Java platform is an implementation of the high-level, dynamic, object-oriented language.

The updates that have been made in this release are:

- Java Collections integration
- Support for the iterators and generators
- Compatible with both JDK1.5 and 1.6

CodeGear's JGear Eclipse Plug-ins Revealed:

A set of specialized plug-ins has been revealed by CodeGear from Borland Corporation for the Eclipse open-source development platform: **JGear. The new JGear has the following features:**

- JGear LiveSource for Eclipse.
- JGear Team for Eclipse (both Client and Server editions).
- JGear Performance for Eclipse – contains features such as memory and CPU profiling and debugging; real-time monitoring of programs' use of virtual machine memory; automatic detection of potential memory leaks etc.

Internationalization

Internationalization is one of the key features of Java language, which makes a java application internationalized. In other words, Internationalization is the process of designing an application, which is able to adapt itself in different countries and regions without recompiling. Normally, software follows the conventions of region or country in which it is developed. This software is supposed to be used by the group of users familiar with this particular convention. For example, an American developer tends to develop software, which displays text in English, take the amount of money in "dollars" etc. On the other hand, a French developer is expected to develop software, which displays text in French, take currency in "franc". Such software can't be considered as internationalized. Java provides a solution of this issue. A truly internationalized program contains no hard coded area for a specific locale. For example, text, currency, date, number formats, audio clips etc., which makes an application locale specific. Instead of hard coding, these elements are stored outside of the program. Now, the program is not required to be compiled again when a new country or region requires support. When discussing internationalization, word "localization" comes to the front, which is the process of adapting software for a specific region or language by adding locale-specific components and translating text. It involves changing the language interaction, display of numbers, dates, currency, and so on. For better visualization, just go through the example below.

Suppose, we have a program "InternationalizationDemo" in which the text to display is hard coded so it always displays the same text in English Language, no matter a Spanish person wants these texts in its own mother language.

InternationalizationDemo.java: (Without Internationalization Support)

```
import java.util.*;
public class
InternationalizationDemo {
public static void main(String[]
args) {
System.out.println("The text
displayed is specific to
locale"+"
("+Locale.getDefault().
getDisplayLanguage()+",
"+Locale.getDefault().
getDisplayCountry ()+").\n");

System.out.println("Hello, how
are you?");
System.out.println("Thanks to
visit.");
}}
```

Now you want this program to get internationalized so that it may response according to the specific region and country i.e. locale (Remember no code changes are required for different locale). Just follow these steps:

1. Create Properties Files:

Create ". properties" file containing a set of key and value pair. Remember to keep the keys same in all the files and provide values according to the locale in different files.

Properties Files Naming Convention:

Creating a default properties file is a good practice, which is available to the program by default. You can give any name to this file but remember to use the same name that your **ResourceBundle** uses (**MessageBundle. properties** file in our example). While naming other properties files follow the syntax:

PropertiesFileNameUsedIn ResourceBundle_language Code_countryCode.properties

Let's create these files. Default file:

Write down the following lines in a plain-text file (Default version) and save it as

MessagesBundle.properties:

```
localeInfo = The text displayed is specific to
localewelcome =  Hello, how are
you?sayThanks =  Thanks to visit.
```

Other properties files:
Write down the following lines in a plain-text file (for the French version) and save it as **MessagesBundle_fr_FR.properties:**

```
localeInfo = Le texte affiché est spécifique
à la scène
welcome =  Bonjour, comment allez-vous ?
sayThanks =  Merci pour visiter.
```

Write down the following lines in a plain-text file (for the Spanish version) and save it as
MessagesBundle_es_ES.properties:

```
localeInfo = El texto mostrado es específico
al lugar
welcome = ¿Hola, ¿Cómo está usted?.
sayThanks = Gracias a visita.
```

Notice that "localeInfo", "welcome", "sayThanks" keys are same in English, French, Spanish files and values are replaced with the converted value of the particular language.

2. Remove hard coded text from the source file:

Our next step is to remove the hard coded text from our source file and use the keys of properties file. These values are picked up at the run time from the properties file according to the locale provided to the program. So this source file doesn't require to be compiled while dealing with the different locales. You can be sure of this fact viewing the code modified below:

InternationalizationDemo.java:
(Internationalization Support)

```
import java.util.*;
public class InternationalizationDemo {
    public static void main(String[] args) {
        String language;
        String country;
        Locale locale;
        ResourceBundle rb;
        if (args.length != 2) {
            language = new String("en");
            country = new String("US");
        }
        else {
            language = new String(args[0]);
            country = new String(args[1]);
        }
 locale = new Locale(language, country);
rb = ResourceBundle.get
Bundle("MessagesBundle", locale);
System.out.println(rb.getString("localeInfo")+"
("+locale.getDisplayLanguage()+",
    "+locale.getDisplayCountry ()+").\n");
System.out.println(rb.getString
("welcome"));
        System.out.println(rb.getString
("sayThanks"));
    }
}
```

A Locale object represents a specific geographical, political, or cultural region. In other words, Locale is an identifier for a particular combination of language and region. One of its constructors is:

Locale(String language, String country)

First argument specifies the language and the second one specifies the country to support. The language argument is a valid ISO language code (two letter code and in lower case) defined by ISO-639. In the same way, country argument is also a valid ISO country code (two letter code but in upper case) defined by ISO-3166. For example, the list of some language code and country code is given below:

Language Code	Country Code	Description
en	US	English & USA
fr	FR	French & France
es	ES	Spanish & Spain
en	GB	English & Great Britain
fr	CA	French & Canada

Find full list of language codes from:

http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

Find full list of country codes from:

http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

Output:

When the set of language code and country code is provided at run time, the output is shown according to that particular locale. For example, in the output below, for "es" and "ES" the output is displayed in Spanish language. The text displayed is fetched from **MessageBundle_es_ES.properties** file. If "fr" and "FR" codes are provided at run time then values are displayed in French language. Now this text is fetched from **MessageBundle_fr_FR.properties** file. Here we are not changing and compiling the source code to work for different locale because the program references the keys, not the values.

```
C:\JavaJazzup>javac InternationalizationDemo.java

C:\JavaJazzup>java InternationalizationDemo
The text displayed is specific to locale (English, United States).

Hello, how are you?
Thanks to visit.

C:\JavaJazzup>java InternationalizationDemo en US
The text displayed is specific to locale (English, United States).

Hello, how are you?
Thanks to visit.

C:\JavaJazzup>java InternationalizationDemo es ES
El texto mostrado es especΦfico al lugar (Spanish, Spain).

¡Hola, ¿Cόmo estβ usted?.
Gracias a visita.

C:\JavaJazzup>java InternationalizationDemo fr FR
Le texte affichθ est spθcifique α la scδne (French, France).

Bonjour, comment allez-vous ?
Merci pour visiter.

C:\JavaJazzup>_
```

JSF Tags: Tomahawk Tags

Tomahawk tags are collection of standard components with extended functionality and many more extra set of components with rich set of functionality. If you are going to create a web application in JSF then Tomahawk will be a great weapon because it contains all the commonly used set of components with large options of functionalities. The best way to justify the above fact is exploring some examples of Tomahawk tags. Have a glance on some good tomahawk tags below.

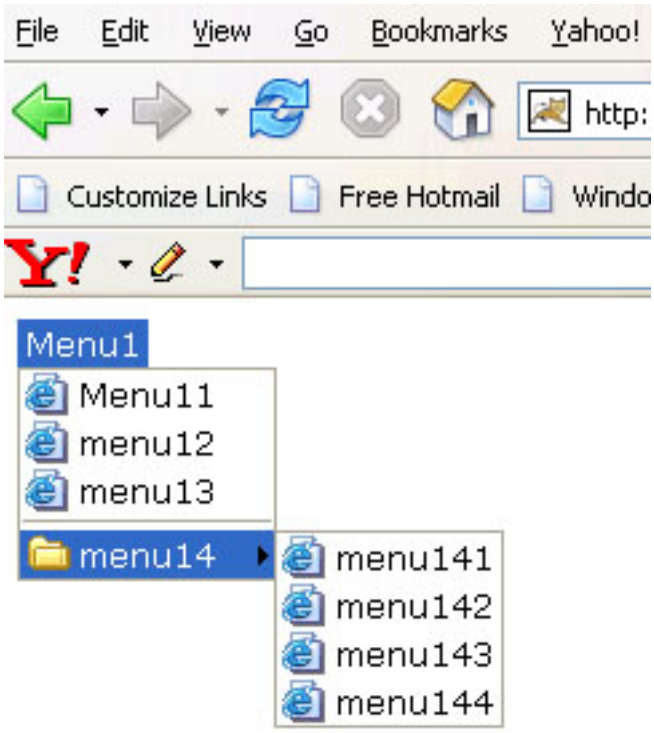
1.Tomahawk jscookMenu tag:

This tag is used to create a menu component in a web page. To display the items navigationMenuItem or navigationMenuItems tag are used. In the following example, we have used theme that looks just as in Internet Explorer.

Code Description:

The **"layout"** attribute provides the layout for the components. The values for this attribute can be hbr, hbl, hur, hul, vbr, vbl, vur, vul. The **"theme"** attribute is used to specify the theme to be applied on the component. The values for this attribute may be ThemeIE, ThemeMiniBlack, ThemeOffice, ThemePanel. "ThemeIE" value gives a look like Internet Explorer. The **navigationMenuItem** tag is used to provide the menu items and sub items. The label for the item is set using **itemLabel** attribute and the action to perform is specified in the **action** attribute. The icon displayed before the item is set to the **icon** attribute. Its **split** attribute is used when we want its sub items.

```
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>
<html>
<head>
<title>t:jscookMenu example</title>
</head>
<body>
<f:view>
<h:form>
<t:jscookMenu layout="hbr"
theme="ThemeIE" >
<t:navigationMenuItem id="nmi1"
itemLabel="Menu1">
 <t:navigationMenuItem id="nmi1_1"
itemLabel="Menu11" action="welcome" />
 <t:navigationMenuItem id="nmi1_2"
itemLabel="menu12" action="welcome" />
<t:navigationMenuItem id="nmi1_3"
itemLabel="menu13" action="welcome" />
<t:navigationMenuItem id="nmi1_4"
itemLabel="menu14" split="true">
<t:navigationMenuItem id="nmi14_1"
itemLabel="menu141" action="welcome" />
<t:navigationMenuItem id="nmi14_2"
itemLabel="menu142" action="welcome"/>
<t:navigationMenuItem id="nmi14_3"
itemLabel="menu143" action="welcome" />
<t:navigationMenuItem id="nmi14_4"
itemLabel="menu144"action="welcome"/>
</t:navigationMenuItem>
</t:navigationMenuItem>
</t:jscookMenu>
</h:form>
</f:view>
</body>
</html>
```



Rendered Output:

2.Tomahawk inputFileUpload tag :

File uploading is the concept of uploading the file to the server. In Tomahawk, the component for this purpose can be created using **<t:inputFileUpload>** tag. Do remember to include **"enctype"** attribute in the form tag and set to **"multipart/ form-data"**. You must enable the MultiPart Filter to make the component work. For this, you have to add the code below in the **web.xml** file.

```
<filter>
<filter-name>extensionsFilter
</filter-name>
<filter-class>
org.apache.myfaces.webapp.filter.ExtensionsFilter
</filter-class>
<init-param>
<description>Set the size limit for
uploaded files.
 Format: 10 - 10 bytes
 10k - 10 KB
 10m - 10 MB
1g - 1 GB
</description>
<param-name>uploadMaxFileSize
</param-name>
<param-value>100m</param-value>
</init-param>
<init-param>
<description>Set the threshold size - files
below this limit are stored in memory, files
above this limit are stored on disk.
Format: 10 - 10 bytes
10k - 10 KB
10m - 10 MB
1g - 1 GB
</description>
<param-name>uploadThresholdSize
</param-name>
<param-value>100k</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>extensionsFilter
</filter-name>
<url-pattern>*.jsf</url-pattern>
</filter-mapping>
<filter-mapping>
<filter-name>extensionsFilter
</filter-name>
<url-pattern>/faces/*</url-pattern>
</filter-mapping>
```

Code Description:

In the code below, **inputFileUpload** tag creates the file upload component. We have also created a button that when clicked, causes a method **upload()** of the backing bean(**FileUploadForm.java**) to be called. In this code, the output text for showing success and failure is rendered based on the condition of success / failure in uploading the file. For this, rendered attribute has been used.

```
<%@taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk" prefix="t"
%>
<html>
<head>
<title>t:inputFileUpload example</title>
</head>
<body>
<f:view>
<h:form id="welcomeForm"
enctype="multipart/form-data">
<t:inputFileUpload id="fileupload"
value="#{FileUploadForm.upFile}
"size="20" />
<p/>
<h:commandButton value="Load the
file"
action="#{FileUploadForm.upload}"
/>
<t:outputText value="File Uploaded
Successfully."
rendered="#{FileUploadForm.rendSuccess}"
style="color:green;font-weight:bold"/>
<t:outputText value="Error in File
Uploading."
rendered="#{FileUploadForm.rendFailure}"
style="color:red;font-weight:bold"/>
</h:form>
</f:view>
</body>
</html>
```

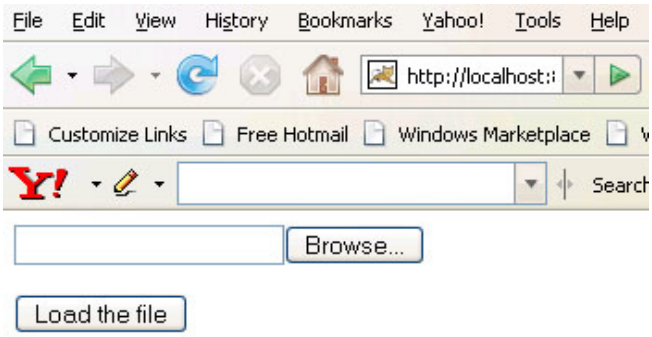
FileUploadForm.java :

```
package net.roseindia.web.ui;

import java.io.*;
import javax.servlet.http.*;
import
org.apache.myfaces.custom.fileupload.UploadedFile;
import javax.faces.context.FacesContext;

public class FileUploadForm{
   private UploadedFile upFile;
    boolean rendSuccess=false;
    boolean rendFailure=false;

    public FileUploadForm(){
     }

      public UploadedFile getUpFile(){
         return upFile;
      }

      public void setUpFile(UploadedFile
upFile){
         this.upFile = upFile;
      }

     public boolean getRendSuccess(){
         return rendSuccess;
      }

     public void setRendSuccess(boolean
rendSuccess){
         this.rendSuccess = rendSuccess;
      }

     public boolean getRendFailure(){
         return rendFailure;
      }

     public void setRendFailure(boolean
rendFailure){
         this.rendFailure = rendFailure;
      }

      public String upload() throws
IOException{
      try {
        InputStream stream =
upFile.getInputStream();
        long size = upFile.getSize();
        byte [] buffer = new byte[(int)size];
        stream.read(buffer, 0, (int)size);
        stream.close();
        rendSuccess=true;
        rendFailure=false;
        System.out.println("File Upload
Successful.");
        return "ok";
      }
      catch (Exception ioe) {
        System.out.println("File Upload
Unsuccessful.");
```

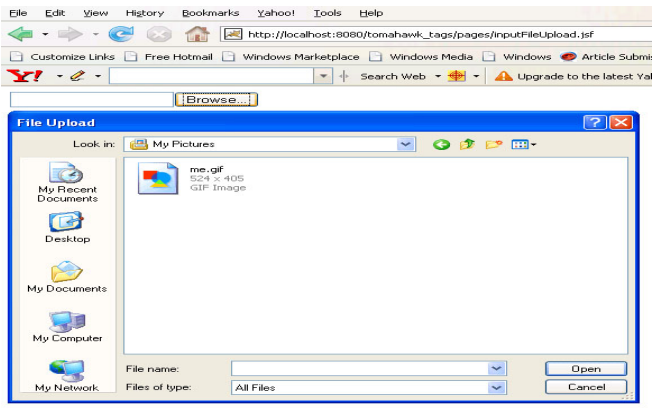
```
    rendSuccess=false;
    rendFailure=true;
    return "no";
      }
  }
}
```

Rendered Output:

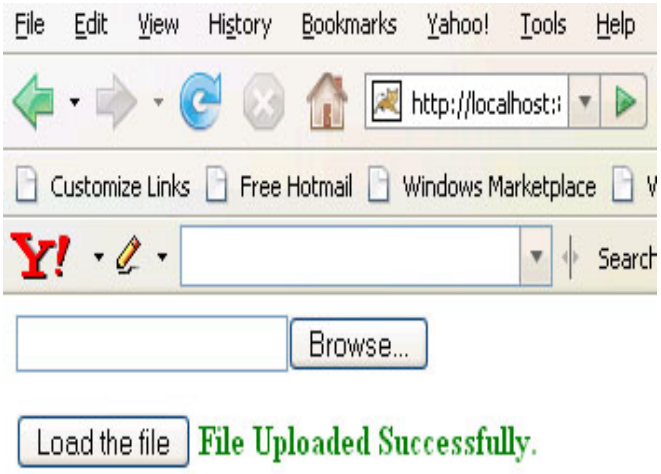
This is the output of the above code. The file upload component is the combination of an input box and a button for choosing a file to be uploaded.



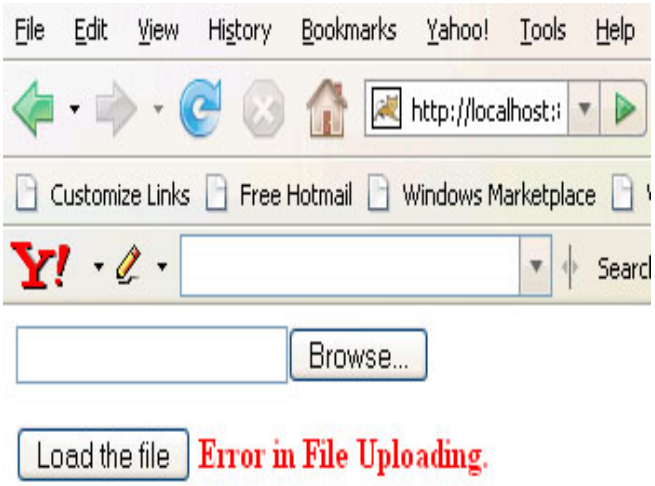
The browse button and select one file to upload it.



The following figure appears when a file is successfully uploaded.



The following figure appears when there is any problem in uploading a chosen file.



3. Tomahawk inputCalendar tag:

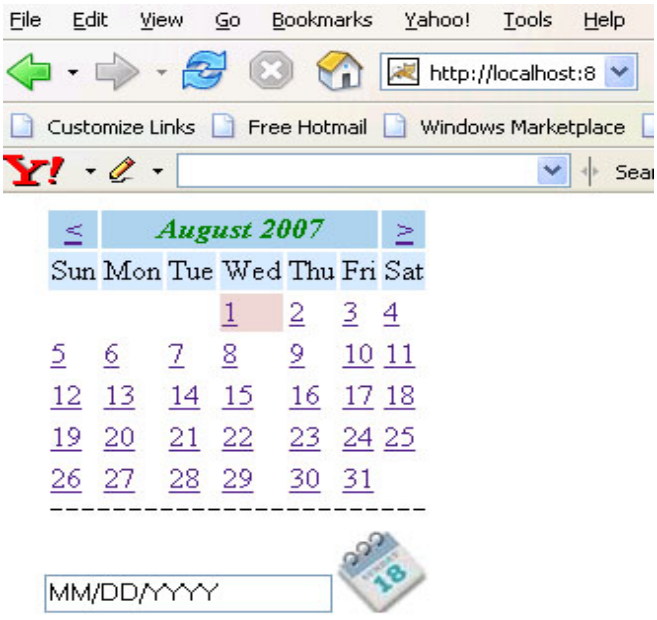
This tag is used to create calendar component in the page. It can be created in different styles depending on the value of **renderAsPopup** attribute of the component. If **renderAsPopup** attribute is set to **true** then it creates an input box along with a button which when clicked renders a popup calendar. We can select any date and it is displayed in the input box in our required format. A help text can also be given in the input box to help the user

to input the date in the specified format. The current date can be displayed in the popup window in our required format with some string. Many attributes are there to use CSS in different parts of the component. In the following example, we can see how the component can be used in our page.

Code Description :

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>

<f:view>
  <t:document>
    <t:documentHead>
      <meta http-equiv="Content-Type"
content="text/html;
          charset=iso-8859-1">
      <title>t:inputCalendar example</
title>
      <style type="text/css">
        <!—
        .yearRowStyle {
            background-color: #A8D1E8;
            color: green;
            text-align: center;
            font-weight: bold;
            font-style:italic;
           }
        .weekRowStyle {
            background-color: #D6EBFC;
           }
        .selectedDayCellStyle {
           background-color: #ECD5D2;
           }
        —>
      </style>
    </t:documentHead>
    <t:documentBody ><center>
      <h:form>

          <t:inputCalendar
monthYearRowClass="yearRowStyle"
weekRowClass="weekRowStyle"
 currentDayCellClass="selectedDayCellStyle"
value="#{Calendar.selectedDate1}"/
>
<t:htmlTag value="pre">——</t:htmlTag>
<t:inputCalendar id="secondOne"
value="#{Calendar.selectedDate2}"
renderAsPopup="true"

popupDateFormat="MM/dd/yyyy"

popupTodayDateFormat="dd-MMM-yyyy"

popupWeekString="Week"

popupTodayString="The date today is :"

renderPopupButtonAsImage="true"

popupButtonImageUrl="images/cal.gif"
helpText="MM/DD/YYYY"
forceId="true"/>

</h:form></center>
</t:documentBody>
</t:document>
</f:view>
```

Rendered Output :



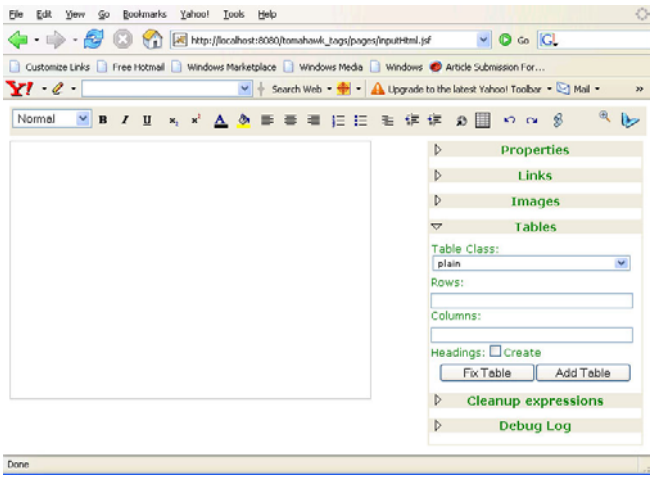
4. Tomahawk inputHtml tag:

This tag is used to create the **Kupu text editor**. To know about this you can visit the web site http://kupu.oscom.org. The important thing about this tag is that it **supports one editor per page**. You can use multiple editors if they are placed in different tags but rendered only one at a time, for example, in many tabbed panes. You can customize this editor according to your requirement. You can add different tool boxes if required. Different look and style can be given by the use of CSS.

Code Description :

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>
<f:view>
<t:document>
<t:documentHead>
<meta http-equiv="Content-Type"
content="text/html;
charset=iso-8859-1">
<title>t:inputHtml example</title>
</t:documentHead>
<t:documentBody >
<h:form>
<t:inputHtml style="height:300px;
color:green"
addKupuLogo="true"
showAllToolBoxes="true" >
</t:inputHtml>
</h:form>
</t:documentBody>
</t:document>
</f:view>
```

Rendered Output :



5. Tomahawk popup tag

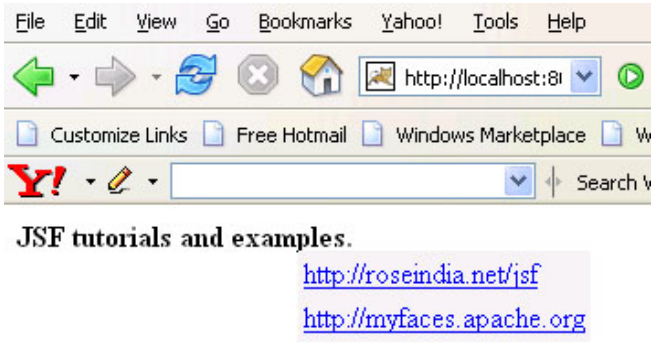
This tag is used to create the **popup window** when user takes the mouse on the element. This popup is created on the **mouse event**. It has many attributes that can give it extra functionalities. It has an attribute responsible for displaying the popup window at a certain place i.e. we can set its **horizontal distance** and **vertical distance** from the triggering element. We can also set when to close this window, either when triggering element is left or the popup window is left.

Code Description:

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>

<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
```

```
<title>t:popup example</title>
<style type="text/css">
<!—
.popClass{
background-color:#F1F1F1;
}
—>
</style>
</head>
<body >
<h:form>
<t:popup styleClass="popClass"
        closePopupOnExitingElement="true"
        closePopupOnExitingPopup="true"
        displayAtDistanceX="0"
        displayAtDistanceY="0">
<h:outputText value="JSF tutorials and
examples." style="font-weight:bold;"/>
<f:facet name="popup">
<h:panelGrid columns="1" >
<h:commandLink value="http://
roseindia.net/jsf" />
<h:commandLink value="http://
myfaces.apache.org" />
</h:panelGrid>
</f:facet>
</t:popup>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output :



6. Tomahawk dataScroller tag :

The data scroller component of tomahawk is one of the very useful component. This component can take the reference of the UIData component like dataTable, dataList in its "for" attribute. **dataScroller** tag renders a component that provides the facility to navigate through the data by scrolling.

Code Description:

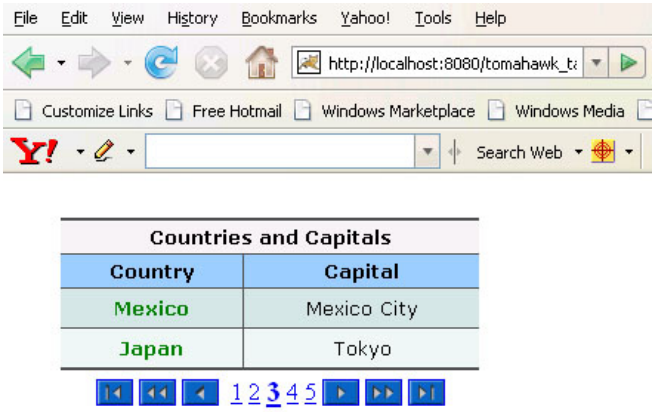
In the code below, the data scroller component for a data table has been created that helps to navigate from one set of data to the other by clicking various navigation links. There are three types of links to **move one by one**, **fast forward** and **move to the last** directly. In this example, **faststep** attribute has been set to the value 2 so **fast forward** link will take you two page ahead directly. **paginatorMaxPages** attribute sets the number of pages to be shown in the component to navigate directly clicking on them.

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:dataScroller example</title>
<style type="text/css">
<!—
body{
margin-top:30;
}
.TableRow1 {
   background-color: #D0E6E0;
}

.TableRow2 {
   background-color: #E8F7F1;
}
.TableColumn {
   text-align: center
}
.TableClass {
```

```
        font-family : verdana, Geneva, Arial,
Helvetica, sans-serif;
        font-size: 13px;
        color: #000000;
    padding: 2;
    border-style: solid;
    border-width: 2px;
}
.TableHeader {
        color: #000000;
    background-color: #F1F1F1;
    padding: 3;
    text-align: center;
    border: none;
}
—>
</style>
</head>
<body ><center>
<f:view>
<h:form id="form1" >
<t:dataTable id="dt1"
value="#{DataScrollerBean.country_Capital}"
var="item" bgcolor="#F1F1F1"
border="10" cellpadding="5"
cellspacing="3" first="0" rows="2"
width="50%" dir="LTR" frame="hsides"
rules="all" summary="This is a JSF code
to create dataTable."
rowClasses="TableRow1,TableRow2"
columnClasses="TableColumn"
styleClass="TableClass"
headerClass="TableHeader">
<f:facet name="header">
<h:outputText value="Countries and
Capitals" />
</f:facet>
<t:column style="color:green; font-
weight:bold"headerstyle="background-
color:#99CCFF;">
<f:facet name="header">
<t:outputText value="Country" />
</f:facet>
<t:outputText
value="#{item.country}"></
t:outputText>
</t:column>
<t:column headerstyle="background-
color:#99CCFF;">
<f:facet name="header">
<t:outputText value="Capital"/>
</f:facet>
<t:outputText value="#{item.capital}">
</t:outputText>
</t:column>
</t:dataTable>
<t:dataScroller id="scroller"
for="dt1" paginator="true"
fastStep="2" paginatorMaxPages="5"
paginatorActiveColumnStyle="font-
size:20px;font-weight:bold;"
immediate="true">
<f:facet name="first" >
<t:graphicImage url="images/first.gif"
border="1" />
</f:facet>
<f:facet name="last">
<t:graphicImage url="images/last.gif"
border="1" />
</f:facet>
<f:facet name="previous">
<t:graphicImage url="images/pre1.gif"
border="1" />
</f:facet>
<f:facet name="next">
<t:graphicImage url="images/next1.gif"
border="1" />
</f:facet>
<f:facet name="fastforward">
<t:graphicImage url="images/next2.gif"
border="1" />
</f:facet>
<f:facet name="fastrewind">
<t:graphicImage url="images/pre2.gif"
border="1" />
</f:facet>
</t:dataScroller>
</h:form>
</f:view>
</center>
</body>
</html>
```

Rendered Output :

The figure below shows that when we click on fast forward element then it moves ahead by 2 pages because "**fastStep**"



attribute has been set to the value "2".

Backing bean (DataScrollerBean.java)

```
package net.roseindia.web.ui;
public class DataScrollerBean {

private CountryCapital[] country_Capital =
new CountryCapital[]{
new CountryCapital("India","New Delhi"),
new CountryCapital("United
States","Washington D.C."),
new CountryCapital("Russia","Moscow"),
new CountryCapital("United
Kingdom","London"),
new CountryCapital("Mexico","Mexico City"),
new CountryCapital("Japan","Tokyo"),
new CountryCapital("Germany","Berlin"),
new CountryCapital("France","Paris"),
new CountryCapital("China","Beijing"),
new
CountryCapital("Denmark","Copenhagen"),
new CountryCapital("Brazil","Brasilia"),
new CountryCapital("Bangladesh","Dhaka"),
new CountryCapital("Australia","Canberra"),
new CountryCapital("Fiji","Suva")

};

public CountryCapital[]
getCountry_Capital() {
return country_Capital;
}

public class CountryCapital {
  String country;
  String capital;
public CountryCapital(String
  country,String capital) {
    this.country = country;
    this.capital = capital;
  }
  public String getCountry() {
  return country;
  }

  public String getCapital() {
  return capital;
  }
}

}
```

7. Tomahawk panelTabbedPane tag :

This tag is used to create a **tabbed pane**. Its has the capability of switching the tabs at **client side** or **server side**. If its "**serverSideTabSwitch**" attribute is set to **true** then switching is server side otherwise it is client side. If we want a specific tab to be opened by default when it is rendered then we can set its "**selectIndex**" attribute to any integer value. Here index value starts from 0 i.e. setting the value to 0 indicates to the first tab and 1 to the second tab and so on. This

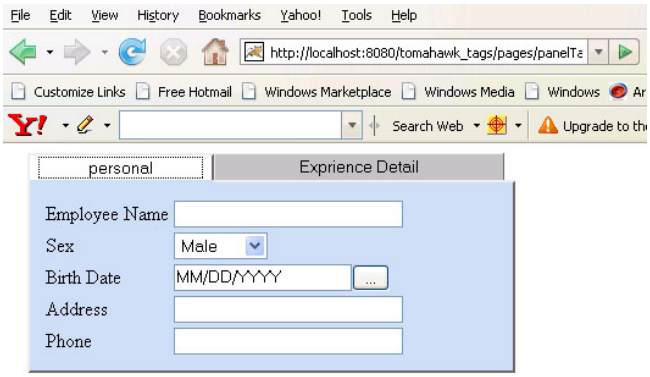
tag offers a lot of attributes to make it according to your need of look and functionality. The "**panelTab**" tag is used with "**panelTabbedPane**" tag to create tabs for panel. It has **label** attribute to give label for the tab

Code Description:

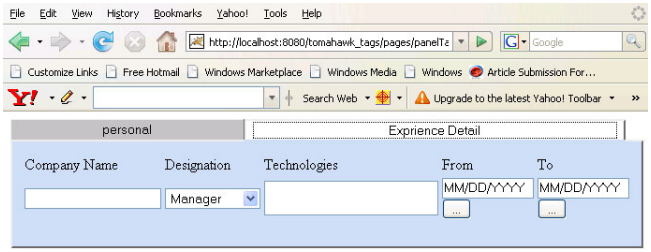
```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f"%>
<%@ taglib uri="http://
myfaces.apache.org/tomahawk"
prefix="t"%>
<html>
<head>
<title>t:panelTab example<title>
</head>
<body>
<center>
<f:view>
<t:panelTabbedPane width="400"
bgcolor="#cddcf6" align="center"
serverSideTabSwitch="false"
selectedIndex="0">
<t:panelTab label="personal"
rendered="true">
<h:form id="form1">
<t:panelGrid columns="2" border="0">
<t:outputText value="Employee Name"/>
<t:inputText id="empFn" size="27" />
<t:outputText value="Sex"/>
<t:selectOneListbox id="empSex"
value="" size="1" required="true" >
<f:selectItem id="sex1" itemLabel="Male"
itemValue="sex1" />
<f:selectItem id="sex2"
itemLabel="Female" itemValue="sex2" />
</t:selectOneListbox>
<t:outputText value="Birth Date"
styleClass="font11bold"/>
<t:inputCalendar id="empDOB"
value="" renderAsPopup="true"
popupDateFormat="MM/dd/yyyy"
helpText="MM/DD/YYYY" forceId="true"/>
<t:outputText value="Address"
styleClass="font11bold"/>
<t:inputText id="empAdd" size="27" />
<t:outputText value="Phone"
styleClass="font11bold"/>
<t:inputText id="empPhone" size="27" />
</t:panelGrid>
</h:form>
</t:panelTab>
<t:panelTab label="Exprience Detail"
rendered="true">
<h:form id="form2">
<t:panelGrid columns="5" border="0">
<t:outputText value="Company Name"/>
<t:outputText value="Designation"/>
<t:outputText value="Technologies"/>
<t:outputText value="From"/>
<t:outputText value="To"/>
<t:inputText id="cmpname" />
<t:selectOneListbox id="designation"
value="" size="1"  required="true">
<f:selectItem id="one"
itemLabel="Manager" itemValue="one" />
<f:selectItem id="two" itemLabel="Jr.
Manager" itemValue="two" />
<f:selectItem id="three" itemLabel="Clerk"
itemValue="three" />
<f:selectItem id="four"
itemLabel="Engineer" itemValue="four" />
</t:selectOneListbox>
<t:inputTextarea id="technologies"
rows="1" />
<t:inputCalendar id="fromdate" value=""
renderAsPopup="true"
 popupDateFormat="MM/dd/yyyy"
helpText="MM/DD/YYYY"
 forceId="true" size="12"/>
<t:inputCalendar id="todate" value=""
renderAsPopup="true"
popupDateFormat="MM/dd/yyyy"
helpText="MM/DD/YYYY"
forceId="true" size="12"/>
</t:panelGrid>
</h:form>
</t:panelTab>
</t:panelTabbedPane>
</f:view>
</center>
</body>
</html>
```

JSF Tags: Tomahawk Tags

Rendered Output:



When we click on the second tab we get the following image as shown below.



Building Projects: Maven2

Learn to Set Up A Maven2 Repository

Maven2 is an Open Source build tool that made the revolution in the area of building projects. Like the build systems as "make" and "ant" it is not a language to combine the build components but it is a build lifecycle framework. A development team does not require much time to automate the project's build infrastructure since maven uses a standard directory layout and a default build lifecycle. Different development teams, under a common roof can set-up the way to work as standards in a very short time. This results in the automated build infrastructure in more stable state. On the other hand, since most of the setups are simple and reusable immediately in all the projects using maven therefore many important reports, checks, build and test animation are added to all the projects. Which was not possible without maven because of the heavy cost of every project setup.

Maven 2.0 was first released on 19 October 2005 and it is not backward compatible with the plugins and the projects of maven1. In December 2005, a lot of plugins were added to maven but not all plugins that exists for maven1 has been ported yet. Maven 2 is expected to be stabilized quickly with most of the Open Source technologies. People are introduced to use maven as the core build system for Java development in one project and a multi-project environment. After a little knowledge about the maven, developers are able to setup a new project with maven and also become aware of the default maven project structure. Developers are easily enabled to configure maven and its plugins for a project. Developers enable common settings for maven and its plugins over multiple projects, how to generate, distribute and deploy products and reports with maven so that they can use repositories to set up a company repository. Developers can also know about the most important plugins and about how to install, configure and use them, just to look for other plugins to evaluate them so that they can be integrated in their work environment.

Maven is the standard way to build projects and it also provides various other characters like clearing the definition of the project, ways to share jars across projects. It also provides the easy way to publish project information (OOS).

Originally maven was designed to simplify the building processes in the Jakarta Turbine project. Several projects were there containing their own slightly different Ant build files and JARs were checked into CVS. An apache group's tool that can build the projects, publish project information, defines what the project consists of and that can share JARs across several projects. The result of all these requirement was the maven tool that builds and manages the java-based-project.

Maven: Features

1. Portable: Maven is portable in nature because:

- Building configuration using maven are portable to another machine, developer and architecture without any effort

- **Non trivial:** Maven is non trivial because all file references need to be relative, environment must be completely controlled and independent from any specific file system.

2. Technology: Maven is a simple core concept that is activated through IoC container (Plexus). Everything is done in maven through plugins and every plugin works in isolation (ClassLoader). Plugings are downloaded from a plugin-repository on demand.

Maven's Objectives: The primary goal of maven is to allow the developers to comprehend the complete state of a project in the shortest time by using easy

build process, uniform building system, quality project management information (such as change Log, cross-reference, mailing lists, dependencies, unit test reports, test coverage reports and many more), guidelines for best practices and transparent migration to new features. To achieve to this goal Maven attempts to deal with several areas like:

- It makes the build process easy
- Provides a uniform building system
- Provides quality related Poject information
- Provides guidelines related to development to meet the best goal.
- Allows transparent migration to new features.

Maven repository Types:

- **Public remote external repository:** This public external repository exists at ibiblio.org and maven synchronizes with this repository.

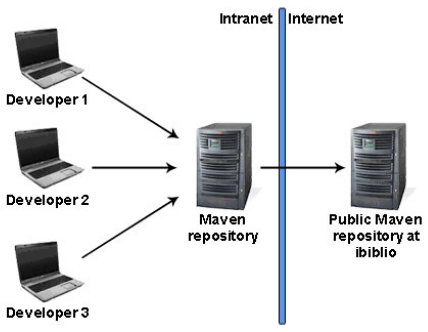
- **Private remote internal repository:** We set up this repository and make changes in the maven's pom.xml or settings.xml file to use this repository.

- **Local repository:** This repository is maintained by the developer and stays on the developer's machine. It is synchronous to the maven repository defined in the settings.xml file that exists in the .m2 directory at its standard location **i.e. C:\Documents and Settings\Administrator.** If no private internal repository is setup and not listed in the pom.xml or in the setting.xml then the local repository exists on the developer's machine is synchronized with the public maven repository at **ibiblio.org.**

Advantages of having an internal private repository :

- Reduces conflicts among likelihood versions.
- To build first time it requires less manual intervention.
- Rather than having several separate independent libraries it provides a single central reference repository for all the dependent software libraries.
- It quickly builds the project while using an internal repository as maven artifacts are retrieved from the intranet server rather than retrieving from the server on internet.

Internal private repository of an organisation:



Use cases for maven repository:

- It creates two sub-repository inside the internal repository.

- Downloads ibiblio-cache from ibiblio for artifacts and make it available publically. This synchronizes with external repository from ibiblio.

- internal-maven-repository: used for internal artifacts of an organization. It contains unique artifacts for the organization and is not synchronized with any repository.

- Alternatively, another sub-repository that is not at ibiblio can be created for artifacts. This does not synchronize with any external repository.

- Browse the remote repository by using a web browser.

- Search the artifacts in the repository.

- Download code from version control and make changes in settings.xml to point to the internal repository and build without any manual intervention.

- Install new version of the artifacts.

- Import artifacts into the repository in bulk.
- Export artifacts from the repository in bulk.
- Setup the task to backup the repository automatically.

Criteria for choosing a maven repository implementation: In ideal condition a maven repository implementation should be:

- Free and open source
- Provide admin tools
- Easy to setup and use
- Provide backup facility
- Able to create, edit and delete sub repositories.
- Anonymous read only access and also access control facility.
- Deployable in any standard web server such as Tomcat or Apache.
- Issue tracker, forums and other independent source of information.
- Active community developers make the product enhanced and bugs fixed.
- Bulk import/export facility to move groups of artifacts into the repository and out of the repository.
- Provide a repository browser: should be a web browser instead of the desktop application.

Getting Hands On Setting Up Internal Private Repository For An Organisation

We have set up an internal Maven Repository for our organisation so that the developers are not wasting time in searching and downloading the required project libraries. This also allows us to have a single company wide repository for project artifacts. The setup steps are not too much complicated but we didn't run into several issues while settingup the local repository server (artifactory), for the first time.

When you're using maven at your company, you almost always want to setup local maven repository as relying on ibiblio for nightly / production builds is not a great idea and it takes time to download the library files if your development team is big. When setting up a local repository you don't want to setup the entire ibiblio repository locally as it is huge and has more libraries than you'll ever be using for your project. Maven-repository (in our case maven artifactory) is a repository server, which acts as your internal maven repository and downloads jars from ibiblio or other public maven repositories on demand and store it for further use in the project builds. And all this is transparent to the developer running a maven build.

The other thing about local maven-repository is that it allows you to neatly separate and organize jars that might not be available on ibiblio i.e. the 3rd-party artifacts (some company specific shared library or a commercial library).

Software requirements to set up the maven repository:

- Artifactory: Download and install the artifactory from the site http://www.jfrog.org/sites/artifactory/latest/. Artifactory comes with the application that can be installed into Tomcat.
- JDK 1.6: Get the information for downloading and installation from the site http://www.jfrog.org/sites /artifactory/latest/install .html.
- Tomcat 6.0

A Quick glance at the steps of how we set up our local maven repository :

1 Download the appropriate Artifactory.zip file. You can get it from http://www.jfrog.org/sites/artifactory/latest/. Download and unzip the file in your directory of choice. We have downloaded artifactory-1.2.1.zip at our end.

2 Lets take the Installation directory as D:\artifactory-1.2.2\artifactory-1.2.2, Extract the artifactory-1.2.1.zip into the <artifactory-install-directory> directory.

3 Create repository configuration file artifactory.config.xml into <artifactory-install-directory>/etc/ directory and paste the following content in the artifactory.config.xml file:

```
<config xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
xmlns="http://artifactory.jfrog.org/xsd/
1.0.0" xsi:schemaLocation="http://
artifactory.jfrog.org/xsd/1.0.0
http://www.jfrog.org/xsd/artifactory-
v1_0_0.xsd">
<!-- Backup every 12 hours -->
<!--<backupCronExp>0 0 /12 * * ?</
backupCronExp>-->

<localRepositories>

<localRepository>
<key>private-internal-repository</key>
<description>Private internal repository</
description>
<handleReleases>true</handleReleases>
<handleSnapshots>true</
handleSnapshots>
</localRepository>

<localRepository>
<key>3rd-party</key>
<description>3rd party jars added
manually</description>
<handleReleases>true</handleReleases>
<handleSnapshots>false</
handleSnapshots>
</localRepository>

</localRepositories>

<remoteRepositories>

<remoteRepository>
<key>ibiblio</key>
<handleReleases>true</handleReleases>
<handleSnapshots>false</
handleSnapshots>
<excludesPattern>org/artifactory/**,org/
jfrog/**</excludesPattern>
<url>http://repo1.maven.org/maven2
</url>
<proxyRef>proxy1</proxyRef>
</remoteRepository>

<remoteRepository>
<key>ibiblio.org</key>
<description>ibiblio.org</description>
<handleReleases>true</handleReleases>
<handleSnapshots>false</
handleSnapshots>
<excludesPattern>org/artifactory/**</
excludesPattern>
<url>http://www.ibiblio.org/maven2</
url>
<proxyRef>proxy1</proxyRef>
</remoteRepository>

<remoteRepository>
<key>java.net</key>
<description>java.net</description>
<handleReleases>true</handleReleases>
<handleSnapshots>false</
handleSnapshots>
<excludesPattern>org/artifactory/**,org/
jfrog/**</excludesPattern>
<url>http://download.java.net/maven/
2</url>
<proxyRef>proxy1</proxyRef>
</remoteRepository>

</remoteRepositories>

<proxies>
<proxy>
<key>proxy1</key>
<host>192.168.10.80</host>
<port>9090</port>
<username></username>
<password></password>
<domain>192.168.10.80</domain>
</proxy>
</proxies>

</config>
```

At our end we are using squid proxy server to connect with the internet. Our artifactory repository server will connect to internet through squid proxy server

running on the machine 192.168.10.80 at port 9090. You can also use artifactory server without proxy. We have made two local and three remote repositories.

4 Install Tomcat 6.0 Download and install tomcat 6.0 on your machine.

5 Copy artifactor.war from artifactory-1.2.1\webapps to the webapps folder of your installed tomcat directory.

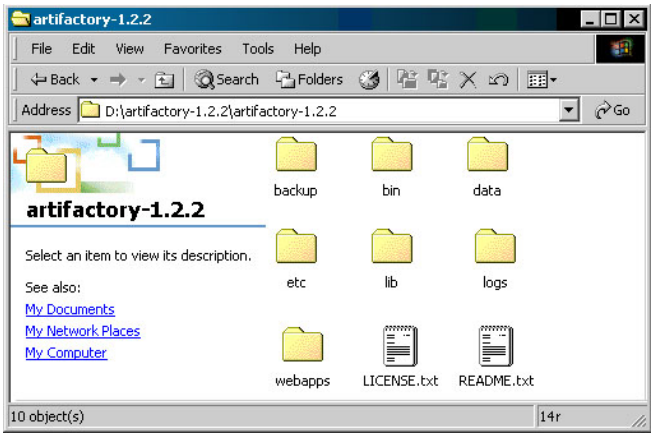
6 Specify the local artifactory installation folder to the tomcat environment.
Go to Start-->Programs --> Apache Tomcat 6.0--> Configure Tomcat and specify the installation folder to the Java Options. -artifactory.home=<artifactory-install-directory>

7 Now start the tomcat and configure your clients to use maven artifactory repository.

8 To access the admin control panel type http://<server>:<port>/artifactory and login as User: admin and Password: password.

Let's Dig the things Deeper

I. Directory Structure of Artifactory-1.2.2 : Here are the folders that are shipped with the Artifactory-1.2.2.zip file.



- **bin:** batch files are used to run the included jetty web server.
- **lib:** Contains all the jar files required to run any application.
- **webapps:** Contains the war files of an application. We can also copy and install it in tomcat.
- **logs:** Includes all the log files.
- **backup:** Backs up the repository. We can use 'cron' expressions to setup the backup policy and Quartz scheduler to run the backup at the specified time. The backup interval is specified in the config.xml file inside the 'ARTIFACTORY_INSTALLATION _FOLDER>/etc/artifactory folder'.
- **data:** Includes the derby database files. If you are interested to clean up the repository then all the things containing in this folder are deleted. In case of new installation process this folder is empty.
- **etc:** Includes the artifactory configuration files "artifactory.config.xml", "jetty.xml" and "log4j.properties".

II. Deployment in Tomcat 6.0 :
Deploy the 'war' file of your application in '<ARTIFACTORY_INSTALLATION _FOLDER>/webapp' to '<TOMCAT_INSTALLATION_FOLDER> / webapps'. There is no need to change the configuration with jdk1.6 and Tomcat 6.0. Tomcat detects the web application and deploy it.

Once the application is deployed successfully, the web application requires the following information:

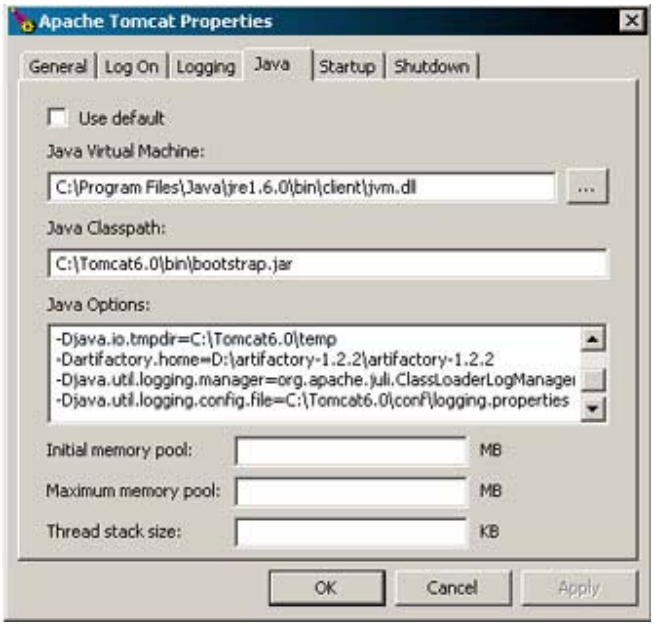
* Database location to store artifacts
* Artifactory config xml file location
* Backup folder location

To specify all the above three information, a single configuration is used. We only need to specify the artifactory installation folder location during Tomcat startup and artifactory does all the rest of the task by

itself. There is another approach, that is, setup the connection to the derby database using jdbc and configure artifactory in the web application (by including artifactory.config.xml in the web application).

We can use the environment variable to specify the location of the artifactory installation folder. In case of Windows, we can add it to Tomcat startup options and Configure Tomcat by specifying the installation folder to the Java Options.

-Dartifactory.home=<artifactory-install-directory>
e.g. -artifactory.home=D:\artifactory-1.2.2\artifactory-1.2.2



III. Setting up the maven repositories:
First create three repositories or we can say sub-repositories given in the following list in our maven repository.

- **Private-internal-repository:** The artifacts which are used only within the organization are contained in this repository. The developer uploads them manually. Since the artifacts in this repository (sub-repository) are private to the organization, this repository does not synchronize with the remote repository with ibiblio.

- **Ibiblio-cache:** This repository is the cache of the artifacts from ibiblio and synchronized with ibiblio.

- **3rd party:** Publicly available artifacts are contained in this repository but not in ibiblio repository. Since ibiblio does not have these jar files therefore this repository is not synchronized with ibiblio.

This is configured in the <ARTIFACTORY_INSTALLATION _FOLDER>/etc/artifactory .config.xml'. The configuration to setup these 3 repositories is shown below:

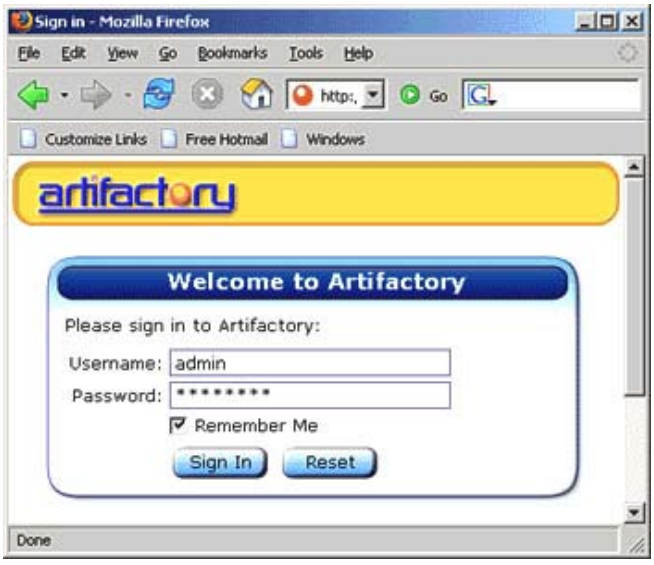
```
<config xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
xmlns="http://artifactory.jfrog.org/xsd/
1.0.0" xsi:schemaLocation="http://
artifactory.jfrog.org/xsd/1.0.0
http://www.jfrog.org/xsd/artifactory-
v1_0_0.xsd">
<!-- Backup every 12 hours -->
<!--<backupCronExp>0 0 /12 * * ?</
backupCronExp>-->

<localRepositories>

<localRepository>
<key>private-internal-repository</key>
<description>Private internal repository</
description>
<handleReleases>true</handleReleases>
<handleSnapshots>true
</handleSnapshots>
</localRepository>

<localRepository>
<key>3rd-party</key>
<description>3rd party jars added
manually</description>
<handleReleases>true</handleReleases>
<handleSnapshots>false
```

```
</handleSnapshots>
</localRepository>

</localRepositories>

<remoteRepositories>

<remoteRepository>
<key>ibiblio</key>
<handleReleases>true</handleReleases>
<handleSnapshots>false</
handleSnapshots>
<excludesPattern>org/artifactory/**,org/
jfrog/**</excludesPattern>
<url>http://repo1.maven.org/maven2</
url>
</remoteRepository>

</remoteRepositories>

</config>
```

IV. Navigate Artifactory:

Start Tomcat, open the browser and navigate **http://localhost:8080/artifactory**

Here is the artifactory home page shown below:



Sign in username as 'admin' and password as 'password'. You can view the content of the repository simply by clicking on the Repository Browser link.

V. Configuring maven to use the new repository:

You can use either of the settings.xml or the pom.xml files to configure maven to use the local repository.

Configure maven using settings.xml file: Maven uses the settings.xml file contained in .m2 folder inside the C\Document and Setting\Administrator to get the location of the maven repository. In case of no repository is specified then maven uses the default repository from ibiblio.org. We will have to make changes in the settings.xml file to use the new repository. Here is the settings.xml shown below:

```
<profiles>
<profile>
<id>dev</id>
<properties>
<tomcat6x.home>C:/InstalledPrograms/
apache-tomcat-6.0</tomcat6x.home>
</properties>
<repositories>
<repository>
<id>central</id>
<url>http://localhost:8080/artifactory/
repo</url>
```

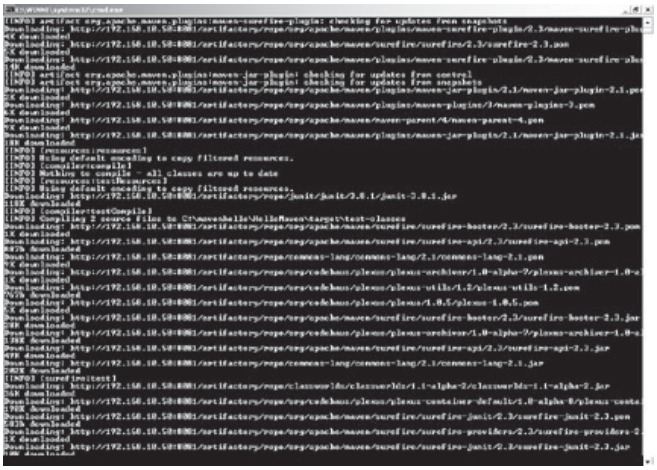
```
<snapshots>
<enabled>false</enabled>
</snapshots>
</repository>
<repository>
<id>snapshots</id>
<url>http://localhost:8080/artifactory/
repo</url>
<releases>
<enabled>false</enabled>
</releases>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>central</id>
<url>http://localhost:8080/artifactory/
repo</url>
<snapshots>
<enabled>false</enabled>
</snapshots>
</pluginRepository>
<pluginRepository>
<id>snapshots</id>
<url>http://localhost:8080/artifactory/
repo</url>
<releases>
<enabled>false</enabled>
</releases>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>
```

Configure maven using project "pom.xml" : We can also set the repository settings through the pom.xml file. Simple pom.xml is shown below:

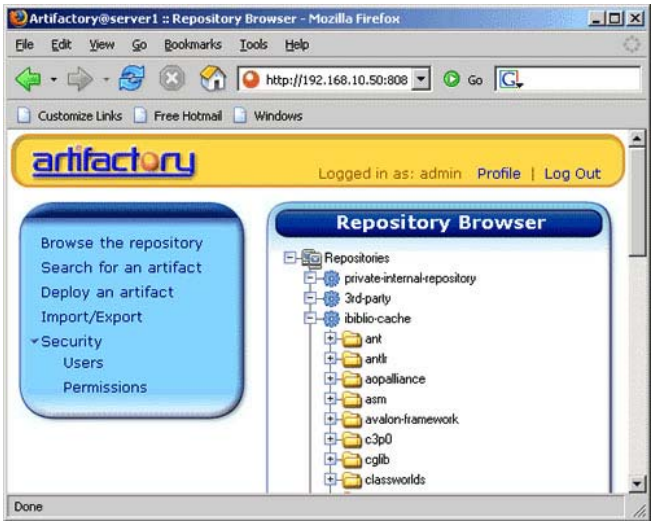
```
<project xmlns="http://
maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xsi:schemaLocation="http://
maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-
v4_0_0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>test</groupId>
<artifactId>test</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>test</name>
<url>http://maven.apache.org</url>
<repositories>
<repository>
<id>central</id>
<url>http://localhost:8080/artifactory/
repo</url>
<snapshots>
<enabled>false</enabled>
</snapshots>
</repository>
<repository>
<id>snapshots</id>
<url>http://localhost:8080/artifactory/
repo</url>
<releases>
<enabled>false</enabled>
</releases>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>central</id>
<url>http://localhost:8080/artifactory/
repo</url>
<snapshots>
<enabled>false</enabled>
</snapshots>
</pluginRepository>
<pluginRepository>
<id>snapshots</id>
<url>http://localhost:8080/artifactory/
repo</url>
<releases>
<enabled>false</enabled>
</releases>
</pluginRepository>
</pluginRepositories>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
</dependencies>
</project>
```

VI. Building using the new maven repository

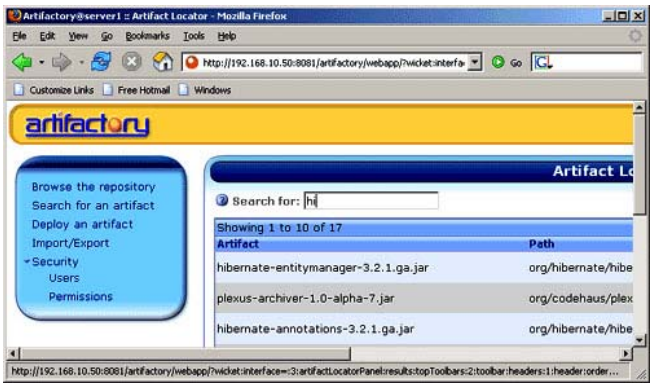
While building the maven project, all the repositories should be downloaded using the new repository. The console will show the server maven uses as shown below:



Login into your new repository through the web browser and you see that the artifactory has downloaded and cached the artifacts from ibiblio.



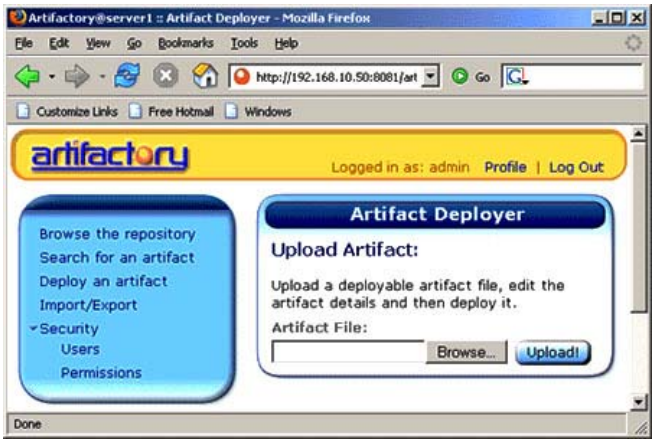
It provides an advance search facility that can be used to search an artifact simply by using the advanced search technique.



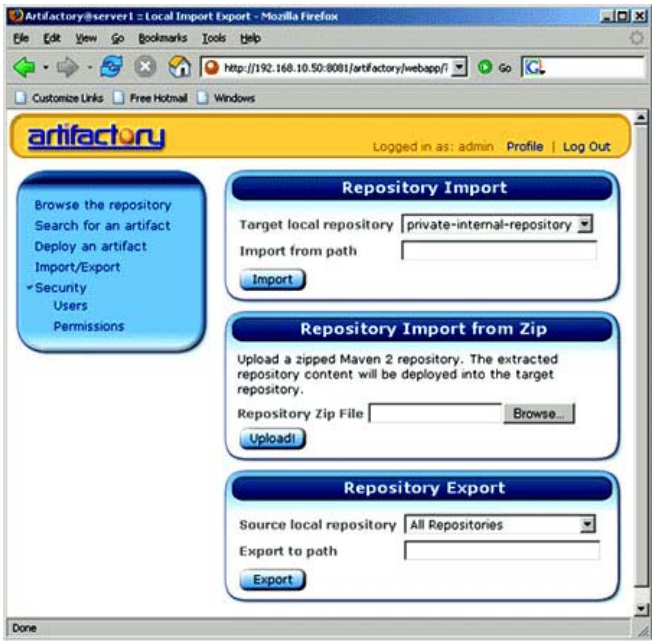
VII. Installing artifacts to the repository :

We can use the web UI or maven command line to install the artifacts. Installation process through the web UI is simpler and faster and does not require any change in the configuration. While installation through the command line requires changes in the configuration in settings.xml and then we can use it in other scripts.
Installing artifacts using the web UI: Steps involved in the installation process are shown below:

1 Upload artifact (e.g. 'jar' file or 'pom' file) and deploy using Artifact Deployer.



2 If you upload the jar file then the artifactory will create the pom.xml file as well as you can also specify which repository to upload to.



3 Once uploading is complete successfully, the artifact along with the 'pom.xml' file created by artifactory appears in the repository.

Installing artifacts from maven command line:Using the command 'maven clean install' maven only packages and installs the artifact to the local repository. We need to add the additional configuration section in the settings.xml file to install it to the local internal repository. Steps involved in whole of the process are shown below:

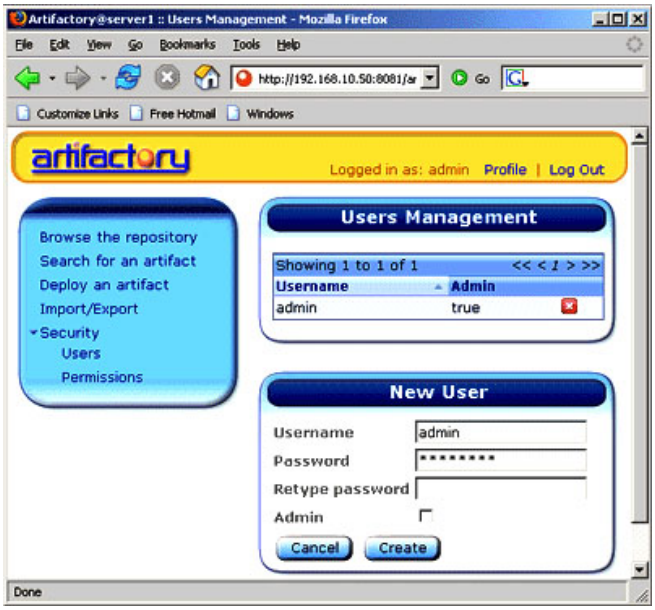
settings.xml

```
<settings>

<servers>
<server>
<id>organisation-internal</id>
<username>admin</username>
<password>password</password>
</server>
</servers>

</settings>
```

The command given below is used to install an artifact to internal maven repository.

```
mvn deploy:deploy-file -
DrepositoryId=organisation-internal -
Durl=http://localhost:8080/artifactory/
private-internal-repository-DgroupId=
test -DartifactId=test -Dversion=
1.1 -Dpackaging=jar -Dfile=target/test-
1.1.jar
```

Both the repository id and the server id defined in the settings.xml should be same. The url should include the repository name to which the artifact is to be installed. The artifactory and the new artifacts appeared in the repository creates the 'pom' (pom.xml) file for us automatically.



VIII. Other Artifactory features:

* Backup the repository: Backup policy is specified in the <ARTIFACTORY_INSTALLATION _FOLDER>/ etc/artifactory .config.xml and 'cron' expression specifies the backup configuration. Backup configuration is shown below:

```
<config xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
xmlns="http://artifactory.jfrog.org/xsd/
1.0.0"
xsi:schemaLocation="http://
artifactory.jfrog.org/xsd/1.0.0
http://www.jfrog.org/xsd/artifactory-
v1_0_0.xsd">
<!-- Backup every 12 hours -->
<backupCronExp>0 0 /12 * * ?
</backupCronExp>
<localRepositories>
<localRepository>
<key>private-internal-repository</key>
<description>Private internal repository
</description>
<handleReleases>true</
handleReleases>
<handleSnapshots>true
</handleSnapshots>
</localRepository>
<localRepository>
<key>3rd-party</key>
<description>3rd party jars added
manually</description>
<handleReleases>true
</handleReleases>
<handleSnapshots>false
</handleSnapshots>
</localRepository>
</localRepositories>
<remoteRepositories>
<remoteRepository>
<key>ibiblio</key>
<handleReleases>true
</handleReleases>
<handleSnapshots>false
</handleSnapshots>
<excludesPattern>org/artifactory/
**,org/jfrog/**</excludesPattern>
<url>http://repo1.maven.org/maven2
</url>
</remoteRepository>
</remoteRepositories>
</config>
```

The directory ‘<ARTIFACTORY_INSTALLATION _FOLDER>/backups’ contains the backups. The local repository on the developer's machine and the backups both have the same format. It allow us the repository contents to migrate easily to another implementation of maven repository.

Other features:

- Use the web UI to delete the artifacts
- Use the web UI to search for artifacts.
- Bulk import/export all artifacts in repository.
- If tomcat is not required then we can use the included jetty web server.

Conclusion: The overall conclusion is that an internal maven repository helps us to avoid conflicts due to different versions of libraries and it also speeds up the build process. Artifactory seems the better product among the 4 common maven repository. It has all the features that a good maven repository should have. The organization will not be locked into this tool since migration of the repository to another implementation is rather easy. A web UI simplifies the use of the repository even for the peoples who don't know the working of the repository.

Maven2 with JPA Example

Getting Started with Maven2 : Download Maven2 from maven.apache.org, unzip the archive into your local directory. Here we are assuming that the local repository already exists.

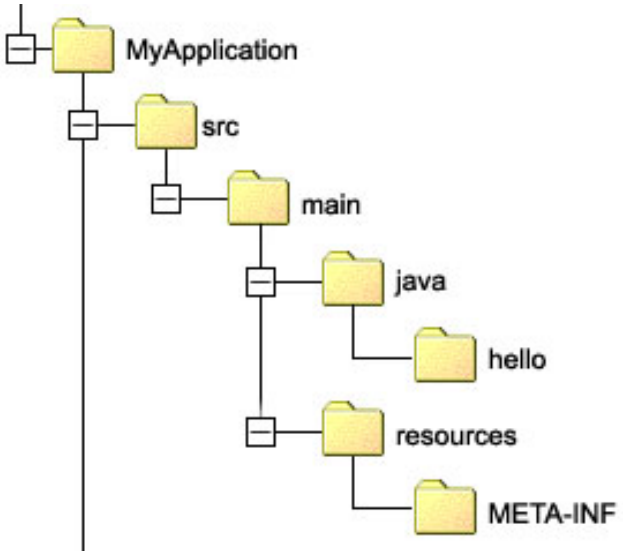
Set the JAVA_HOME variable to point to the JDK installation directory and MAVEN_HOME to point to the Maven directory and add the MAVEN_HOME/bin to the PATH environment variable.

To test, whether the path has been set properly. Type mvn -version on the command prompt.

```
C:\>mvn -versionMaven
version: 2.0.7Java version:1.5.0OS name:
"windows 2000" version: "5.0" arch:"x86"
```

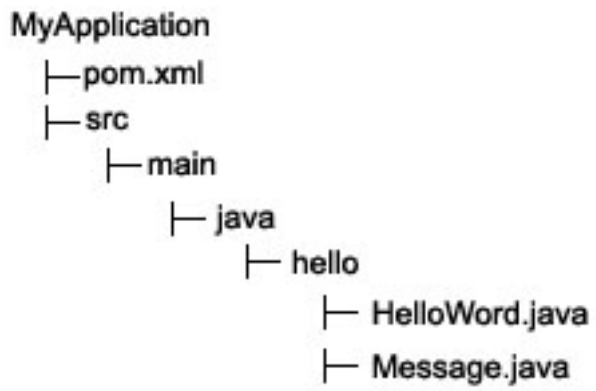
Developing java presistence example with Maven2

Here I will show, how to build a very simple example that uses java persistence API. Maven2 highly simplifies the build process, all the required libraries are downloaded automatically only when it really requires.



Directory Structure of the example: Our application must maintain a directory structure understandable by the maven project build tool. It is shown below:

Our Application includes: Our application includes the source folder "src" and an pom.xml file. Source folder contains java files, here HelloWorld.java and Message.java files are contained in the hello folder. Resource folder exists parallel to the java folder inside the main folder and contains log4j.properties file and the META-INF folder. META-INF folder contains persistence.xml file. "pom.xml" is the main file for the maven build system that includes information about the tools required to build the application.



1. Create a java class (lets take it Message.java), where you map java class/ properties to database table/fields. Java persistence annotations will do this job for us:

```
package hello;
import javax.persistence.*;
@Entity
@Table(name = "MESSAGES")
public class Message {
@Id @GeneratedValue @Column(name =
"MESSAGE_ID")
private Long id;
```

```
@Column(name = "MESSAGE_TEXT")
private String text;
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name =
"NEXT_MESSAGE_ID")
private Message nextMessage;

public Message() {}

public Message(String text) {
this.text = text;
}

public Long getId() {
return id;
}

private void setId(Long id) {
this.id = id;
}

public String getText() {
return text;
}
public void setText(String text) {
this.text = text;
}

public Message getNextMessage() {
return nextMessage;
}

public void setNextMessage(Message
nextMessage) {
this.nextMessage = nextMessage;
}
}
```

Here "Message" class is mapped to "MESSAGES" table, "id", "text" and "nextMessage" properties - to "MESSAGE_ID", "MESSAGE_TEXT" and "NEXT_MESSAGE_ID" fields.

2. Now create a simple program (HelloWorld.java) that uses persistent "Message" object:

```
HelloWorld.java
package hello;
import java.util.*;
import javax.persistence.*;
public class HelloWorld {
public static void main(String[] args) {
// Start EntityManagerFactory
EntityManagerFactory emf =
Persistence.createEntity
ManagerFactory("helloworld");
// First unit of work
EntityManager em =
emf.createEntityManager();
EntityTransaction tx =
em.getTransaction();
tx.begin();

Message message = new Message("Hello
World with JPA");
em.persist(message);

message = new Message("This is message
2");
em.persist(message);

message = new Message("This is message
3");
em.persist(message);
tx.commit();
em.close();
// Second unit of work
EntityManager newEm =
emf.createEntityManager();
EntityTransaction newTx =
newEm.getTransaction();
newTx.begin();

List messages =
newEm.createQuery("select m from
Message m order by m.text
asc").getResultList();

System.out.println( messages.size() + "
message(s) found:" );

for (Object m : messages) {
Message loadedMsg = (Message) m;
System.out.println(loadedMsg.getText());
}
```

```
newTx.commit();
newEm.close();
// Shutting down the application
emf.close();
}
}
```

This example does not refer to any persistent framework directly. Instead, it uses symbolic names to access the framework in indirect way. In the above mentioned example we have used "helloworld" name to refer to the ORM framework.

3. In this example we used Hibernate (http://hibernate.org) as persistence framework and hsqldb (http://hsqldb.org) as database. Let's take a look at the hibernate configuration file (persistence.xml) where we describe "helloworld" factory:

persistence.xml

```
<persistence xmlns="http://java.sun.com/
xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/
xml/ns/persistence
http://java.sun.com/xml/ns/persistence/
persistence_1_0.xsd"
version="1.0">

<!-- persistence.xml -->

<persistence-unit name="helloworld">

<!-- The provider only needs to be set if
you use several JPA providers -->
<provider>
org.hibernate.ejb.HibernatePersistence
</provider>
<properties>
<!-- Scan for annotated classes and
Hibernate mapping XML files -->
<property
name="hibernate.archive.autodetection"
value="class, hbm"/>
<!-- SQL stdout logging -->
<property name="hibernate.show_sql"
value="true"/>
<property name="hibernate.format_sql"
value="true"/>
<property name="use_sql_comments"
value="true"/>
<property name="hibernate.dialect"
value="org.hibernate.dialect.HSQLDialect"/>
<property
name="hibernate.connection.driver_class"
value="org.hsqldb.jdbcDriver"/>
<property
name="hibernate.connection.url"
value="jdbc:hsqldb:file:persistence-db/
test"/>
<property
name="hibernate.connection.username"
value="sa"/>
<property
name="hibernate.hbm2ddl.auto"
value="create"/>
<property name="hibernate.c3p0.min_size"
value="5"/>
<property name="hibernate.c3p0.max_size"
value="20"/>
<property name="hibernate.c3p0.timeout"
value="300"/>
<property
name="hibernate.c3p0.max_statements"
value="50"/>
<property
name="hibernate.c3p0.idle_test_period"
value="3000"/>
</properties>
</persistence-unit>
</persistence>
```

persistence.xml should be located on your CLASSPATH within META-INF directory. "hibernate.hbm2ddl.auto" property will take care of creating database table automatically.

4. Create a Maven2 file i.e. pom.xml file, responsible for downloading all dependent libraries, building correct CLASSPATH for the project and running the example. POM is the fundamental unit of work in Maven. It is an xml file that contains information about the

project and configuration details used by Maven to build the project. It contains default values for most projects. For example, the build directory, which is "target", the source directory, which is "src/main/java" the test source directory, which is "src/main/test" and so on. POM also contains the goals and plugins. So, while executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal. Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>persistence-deps</groupId>
<artifactId>persistence-deps</artifactId>
<version>1.0</version>
<dependencies>

<dependency>
<groupId>commons-logging</groupId>
<artifactId>commons-logging</artifactId>
<version>1.0.4</version>
</dependency>

<dependency>
<groupId>hsqldb</groupId>
<artifactId>hsqldb</artifactId>
<version>1.8.0.7</version>
</dependency>

<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate</artifactId>
<version>3.2.2.ga</version>
</dependency>

<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-annotations
</artifactId>
<version>3.2.1.ga</version>
</dependency>

<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-entitymanager</
artifactId>
<version>3.2.1.ga</version>
</dependency>

<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-tools</artifactId>
<version>3.2.0.beta9a</version>
</dependency>

<dependency>
<groupId>c3p0</groupId>
<artifactId>c3p0</artifactId>
<version>0.9.1</version>
</dependency>
</dependencies>

<build>
<defaultGoal>compile</defaultGoal>

<plugins>
<plugin>
<groupId>org.apache.maven.plugins</
groupId>
<artifactId>maven-compiler-plugin</
artifactId>
<configuration>
<source>1.5</source>
<target>1.5</target>
</configuration>
</plugin>

<plugin>
<groupId>org.codehaus.mojo</groupId>
<artifactId>exec-maven-plugin</
artifactId>
<executions>
<execution>
<goals>
<goal>java</goal>
</goals>
</execution>
```

```
</executions>
<configuration>
<mainClass>hello.HelloWorld</mainClass>
</configuration>
</plugin>
</plugins>
</build>
<!--
<repositories>
<repository>
<id>scriptlandia-repo</id>
<name>Scriptlandia Maven2 repository</
name>
<url>http://scriptlandia-
repository.googlecode.com/svn/trunk/
tools</url>
</repository>
</repositories>
-->
</project>
```

4. Open the command prompt. Go inside the project directory of your application (MyApplication in this case) and run the "mvn compile exec:java" command.

```
C:\MyApplication >mvn compile exec:java
```

After successfully running the command, your command prompt will display the output on the console shown below:

```
Hibernate:
insert
into
MESSAGES
(MESSAGE_ID, MESSAGE_TEXT,
NEXT_MESSAGE_ID)
values
(null, ?, ?)
Hibernate:
call identity()
Hibernate:
insert
into
MESSAGES
(MESSAGE_ID, MESSAGE_TEXT,
NEXT_MESSAGE_ID)
values
(null, ?, ?)
Hibernate:
call identity()
Hibernate:
insert
into
MESSAGES
(MESSAGE_ID, MESSAGE_TEXT,
NEXT_MESSAGE_ID)
values
(null, ?, ?)
Hibernate:
call identity()
Hibernate:
select
message0_.MESSAGE_ID as
MESSAGE1_0_,
message0_.MESSAGE_TEXT as
MESSAGE2_0_,
message0_.NEXT_MESSAGE_ID as
NEXT3_0_
from
MESSAGES message0_
order by
message0_.MESSAGE_TEXT asc
3 message(s) found:
Hello World with JPA
This is message 2
This is message 3
Aug 21, 2007 3:18:06 AM
org.hibernate.impl.SessionFactoryImpl
close
INFO: closing
[INFO] ------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------
[INFO] Total time: 34 seconds
[INFO] Finished at: Tue Aug 21 03:18:06
GMT+05:30 2007
[INFO] Final Memory: 6M/11M
[INFO] ------------------------------
```

On building successfully, our application displays the messages given below on the console:

Hello World with JPA
This is message 2
This is message 3

Database Testing with DbUnit

DbUnit :

DbUnit is an open source Framework created by Manuel Laflamme. This is a powerful tool for simplifying Unit Testing of the database operations.

It extends the popular JUnit test framework that puts the database into a known state while the test executes. This strategy helps to avoid the problem that can occur when one test corrupts the database and causes subsequent test to fail. DbUnit provides a very simple XML based mechanism for loading the test data, in the form of data set in XML file, before a test runs. Moreover the database can be placed back into its pre-test state at the completion of the test.

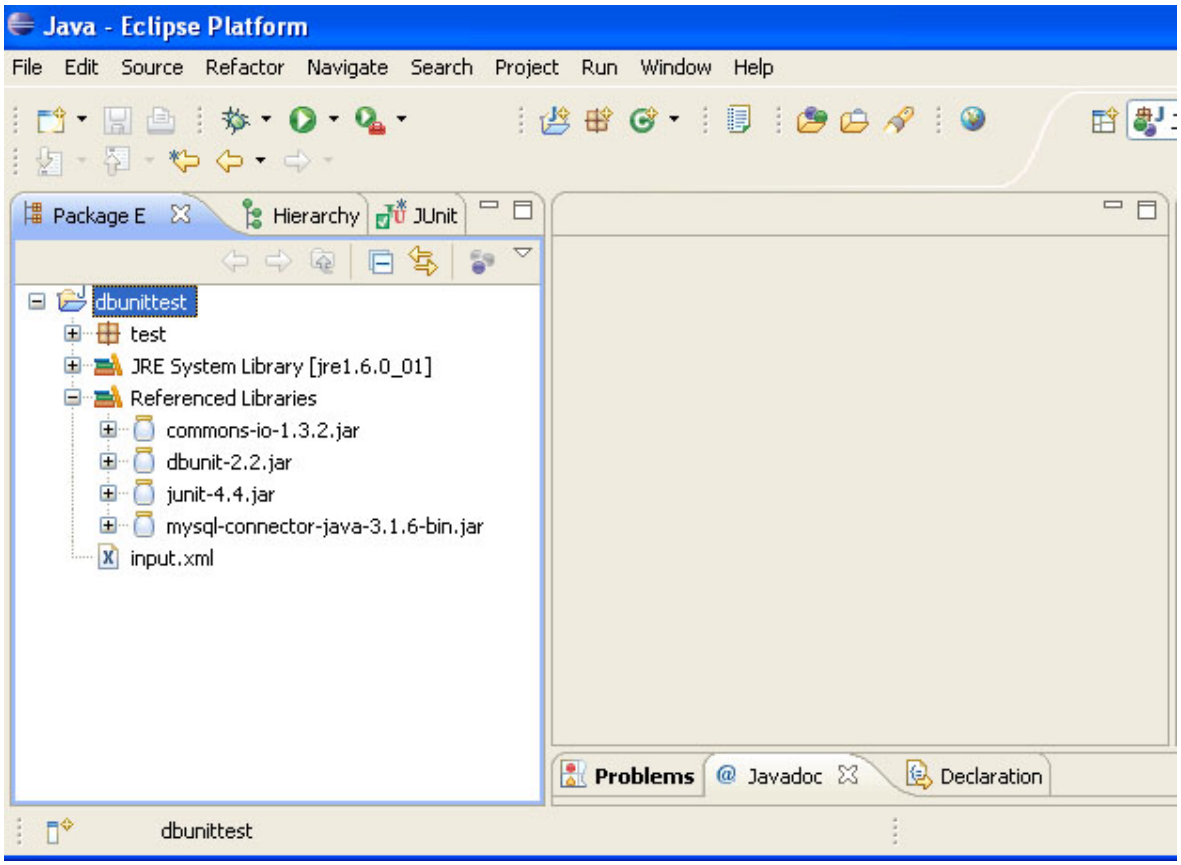
Advantages of DbUnit :

The reasons to use this testing tool can be summarized as follows :

1 A framework which simplifies operations for each stage in the life cycle of individual database tests.
2 It provides a very simple XML based mechanism for loading test data.
3 It provides equally a simple mechanism to export existing test data into the XML format for subsequent use.
4 It can work with very large datasets.
5 It can help verify your data matches an expected set of values.
6 It provides methods for comparing data between flat files, queries and database tables.

Setting up the environment :

To run the example of DbUnit in Eclipse, we need jar files of DbUnit, JUnit and Jakarta Commons IO. DbUnit is available from DbUnit web site, http://www.dbunit.org. JUnit is available from http://www.junit.org. commons-io-1.3.2.jar can be found from http://mirrors.kahuki.com/apache/commons/io/binaries/commons-io-1.3.2-bin.zip. Now these jar files are required to be added in the referenced library of your testing directory.



DbUnit test Life Cycle :

DbUnit framework follows some steps in its life cycle :

1 Removing the old data left in the database from previous tests.
2 Loading some data from XML file into the database.
3 Running the test.

DatabaseTestCase class provides two methods setUp() and TearDown() which internally call getSetUpOperation() and getTearDownOperation() methods respectively. setUp() method provides the setup operation DatabaseOperation.CLEAN_INSERT or DatabaseOperation.REFRESH. DatabaseOperation.CLEAN_INSERT operation is the combination of two operations DELETE_ALL and INSERT. So data defined in the XML file is loaded in the database by this operation. First two steps of the life cycle are executed when executing the setUp() method before running the test. These steps allow you not to waste time in writing code to restore state in the database.

DatabaseOperation.REFRESH updates the desired database with the data found in the XML file. The getTearDownOperation() performs a NONE operation which does nothing.

```
protected void setUp() throws Exception{
        super.setUp();
     executeOperation(getSetUpOperation());
        }
protected void tearDown() throws
Exception{
         super.tearDown();
     executeOperation(getTearDownOperation());
        }
protected DatabaseOperation
getSetUpOperation() throws Exception{
        return
DatabaseOperation.CLEAN_INSERT;
        }
protected DatabaseOperation
getTearDownOperation() throws Exception{
        return DatabaseOperation.NONE;
}
```

DbUnit can work with default behavior, however, you can override the methods according to the s requirement.

Getting Started : For the database purpose we have used MySQL

1 Create a table "login" in the database "hrapptest" in MySQL like below :

login Table in database:

Field	Type	Collation	Null	Key	Default
id	bigint(20)	(NULL)	NO	PRI	
empcode	varchar(15)	latin1_swedish_ci	YES		(NULL)
loginname	varchar(30)	latin1_swedish_ci	NO		
password	varchar(30)	latin1_swedish_ci	NO		
loginenabled	varchar(1)	latin1_swedish_ci	NO		

2 Create XML file (for example, "input.xml") representing the database tables and the data within it. Put a data set in this file like below. In this file "login" is the table name in the database and "id", "empcode" etc are the columns in the table. Put values for the fields in this file.

input.xml :

```
<?xml version='1.0' encoding='UTF-8'?>
<dataset>
<!--Login Table -->
<login id="1" empcode="E005"
loginname="chandan"
password="chandan" loginenabled="y"/>
<login id="2" empcode="E006"
loginname="deepak" password="deepak"
loginenabled="n"/>
</dataset>
```

3 DbUnit framework provides an abstract class named "DatabaseTestCase" which is a sub class of JUnit's "TestCase" class. So instead of creating a subclass of

TestCase class we need to extend DatabaseTestCase class. This class provides two abstract methods "getConnection()" and "getDataSet()".

```
IDatabaseConnection getConnection()
throws Exception
protected IDataSet getDataSet() throws
Exception.
```

Because of its being an abstract class we need to implement these two methods:

TestDbUnit.java :

```
.....
.....

 // Provide a connection to the database
 protected IDatabaseConnection

getConnection() throws Exception{
Class driverClass = Class.forName
("com.mysql.jdbc.Driver");

Connection jdbcConnection =
DriverManager.getConnection
("jdbc:mysql://localhost:3306/hrapptest",
"root", "root");

return new

DatabaseConnection(jdbcConnection);
    }
// Load the data which will be inserted for
the test

protected IDataSet getDataSet() throws
Exception{   loadedDataSet =
new FlatXmlDataSet(this.getClass()
.getClassLoader().getResourceAsStream
("input.xml"));
        return loadedDataSet;
    }
.....
.....
```

getConnection() method returns IDatabaseConnection object that represents database connection created using DriverManager class. In the above code, IDatabaseConnection represents MySQL database where hrapptest is the name of database where username and password both are "deepak".

getDataSet() method uses the FlatXmlDataSet class to load "input.xml" file and return this loaded data set as an object implementing IDataSet interface. IDataSet provides many useful methods to return data sets.

4. Writing Test :

Now, write test to check that the data has been loaded in TestDbUnit.java file:

```
.....
.....
public void testCheckLoginDataLoaded()
throws Exception{
assertNotNull(loadedDataSet);
int rowCount = loadedDataSet.getTable
(TABLE_LOGIN).getRowCount();
assertEquals(2, rowCount);
}
.....
.....
```

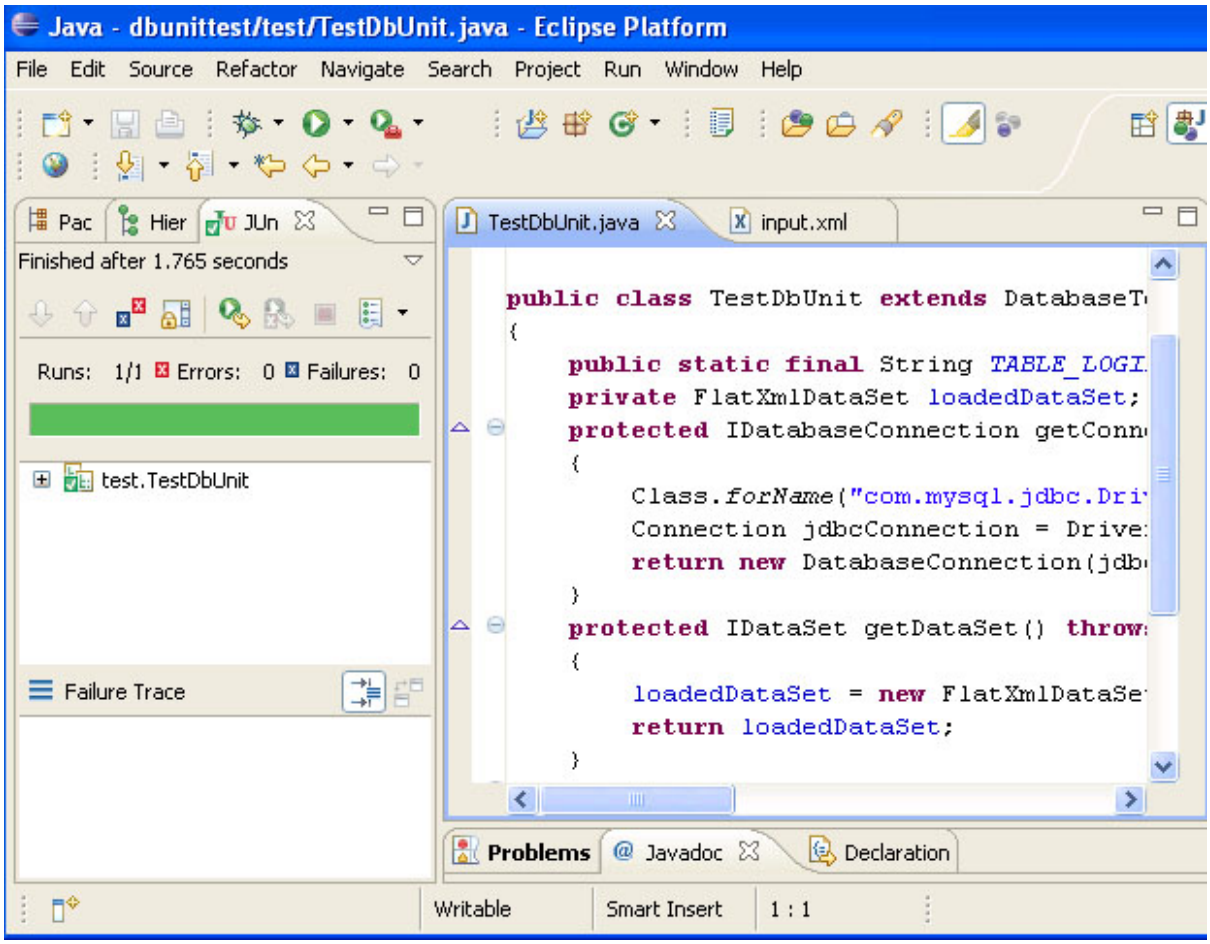
Combining all of the above functionalities into one TestDbUnit.java file, you will find it as follows :

TestDbUnit.java :package test;

```
import java.sql.Connection;
import java.sql.DriverManager;
import org.dbunit.DatabaseTestCase;
import
org.dbunit.database.DatabaseConnection;
import
org.dbunit.database.IDatabaseConnection;
import org.dbunit.dataset.IDataSet;
import
org.dbunit.dataset.xml.FlatXmlDataSet;
public class TestDbUnit extends
DatabaseTestCase{
public static final String TABLE_LOGIN =
```

```
"login";

private FlatXmlDataSet loadedDataSet;

 // Provide a connection to the database
protected IDatabaseConnection
getConnection() throws Exception{
Class.forName("com.mysql.jdbc.Driver");
Connection jdbcConnection =
DriverManager.getConnection("jdbc:mysql:/
/localhost:3306/hrapptest", "root", "root");
return new
DatabaseConnection(jdbcConnection);
}

 // Load the data which will be inserted for
the test
protected IDataSet getDataSet() throws
Exception{
loadedDataSet = new
FlatXmlDataSet(this.getClass().getClassLoader().
getResourceAsStream("input.xml"));
return loadedDataSet;
    }
// Check that the data has been loaded.
public void testCheckLoginDataLoaded()
throws Exception{
 assertNotNull(loadedDataSet);
 int rowCount =
loadedDataSet.getTable(TABLE_LOGIN).
getRowCount();
      assertEquals(2, rowCount);
    }
}
```

Running Test :
Now, in Eclipse, go to Run->Run As and click "JUnit Test" to run tests. If testing is successful then a green strip appears at the left of the eclipse window. If any of the test fails then it turns into a red strip indicating failure of any test.



Creational Design Patterns

Singleton Pattern:

The singleton design pattern deals with one and only one instance of an object that encapsulates the control of the object from a common place. There are various ways of achieving this pattern.

For example, create an exception that is thrown by a class if the class is instantiated more than once. Next declare a boolean variable and make it static so that it can be shared among all the instances of a class. Initialize this static variable inside the constructor of this class. The constructor would throw an exception if it is initialized second time. It is the best technique of using the singleton pattern. But take care to set the variable whenever destroying during the finalize method call.

Another approach of creating Singleton method includes a static method and private constructor in a class. Declaring the constructor as private ensures that the instance variable can be created only from inside the method of the class. The static method sets the variable as boolean that indicates the creation of the instance and returns an instance.

Benefits: Singleton pattern controls access to unique instances, reduce name space, permits a variable number of instances, allows refinement of operations and representations, and provides more flexibility than class operations.

Usage: These are used in those places where there is only one instance of a class. The Singleton pattern is mostly used in multi-threaded applications. Singleton patterns are often used as global variables because the global variables permit allocation and initialization whenever required. They don't permit to pollute the global namespace with unnecessary variables.

```
package singleton;
import org.apache.log4j.Priority;
import java.util.GregorianCalendar;
import java.text.SimpleDateFormat;
import java.io.FileOutputStream;
import java.util.Properties;
import java.io.PrintStream;
import java.io.InputStream;
import java.io.IOException;


public class Logger {
private String fileName;
private Properties properties;
private Priority priority;

private Logger() {
logger = this;
}

public int getRegisteredLevel() {
int i = 0;
try {
InputStream inputstream =
getClass().getResourceAsStream
("Logger.properties");
properties.load(inputstream);
inputstream.close();
i = Integer.parseInt(properties.getProperty
("logger.registeredlevel"));
if(i < 0 || i > 3)
i = 0;
}
catch(Exception exception) {
System.out.println("Logger: Failed in the
getRegisteredLevel method");
exception.printStackTrace();
}
return i;
}

private String
getFileName(GregorianCalendar gc) {
SimpleDateFormat dateFormat1 = new
SimpleDateFormat("dd-MMM-yyyy");
String dateString =
dateFormat1.format(gc.getTime());
String fileName =
"C:\\prashant\\patterns\\log\\PatternsExceptionLog-
" + dateString + ".txt";
```

```
return fileName;
}

public void logMsg(Priority p, String
message) {
try {
GregorianCalendar gc = new
GregorianCalendar();
String fileName = getFileName(gc);
FileOutputStream fos = new
FileOutputStream(fileName, true);
PrintStream ps = new PrintStream(fos);
SimpleDateFormat dateFormat2 = new
SimpleDateFormat("EEE, MMM d, yyyy 'at'
hh:mm:ss a");
ps.println("<"+dateFormat2.format
(gc.getTime())+">["+message+"]");
ps.close();
}
catch (IOException ie) {
ie.printStackTrace();
}
}

public static void initialize() {
logger = new Logger();
}
// singleton - pattern
private static Logger logger;
public static Logger getLogger() {
return logger;
}
}
```

Difference between static class and static method approaches: One question that comes to most of the people's mind is that " What's the difference between a static class and a singleton class?" The answer is static class is one of the approaches that makes a class "Singleton".

We can create and declare a class as "final" simply by declaring all its methods as "static". In this case, you can call the static methods directly but can't create any instance of class.

Example:

```
final class Logger {
//implementation of a static class of a
Singleton pattern
static public void logMessage(String s) {
System.out.println(s);
}
}

public class StaticLogger {
public static void main(String args[]) {
Logger.logMessage("This is SINGLETON");
}
}
```

The advantage of this static approach is that it's easier to use. The disadvantage of course is that if in future you do not want the class to be static anymore, you will have to do a lot of recoding.

Builder Pattern:

This design pattern allows the client to construct a complex object based on its type and content. One can achieve the way of constructing the objects by using the factory pattern. This is similar to the abstract factory pattern as both returns a group of related objects. The only difference between these two patterns is that Builder pattern makes a complex object step by step on the basis of data passed to it.

Benefits: These types of patterns provide greater control over construction process, separation between the construction and representation, and support to change the internal representation of objects.

Usage: Builder is useful in those conditions where you need to assemble several different kinds of complex objects at run-time. Once it is created it isolates the building process of object, from the object itself. Different objects may be constructed by using similar methods but once the

object is constructed it may exhibit different behavior.

First lets create an interface named Item which contains the two public methods one for packaging the item and other for defining the price of each item.

We are putting all the classes in the package builder.

```
package builder;

public interface Item {

public Packing pack();

public int price();

}
```

Now, we define the classes for each of the item, as pizza, ice cream and cold drink. All these classes will implement the Item interface.

Lets start with Pizza. Here we are making the pizza.java class as abstract because we will implement price method according the type of the pizza. A pizza is wrapped in the paper and is served. The class Wrapper is sub-class of Packing interface.

Lets consider this class as Pizza.java

```
package builder;

public abstract class Pizza implements
Item {

public Packing pack() {
return new Wrapper();
}
public abstract int price();
}
```

Now the class pizza further extended to Italianpizza, Cheesepizza etc. All these classes will implement the price() method and return a price for each type of pizza. In this example we are implementing the Italianpizza class of the Pizza class.

Italianpizza.java

```
package builder;

public class Italianpizza extends Pizza {

public int price() {
return 200;
}
}
```

Now lets consider the item Ice Cream

```
IceCream.java
package builder;

public class IceCream implements Item {

public Packing pack() {
return new Envelop();
}

public int price() {
return 20;
}
}
```

Now, let's see the Builder class, MealBuilder. This is the class which serves the meal.

```
MealBuilder.java
package builder;

public class MealBuilder {
public Packing additems() {
Item[] items = {new Italianpizza(), new
IceCream(), new Coke()}

return new MealBox().addItems(items);
}
```

```
public int CalculatePrice() {
int totalPrice = new Italianpizza().price() +
new Coke().price() + new
IceCream().price();

return totalPrice;
}
}
```

This class calculates the total meal and its total price. Here, we have extracted the price calculation and meal package building activity, that is a meal box. The Builder pattern hides the internal details of how the product is built. Since each builder is independent of others therefore it improves modularity and makes the building of other builders easy. Because, each builder builds the final product step by step, we have more control on the final product.

The conclusion is that in Builder Pattern, the client instructs the builder class what it needed and asks for the result, the client is not interested how the builder class will create the objects.

The Prototype Pattern:

This pattern enables you to copy or clone of an existing object instead of creating the new one and may also customized as per the requirement. It copies the existing object to avoid so much overhead. We can also use the clone method by implementing the Clonable interface to create the copy of the existing object.

Benefits: It supports for specifying the new objects with varying structure and varying values, adding and removing products at runtime, dynamically configuring the classes for an application and reducing sub-classing.

Usage: These are used when you are not interested in constructing a class hierarchy of factories which is parallel to the class hierarchy of products. Instances of a class can have only one combination of state, the classes are instantiated at run-time.

Example: Let's create an interface and implement it in the various classes.

Shape.java

```
interface Shape {
public void draw();
}

Square.java
class Square implements Shape {
public void draw() {
System.out.println("square");
}
}

Circle.java
class Circle implements Shape {
public void draw() {
System.out.println("circle");
}
}
Painting.java
class Painting {
public static void main(String[] args) {

Shape s = new Square();
Shape c = new Circle();

paint(s);
paint(c);
}
static void paint(Shape s1) {
s1.draw();
}
}
```

At the runtime, the paint method takes a variable of Shape type and the draw method is called accordingly.

Overloading method is a kind of prototype pattern too.

Painting.java

```
class Painting {
public void draw(Point p, int x, int y) {
}
public void draw(Point p, int x) {
}
}
```

To draw the related shape the draw method is called on the bases of the parameters passed to it.

Web services with Lomboz

Lomboz plug-in

In this article, we will discuss the Lomboz plug-in and its usability in developing web services.

Lomboz is an open source and free JEE development environment used for businesses and individual purposes.

With Lomboz , one can develop, test, profile and deploy Web Services, Java, JEE and EJB applications. It provides comprehensive support for most of the JEE standard application server runtimes, and supports other vendor runtime environments too.

Lomboz is built on the Eclipse open source platform and the Web Tools Platform (WTP) projects. It is a free Eclipse plug-in for JEE developers. Lomboz is integrated with many popular open source JEE tools such as: Jasper, XDoclet, Axis and Ant and even more naturally with Eclipse and the Eclipse Java Development Toolkit (JDT).

Lomboz implements an end-to-end JEE application development by providing support for the complete development cycle: Code, deploy, test and debug.

Salient features of Lomboz :

Wizards for rapid coding: Containers, Servlets, JSPs, and EJBs

- Flexibility to manipulate the launching and deployment behaviors by allowing developers to modify corresponding XML scripts
- JEE Project Outlining View: Allowing developers to view and manipulate containers in projects
- EJB code generators
- Improved EJB and XDoclet wizards

A whole new JSP editor with

- Java™ Servlet previews
- JSP and HTML code assist with syntax checking and highlighting
- Tracking of JSP errors with standard Eclipse problem markers
- The ability to edit a JSP file regardless of how it is packaged in web modules.
- Code assists for TLDs and taglibs.

Application Server support

- Application server launchers and stoppers for and almost any J2EE compliant server including: JBoss, Resin, Tomcat, JRun, BEA WebLogic Server, new: IBM WebSphere, Oracle 9iAS, Orion, and -new- JOnAS, the open source application server of ObjectWeb
- Manage and launch multiple application-server configurations inside Eclipse in Run or Debug modes
- JSP debugging for Tomcat and BEA WebLogic Server
- Manage multiple Web, EJB and Application (EAR) modules inside the same project.
- Ability to target multiple application servers from the same project

Web Services Support

- Web Services wizards
- Web Services code generators

EAR support

Create application archives quickly and deploy them on application servers using Lomboz actions. A graphical application.xml editor allows you to edit the contents of an application rapidly.

Developing web services with Lomboz plug-in

Let's develop web services with Lomboz

Now let's create a simple web service and a client web application using **eclipse IDE** along with **Lomboz plug in.** We will deploy and test the web service on Tomcat 5.5.4 web application server.

I. Environment

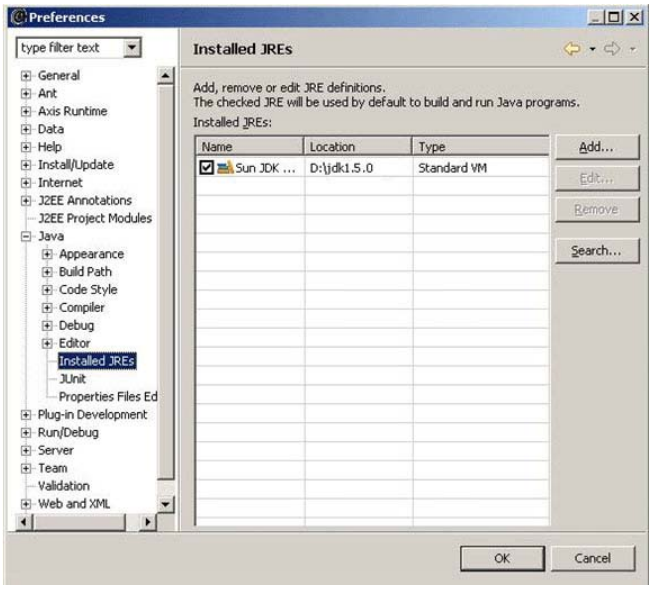
Jdk1.5.0 gets it at: http://java.sun.com/
Eclipse 3.1 gets it at: http://www.eclipse.org/
Tomcat 5.5.4 gets it at: http://tomcat.apache.org/
Lomboz 3.1RC2 gets it at: http://lomboz.objectweb.org/

II. Installation

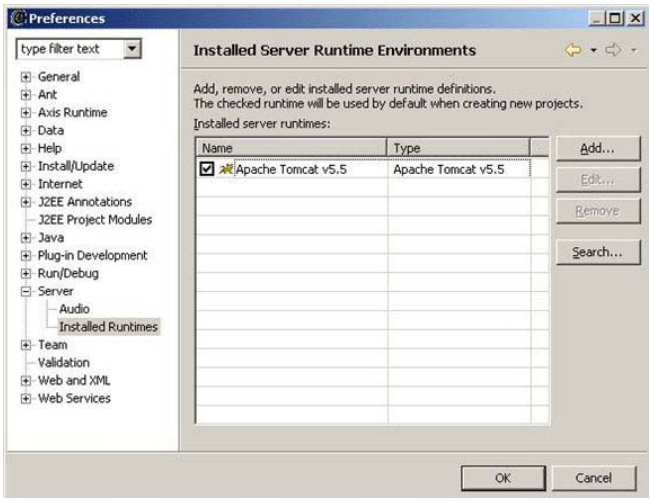
Install JDK (in D:\ jdk1.5.0)
Install Tomcat (in E:\Tomcat5.5)
Install Eclipse (in E:\Eclipse3.1)
Install Lomboz (in E:\Eclipse3.1)

III. Set Ups

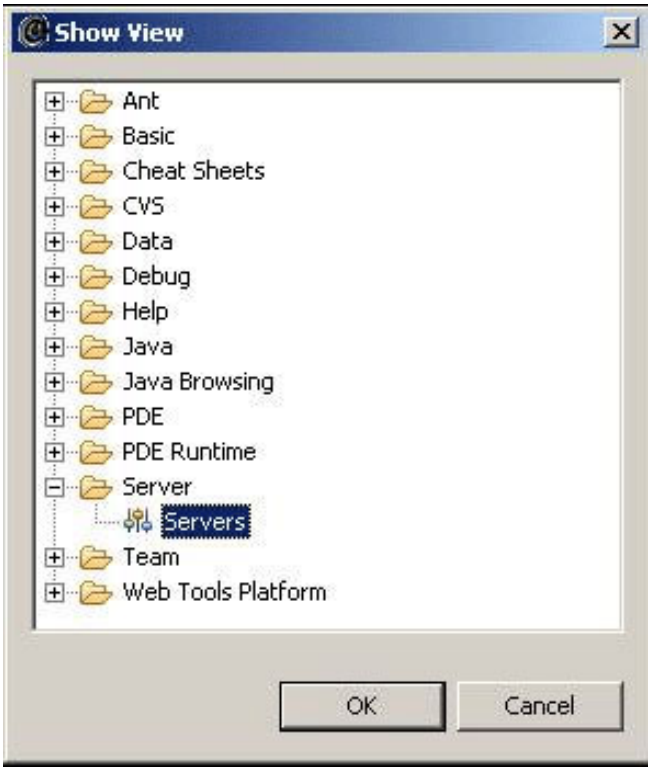
1. Set up the installed JRE in eclipse (Windows -> Preferences -> Java -> Installed JREs)



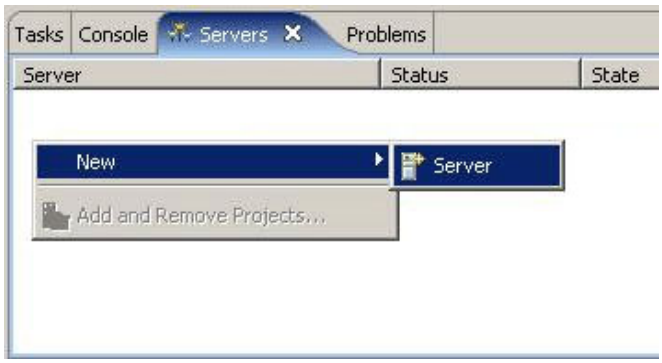
2. Set up the installed runtime for server in eclipse (Windows -> Preferences -> Server -> Installed Runtimes)

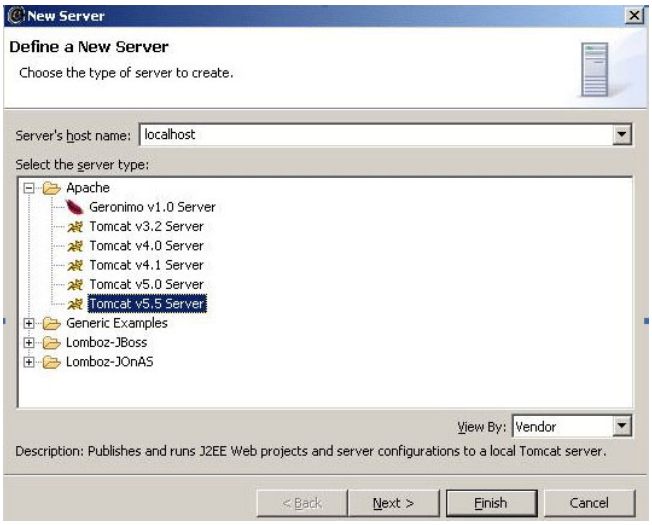


3. Set up the Server view in eclipse (Windows -> Show View -> Other)



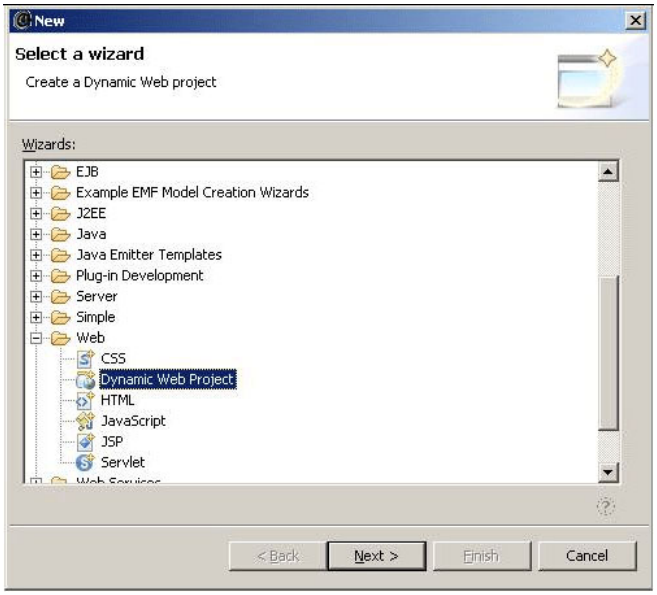
4. Set up the Tomcat Server by right clicking and selecting New -> Server option from the Server view in eclipse



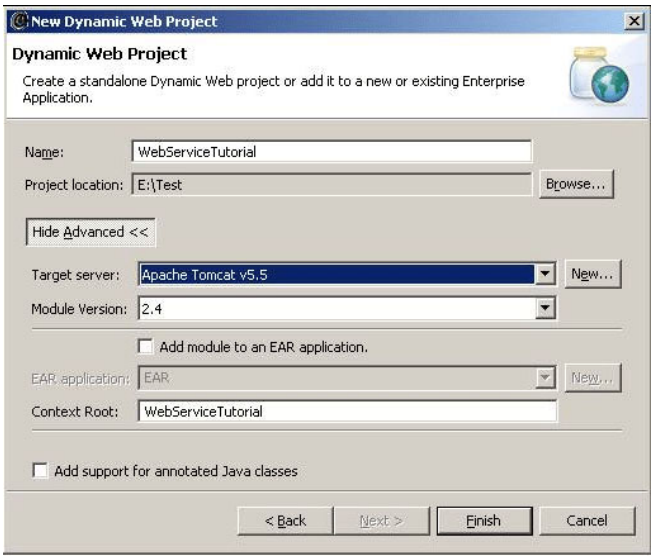


VI. Creating a Web service

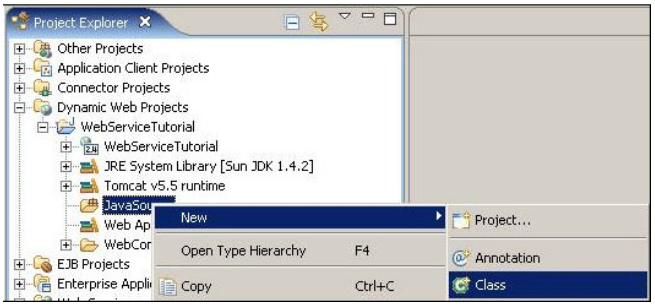
1. Create a new Dynamic Web Project in eclipse (File -> New -> Other)



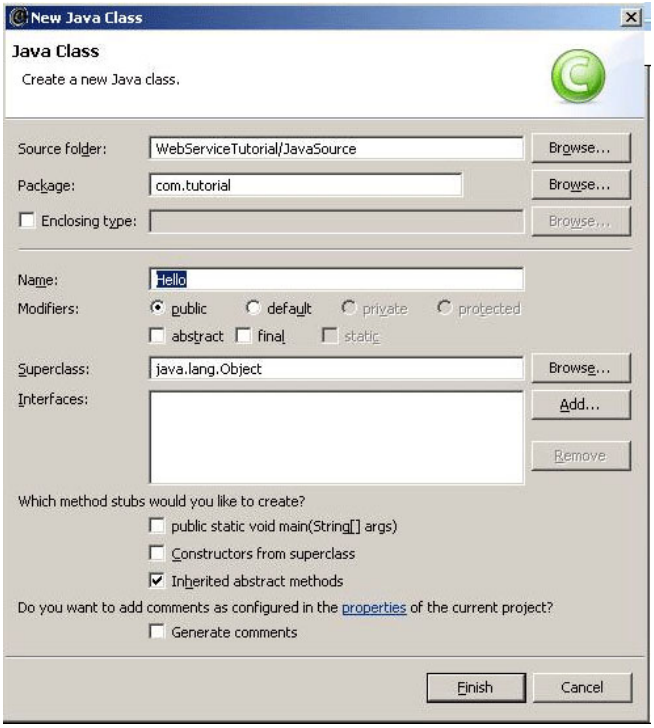
2. Enter name as "WebServiceTutorial", select project location as "E:\Test" and select Apache Tomcat v5.5 as the Target server.



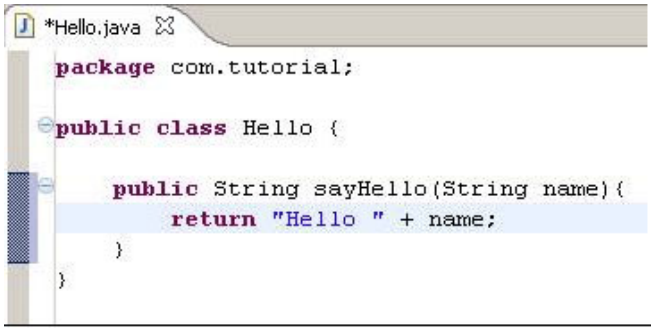
3. Now create a new Java class from the Project Explorer (Dynamic Web Projects -> Java Source -> New -> Class



4. Enter name as "Hello" and package as "com.tutorial".

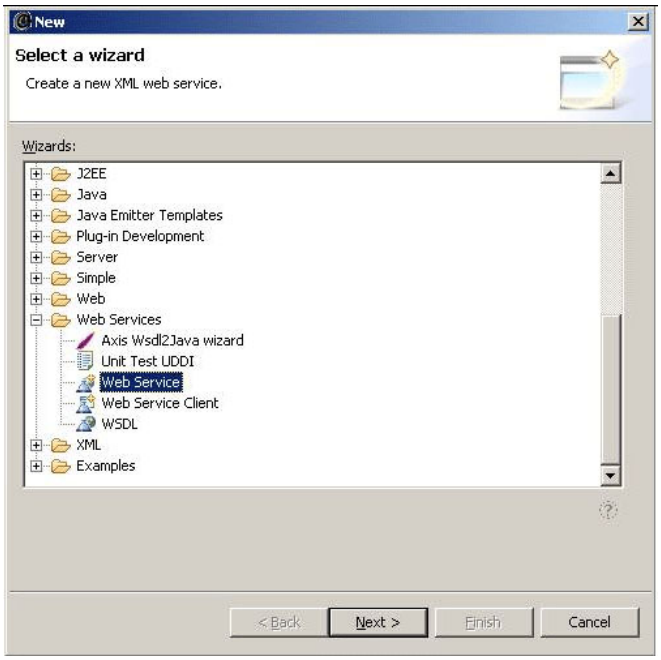


5. Add a simple method in the "Hello" class as shown below.



6. Save and build the project.

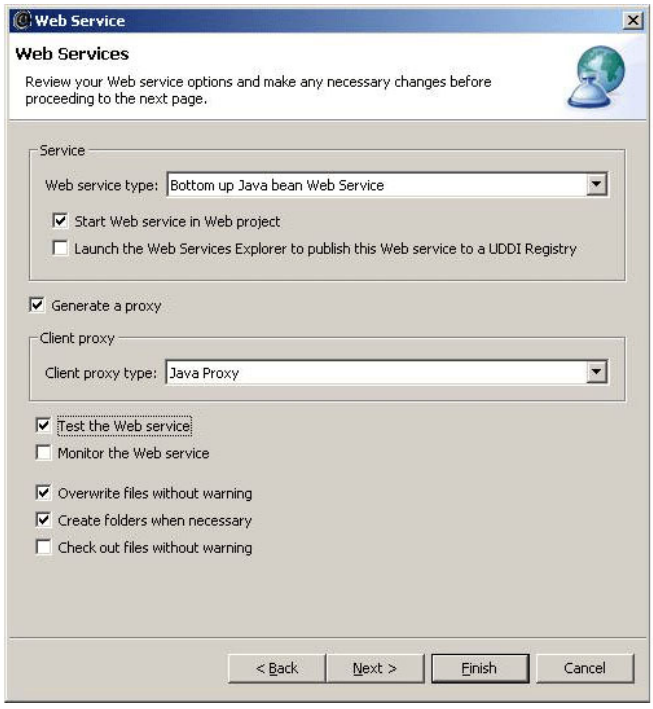
7. Create a new Web service in eclipse (File -> New -> Other)



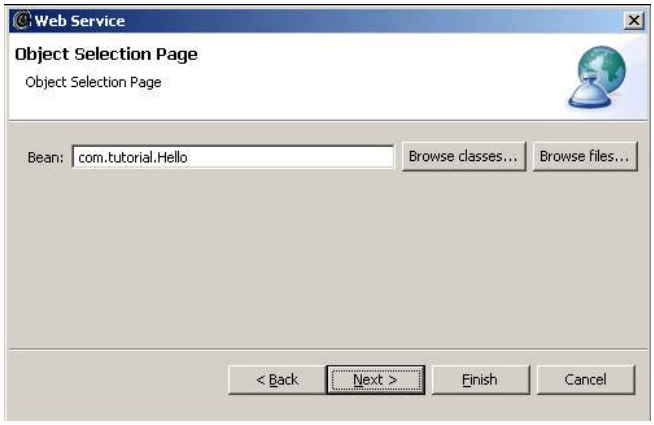
8. Select Generate a proxy.

9. Select Test the Web service.

10. Select Overwrite files without warning.



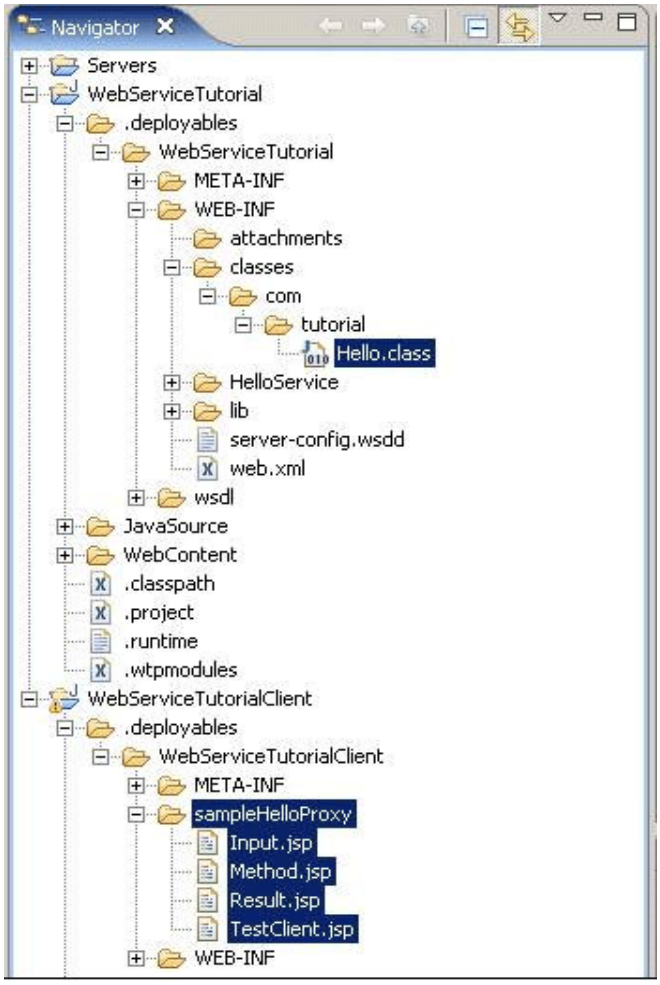
11. Select or enter the Bean name as "com.tutorial.Hello". This is the java class that we just now created.



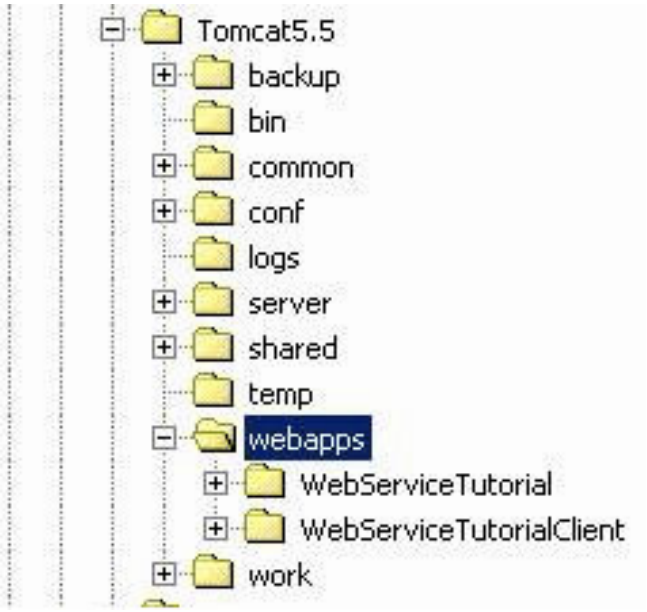
12. Continue the wizard by clicking next and finish.

13. On Finish, the Tomcat server starts up and launches the Test client.

14. Verify the generated contents. Look for Hello.class and the generated JSPs as below.

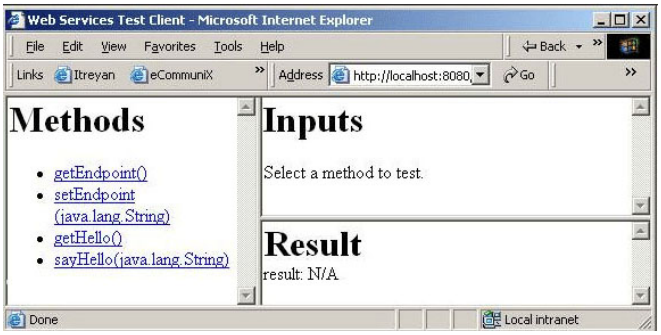


15. Verify the Tomcat folder and ensure the newly created web applications – WebServiceTutorial, WebServiceTutorialClient.

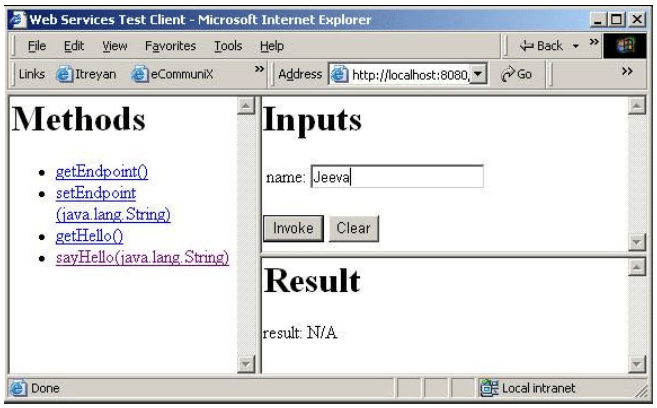


16. We can also run the following url from the browser to access/test the Web service.
http://localhost:8080/WebServiceTutorialClient/sampleHelloProxy/TestClient.jsp
If servlet error "org.eclipse.jst.ws.util.JspUtils cannot be resolved or is not a type" is thrown on the browser, then copy the webserviceutils.jar file from the E:\Eclipse3.1\eclipse\plugins\org.eclipse.jst.ws.consumption_0.7.0 into the WEB-INF\lib folder of the WebServiceTutorialClient application and restart the Tomcat server.

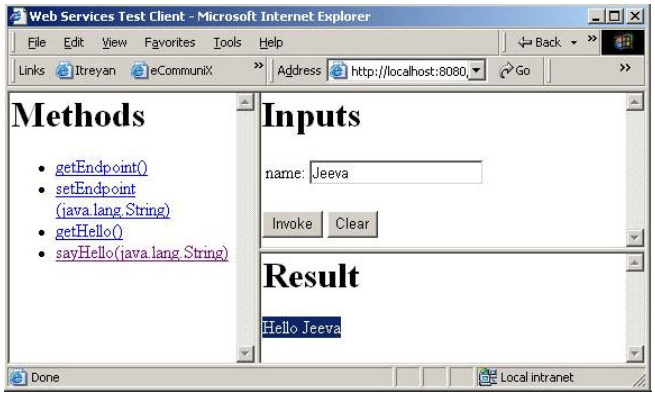
17. The browser displays the methods available in the web service.



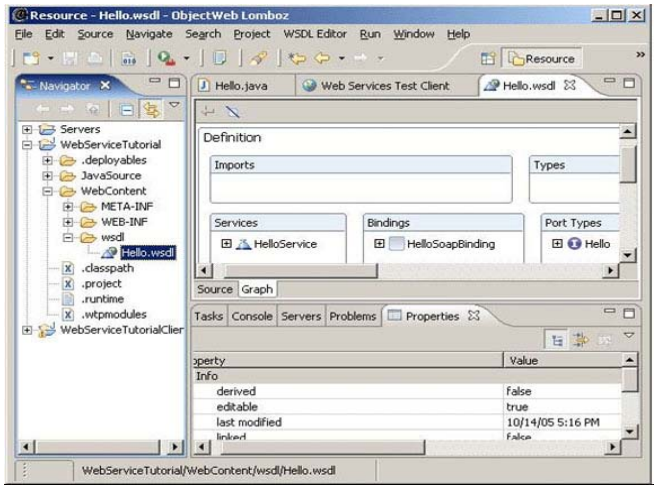
18. Click on the **sayHello(..)** method, enter your name (for e.g. "Jeeva") in the inputs section and click "Invoke".



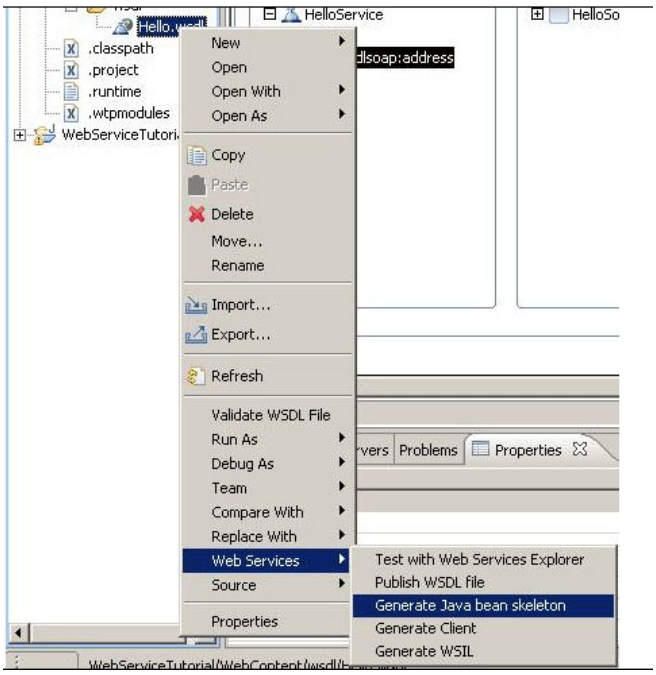
19. The browser greets using the web service.



20. The WSDL for the Hello Web service can be found in E:\Test\WebServiceTutorial\ WebContent\wsdl\Hello.wsdl. On double-click, the WSDL opens in a graphical editor.



21. Right-click on the WSDL file and explore the options to test the web service / publish the WSDL file / generate client / etc.



Conclusion

This article provides a good introduction to Web service development using one of the Web development tools available.

We hope this article would help you create a simple web service and a client web application using **eclipse IDE** along with **Lomboz plug in,** very convieniently**.** We have deployed and tested the web service on Tomcat 5.5.4 web application server.

Spring Framework

Spring Core with Data Access Framework

In this article, we will move on to the main process of any enterprise application: Data Persistence. For this we have to initialize our data access framework, manage resources, handle various exceptions and if anything goes wrong, we must roll-back so as to save the existing data.

Spring comes with a family of data access frameworks that integrates well will variety of data access technologies like JDBC, Java Data Objects and Object Relational Mapping (ORM) tools like Hibernate, OJB, iBatis etc.,

Many JEE application servers and even web servers provide a 'dataSource' via JNDI name. To configure the spring bean with the JNDI name of our 'dataSource' and use its connection pooling facility, **'JndiObjectFactoryBean'** is used. When a DataSource is not present, we need a connection pooling bean that implements 'dataSource'. For this purpose we use **'dbcp.BasicDataSource'**. By using this we can have a 'dataSource' with connection pooling independent of application server.

To perform unit-tests in our data access code, spring comes with a very lightweight 'dataSource' implementation class: '**DriverManagerDataSource**'. This class can be easily configured for unit tests as shown:

```
…
   …
DriverManagerDataSource   dataSource =
new   DriverManagerDataSource();
dataSource.setDriverClassName(driver);
dataSource.setUrl(url);
dataSource.setUsername(username);
dataSource.setPassword(password);
…
```

These properties can be configured in the spring configuration file also.

Spring's own Data Access Framework

Spring comes with its own data access framework. Spring separates the fixed and variant parts of data access process into two distinct classes: **template and callback.** Template manages the fixed part of our framework like data connection, managing resources, controlling transaction etc., while the Callback defines the things that are specific to our application like creating statements, binding parameters etc.,

The template class of Spring is **'JdbcTemplate'.** A **'dataSource'** is provided inside JdbcTemplate.

1 An example of database connection using 'JdbcTemplate' is shown below. Here we are using **'MySql'** database. The MySql database can be downloaded from **http://www.mysql.com.** Download latest MySql version and mysql-connector-java-3.1.6-bin.jar. Install them in the hard disk.

Installing MySql:

1. For Mysql give a username('root') and a password ('root').

2 Then start the 'My Sql Console Line Client' from programs and type the password. The prompt will be changed to mysql,mysql> show databases; Few databases will be present by default like mysql and test. Let's use 'test' for our purpose.mysql> use test;

3 We will get a message as **'Database changed'**.

4 Next create a table in 'test' database as follows
mysql> create table table1(name text, place text);

You will get the message '**Query OK, 0 rows affected'**.

Now we have created a table in mysql database with two fields: Name and City.

Setting up the environment variables: As the entire Spring Framework is included in **spring.jar**. We use it to run our examples.

1 Copy **spring.jar** from **spring1.2.9\dist** folder to the working folder (**springdemo**) , **mysql-connector-java-3.1.6-bin.jar,** also copy **commons-logging.jar** from **apache-tomcat-6.0.10** to the working directory.

2 **Set path for jdk1.4.2 and above versions only**

3. Now set the classpath as shown:

```
D:\>springdemo >set
classpath=D:\springdemo;
D:\springdemo\spring.jar;
D:\springdemo\commons-logging.jar;
D:\springdemo\ mysql-connector-java-
3.1.6-bin.jar;
```

4. For a typical Spring Application we need the following files:

i. An **interface** that defines the functions.
ii. An **Implementation** that contains properties, its setter and getter methods, functions etc.,
iii. A XML file called **Spring configuration file.**
iv. Client program that uses the function.

Create the following files

1. datacon.java
2. dataconimpl.java
3. datacon.xml
4. helloclient.java

1. D:\springdemo\ datacon.java

```
import javax.sql.*;
public interface datacon{
public DataSource dbcon();
}
```

2. D:\springdemo\ dataconimpl.java import

```
org.springframework.jdbc.core.*;
import
org.springframework.jdbc.datasource.*;
import org.springframework.jdbc.object.*;
import
org.springframework.jdbc.support.*;
import javax.sql.*;
public class dataconimpl implements
datacon{
private DataSource dataSource;
public void setDataSource(DataSource ds)
{
 dataSource = ds;
}
public DataSource dbcon()  {
return dataSource;
}
}
```

3. D:\springdemo\datacon.xml

```
<? xml version="1.0" encoding="UTF-
8"?><!DOCTYPE beans PUBLIC "-//
SPRING//DTD BEAN//EN""http://
www.springframework.org/dtd/spring-
beans.dtd"><beans><bean
id="dataSource"
class="org.springframework.jdbc.datasource.
DriverManagerDataSource">
<property name="driverClassName">
<value>com.mysql.jdbc.Driver</value>
</property>
<property name="url">
<value>
jdbc:mysql://localhost:3333/test
</value>
</property>
<property name="username">
<value>root</value>
</property>
```

```
<property name="password">
<value>root</value>
</property></bean>
<bean id="datacon"
class="dataconimpl">
<property name="dataSource">
<ref local="dataSource"/>
</property>
</bean>
</beans>
```

4. D:\springdemo\helloclient.java

```
import java.io.*;
import javax.sql.*;
import java.sql.*;
import java.util.*;
import
org.springframework.beans.factory.*;
import
org.springframework.beans.factory.xml.*;
import org.springframework.core.io.*;
import
org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.core.*;
import
org.springframework.jdbc.datasource.*;
import org.springframework.jdbc.object.*;
import
org.springframework.jdbc.support.*;
public class helloclient {
public static void main(String args[])
throws Exception{
try {
String a="Amit";
String b="Dehradun";
System.out.println("Wait...");
Resource res = new
ClassPathResource("datacon.xml");
System.out.println("Please Wait...");
BeanFactory factory = new
XmlBeanFactory(res);
datacon bean1 =
(datacon)factory.getBean("datacon");
DataSource ds=bean1.dbcon();
JdbcTemplate jt = new
JdbcTemplate(ds);
jt.execute("insert into table1 values('"+a
+"','"+ b+"') ");
System.out.println("Record Added");
}
catch(Exception e1) {
System.out.println(""+e1);
}
}
}
```

Now, execute the programs:

```
D:\springdemo>javac datacon.java
D:\springdemo>javac dataconimpl.java
D:\springdemo>javac helloclient.java
D:\springdemo>java helloclient
```

We will get the output as follows:

```
Wait... Please Wait...Aug 27, 2007 4:35:00
PM
org.springframework.core.CollectionFactory
<clinit>INFO: JDK 1.4+ collections
availableAug 27, 2007 4:35:01 PM
org.springframework.beans.factory.xml.
XmlBeanDefinitionReader
loadBeanDefinitionINFO: Loading XML bean
definitions from class path resource
[datacon.xml]Aug 27, 2007 4:35:01 PM
org.springframework.jdbc.datasource.Driver
ManagerDataSource
setDriverClassNameINFO: Loaded JDBC
driver: com.mysql.jdbc.DriverRecord
Added{name=Amit, place=Dehradun}
```

Spring also provides integration for many of the ORM frameworks like **Hibernate, JDO, Apache OJB and iBATIS SQL Maps.**

For mapping the hibernate resources, an instance of '**SessionFactory**' is needed, **'LocalSessionFactoryBean'** is used for this purpose and its properties **'hibernateProperties', 'mappingResources'** and **'mappingDirectoryLocation'** are set. Like spring's DAO framework, here we have **'HibernateTemplate'** to create an object of **'SessionFactory'.** To access the data with 'HibernateTemplate' **'execute(HibernateCallback)'** method is used.

Similar to the '**SessionFactory'** of hibernate, JDO has **'PersistenceManager Factory'.** It can be configured by using **'LocalPersistenceManagerFactoryBean'.** Also **'JDOTemplate'** to create an object of **'PersistenceManagerFactory'.** To access the data with 'JDOTemplate' **'execute(JDOCallback)'** method is used.
For iBATIS, we have to configure a **'SQLMapClient'** by using **'SQLMapClientFactoryBean'** and its properties 'configLocation' and 'dataSource' are set. Here also we have **'SQLMapClientTemplate'.**
To access the data **'execute(SQLMapClientCallback)'** method is used.
The only property that we need to change to integrate Spring with OJB is **'ConnectionFactoryClass'** and it is done by using **'LocalDataSourceConnectionFactory**'.

Web 3.0

Web 3.0 is a term, which definition is not confirmed or defined so far as several experts have given several meaning, which do not match to each other, but sometimes it is referred to as a Semantic Web. In the context of Semantic Web, ***Web 3.0 is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, bu t also in a form that can be understood, interpreted and used by software agents, thus permitting them to find, share and integrate information more easily.***

Tim Berners-Lee, the inventor of first World Wide Web has coined the term Semantic Web. But the concept of Web 3.0, first entered among the public in 2001, when a story appeared in scientific article written by American Coauthored Berners-Lee that described this term as a place where machines can read Web pages as much as humans read them e.g. web connected bathroom mirrors, which can read the news coming through on the web.

Definitions and Roadmap

There are several definitions of the web, but usually Web 3.0 is defined as *a term, which has been coined with different meanings to describe the evolution of web usage and interaction among the several separate paths.* These include transforming the Web into a database, a move towards making content accessible by multiple non-browser applications, the leveraging of artificial intelligence technologies, the Semantic web, or the Geospatial Web.

According to Wikipedia, an online encyclopedia, "Web 3.0 is a third generation of Internet based Web services, which emphasize machine-facilitated understanding of information in order to provide a more productive and intuitive user experience.". The third generation of Internet services is collectively consists of semantic web, microformats, natural language search, data-mining, machine learning, recommendation agents that is known as ***Artificial Intelligence technologies*** or ***Intelligent Web.***

According to some experts, "Web 3.0 is characterized and fueled by the successful marriage of artificial intelligence and the web". While some experts have summarized the definition defining as "Web 3.0 is the next step in the progression of the tubes that are the Internets".

According to Nova Spivack, the CEO of Radar Networks, one of the leading voices of this new-age Internet, "Web 3.0 is a set of standards that turns the Web into one big database."

Steve, a famous Blog author has defined the term Web 3.0 as, " Web 3.0 is highly specialized information structures, moderated by a group of personality, validated by the community, and put into context with the inclusion of meta-data through widgets".

While Leiki, the Finland based pioneer company of Semantic Web describes: "Web 3.0 makes the discovery of content streams effortless. It introduces automatic discovery of like-minded users and automatic tagging."

History

The term 'Web 3.0' was first coined by John Markoff of the New York Times in 2006, while it first appeared prominently in early 2006 in a Blog article written by Jeffrey Zeldman in the "Critical of Web 2.0 and associated technologies such as Ajax".

The debate originates in summit named Technet Summit in November 2006, in which various software tycoons expressed their views. e.g.

Jerry Yang, founder and Chief of Yahoo, stated:

"Web 2.0 is well documented and talked about. The power of the Net reached a critical mass, with capabilities that can be done on a network

level. We are also seeing richer devices over last four years and richer ways of interacting with the network, not only in hardware like game consoles and mobile devices, but also in the software layer. You don't have to be a computer scientist to create a program. We are seeing that manifest in Web 2.0 and 3.0 will be a great extension of that, a true communal medium…the distinction between professional, semi-professional and consumers will get blurred, creating a network effect of business and applications. "

-- Jerry Yang

While Reed Hastings, the founder and CEO of Netflix, stated a simpler formula for defining the phases of the Web in the same Technet Summit:

" Web 1.0 was dial-up, 50K average bandwidth, Web 2.0 is an average 1 megabit of bandwidth and Web 3.0 will be 10 megabits of bandwidth all the time, which will be the full video Web, and that will feel like Web 3.0."

--Reed Hastings

Before this people were very curious about 'Web 3.0' as they asked to Tim Berener about the full-fledged information of Web 3.0 as Tim Berners-Lee stated in May 2006:

"People keep asking what Web 3.0 is. I think maybe when you've got an overlay of scalable vector graphics - everything rippling and folding and looking misty - on Web 2.0 and access to a semantic Web integrated across a huge space of data, you'll have access to an unbelievable data resource."

--Tim Berners-Lee,
A 'more revolutionary' Web

The term Web 3.0 has became a subject of interest and debate since late 2006 to till date. But no exact definition has been created that everyone accepts it.

Web 3.0 Debates over Definition

Since the origins of the concept of Web 3.0, the debate continues goes on about exactly what the term Web 3.0 means, and what a suitable definition might be. As emerging the new technology, a new definition emerged:

Transforming the Web into a database

Transforming the Web into database is the beginning step towards transforming definition of Web 3.0 when the technology of 'Data Web' emerged as structured data records that can be published to the Web in reusable and remotely queryable formats, such as XML, RDF and microformats. The Data Web is the initial step in the way of full Semantic web that enables a new level of data integration and application interoperability, which makes the data openly accessible and linkable as Web pages. To make available structured data using RDF is primarily focused in Data Web phase. The full Semantic Web stage will so expand the scope that both structured and semi-structured or unstructured content will be widely available in RDF and OWL semantic formats.

An evolutionary path to artificial intelligence

Web 3.0 has also been used to describe the trend of artificial intelligence, which is being popular in the web like a quasi-human fashion. Some cynic believes that it is an unobtainable vision. However, this is being used new technologies on mass level that yields amazing information like making predictions of hit songs from mining information on college music Web sites. There is also debate on the driving force behind Web 3.0. Will it be the intelligent systems, or whether intelligence will emerge in a more organic fashion and how people interact with it?

The realization of the Semantic Web and Service Oriented Architecture

Another debate originates over the artificial intelligence direction in which Web 3.0 can be extent to Semantic web concept. Academic

research is going on to develop such reasoning software that must be based on description logic and intelligent agents. These sorts of applications can perform logical reasoning operations through using sets of rules expressing logical relationships between concepts and data on the Web.

But some critics are disagree on the viewpoint, which describes that Semantic Web would be the core of the 3rd generation of the Internet and suggests a formula to summarize Web 3.0.

Web 3.0 has also been associated to a possible hub of SOA (Service Oriented Architecture) and Semantic web.

Evolution towards 3D

The evolution of 3D technology is also being connected to Web 3.0 as Web 3.0 may be used on massive scale due to its characteristics. In this process Web 3.0 would transform into a series of 3D spaces, taking the concept realized by Second Life expansion. This could open up new ways to connect and collaborate using 3D shared spaces.

Proposed Expanded Definitions of Web 3.0

Nova Spivack has proposed the expanded definition of Web 3.0 that indulge in itself the collection of various foremost harmonizing technology developments that are growing to a new level of maturity simultaneously includes:

- **Ubiquitous Connectivity**, broadband adoption, mobile Internet access and mobile devices
- **Network computing**, software-as-a-service business models, Web services interoperability, distributed computing, grid computing and cloud computing
- **Open technologies,** Open APIs and protocols, open data formats, open-source software platforms and open data (e.g. Creative Commons, Open Data License)
- **Open identity,** OpenID, open reputation, roaming portable identity and personal data
- **The intelligent web,** Semantic web technologies such as RDF, OWL, SWRL, SPARQL, Semantic application platforms, and statement-based datastores
- **Distributed databases,** the "World Wide Database" (enabled by Semantic Web technologies)
- **Intelligent applications,** natural language processing, machine learning, machine reasoning, and autonomous agents

Web 3.0 as Different Formats of Web

The Semantic Web

The term 'Semantic Web' refers to "Defined" Web that is an alliance of World Wide Web Consortium (W3C) and others to provide a standard for defining data structures on the Web. Semantic Web refers to the use of XML-tagged data that matches the Resource Description Framework (RDF).

Sometimes it is refers to "Web 3.0," that is a debatable topic, but in the form of Web 3.0, the main goal of the Semantic Web becomes to identify exact required data that matches the keywords. e.g. if we search Web 3.0 in Google / yahoo / msn or any advance search engines using specific key words, there are millions of web pages appears on the window in which only very few have some information and all other pages are worthless.

Web 3.0 in terms of Semantic Web is the third generation of World Wide Web in which machines can read sites similar to human being and also follows your instructions. For example if you order to check your schedule against the schedules of all the dentists and doctors within a 10-mile radius if follows tour order and provide the appropriate information.

Web 3.0

The 3D Web

3-D refers to the three dimensional design that represents the virtual looks of any object from three different sides simultaneously. A user can view the true picture of any building, any location or any object and walk through the location without leaving the computer desk on his/her system. Though these are the virtual pictures but seem to be real. These technologies are extensively being used in a wide range of services like computer games, Virtual Reality (VR) models and Multimedia solutions.

Now, 3-D technology has come on the Internet and has become a new trend of Web. Now user can go house hunting across town or take a tour of the world or can walk through a Second Life–style virtual world, surfing for data and interacting with others in 3D. The 3d web is being used massively in online computer games, virtual world tour, Geospatial engineering, online high tech research, online software development, online shopping, online telecommunication and social networking sites. Google Earth, Wiki Earth, MySpace, You Tube are the biggest examples of 3D web users.

The Media-Centric Web

The terms Media-Centric Web refers to the web where users can find true similar graphics and sound on the other media, not just the keywords. e.g. if users searches any favorite movie/ graphics/ music in the search engines, can find the exact desired thing on the other media.

The Pervasive Web

The pervasive web refers the uses of web in the wide range of area in which the web has now been reached not only in computers and cell phones but also in clothing, appliances, and automobiles and much more, e.g. web based bedroom windows that checks weather and self open or close it according to climate.

Web 3.0 in terms of pervasive web refers to those websites, which are going to be transformed into web services and will depict and expand their information to the world.

Overview

As the times goes and the technology enriches, the experts feels to develop some thing better that can be more fruitful, advance, user friendly and intelligent. Thus originates the concept of web 3.0 and now it is taking a handsome shape. Web 3.0 has some more features including the feature of Web 2.0.

Web 3.0 sites will only allow collaboration of content generated from an approved pseudo-random sequence of characters. Web 3.0 would have three main objectives:

1- Seeking Information
2- Seeking Validation
3- Seeking entertainment

Seeking Information

Searching information would be more compact in Web 3.0. Till now, the web uses keywords in order to comprehensive data into usable chunks. Search engines index the Internet in proper order and present it to the end user in order of relevance. The users select the information that is nearer to their requirement. Sometimes this becomes a very hectic process. But Web 2.0 goes one step ahead and brought us a change in the basic way of searching. It applies the tags in the searching data e.g. if anyone wants to look for car. He/she types the word in the specified space of the search engine. The search engine displays many webs, but if the user type BMW cars, it displays all the relevant site onlk\y related to BMW cars. So BMW works as a tag.

Web 3.0 will be more advance in searching the information for example of Cars, Web 3.0 uses the further research beyond the engines, it also uses the

subsearch engines that would provide more compact information and user can find the nearest desired data. It would go to all major categories like pictures, videos, blog posts, news articles, commerce etc. Each of these would happed because of RSS feed so that user can get alerts when something new would added to his/her search profile.

Seeking Validation

If the user wants to go the news not the information, it will work in a different way. It would provide the exact data what user wants. It would also search the available people on the net. The user have to type the words what he/she wants to access, Web 3.0 would provide the relevant information in order of its proximity, algorithms, tagging, and validation through user voting.

Seeking Entertainment

Entertainment, the most popular trend of Web 2.0 would be more advance in Web 3.0 as it would be based around the sect of the personality. People Search will replace the social networks that are most popular fashion in this generation of web. For searching about any person, just type the name and all the information related to regarding person would be displayed with some attached tags. If would display the total wiki profile, in which all the data would be specified whether the user would have created it or anyone else. All the related deeds would also show in the profile. Then People would be more universal rather than now.

The looks and shape of the blogging would be also changed; the current weblogs would be converted in to Microblogging. People will be able to blog from anywhere, without having to spend hours writing a properly formatted post. Web 3.0 will see a more complete integration between devices like cell phones and the World Wide Web. Posting pictures, videos and text from anywhere, anytime would be more tussles free.

Commerce

Here the terms of commerce means the criteria of earning that will be more advance, but the whole criteria would not primarily change. The product will carry on to sell online. "Conversational advertising" and Advertainment will take the place of stock ads and promotions. Sect of personality and their sponsorships will also be more specific as the advertisement companies will be narrower because of categorizing of the people.

The entire advertising landscape will change; the ultra specialized subengines will search the tightly focused target audience to selling the product. Contextual advertisement will take second seat to product placements on sites, search results and subengines.

Web 3.0 Design

REST, AJAX, Silverlight, Widget Enabled, Taggable, Searchable everything…

RSS. A Web 3.0 Driver

In the coming ten years RSS and its related technologies will become the single most important Internet technology because of its specific quality to development of the new web as it's really very simple. Any person who has a little bit knowledge of coding can generate an extensible, standards based database of information that can be transferred to almost any other modern web site.

If Web 3.0 is the Semantic Web, where machine read content like human beings then RSS will be its eyes. RSS technology is still in vast uses especially in the online news portals. The entire business models have already being created around aggregating meta-data. IGoogle, MyIndiaTims and Netvibes allow the users to create their own personal homepage, drawing much of its content from RSS feeds that users select.

The trend of RSS tool will be increased in the future in which user can include a

host of data-points. Each blog post, the future microblogging feed can be personalize according to users' desire as every picture, every video clip, every music will be searchable, taggable and XML based collaborate. The biggest example of it's already exists in a web portal named 'MyIndiaTims.com'.

The real power of Web 3.0 will be in the used in creating data and transferring it effectively.

Candidate Web 3.0 technologies

Web 3.0 would be used in various technologies of computer and Internet. Here is the list of web 3.0 users:

- Artificial intelligence
- Automated reasoning
- Cognitive architecture
- Composite applications
- Distributed computing
- Knowledge representation
- Ontology (computer science)
- Recombinant text
- Scalable vector graphics
- Semantic Web
- Semantic Wiki
  Software agents

Tips 'n' Tricks

1. Download files data from many URLs and save them to the local files in the directory of your choice:

This Java program lets you download files from one or more URLs and save them in the directory where you want. This program takes destination directory for the files to save as first command line argument and URLs for the files as next command line arguments separated by space. Java provides URLConnection class that represents a communication link between the application and a URL. Invoking the openConnection method on a URL creates URLConnection object. Now get InputStream object from that connection and read the data. Finally write the data to the local file.

FileDataDownload.java:

```
import java.io.*;
import java.net.*;

public class FileDataDownload {
        final static int size=1024;
        public static void
FileDownload(String fileAddress, String
localFileName, String destinationDir) {
OutputStream os = null;
URLConnection URLConn = null;

// URLConnection class represents a
communication link between the
 // application and a URL.

InputStream  is = null;
try {
URL fileUrl;
byte[] buf;
int ByteRead,ByteWritten=0;
fileUrl= new URL(fileAddress);
os = new BufferedOutputStream(new
FileOutputStream(destinationDir+"\\"+
localFileName));

//The URLConnection object is
created by invoking the
// openConnection method on a URL.
URLConn = fileUrl.openConnection();
is = URLConn.getInputStream();
buf = new byte[size];
while ((ByteRead = is.read(buf)) != -1) {
os.write(buf, 0, ByteRead);
ByteWritten += ByteRead;
}

System.out.println("Downloaded
Successfully.");
System.out.println("File name
:\""+localFileName+ "\"\nNo of
 bytes :" + ByteWritten);
}
catch (Exception e) {
e.printStackTrace();
}
finally {
try {
is.close();
os.close();
}
catch (IOException e) {
e.printStackTrace();
}
}
}

public static void fileDownload(String
fileAddress, String destinationDir)
 {

// Find the index of last occurance of
character '/' and '.'.

int lastIndexOfSlash =
fileAddress.lastIndexOf('/');

int lastIndexOfPeriod =
fileAddress.lastIndexOf('.');

// Find the name of file to be downloaded
from the address.
String fileName=fileAddress.substring
(lastIndexOfSlash + 1);

// Check whether path or file name is given
correctly.
```

```
if (lastIndexOfPeriod >=1 &&
lastIndexOfSlash >= 0 &&
lastIndexOfSlash < fileAddress.length() -
1)
{
FileDownload(fileAddress,fileName,destinationDir);
}
else
{

System.err.println("Specify correct path or
file name.");

}
}
        public static void main(String[] args)
{

// Check whether there are atleast
two arguments.

if(args.length==2)
{
for (int i = 1; i < args.length; i++) {
fileDownload(args[i],args[0]);
}
}

else{System.err.println("Provide
\"Destination directory path\" and \"file
                                    names\"
separated by space.");
}
}
}
```

Compile and Run:

In this example, the location for the directory where file "**jsf.htm**" is to be saved is "**c:\download**".

```
C:\JavaJazzup>javac
FileDataDownload.java
C:\JavaJazzup>java FileDataDownload
c:\download http://localhost:8080/
tomahawk_tags/pages/jsf.htm
```

Output:

```
Downloaded Successfully.
File name :"jsf.htm"
No of bytes :82452
```

2. Using Swing Timer:

A Swing Timer fires an event after a specified delay of time. Swing Timer can be used to perform a task once, after a delay and to perform a task repeatedly. For example, determining when to display and hide a tool tip, updating a component that displays the progress. Swing timers are very easy to use. Create **actionPerformed()** method, which is called when action listener is notified. So write the code for the task to be performed, in **actionPerformed()** method. When creating timer by instantiating the Timer class, we specify the time in milliseconds which is the time interval of invoking the **actionPerformed()** each time. To start the timer, just call its **start()** method and to stop it doing anything, call **stop()** method. To understand how the swing timer can be used with the progress bar component showing its status is given below:

SwingTimer.java:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.html.*;

public class SwingTimer{
    final static int interval = 1000;
    int i;
    JLabel label;
    JProgressBar pb;
    Timer timer;
    JButton button;
    public SwingTimer() {
JFrame frame = new JFrame("Swing Timer
Example");
button = new JButton("Start");
button.addActionListener(new
ButtonListener());
```

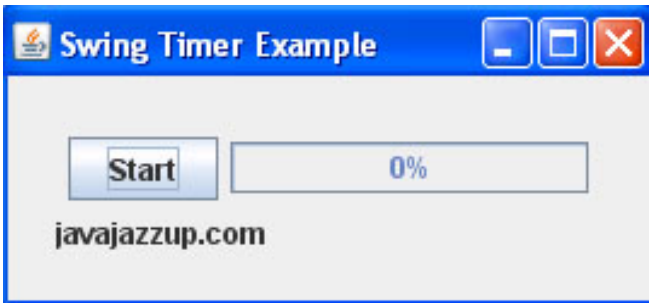
```
pb = new JProgressBar(0, 20);
pb.setValue(0);
pb.setStringPainted(true);
label = new JLabel("Roseindia.net");
JPanel panel = new JPanel();
panel.add(button);
panel.add(pb);
JPanel panel1 = new JPanel();
panel1.setLayout(new BorderLayout());
panel1.add(panel, BorderLayout.NORTH);
panel1.add(label, BorderLayout.CENTER);
panel1.setBorder
(BorderFactory.createEmptyBorder(20, 20,
20, 20));
frame.setContentPane(panel1);
frame.pack();
frame.setVisible(true);
frame.setDefaultCloseOperation
(JFrame.EXIT_ON_CLOSE);

//Create a timer.
timer = new Timer(interval, new
ActionListener() {
public void actionPerformed(ActionEvent
evt) {
if (i == 20){
timer.stop();
button.setEnabled(true);
pb.setValue(0);
String str = "<html>" + "<font
color=\"#FF0000\">" + "<b>" +
"Downloading completed." + "</b>" + "</
font>" +   "</html>";
label.setText(str);

i = i + 1;
pb.setValue(i);
}
}
);
}

class ButtonListener implements
ActionListener {
public void actionPerformed(ActionEvent
ae) {
button.setEnabled(false);
i = 0;
String str = "<html>" + "<font
color=\"#008000\">" + "<b>" +
"Downloading is in process....." + "</b>"
+ "</font>" +"</html>";
label.setText(str);
timer.start();
}
}
public static void main(String[] args) {
SwingTimer spb = new SwingTimer();
}
}
```

Compile and Run:

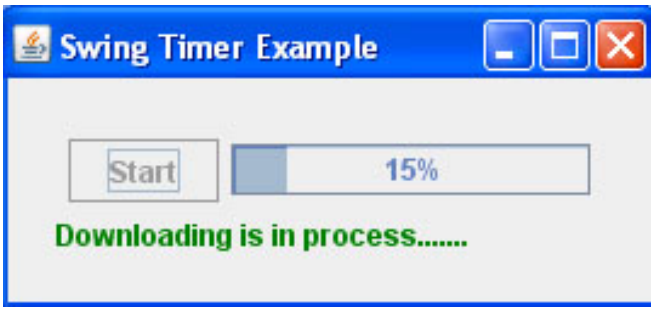
C:\JavaJazzup>javac SwingTimer.java
C:\JavaJazzup>java SwingTimer

Output:

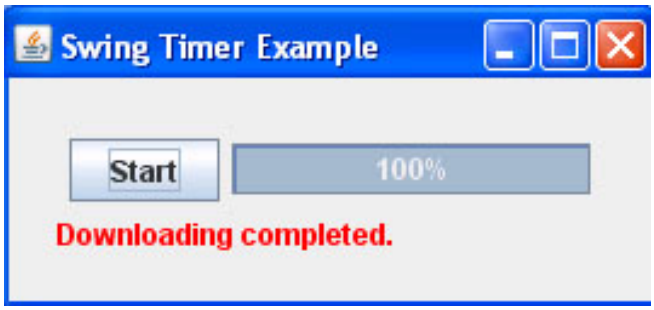
The output of the above program appears as shown in the figure below.



When the start button is clicked, a string "Downloading is in process…." appears at the bottom as in the figure below.



The above string is replaced by the string "Downloading completed." when the task is completed.



3. Creating Indeterminate Progress Bar in Java:

Sometimes, at the time of downloading or transferring a file, we see a component showing how much the particular task has been completed. For this purpose, Java provides a Progress Bar component to convey the progress of completing a task in a GUI. One way of showing this progress is in percent format. But suppose a situation where extent of the task is unknown and so we are not able to present it in percent format. To handle this situation java provides a solution by putting the progress bar in indeterminate mode. An indeterminate progress bar animates constantly. By default, every progress bar is determinate. You may make any JProgressBar indeterminate pass a "true" boolean value to setIndeterminate() method. Just have a look on the code below:

IndeterminatePB.java:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class IndeterminatePB extends
JFrame {
JProgressBar ipb;
int number = 0;
public IndeterminatePB() {
super("Indeterminate Progress Bar
Example");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JPanel pane = new JPanel();
ipb = new
JProgressBar(JProgressBar.HORIZONTAL,0,
1000);
//make any JProgressBar indeterminate
passing a true boolean value to
setIndeterminate() method.

ipb.setIndeterminate(true);
pane.add(ipb);
setContentPane(pane);
}
public void iterate() {
while (number <= 1000) {
ipb.setValue(number);

//Sets the progress bar's current value to n.

try {
Thread.sleep(10);
}
catch (InterruptedException e) { }
number++;
}
//To stop the animation make the progress
bar determinate.

ipb.setIndeterminate(false);
}
public static void main(String[] args) {
IndeterminatePB frame = new
IndeterminatePB();
frame.pack();
frame.setVisible(true);
frame.iterate();
}
}
```

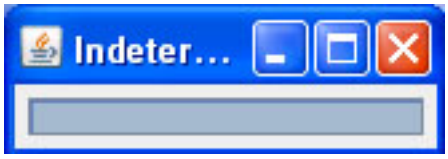
Compile and Run:

```
C:\JavaJazzup>javac IndeterminatePB.java
C:\JavaJazzup>java IndeterminatePB
```

Output:
The output below appears until the completion of the task. The dark portion waves from one side to the other.

When the task ends, progress bar

appears like this.



4. Wishing New Year with count down in full screen window:

This program displays Frame in full screen window in which the remaining number of seconds for the New Year to come is displayed. Screen is updated every second and finally the message "Happy New Year 2008" is displayed.

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.text.NumberFormat;
public class GreetingNewYear
implements Runnable{

JFrame frame;
JLabel label;
long timeInMillis;
String greetingMessage;

static NumberFormat formatter =
NumberFormat.getInstance();
public GreetingNewYear (JFrame frame,
JLabel label)
        {
this.frame = frame;
this.label = label;

// Set values for New Year.

Calendar calendar = new
CregorianCalendar();
int newYear = calendar.get(Calendar.YEAR)
+ 1;
calendar.set(Calendar.YEAR, newYear);
calendar.set(Calendar.MONTH,
Calendar.JANUARY);
calendar.set(Calendar.DAY_OF_MONTH, 1);
calendar.set(Calendar.HOUR_OF_DAY, 0);
calendar.set(Calendar.MINUTE, 0);
calendar.set(Calendar.SECOND, 0);
timeInMillis = alendar.getTime().getTime();
// set the message to be displayed for
greeting.
greetingMessage = "Happy New Year " +
newYear;
}
public static void main(String[] args)  {
JFrame frame = new JFrame();
frame.setUndecorated(true);
JLabel label = new JLabel(".");
// Set the alignment of the text to be
displayed.
label.setHorizontalAlignment
(SwingConstants.CENTER);
frame.getContentPane().add(label);
// Set the size of window to full of the
screen.
GraphicsEnvironment.getLocal
GraphicsEnvironment().
getDefaultScreenDevice().
setFullScreenWindow(frame);
// Set the style, type and size for the
font of the label.
label.setFont(new
Font("monospaced",Font.BOLD,100));
new HappyNewYear(frame, label).run();
}
public void run()
{
boolean isNewYear = false;
do
{
long timeLeft= (timeInMillis -
System.currentTimeMillis()) /1000L;
String displayMessage;
// Check for New Year.
if (timeLeft< 1){
isNewYear = true;
displayMessage = greetingMessage;
}
else
{
displayMessage =
formatter.format(timeLeft);
// Set the above displayMessage
value to the label text.
label.setText(displayMessage);
```

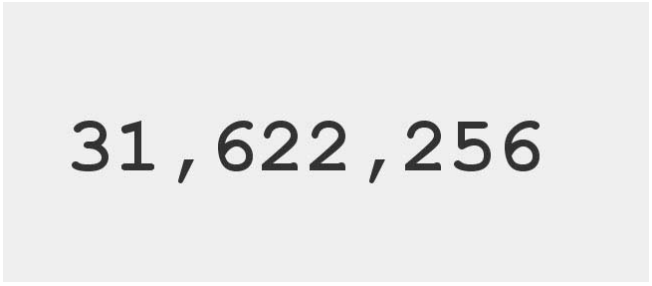
```
// Make the thread sleep for one second.
try{
Thread.sleep(1000);
}
catch (InterruptedException ie)
{
ie.printStackTrace(System.err);
}
}
while (!isNewYear);
}
}
```

Compile and Run:

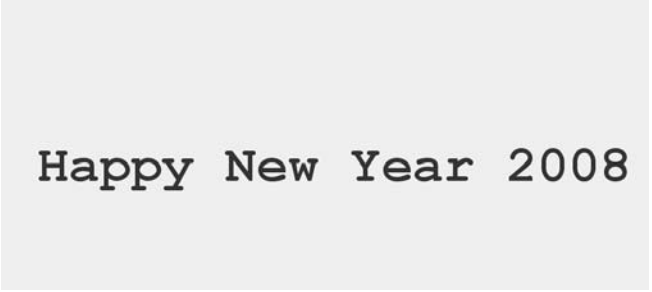
C:\JavaJazzup>javac GreetingNewYear.javaC:\JavaJazzup>java GreetingNewYear

Output:

The output shows the remaining number of seconds. This screen gets updated every second and no of seconds displayed in the screen becomes one less than previous. So this count down goes on till New Year comes.



When New Year comes, it wishes for the New Year like in the figure below:



5. Save your documents in the directory of your choice in zipped form:

Zip file is a way of packaging a set of files in the compressed form. When a developer needs to use a set of files together, bundling them together in zipped form is one of the most frequently adopted way. When we need to download a group of files from the web, we create a zip file to transport them as a single unit. In this section, we will learn creating zip file containing two files of your choice in the specified directory.

CreateZip.java:

```
import java.io.*;
import java.util.zip.*;
public class CreateZip{
 public static void main(String args[]){
//Collect file names to be zipped in an array by command line arguments.
String[] filesToZip = new String[]{args[0], args[1]};
// Create a buffer for reading the files
byte[] buffer = new byte[1024];
try {
// Specify the location of the zipped file to be stored.
 String zip = "c:\\MyDocuments\\MyZippedDocuments.zip";
//Creates a new ZIP output stream.
ZipOutputStream out = new ZipOutputStream(new FileOutputStream(zip));
// Compress the files  for (int i=0; i<filesToZip.length; i++) {
FileInputStream in = new FileInputStream(filesToZip[i]);
// Add ZIP entry to output stream.
out.putNextEntry(new ZipEntry(filesToZip[i]));
//ZipEntry(String name)Creates a new zip entry with the specified name.
// Transfer bytes from the file to the ZIP file
int len;
while ((len = in.read(buffer)) > 0) {
```

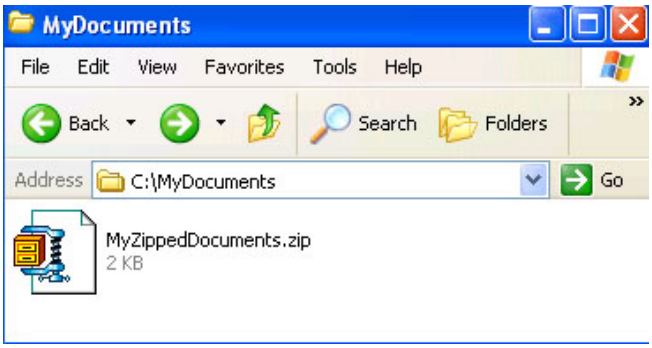
```
out.write(buffer, 0, len);
//Writes an array of bytes to the
current
// ZIP entry data.                }
out.closeEntry();
//Closes the current ZIP entry and
positions the stream
//for writing the next entry.

in.close();
}
out.close();

// Closes this output stream and
releases any system resource
//associated with the stream.
}
 catch (Exception e) {
 e.printStackTrace();
}
}
}
```

Compile and Run:

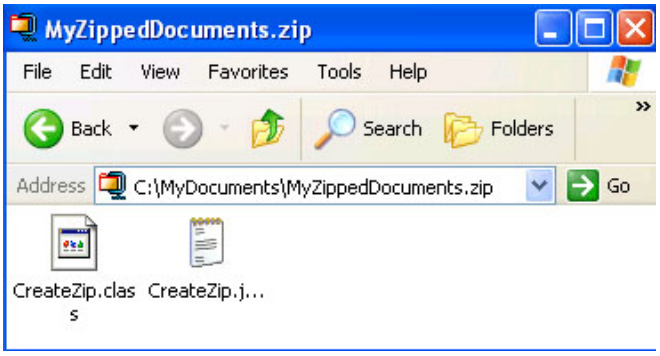
C:\JavaJazzup>javac CreateZip.java
C:\JavaJazzup>java CreateZip
CreateZip.java CreateZip.class

Output:

Running the above program creates a zip file named "MyZippedDocuments.zip" which contains two files **"CreateZip.java"** and **"CreateZip.class"**. You can take a look below:



Exploring the above file shows two files inside it. You can see this in the figure below:



6. Want to scribble on the Applet window?

Do you know how to write with the mouse on the applet window? It's an easy task. Select and save position (X and Y coordinates) of the mouse on mouse down event and position on mouse drag. Just draw a line between these two points. This process goes on until mouse is dragged. When the mouse is pressed down again the line is drawn again on the applet window.

MouseScribbling.java:

```
import java.awt.*;
import java.applet.*;
public class MouseScribbling extends
Applet {
// Declare variables to store mouse
point location.
int startX,startY;
// This method is called when the
user press down the mouse button.
public boolean mouseDown
(Event e, int startX, int startY)
{
// Store the point location.
this.startX = startX;
this.startY = startY;
return true;
}
// This method is called when the
user drags the pressed mouse.
```

```
public boolean mouseDrag(Event e, int
endX, int endY)
{
Graphics g = getGraphics();
// Draw a line from start point to the
end point.
g.drawLine(startX, startY, endX, endY);
// Change the start point with the end
point.
startX = endX; startY = endY;
return true;
}
}
```

MouseScribbling.html:

```
<HTML>
<BODY>
<APPLET CODE="MouseScribbling"
WIDTH="200"
HEIGHT="300">
</APPLET>
</BODY>
</HTML>
```

Compile and Run:

C:\JavaJazzup>javac MouseScribbling.javaC:\JavaJazzup>appletviewer MouseScribbling.html

Output:

Running the applet creates an applet window like below where you can write anything by pressing and dragging the mouse. For instance, in the figure below "Java" word is written in the applet window.

7. Learn running external programs (Notepad, Calculator, browser etc.) by a Java program:

Opening any sub process like Notepad, Calendar, browser etc from a java program feels good for a new java programmer. The class Runtime provides a static method **getRuntime()**, which helps to work with Java Runtime Environment. This is the only way to get the reference to the Runtime object. Now, by invoking its method **exec()**, external programs can be invoked. Here is a simple demonstration for performing this operation from a Java program.

CreateSubProcess.java:

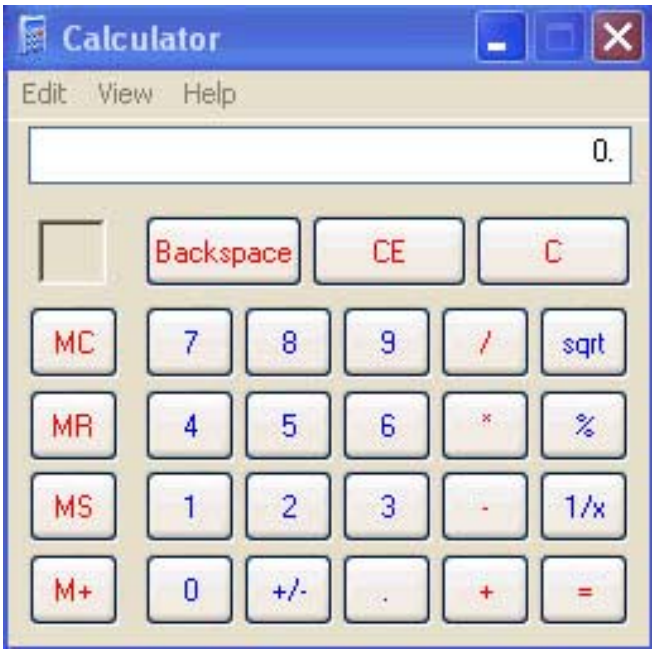
```
public class CreateSubProcess
 {
public static void main(String[] args) {
 try {
//Get the current runtime to interface
with the environment in
//which the application is running.

Runtime runTime = Runtime.getRuntime();
System.out.println("Solve the problem
within 1 minute.");
Process process = runTime.exec("calc");

//Executes the specified string
// command  in a separate process.

try {
Thread.sleep(60*1000);
}
catch (InterruptedException e) {
e.printStackTrace();
}
process.destroy();
// Kills the subprocess forcibly.

int exitValueCalc = process.exitValue();

//The exit value 0 indicates normal
termination.

System.out.println("Process exitValue: " +
exitValueCalc);
}
catch (Throwable e) {
e.printStackTrace();
}
}}
```

Compile and Run:

C:\JavaJazzup>javac CreateSubProcess.javaC:\JavaJazzup>java CreateSubProcessSolve the problem within 1 minute.

Output:

Running the above program prints the message "**Solve the problem within 1 minute.**" and opens a calculator.



8. Can you find the name of Operating System you are working on and environment variables by a Java program?

Environment variables are **key/value** pairs, which are available to all the programs running within DOS environment or top of it, including Windows. Because these can be changed so they are called variables. Java provides a way to access these variables by providing System class. To get the value of a single variable, call its static method **getProperty("property_name")**. To get value of all the variables, use static method **getenv()** which returns values as a Map. Now using collection methods, keys and values can be displayed.

EnvironmentInformation.java:

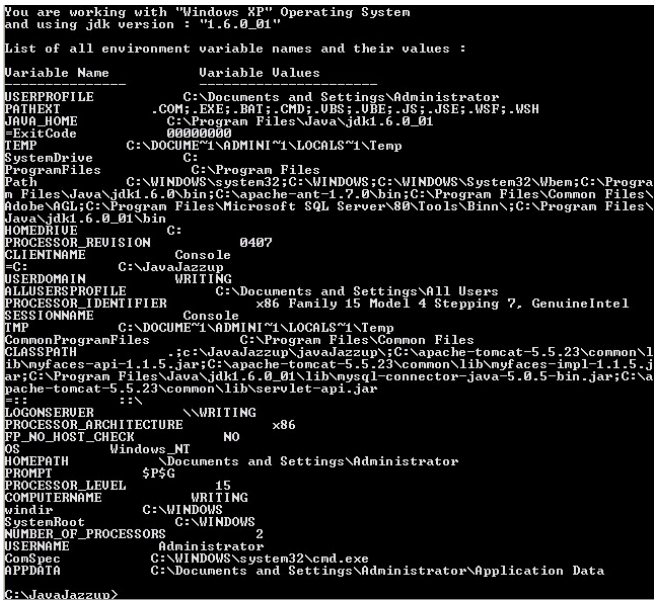
```
import java.util.Map;
import java.util.Set;
import java.util.Iterator;
public class EnvironmentInformation{
public static void main(String[] args){
System.out.println("\n"+"You are working
with
\""+System.getProperty("os.name")+"\"
Operating System ");
System.out.println("and using jdk version :
\""+System.getProperty("java.version")+"\"\n");
Map map = System.getenv();
Set keys = map.keySet();
Iterator iterator = keys.iterator();
System.out.println("List of all environment
variable names and their values :"+"\n");
System.out.println("Variable Name
Variable Values");
System.out.println("————————\t\t————
————————");
while (iterator.hasNext())
{
String key = (String) iterator.next();
String value = (String) map.get(key);
System.out.println(key + "          " +
value);
}
}
}
```

Compile and Run:

C:\JavaJazzup>javac
EnvironmentInformation.javaC:\JavaJazzup>java
EnvironmentInformation

Output:

Above program shows environment variables as it is shown in the figure below.



Are you really confident about the execution flow of a java program?
Sometimes, even a simple java program, can make you confused about the flow of initialization of variables and flow of program. If you are clear with the concept of "static" keyword and flow of constructor calling (constructor chainging) then the program below is simple to understand. The compiler combines multiple static initializer blocks into a single initialization procedure (which is executed one time only when the class is loaded) so the order of execution of program code will not necessarily be the same as the order in which the code appears in the program i.e. non-static code can separate multiple static initializer blocks. The static initializer blocks are executed in the order in which they appear in the code, regardless of the other code that may separate them. This causes the program to be a little difficult to understand.

In the program below first static variables are initialized. Then main method is called which calls the SubClass constructor. Here constructor chaining is applied in which every constructor calls its super class constructor. So before entering into constructor of SubClass, its instance variables are initialized. Now, control goes to call the constructor of

SuperClass without entering into constructor of SubClass. Here also the instance variables of SuperClass are initialized and then super class's constructor is called then control comes back to call the constructor of SubClass.

SubClass.java:

//This example demonstrates the execution order of a java program.

```
class SuperClass {
private String supVar1=staticPrintMethod
("\"SuperClass.supVar1\" initialized");
public int supVar2;
superClass(){
System.out.println("In \"SuperClass\"
constructor.");
supVar1="java";
supVar2=10;
System.out.println("supVar1 = " + supVar1
+ ", supVar2 = " + supVar2);
}
private String staticVarInSuperClass =
staticPrintMethod("static
\"SuperClass.staticVarInSuperClass\"
initialized");
static String staticPrintMethod(String str) {
System.out.println(str);
return "success";
}}
public class SubClass extends SuperClass {
private String subVar1 =
staticPrintMethod("\"SubClass.subVar1\"
initialized");
public SubClass() {
System.out.println("In \"SubClass\"
constructor.");
System.out.println("subVar1 = " +
subVar1);
}
private static String staticVarInSubClass =
staticPrintMethod("static
\"SubClass.staticVarInSubClass\"
initialized");
public static void main(String[] args) {
System.out.println("In main() method.");
SubClass b = new SubClass();
}}
```

Compile and Run:

C:\JavaJazzup>javac SubClass.javaC:\JavaJazzup>java SubClass

Output:

```
static "SuperClass.staticVarInSuperClass" initialized
static "SubClass.staticVarInSubClass" initialized
In main() method.
"SuperClass.supVar1" initialized
In "SuperClass" constructor.
supVar1 = java, supVar2 = 10
"SubClass.subVar1" initialized
In "SubClass" constructor.
subVar1 = success
```

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

Valued Java Jazz Up 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 Reader's Forum

If there's something you're curious about, we're confident that your curiosity, combined with the knowledge of other participants, will be enough to generate a useful and exciting Reader's Forum. If there's a topic you feel needs to be discussed at JavaJazzup, it's up to you to get it discussed.

Convene a discussion on a specific subject

If you have a topic you'd like to talk about. Whether it's 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 Reader's Forum. If you like, you can prepare a really a good article to explain what you're interested to tell java technocrates about.

Sharing Expertise on Java Technologies

If you're a great expert on a subject in java, the years you spent developing that expertise and want to share it with others. If there's something you're an expert on that you think other technocrates might like to know about, we'd love to set you up in the Reader's 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

we're inviting people from lots of different worlds. We do not expect everybody at Reader's 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 reader's around the globe.

Post resourceful URLs

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

Anything else

if you have another idea for something you'd like to do, talk to us. If you want to do something that we haven't thought of, have a crazy idea, we'd really love to hear about it. We're open to all sorts of suggestions, especially if they promote reader's participation.