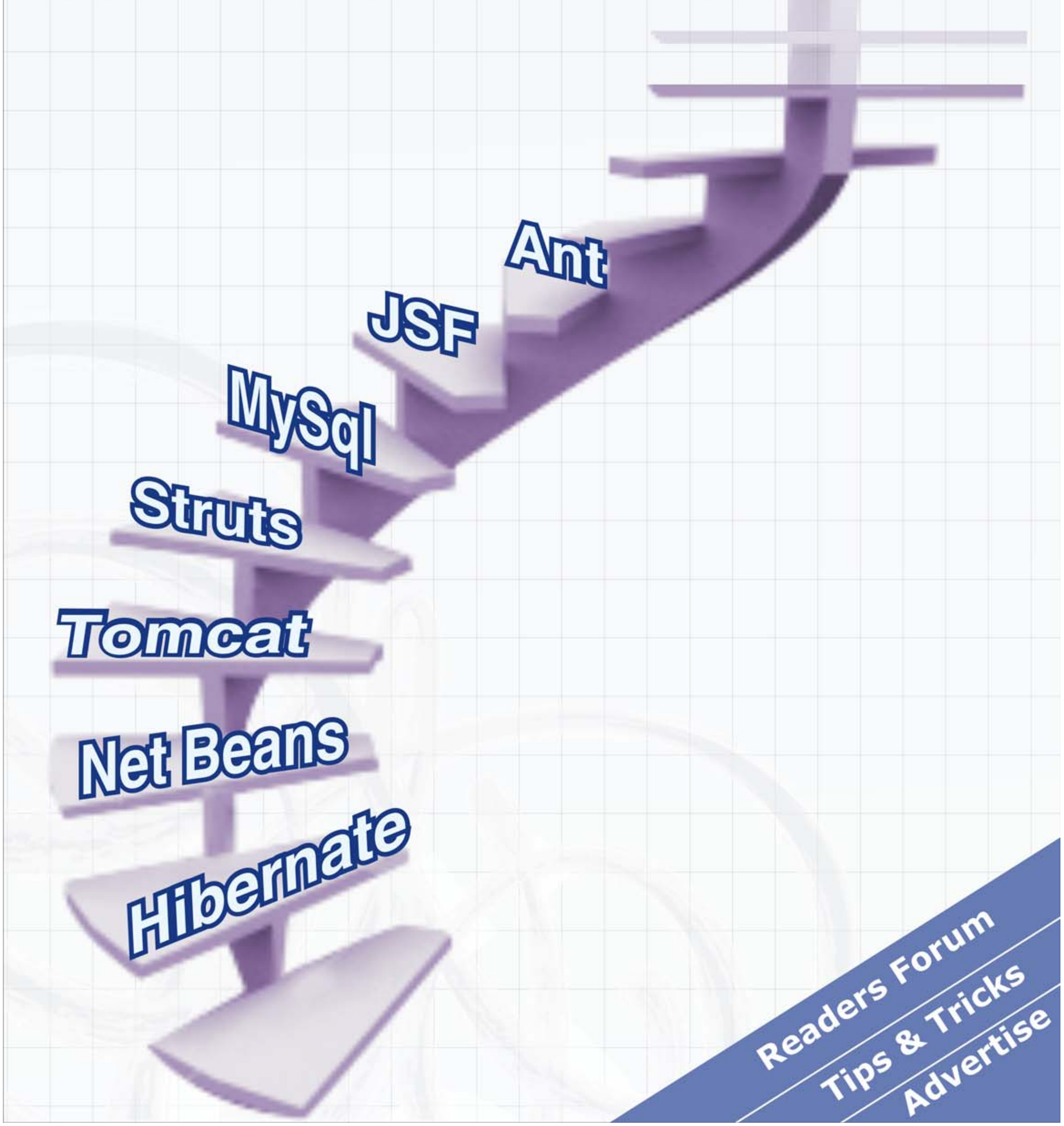
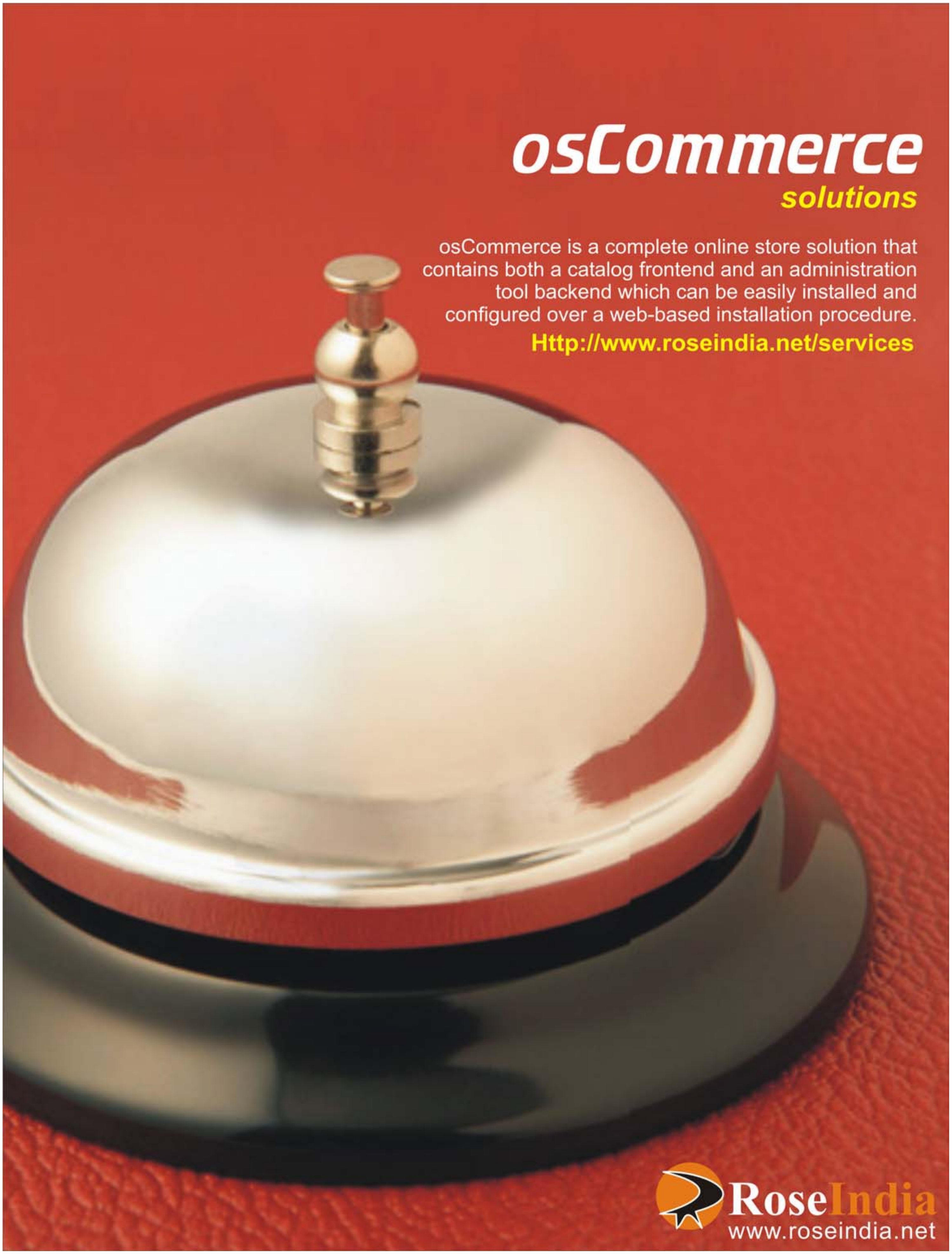
Dec 07 | Volume 1 |Issue 6 | **Free**

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING





osCommerce

solutions

osCommerce is a complete online store solution that contains both a catalog frontend and an administration tool backend which can be easily installed and configured over a web-based installation procedure.

Http://www.roseindia.net/services

RoseIndia
www.roseindia.net

Java Jazz up
A BETTER WAY TO LEARN PROGRAMMING

December 2007 Volume I Issue VI

"Learning new technologies demands individuals to work together to construct shared understandings and knowledge."

Published by

RoseIndia

JavaJazzUp Team

Editor-in-Chief

Deepak Kumar

Editor-Technical

Ravi Kant
Ashish Srivastava
Vinod Kumar

Sr. Graphics Designer

Suman Saurabh

Graphics Designer

Santosh Kumar

Register with JavaJazzUp

and grab your monthly issue

"Free"

Editorial

Dear Readers,

We are back here with the Dec' 07 issue of Java Jazz up. The current edition is specially designed for the sprouting technocrats. This issue highlights the interesting Java technologies especially for the beginners.

Though it was a hard job to simplify the complexities of the technologies like Tomcat, Hibernate, MySQL, struts 2, JSF and Design Patterns. Still our team has done a marvelous work in making it easy and simpler for the new programmers regime. This issue reflects our consistent attempts to avail the quality technological updates that enforce the readers to appreciate it a lot and be a part of its Readers Community.

Java News and Updates section provides the latest updates of the things happening around the globe making the readers aware of the java technological advancement. In this section, you will know the new features introduced in the existing tools, utilities, application servers, IDEs, along with the Java API updates.

The set of articles conferring technologies like Design patterns, JSF, Hibernate, MySQL Database etc. are provided in such a manner that even a novice learns and implements the concepts in a very easy manner.

We are providing it in a PDF format so that you can view and even download it as a whole and 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-in-Chief
Deepak Kumar
Java Jazz up

Content

05 **Java News |** Red Hat and Sun Collaborate on Java Development: Open source solution vendor Red Hat has signed an OpenJDK Community agreement and joined the Open Java project.

07 **NetBeans IDE |** The NetBeans IDE, product of Sun Microsystems, is a free, open-source integrated development environment written entirely in Java for software developers.

12 **Introduction to Tomcat Server |** Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies.

18 **Introduction to MySQL |** MySQL is an open source Relational Database Management System based on the Structured Query Language (SQL). It is very fast reliable and flexible Database Management System based on relation model that is developed to manage large volumes of data at very high speed with security.

26 **Introduction to Hibernate |** When an application is developed, a major portion of that application involves the creation and maintenance of the persistence layer to store and retrieve objects from the database. Whenever any changes are made to the underlying database schema, it can be expensive to spread those changes to the rest of the application.

35 **ANT |**Initially, it was not an easy task to work on the large java projects but thanks to ant for simplifying the complex details apart from the development concepts that the software developers have to keep in regard while developing the java software.

41 **Struts2 Tags |** Apache Struts is an open-source framework used to develop Java web applications. Strut1 with all standard Java technologies and packages of Jakarta assists in creating an extensible development environment.

51 **Integrating Struts and Hibernate|** This article explains the integration of Struts and Hibernate, two worldwide-accepted frameworks in the landscape of development of enterprise applications. Completing this article will make you able to use Hibernate in your Struts project. We will be using Hibernate Struts plug-in to write high performance application using Struts and Hibernate.

62 **JSF Application |** not only let the developers use converters provided already by the implementation but also create their own converters according to the requirement of the application. This topic explains about how to create custom converter.

65 **Design Pattern |** Behavioral patterns are those patterns, which are specifically concerned with communication (interaction) between the objects. The interactions between the objects should be such that they are talking to each other and are still loosely coupled. The loose coupling is the key to n-tier architectures.

70 **Tips & Tricks|** Java provides a lot of fun while programming. This article shows you how to use Java.awt.Robot class that is both useful and fun. Robot class is used to take the control of mouse and keyboard, which lets you do any mouse and keyboard related operation through java code for the purposes of test automation, self-running demos.

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

80 **Valued JavaJazzup Readers Community |** We invite you to post Java-technology oriented stuff. It would be our pleasure to give space to your posts in JavaJazzup.

Java News and Releases

1. Red Hat and Sun Collaborate on Java Development

Open source solution vendor Red Hat has signed an OpenJDK Community agreement and joined the Open Java project. Now, all Red Hat engineers can join any of Sun-led free and open source projects and also get access to Sun's Java test suite to ensure complete compatibility for developed Java applications. Now the community gets a more open, collaborative development process on Java, which will result in faster release cycles. "Through these strategic agreements, Red Hat commits to contribute to the Java platform and distribute a compatible, open source Java software implementation.", said Sacha Labourey, chief technical officer at JBoss, a division of Red Hat.

2. Eclipse, a new member of JCP

Now, Eclipse Foundation, open-source tooling organization, is one of the newly elected members of Java Community Process (JCP) executive committee.

Milinkovich, executive director of Eclipse, said, "We are very honored to be elected to the JCP executive committee. The success of the Java community is very important to the long-term success of Eclipse. Therefore, I expect to be an active participant in ensuring the JCP moves towards a more open and collaborative environment for setting the future direction of Java."

Patrick Curran, director of the JCP Program at Sun, said, "Eclipse adds open-source expertise, which will help the JCP broaden its push toward greater collaboration and openness."

3. Google Releases Android SDK Preview

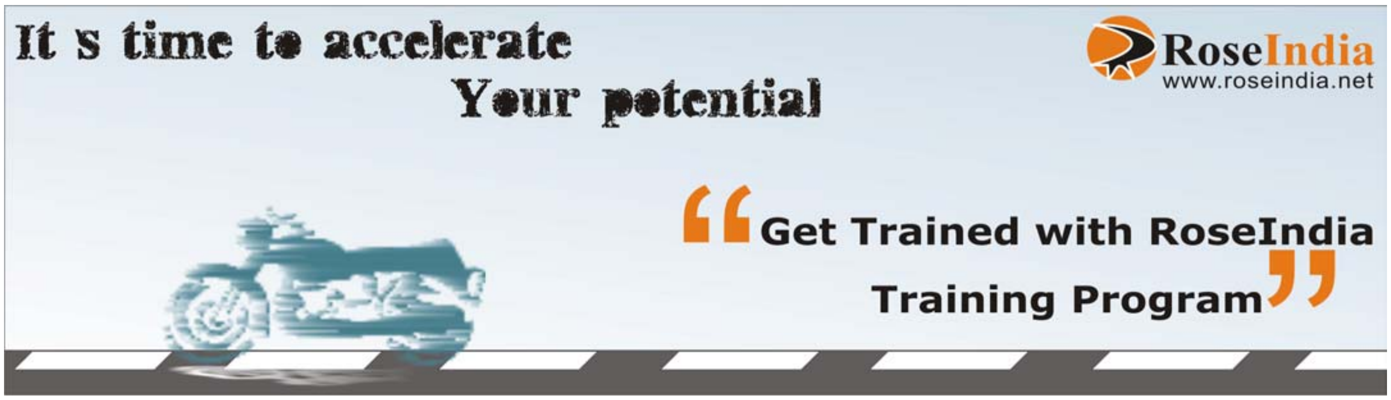
Google has released an "early look" version of its SDK (software development kit) for mobile phones. Android SDK is Eclipse-based that lets the user write Java applications. These applications run on Dalvik, which is a virtual machine, designed to run on top of Linux in embedded applications.

4. Spring Framework 2.5 Released

SpringSource, an open source software provider, formerly known as Interface21, has released Spring Framework 2.5, the latest upgrade to its Java/J2EE application framework. It adds configuration approaches with broad support for configuration annotations like JSR 250's @PostConstruct, @PreDestroy and @Resource, Java EE 5's @WebServiceRef and @EJB and Spring's @Autowired. It adds full Java 6 and Java EE 5 support. Additional messaging support, testing capabilities have been improved with this release.

5. Jasypt 1.4 released

Jasypt 1.4 (Java Simplified Encryption) has been released. New features in version 1.4 include encryptable properties files, encryptable Hibernate data source configuration, new command line tools, Apache wicket integration for URL encryption and upgraded documentation. Jasypt is a java library that enables developers to add basic encryption capabilities to their projects in a simple way without having to know much about cryptography.



Java news and releases

6. Apache Maven Continuum 1.1 Final

The Apache Maven Continuum team announced the Continuum 1.1 final release. This new release highlights mainly on bug fixes and adds new backup tool with xmlrpc for continuum db. Continuum is a continuous integration server for building Java based projects. It supports a wide range of projects such as Maven 1, Maven 2, Ant, and Shell scripts and provides features like easy installation, easy configuration, SCM support, Change set support, Build notification, Build tool support etc.

7. JRuby 1.1 Beta 1 Released

JRuby, Java powered Ruby implementation, has released version 1.1 beta 1. This is the most compatible release with the stable Ruby 1.8 compiler. It was initially released with the version 1.0 in June. According to Lead developer Thomas Enebo, "Ruby code can completely compile in an Ahead Of Time (AOT) or Just In Time (JIT) mode; yielding a faster Ruby! It uses less memory than our previous releases."

8. Apache Cocoon 2.2-RC2 Released

Apache Cocoon Community announced the second release candidate of Cocoon 2.2. Apache Cocoon is a Spring-based framework built around the concepts of separation of concerns and component-based development. Cocoon 2.2 introduces blocks. A block is the unit of modularization and packaged as a Java archive following certain conventions concerning the directory structure.

NetBeans IDE

The NetBeans IDE, product of Sun Microsystems, is a free, open-source integrated development environment written entirely in Java for software developers. NetBeans IDE supports developers providing all the tools needed to create all Java application types i.e. J2SE, web, enterprise and mobile applications. It runs on many platforms like Windows, Linux, Solaris and MacOS. It's very easy to install the NetBeans IDE. The current version is NetBeans IDE 5.5.1, which was released in May 2007.

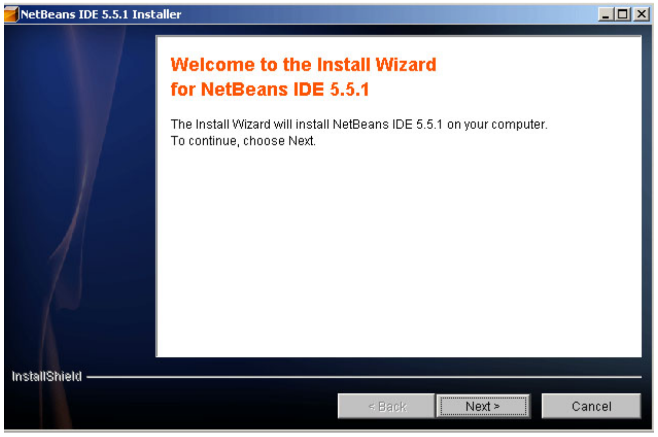
This topic explains about installation steps of NetBeans 5.5.1 followed by creation of a simple application displaying "Hello World" in the standard output. This application will help you understanding the workflow of the application and IDE.

Working with NetBeans:
To use NetBeans in your system, it is required to download its appropriate copy along with java runtime environment. Downloading and installation steps has been summarized below:

Downloading NetBeans:
NetBeans can be obtained from http://www.netbeans.info/downloads/index.php. Download the latest release of NetBeans as **.exe** file named as "**netbeans-5_5_1-windows.exe" at any place of your choice**.

Installing NetBeans:

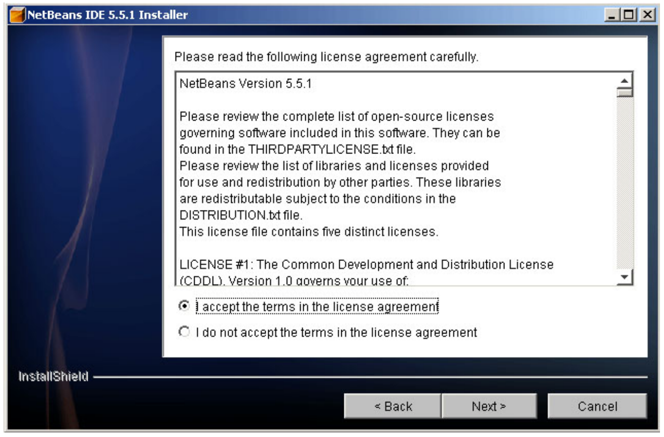
1. Double click the exe file to launch the installer and click next.



Welcome to the Install Wizard for NetBeans IDE 5.5.1

The Install Wizard will install NetBeans IDE 5.5.1 on your computer. To continue, choose Next.

2. Accept the license agreement and click Next button.



Please read the following license agreement carefully.

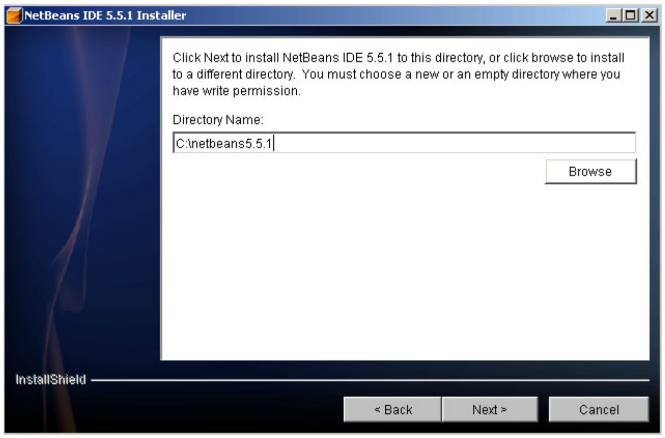
NetBeans Version 5.5.1

Please review the complete list of open-source licenses governing software included in this software. They can be found in the THIRDPARTYLICENSE.txt file.
Please review the list of libraries and licenses provided for use and redistribution by other parties. These libraries are redistributable subject to the conditions in the DISTRIBUTION.txt file.
This license file contains five distinct licenses.

LICENSE #1: The Common Development and Distribution License (CDDL), Version 1.0 governs your use of:

[✓ I accept the terms in the license agreement]
I do not accept the terms in the license agreement

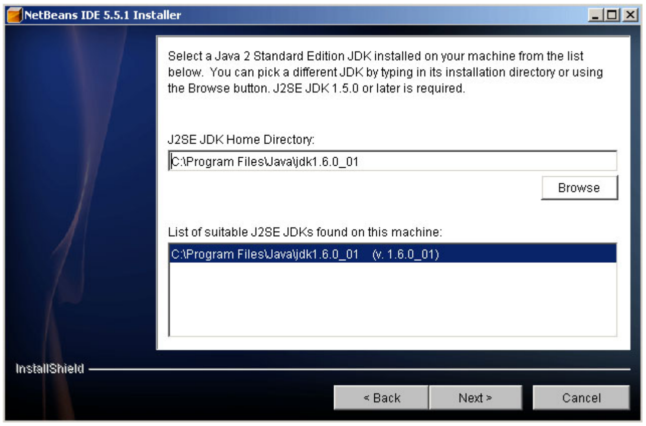
3. Select the directory of your choice clicking the browse button where you want to install the NetBeans IDE (**netbeans5.5.1**, in our case) and click Next button.



Click Next to install NetBeans IDE 5.5.1 to this directory, or click browse to install to a different directory. You must choose a new or an empty directory where you have write permission.

Directory Name: C:\netbeans5.5.1

4. Select the suitable JDK from the list that will be used by the IDE and click Next.

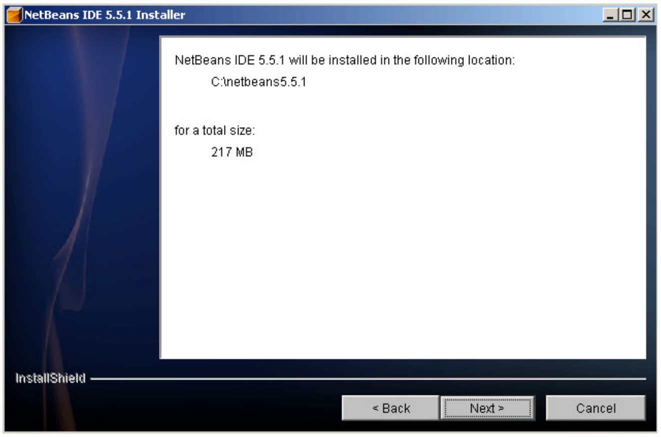

NetBeans IDE 5.5.1 Installer

Select a Java 2 Standard Edition JDK installed on your machine from the list below. You can pick a different JDK by typing in its installation directory or using the Browse button. J2SE JDK 1.5.0 or later is required.

J2SE JDK Home Directory:
C:\Program Files\Java\jdk1.6.0_01

List of suitable J2SE JDKs found on this machine:
C:\Program Files\Java\jdk1.6.0_01 (v. 1.6.0_01)

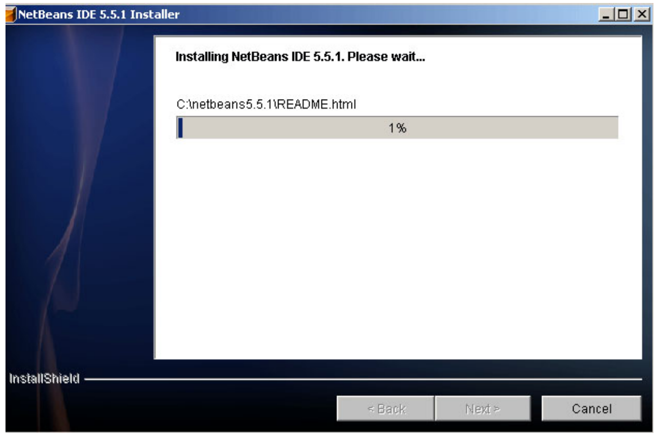
5. Verify the installation location and available space on the system for the installation.


NetBeans IDE 5.5.1 Installer

NetBeans IDE 5.5.1 will be installed in the following location:
C:\netbeans5.5.1

for a total size:
217 MB

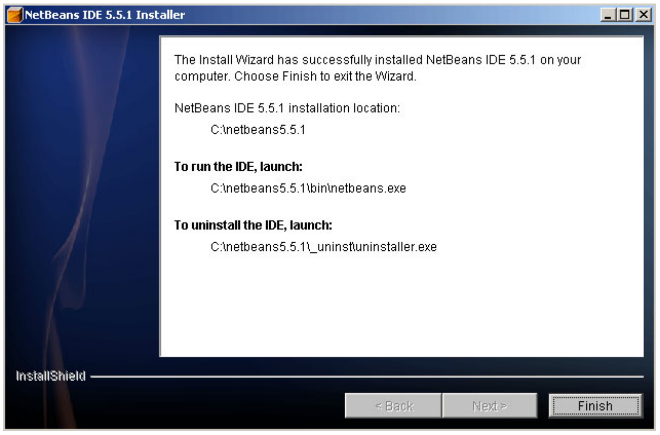
6. Clicking next starts installing the NetBeans IDE 5.5.1 to the appropriate place (**netbeans5.5.1**, in our case).


NetBeans IDE 5.5.1 Installer

Installing NetBeans IDE 5.5.1. Please wait...

C:\netbeans5.5.1\README.html
1%

7. When the installation is completed successfully, click Finish button to exit the wizard.


NetBeans IDE 5.5.1 Installer

The Install Wizard has successfully installed NetBeans IDE 5.5.1 on your computer. Choose Finish to exit the Wizard.

NetBeans IDE 5.5.1 installation location:
C:\netbeans5.5.1

To run the IDE, launch:
C:\netbeans5.5.1\bin\netbeans.exe

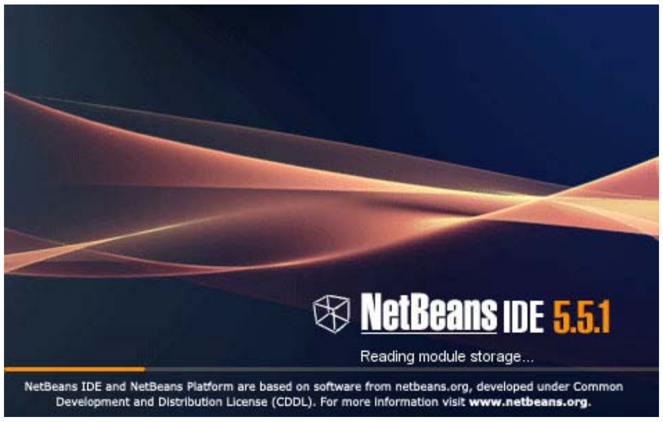
To uninstall the IDE, launch:
C:\netbeans5.5.1_uninst\uninstaller.exe

Launching the IDE:

There are three ways to start the IDE:

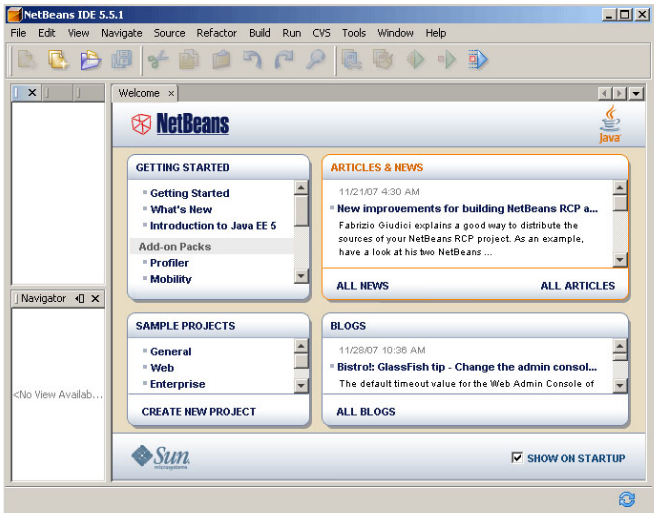
1. Double click the NetBeans IDE icon on your desktop
2. Go to Start menu->Programs->NetBeans 5.5.1 ->NetBeans IDE and click it and
3. Go to the bin folder of installation directory of NetBeans (**C:\netbeans5.5.1\bin**, in our case) and double click **netbeans.exe**.

Any one of the above starts launching the IDE as shown below:


NetBeans IDE 5.5.1
Reading module storage...
NetBeans IDE and NetBeans Platform are based on software from netbeans.org, developed under Common Development and Distribution License (CDDL). For more information visit www.netbeans.org.

NetBeans IDE

Finally, a window appears like the figure shown below.



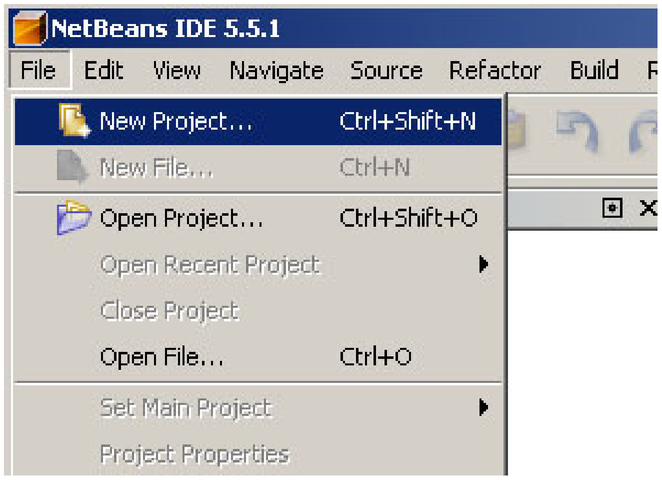
11/21/07 4:30 AM
New improvements for building NetBeans RCP a...
Fabrizio Giudici explains a good way to distribute the sources of your NetBeans RCP project. As an example, have a look at his two NetBeans ...

11/28/07 10:36 AM
Bistro!: GlassFish tip - Change the admin consol...
The default timeout value for the Web Admin Console of

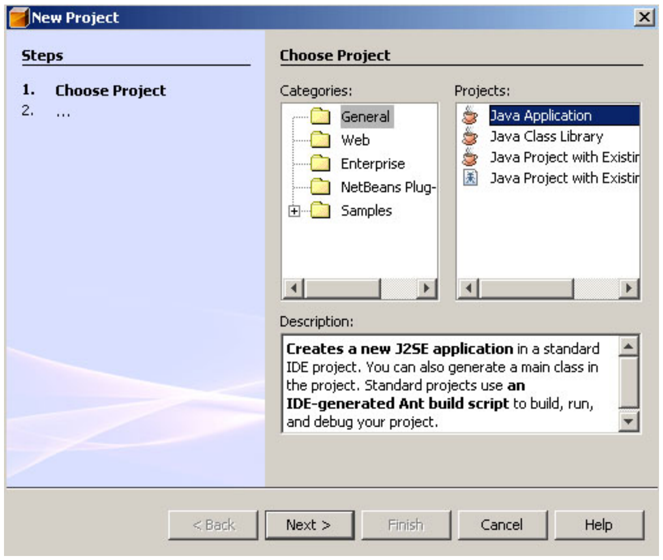
Creating the project:

1. Start the NetBeans IDE.

2. Go to File->New Project and click it.

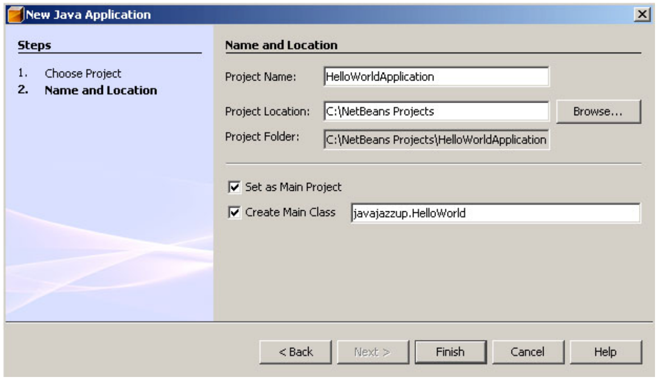


3. A new window opens for selecting the type of category and project. Select Java Application of General category as shown in the figure below and click Next button.



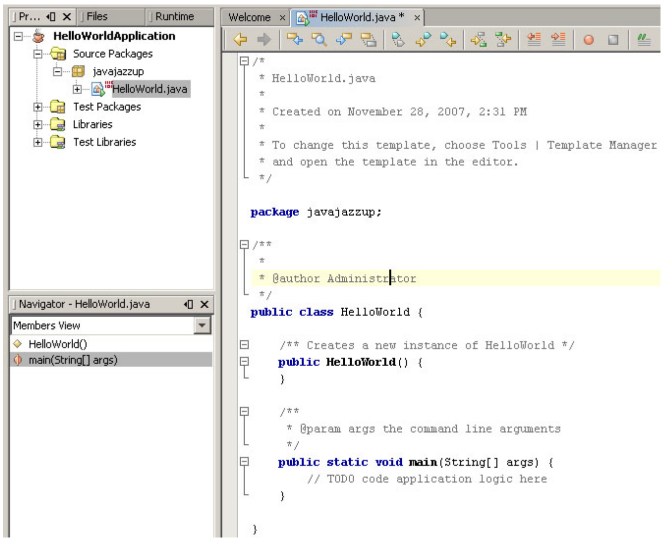
Creates a new J2SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

4. Type project name, location and main class name with the package in the appropriate place. In our case, Project Name is HelloWorldApplication, Project Location is C:\NetBeans Projects, which can be reached clicking the Browse button. Leave the Create Main Class checkbox selected. Specify the name of main class as HelloWorld in the javajazzup package. Click Finish button.



Project Name: HelloWorldApplication
Project Location: C:\NetBeans Projects
Project Folder: C:\NetBeans Projects\HelloWorldApplication
☑ Set as Main Project
☑ Create Main Class javajazzup.HelloWorld

5. Project is created and the source code of "HelloWorld.java" is opened in the IDE.



```
/*
 * HelloWorld.java
 *
 * Created on November 28, 2007, 2:31 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package javajazzup;

/**
 *
 * @author Administrator
 */
public class HelloWorld {

    /** Creates a new instance of HelloWorld */
    public HelloWorld() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

6. The above skeleton is created by the IDE according to the information provided before. Now you can add your lines of code in the appropriate place of the file. For our application, the line **"// TODO code application logic here"** has been replaced by the line **"System.out.println("Hello World");"**

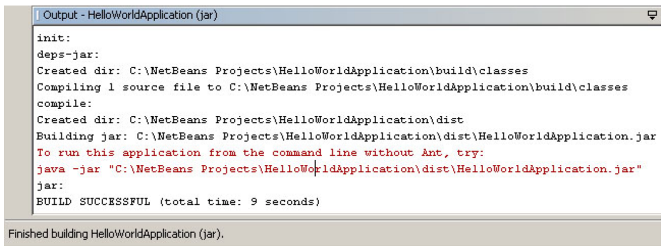
Finally, HelloWorld.java file will look like the following:

```
/*
 * HelloWorld.java
 *
 * Created on November 28, 2007, 2:31
   PM
 *
 * To change this template, choose Tools |
   Template Manager
 * and open the template in the editor.
 */

package javajazzup;

/**
 * This application displays "Hello World" to
   the standard output.
 */
public class HelloWorld {

    /** Creates a new instance of
    HelloWorld */
    public HelloWorld() {
    }

    /**
     * @param args the command line
    arguments
     */
    public static void main(String[] args) {
             System.out.println("Hello
    World");
    }

}
```

7. Save the changes by choosing File->Save or pressing keyboard keys Ctrl+S.

8. Compile the source file by selecting **"Build -> Build Main Project"** from the main menu.

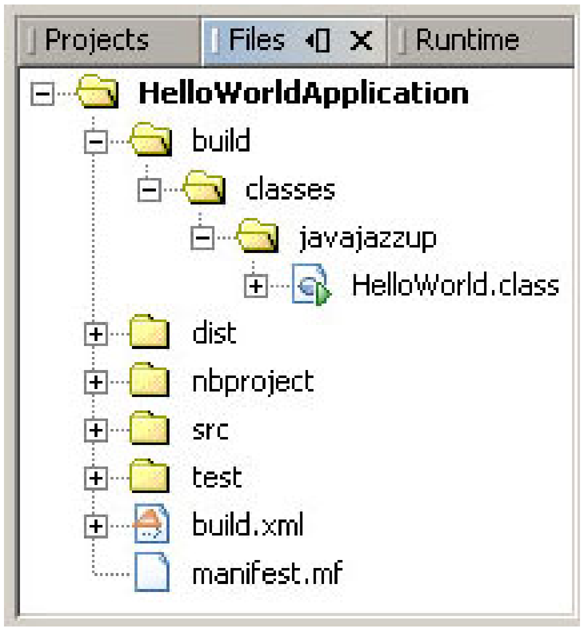
9. The result of compilation appears in the Output window as shown below:



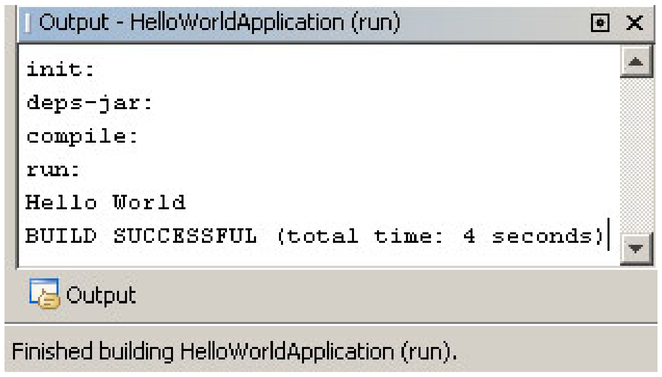
```
init:
deps-jar:
Created dir: C:\NetBeans Projects\HelloWorldApplication\build\classes
Compiling 1 source file to C:\NetBeans Projects\HelloWorldApplication\build\classes
compile:
Created dir: C:\NetBeans Projects\HelloWorldApplication\dist
Building jar: C:\NetBeans Projects\HelloWorldApplication\dist\HelloWorldApplication.jar
To run this application from the command line without Ant, try:
java -jar "C:\NetBeans Projects\HelloWorldApplication\dist\HelloWorldApplication.jar"
jar:
BUILD SUCCESSFUL (total time: 9 seconds)
```

Finished building HelloWorldApplication (jar).

If you get BUILD SUCCESSFUL statement then you have compiled the file successfully and if you get BUILD FAILED then there may be any syntactical error. Errors are shown in the Output window as hyper-linked text. Click the link to reach the place of error in the code. Remove the error from the code and compile it again. After successful compilation of the code the bytecode file HelloWorld.class is generated which can be seen by expanding the File window as shown below:



10. Now, you can run the program by choosing **Run->Run Main Project** from the main menu. The output of the program "Hello World" is shown in the Output window as shown below.



```
init:
deps-jar:
compile:
run:
Hello World
BUILD SUCCESSFUL (total time: 4 seconds)
```

Finished building HelloWorldApplication (run).

Introduction to Tomcat Server

Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Weblogic, is one of the popular application server).

Installation and Configuration of the Tomcat Server

Here we are illustrating the guidelines to install and configure the Tomcat 6.0.10, as a stand-alone web server only for Microsoft Windows. Installing Tomcat on Windows can be done easily using the Windows installer. Sometime Apache Tomcat is referred, as the Jakarta Tomcat so don't confuse between the two.

Steps involved in installation and configuration process for Tomcat 6.0.10 are illustrated below:

Step 1: Installation of JDK: Installation of Tomcat 6.0.10 requires the JVM compatible with Java 1.5 (Java 5) and Java 1.6 (Java 6) so don't forget to install the needed JDK on your system (if not installed) and to set the classpath.

Step 2: Setting the class path variable for JDK: There are two methods to set the classpath.

1. Set the class path using the following command.

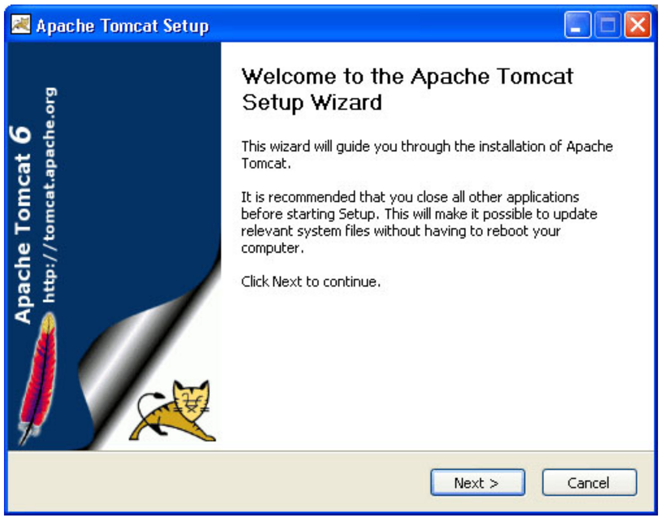
setPATH="C:\Program Files\Java\jdk1.5.0_08\bin";%PATH%

2. The other way of setting the class path variable are:

Right click on the My Computer->properties ->advance->Environment Variables-> path.

Set bin directory path of JDK in the path variable.

Step 3: Now it's time to shift on to the installation process of Tomcat 6.0.10. It takes various steps for installing and configuring the Tomcat 6.0. For Windows, Tomcat comes in two forms: **.**zip file and the Windows installer (**.**exe file). Here we are exploring the installation process by using the **.**exe file. The directory C:\apache-tomcat-6.0.10 is the common installation directory as it is pre-specified C:\ as the top-level directory. First unpack the zipped file and simply execute the **.**exe file.



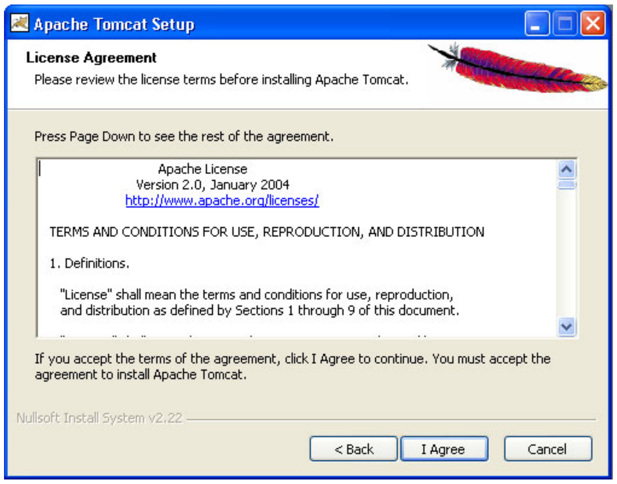
Welcome to the Apache Tomcat Setup Wizard

This wizard will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

The above shown screen shot is the first one shown in the installation process. Just click on the **Next** button to continue the installation process.



License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

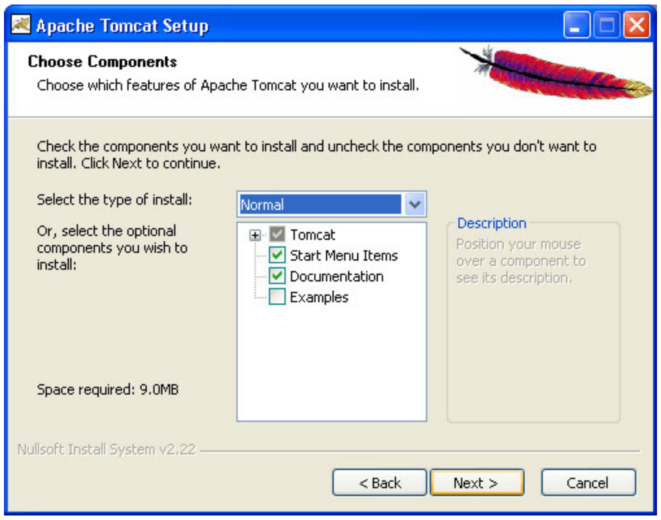
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Tomcat Server

Click "**I Agree**" button to continue the installation process.


Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Normal

Or, select the optional components you wish to install:

- [✓] Tomcat
- [✓] Start Menu Items
- [✓] Documentation
- [ ] Examples

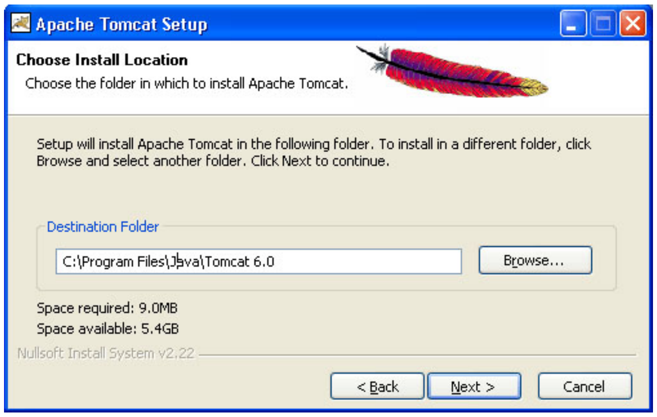
Description

Position your mouse over a component to see its description.

Space required: 9.0MB

Nullsoft Install System v2.22

Click **Next** to go with the default components chosen.


Apache Tomcat Setup

Choose Install Location

Choose the folder in which to install Apache Tomcat.

Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

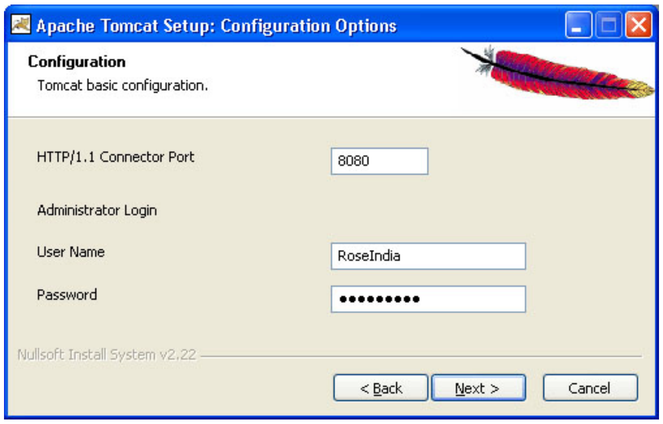
Destination Folder: C:\Program Files\Java\Tomcat 6.0

Space required: 9.0MB

Space available: 5.4GB

Nullsoft Install System v2.22

Choose the location for the Tomcat files as per your convenience. You can also choose the default location.


Apache Tomcat Setup: Configuration Options

Configuration

Tomcat basic configuration.

HTTP/1.1 Connector Port: 8080

Administrator Login

User Name: RoseIndia

Password: ••••••••••

Nullsoft Install System v2.22

Now choose the port number on which you want to run the tomcat server. Tomcat uses the port number **8080** as its default value. But Most of the people change the port number to 80 because in this case the user is not required to specify the port number at request time. But we are using here the default port number as 8080. Choose the user name and password as per your convenience. We can change the port number even the installation process is over. For that, go to the specified location as " Tomcat 6.0 \conf \server.xml ". Within the server.xml file choose "Connector" tag and change the port number.

e.g While using the port number 8080, give the following request in the address bar as:

Default Port: http//localhost:8080/index.jsp

In case of port number number 80 just type the string illustrated below in the address bar:

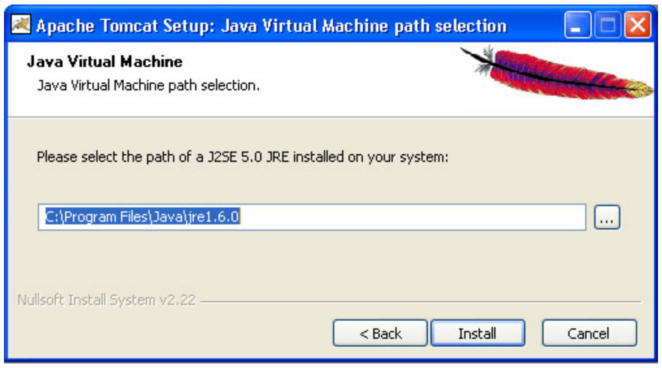
New Port: http://localhost/index.jsp

Note that we do no need to specify any port number in the URL.

Now click on the **Next** button to continue the installation process.

The installation process shows the above screen as the next window. This window asks for the location of the installed Java Virtual Machine. Browse the location of the JRE folder and click on the Install button. This will install the Apache tomcat at the specified location.

Tomcat Server


Apache Tomcat Setup: Java Virtual Machine path selection

Java Virtual Machine

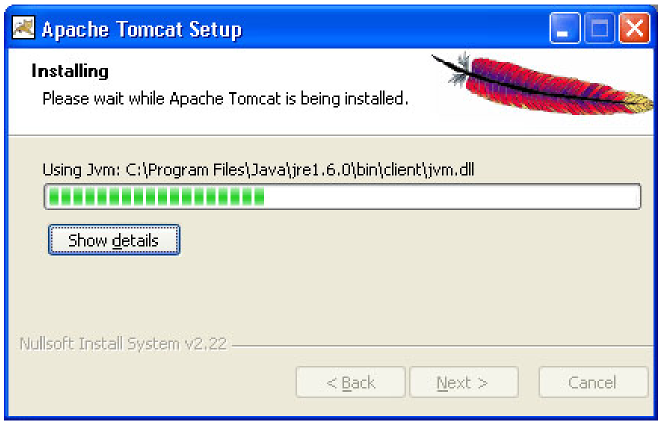
Java Virtual Machine path selection.

Please select the path of a J2SE 5.0 JRE installed on your system:

C:\Program Files\Java\jre1.6.0

Nullsoft Install System v2.22

To get the information about installer click on the "Show details" button.


Apache Tomcat Setup

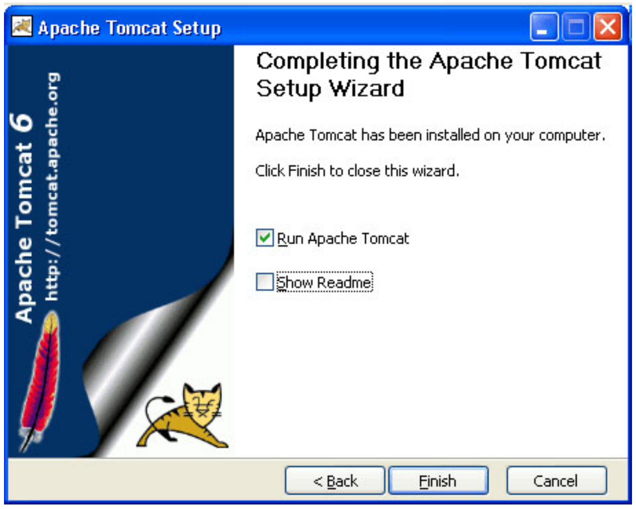
Installing

Please wait while Apache Tomcat is being installed.

Using Jvm: C:\Program Files\Java\jre1.6.0\bin\client\jvm.dll

Nullsoft Install System v2.22

After completion of installation process it will display the window like the above one.


Apache Tomcat Setup

Completing the Apache Tomcat Setup Wizard

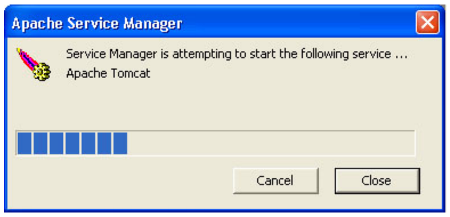
Apache Tomcat has been installed on your computer.

Click Finish to close this wizard.

[✓] Run Apache Tomcat

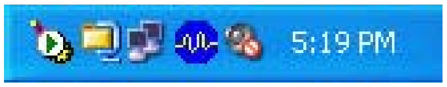
[ ] Show Readme

On clicking at **Finish** button, a window like the above one will display a message printed on the window given below.

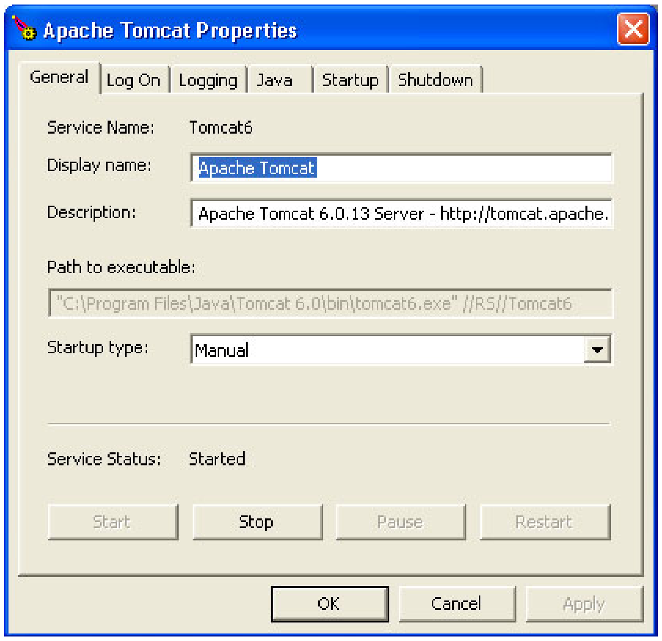

Apache Service Manager

Service Manager is attempting to start the following service ...
Apache Tomcat

After successfully installing, a shortcut icon to start the tomcat server appears in the icon tray of the task bar as shown above. Double clicking the icon, displays the window of Apache Manager for Tomcat. It will show the "Startup type" as manual since we have changed the destination folder for tomcat during the installation process. Now we can configure the other options like "Display name" and "Description" .We can also start, stop and restart the service from here.



If installation process completes successfully then a window as shown below will appear.


Apache Tomcat Properties

Service Name: Tomcat6

Display name: Apache Tomcat

Description: Apache Tomcat 6.0.13 Server - http://tomcat.apache.

Path to executable:

"C:\Program Files\Java\Tomcat 6.0\bin\tomcat6.exe" //RS//Tomcat6

Startup type: Manual

Service Status: Started

Now, set the environment variable for tomcat:
Step 4: Setting the JAVA_HOME Variable: Purpose of setting the environment variable JAVA_HOME is to specify the location of the java run time environment needed to support the Tomcat else Tomcat server does not run. This variable contains the path of JDK installation directory. Note that it should not contain the path up to bin folder.

Set JAVA_HOME=C: \Program Files\Java\jdk1.5.0_08

Here, we have taken the URI path according to our installation convention

For Windows XP, Go through the following steps:

Start menu->Control Panel->System->Advanced tab->Environment Variables->New set the Variable Name as **JAVA_HOME** and Variable Value as **C:\Program Files\Java\jdk1.6.0** and then click on all the three **Ok** buttons one by one. It will set the JDK path.

For Windows 2000 and NT, follow these steps:

Start->Settings->Control Panel->System->Environment Variable->New->set the Variable Name as JAVA_HOME and Variable Value as **C:\Program Files\Java\jdk1.6.0** and then click on all the three ok button one by one. It will set the JDK path.

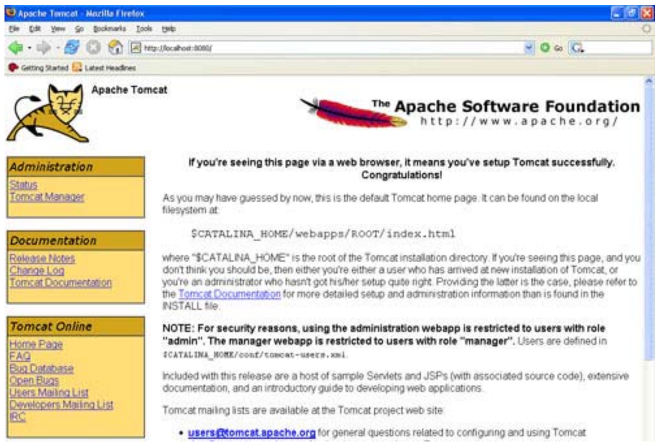
Now, **Start the Tomcat Server:** Start the tomcat server from the bin folder of Tomcat 6.0 directory by double clicking the "tomcat6.exe " file. You can also create a shortcut of this .exe file at your desktop.

Stop the Tomcat Server: Stop the server by pressing the "**Ctrl + c**" keys.

Directory Structure of Tomcat Server

In the previous section, you learned the installation of Tomcat version 6.0.10. Now, lets quickly understand the directory structure of Tomcat 6.0 that are automatically built up after installation.

Here is the directory structure of Tomcat 6.0 shown as:



If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

As you may have guessed by now, this is the default Tomcat home page. It can be found on the local filesystem at:

```
$CATALINA_HOME/webapps/ROOT/index.html
```

where "$CATALINA_HOME" is the root of the Tomcat installation directory. If you're seeing this page, and you don't think you should be, then either you're either a user who has arrived at new installation of Tomcat, or you're an administrator who hasn't got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the administration webapp is restricted to users with role "admin". The manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Tomcat mailing lists are available at the Tomcat project web site:

- users@tomcat.apache.org for general questions related to configuring and using Tomcat

bin:

The bin directory includes all batch files that are essential to startup and shutdown the Tomcat server.

conf:

In this directory, all configuration files are included that are essential to configure the server. These files are context.xml, logging properties file, sever.xml, etc.

lib:

This directory includes all java libraries in JAR format that Tomcat uses. These files are essential to run a web application (like a servlet, jsp, etc.). Before running any type of web application, the path must be set related to that application from the lib directory.

logs:

The log directory contains log files according to the date in which, the information of a web application is put up each time when the application is being run. This information includes user-log (such as admin, host-manager, local host, etc.), servlet-context information, etc.

temp:

This is a directory used by the **AutoDeployer** to store files temporarily.

webapps:

This directory may be considered as a main directory; in which, all deployed web applications are put up including necessary files such as class file, xml file, html file, jar file, etc. These files are put up restrictedly according to the deployment directory structure.

work:

This folder is automatically generated by Tomcat, where Tomcat places intermediate files (such as compiled JSP files) during its work. You will not be able to execute JSP pages if you delete this directory while Tomcat is running.

Running a Servlet on the Tomcat Server

Now, lets see how a web application is run on the Tomcat server taking an example of servlet.

Once you have installed the Tomcat then do the following steps:

1 Set all three paths (such as JDK path, classpath, and Java_Home path) as:

```
set path =%path% C:\Program
Files\Java\jdk1.6.0\bin;
set classpath =%CLASSPATH%
C:\Tomcat6.0\lib\ servlet-api.jar;
set JAVA_HOME=C:\Program
Files\Java\jdk1.6.0;
```

2 Create a servlet class as **HelloWorld.java** and compile it to make a class file of it.

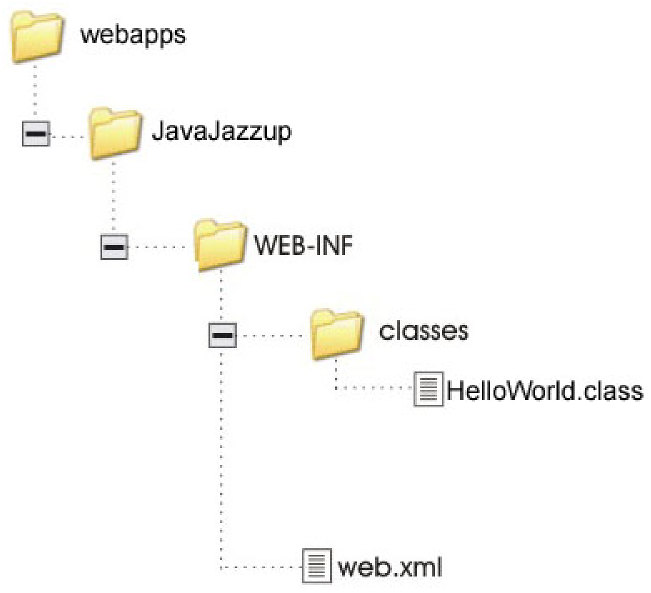
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet{
public void doGet(HttpServletRequest
request, HttpServletResponse response)
throws ServletException,IOException{
    response.setContentType("text/html");
    PrintWriter pw = response.getWriter();
   pw.println("<html>");
   pw.println("<head><title>Hello World
    </title></title>");
   pw.println("<body>");
   pw.println("<h1>Hello World</h1>");
   pw.println("</body></html>");
  }
}
```

3 Create an xml descriptor file as **web.xml**.

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<!—<!DOCTYPE web-app
 PUBLIC "-//Sun Microsystems, Inc.//DTD
Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-
app_2_3.dtd"> —>

<web-app>
<servlet>
<servlet-name>Hello</servlet-name>
<servlet-class>HelloWorld</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>Hello</servlet-name>
<url-pattern>/HelloWorld</url-pattern>
</servlet-mapping>
</web-app>
```

4 Make a Web-application directory structure under the webapps directory shown as

Tomcat Server



A web application is a structured hierarchy of directories. The folder **webapps** is the home directory of Tomcat web server. This is the main folder for the **"public"** section. The MI framework and applications will be stored in the **JavaJazzup** folder and its subfolders.

A special directory exists within this application hierarchy called **WEB-INF**. The WEB-INF folder is the main folder of the "**private**" section of a Web application.

5. Take your compiled servlet class file as "**HelloWorld.class**" and place them in the **WEB-INF/classes** directory. If you have defined your servlet to belong in a package, you must follow the standard Java rules and create the appropriate subdirectories so the JVM will be able to find your classes.

6. Put the created xml file as "**web.xml**" file in the **WEB-INF** directory. This XML file is the deployment descriptor for servlets and other components that make up the application.

7. To test your servlet, start the Tomcat server clicking the **startup.bat** file from the bin directory.

8. Once the server is start, open a Web browser, and open a URL of the following form:

http://{address}:{port}/{servletName}

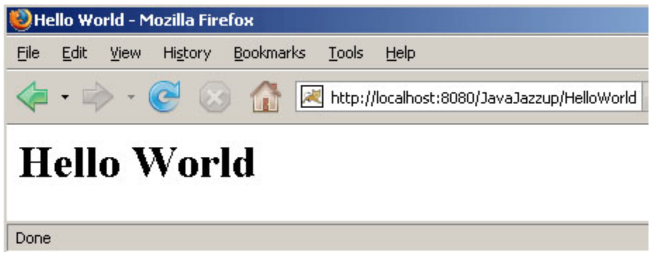
where:

- **address** is the name or IP address of the machine running Tomcat. You can use localhost if the browser is running on the same machine as Tomcat.
- **port** is the port on which Tomcat is listening. By default, that is port 8080.
- **servletName** is the name of the servlet you want to invoke. That should match the value contained in the <url-pattern></url-pattern> tags in the **web.xml** deployment descriptor file.

For example, if Tomcat is running on the same machine as the browser and listening on the default port (8080), you can test **HelloWorld** servlet by opening the following URL:

http://localhost:8080/JavaJazzup/HelloWorld

The output of the program is given below:


Hello World - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/JavaJazzup/HelloWorld
Done

Hello World

Introduction to MySQL

MySQL is an open source Relational Database Management System based on the Structured Query Language (SQL). It is very fast reliable and flexible Database Management System based on relation model that is developed to manage large volumes of data at very high speed with security. MySQL can be used for verity of applications but it is one of the most popular RDBMS used for the web applications on the Internet.

It is referred as open source because it can be run on different platform such as Unix, Linux, Windows, OS/2, and etc. It is possible for anyone to use and modify the software. If you wish, you may study the source code and change it to suit your needs.

MySQL is based on a client/server model; its database package includes MySQL server and MySQL client program. It supports all functionalities of a DBMS, such as standard data types; multi-line commands and ensures that transactions are complete within the ACID rules.

MySQL Features

- MySQL are very fast and much reliable for any type of application.
- It is very Lightweight Database application.
- Its command line tool is very powerful and can be used to run SQL queries against database.
- It Supports indexing and binary objects.
- It allows DBA to change the structure of table while server is running.
- It is a very fast multithreaded-based memory allocation system.
- MySQL is available as a separate program for use in a client/server network environment.
- The MySQL available for the most of all operating system platforms.
- Programming libraries for C, Python, PHP, Java, Delphi etc. are available to connect to MySQL database.

MySQL 5 Features:

In traditional version of MySQL, there are some drawbacks that, it doesn't support Procedures and Functions as well as Triggers. Now these features are included in the new version of MySQL that is **MySQL 5.0**. Its features are:

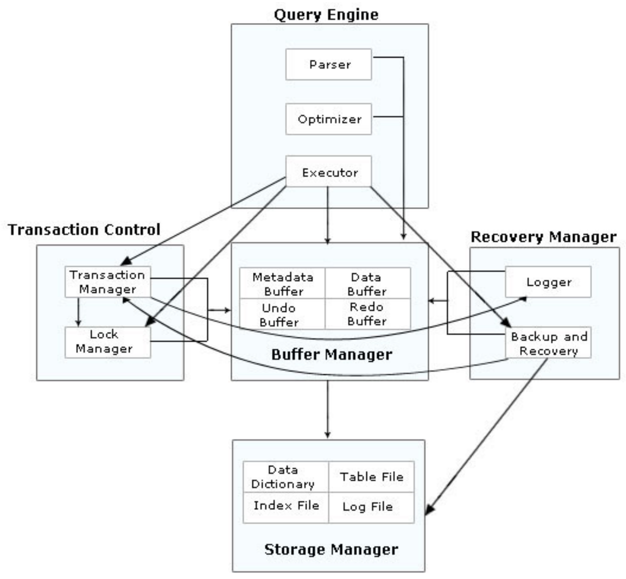
1. Views
Views is a virtual table, which acts as a table, but it contains no data. Views are created using columns from one or more tables.

2. Stored Procedures and Functions
MySQL 5.0 now support Stored Procedures and Functions. This allows you to embed business logic at database level.

3. Triggers
The Triggers is another very imported feature available with MySQL 5.0. Now we can add some business logic whenever data is inserted, Deleted or updated in the table.

High Level Conceptual Architecture of MySQL

At the highest level of abstraction MySQL consists of five major sub-systems.



Query Engine

The *Query Engine* receives all the SQL commands from the user. Its sub-system consists of a *Parser*, *Optimizer* and *Executor*. The **parser** checks the syntax of SQL commands, checks the existence of requested tables and columns, and checks user privileges to perform the operation.

An **optimizer** checks the existence of indexes to satisfy the query chooses among several alternate ways to execute the command and prepares an best possible plan of execution.
The **executor** takes the actions required to execute the plan or requests.

Buffer Manager

The *Buffer Manager* lies in between the query engine and storage manager. This sub-system is responsible for memory management. It keeps the active part of the data from the data files in main memory and performs the necessary replacement of data to and from the storage manager, and supplies the requested data to the query engine. Its sub-system consists of a *data buffer* for table and index file, a *metadata buffer* for data dictionary files and *redo buffer* for the log files. In addition the buffer manager also contains one more component the *undo buffer* that keeps copies of data currently being modified by active transactions.

Storage Manager

The *Storage Manager* is the backend of the system. It manages different categories of data files in the file system. This system deals with efficient storage and retrieval of data to and from the OS file systems. Its sub-system consists of the *data files* containing the relational tables, *data dictionary* that is also known as metadata, contains list of all the tables and indexes, privileges of the users in the data, business rules associated with the data etc., and *log files* generated by the recovery manager.

Transaction Control

The purpose of *Transaction Control* is to provide transactions and concurrency. Transaction allows the users to manipulate data atomically and concurrency where multiple concurrent users can access the data in a consistent way. The transaction control sub-system consists of *Transaction Manager* and *Lock Manager* components. The transaction manager takes care of atomic manipulation of data by temporarily storing copies of data. The lock manager sets locks in tables or records of a table.

Recovery Manager

The *Recovery Manager* is responsible for keeping copies of data and record in which changes has been done. So that in case of primary data files crash, the database can be restored in a consistent state as just before the crash occurred.

Its sub-system consists of the *Logger* that records any modifications done in the database in log files, and *Backup & Recovery* through which a DBA can save copies of the data files that can be used later in concurrence with log files to restore the database in a consistent state.

Introduction to Basic SQL statements

As we have discussed that, most commercial RDBMS's use the **Structured Query Language (SQL)** to access the database.
SQL (Structured Query Language) is a standard language for making interactive queries from and updating a database such as IBM's DB2, Microsoft's Access, and database products from Oracle, Sybase, and Computer Associates.

Queries are the backbone of SQL. It is a term that refers to a widely available set of SQL commands called **clauses**. Each clause (command) performs some sort of function against the database.

For instance, the create clause creates tables and databases and the select clause selects rows that have been inserted into your tables.

SQL statements are divided into two major categories:

Data Definition Language (DDL)
Data Manipulation Language (DML)

Data Definition Language (DDL):
DDL statements are used to define and modify the database structure of your tables or schema. When you execute a DDL statement, it takes effect immediately.
Some commands of DDL are:

- **CREATE** - to create table (objects) in the database
- **ALTER** - alters the structure of the database
- **DROP** - delete table from the database
- **TRUNCATE** - remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT** - add comments to the data dictionary
- **RENAME** - rename a table

1. The create table statement (query) to create a table is given below:

```
CREATE TABLE <table name> (
<attribute name 1> <data type 1>,
...
<attribute name n> <data type n>);
```

Example:

```
CREATE TABLE STUDENT ( StudID NUMBER,
Name VARCHAR);
```

The data types that you will use most frequently are character strings, which might be called VARCHAR or CHAR for variable or fixed length strings; numeric types such as NUMBER or INTEGER, which will usually specify a precision; and DATE or related types. Data types are differ according to the databases software whatever you are using to your system.

2. The alter table statement to make modifications to the table structure such as Key constraints, Column size, etc.

ALTER TABLE **<table name>**
ADD CONSTRAINT <constraint name>
PRIMARY KEY (<attribute list>);

Example:

```
ALTER TABLE STUDENT
ADD CONSTRAINT NOT NULL PRIMARY KEY
(StudID);
```

3. The delete table statement (query) to delete a table is given below:

```
DROP TABLE <table name>;
```

Example:

```
DROP TABLE STUDENT;
```

Data Manipulation Language

Data Manipulation Language (DML) statements are used for managing data within tables. Some commands of DML are:

- **SELECT -** retrieve data from the a database
- **INSERT -** insert data into a table
- **UPDATE -** updates existing data within a table
- **DELETE -** deletes all records from a table, the space for the records remain
- **MERGE -** UPSERT operation (insert or update)
- **CALL -** call a PL/SQL or Java subprogram
- **LOCK TABLE -** control concurrency

1. The insert statement is used to add new row to a table.

```
INSERT INTO <table name>
VALUES (<value 1>, ... <value n>);
```

Example:

INSERT INTO STUDENT VALUES (1001, 'Ram');

The inserted values must match the table structure exactly in the number of attributes and the data type of each attribute. Character type values are always enclosed in single quotes; number values are never in quotes; date values are often (but not always) in the format 'yyyy-mm-dd' (for example, '2006-11-30').

2. The update statement is used to change values that are already in a table.

```
UPDATE <table name>
SET <attribute> = <expression>
WHERE <condition>;
```

Example:

```
UPDATE STUDENT SET Name = 'Amar'
WHERE StudID=1001;
```

The update expression can be a constant, any computed value, or even the result of a SELECT statement that returns a single row and a single column.

3. The delete statement deletes row(s) from a table.

```
DELETE FROM <table name>
WHERE <condition>;
Example:
DELETE FROM STUDENT WHERE
StudID=1001;
```

If the WHERE clause is omitted, then every row of the table is deleted that matches with the specified condition.

4. The SELECT statement is used to form queries for extracting information out of the database.

```
SELECT <attribute>, …..,
<attribute n> FROM <table name>;
```

Example:

```
SELECT StudID, Name FROM STUDENT;
```

Apart from these statements, some statements are also used to control the transaction made by DML statements. The commands used for this purpose are called Transaction Control (TCL) statements. It allows statements to be grouped together into logical transactions. Some commands of TCL are:

- **COMMIT -** save work done.
- **SAVEPOINT -** identify a point in a transaction to which you can later roll back.
- **ROLLBACK -** restore database to original since the last COMMIT.

In this lesson you will read about the configuration of MySQL. The MySQL server configuration normally started during installation process.

MySQL server configuration wizard window

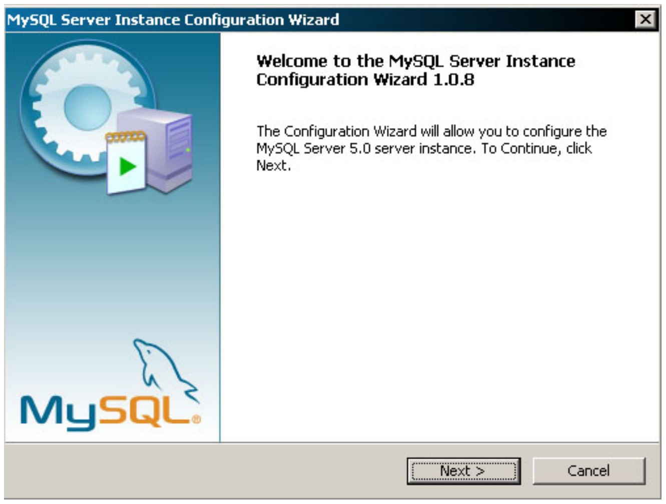
To start the configuration wizard of MySQL, click the MySQL server instance configuration wizard entry that is available in the MySQL Server 5.0 section of the window's start menu. Another way to start the configuration wizard, open the **MySQLInstanceConfig.exe** file directly from bin directory of your MySQL installation. The MySQL server configuration wizard sets configuration variables values in **my.ini** file in the installation directory for the MySQL server.

```
—defaults-file="C:\Program
Files\MySQL\MySQL Server 5.0\my.ini"
```

Path "C:\Program Files\MySQL\MySQL Server 5.0" is installation directory of MySQL server.

The —defaults-file option instructs the MySQL server to read the specified file for configuration options when it starts. The MySQL client and utilities like mysql and mysqldump command-line client are unable to locate the my.ini file locate in the server installation directory. MySQL server configuration wizard will configure MySQL to work as windows services.

1. Once you have opened the configuration wizard, a "Welcome to MySQL Configuration Wizard" dialog will appear. Click the Next button to continue.

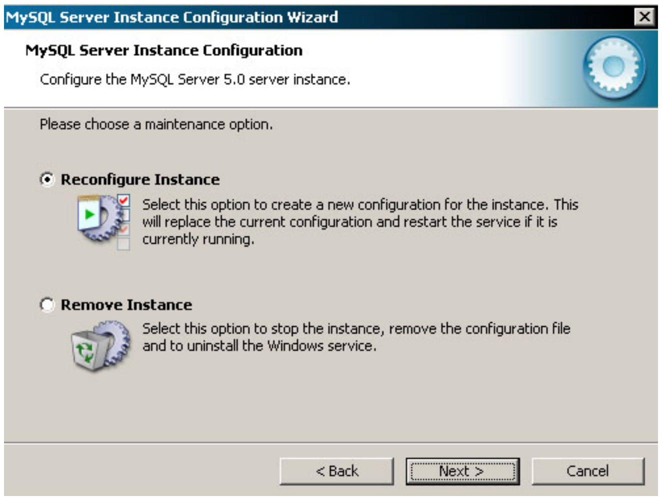

MySQL Server Instance Configuration Wizard

Welcome to the MySQL Server Instance Configuration Wizard 1.0.8

The Configuration Wizard will allow you to configure the MySQL Server 5.0 server instance. To Continue, click Next.

2. In second step, a Maintenance Option dialog appears where the MySQL Server Configuration Wizard detects an existing configuration file.

3. The Remove Instance option button is selected if you want to remove the existing server instance. To reconfigure an existing server, choose the Re-configure Instance option and click the Next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please choose a maintenance option.

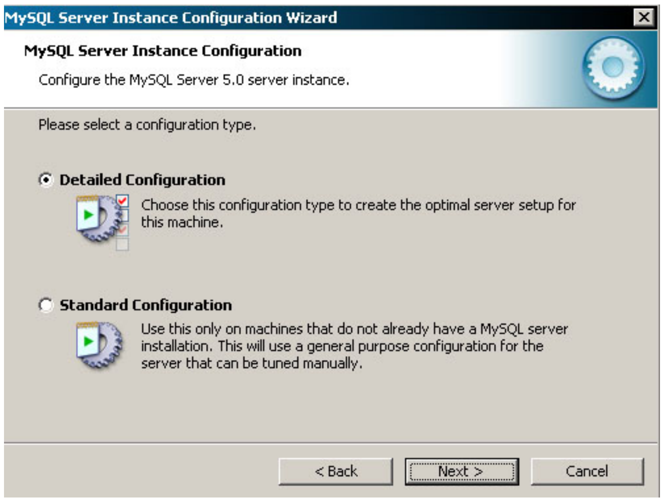
[✓ Reconfigure Instance]

Select this option to create a new configuration for the instance. This will replace the current configuration and restart the service if it is currently running.

Remove Instance

Select this option to stop the instance, remove the configuration file and to uninstall the Windows service.

4. After clicking Next button, you come up to the Configuration Type dialog. There are two configuration types available: **Detailed Configuration** and **Standard Configuration**. The Standard Configuration option is intended for new users who don't already have MySQL server installation. If you have an existing MySQL installation on your system, choose the Detailed Configuration option to configure and click the Next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please select a configuration type.

[✓ Detailed Configuration]

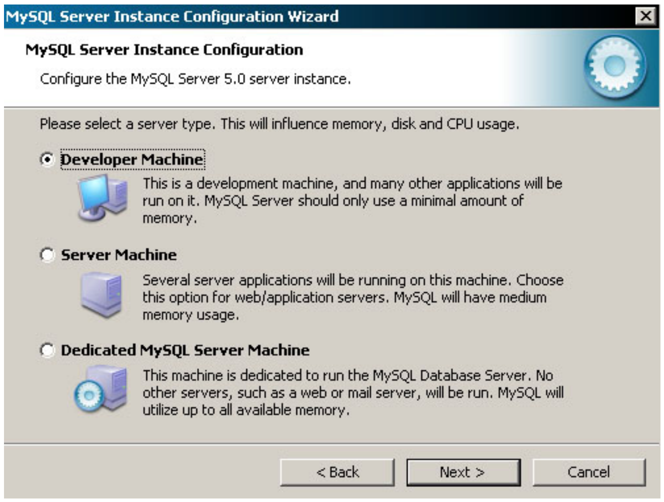
Choose this configuration type to create the optimal server setup for this machine.

Standard Configuration

Use this only on machines that do not already have a MySQL server installation. This will use a general purpose configuration for the server that can be tuned manually.

5. After clicking Next button, you come up to the Server Type Dialog. The server type that you choose affects the decisions of the MySQL Server Configuration Wizard that are taken with regard to memory, disk, and processor usage.

There are three different server types available to choose from. Select Developer Machine option for personal use and click the Next button.


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

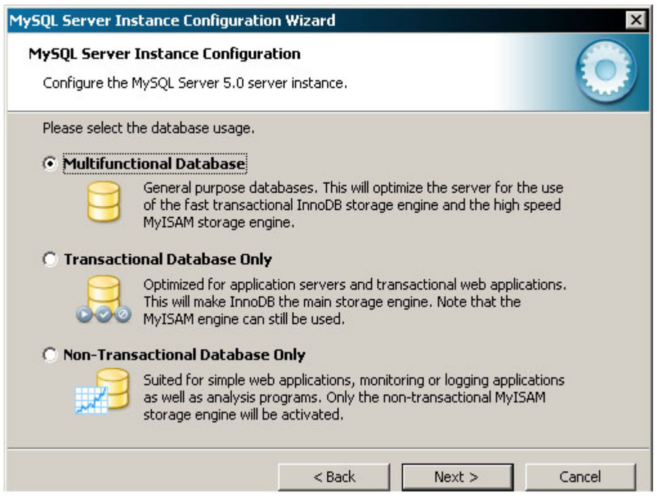
Please select a server type. This will influence memory, disk and CPU usage.

(●) Developer Machine
This is a development machine, and many other applications will be run on it. MySQL Server should only use a minimal amount of memory.

( ) Server Machine
Several server applications will be running on this machine. Choose this option for web/application servers. MySQL will have medium memory usage.

( ) Dedicated MySQL Server Machine
This machine is dedicated to run the MySQL Database Server. No other servers, such as a web or mail server, will be run. MySQL will utilize up to all available memory.

6. In this step, you come up to the Database Usage Dialog. It allows you to indicate the storage engines that you expect to use when creating MySQL tables. The option you choose determines whether the InnoDB storage engine is available and what percentages of the server resources are available to InnoDB. **Choose** Multifunctional Database **option and click the** Next **button.**


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

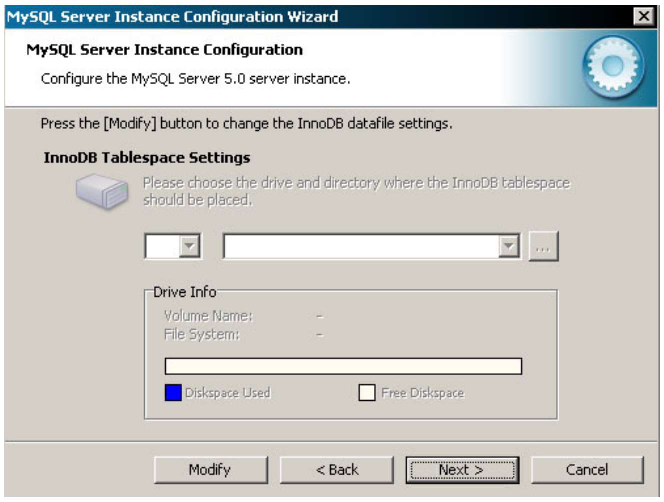
Please select the database usage.

(●) Multifunctional Database
General purpose databases. This will optimize the server for the use of the fast transactional InnoDB storage engine and the high speed MyISAM storage engine.

( ) Transactional Database Only
Optimized for application servers and transactional web applications. This will make InnoDB the main storage engine. Note that the MyISAM engine can still be used.

( ) Non-Transactional Database Only
Suited for simple web applications, monitoring or logging applications as well as analysis programs. Only the non-transactional MyISAM storage engine will be activated.

7. After clicking Next button, the InnoDB Tablespace Dialog appears. Select the drive where the database files will be stored. To change the default location for the InnoDB tablespace files, choose a new drive from the drop-down list of drive letters and choose a new path from the drop-down list of paths. **To create a custom path, click the ... button. After setting the path, click the** Next **button to continue.**


MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Press the [Modify] button to change the InnoDB datafile settings.

InnoDB Tablespace Settings

Please choose the drive and directory where the InnoDB tablespace should be placed.

Drive Info

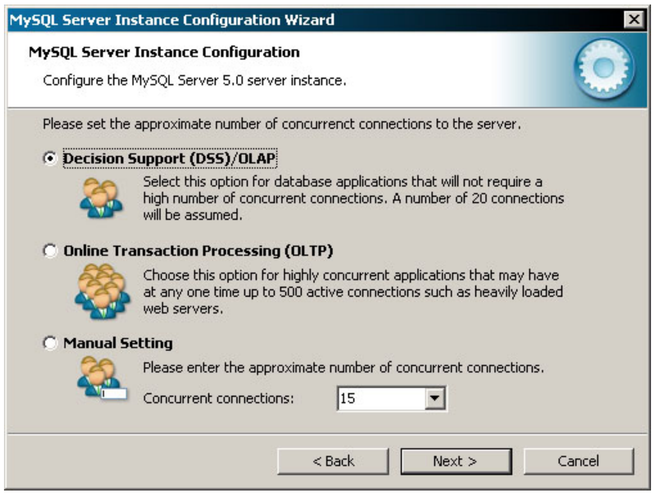
Volume Name: -

File System: -

■ Diskspace Used   □ Free Diskspace

8. **In this step, the** Concurrent Connections dialog **will appears. It allows you to choose the expected usage of your server, and sets the limit for concurrent connections accordingly. It is also possible to set the concurrent connection**

limit manually. Select Decision Support (DSS)/OLAP **option and click** Next **button.**



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please set the approximate number of concurrenct connections to the server.

[✓ Decision Support (DSS)/OLAP]

Select this option for database applications that will not require a high number of concurrent connections. A number of 20 connections will be assumed.

Online Transaction Processing (OLTP)

Choose this option for highly concurrent applications that may have at any one time up to 500 active connections such as heavily loaded web servers.

Manual Setting

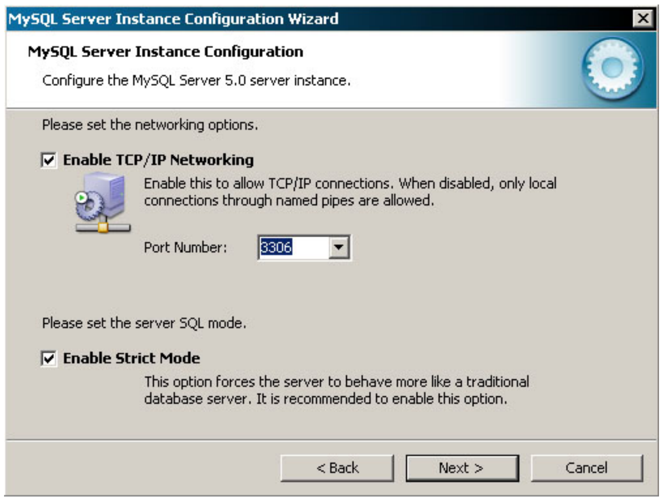
Please enter the approximate number of concurrent connections.

Concurrent connections: 15

< Back | Next > | Cancel

9. In this step, the Networking Options dialog will appears that allows to enable or disable TCP/IP networking and to configure the port number that is used to connect to the MySQL server.

TCP/IP networking is enabled by default. To disable TCP/IP networking, uncheck the box next to the Enable TCP/IP Networking option. Port 3306 is used by default. To change the port used to access MySQL, choose a new port number from the drop-down box or type a new port number directly into the drop-down box. At last click the **Next** button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please set the networking options.

[✓ Enable TCP/IP Networking]

Enable this to allow TCP/IP connections. When disabled, only local connections through named pipes are allowed.

Port Number: 3306

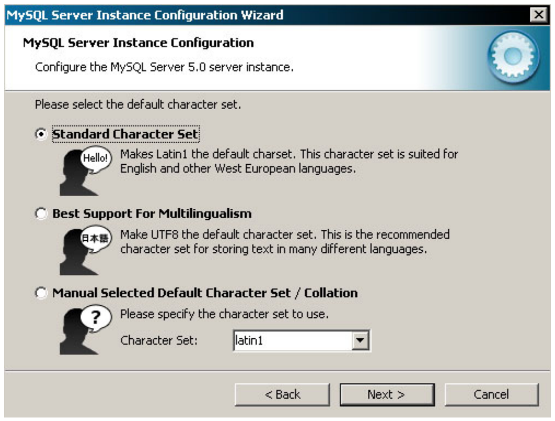
Please set the server SQL mode.

[✓ Enable Strict Mode]

This option forces the server to behave more like a traditional database server. It is recommended to enable this option.

< Back | Next > | Cancel

10. After clicking Next button, The Character Set Dialog will appears. Choose Standard Character Set option if you want to use latin1 as the default server character set. latin1 is used for English and many Western European languages. **Click the** Next **button.**



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Configure the MySQL Server 5.0 server instance.

Please select the default character set.

[✓ Standard Character Set]

Makes Latin1 the default charset. This character set is suited for English and other West European languages.

Best Support For Multilingualism

Make UTF8 the default character set. This is the recommended character set for storing text in many different languages.

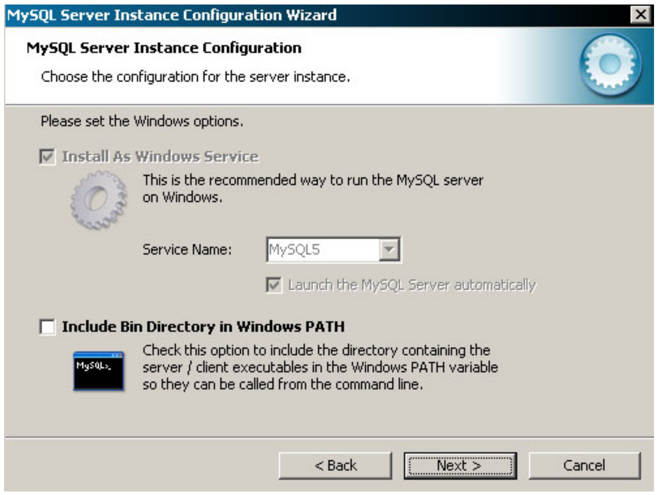
Manual Selected Default Character Set / Collation

Please specify the character set to use.

Character Set: latin1

< Back | Next > | Cancel

11. After that you come up to the Service Options Dialog. The MySQL Server Configuration Wizard installs the MySQL server as a service by default, using the service name MySQL. If you do not wish to install the service, uncheck the box next to the Install As Windows Service option. You can change the service name by picking a new service name from the drop-down box provided or by entering a new service name into the drop-down box and click the Next button.



MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration

Choose the configuration for the server instance.

Please set the Windows options.

[✓ Install As Windows Service]

This is the recommended way to run the MySQL server on Windows.

Service Name: MySQL5

[✓ Launch the MySQL Server automatically]

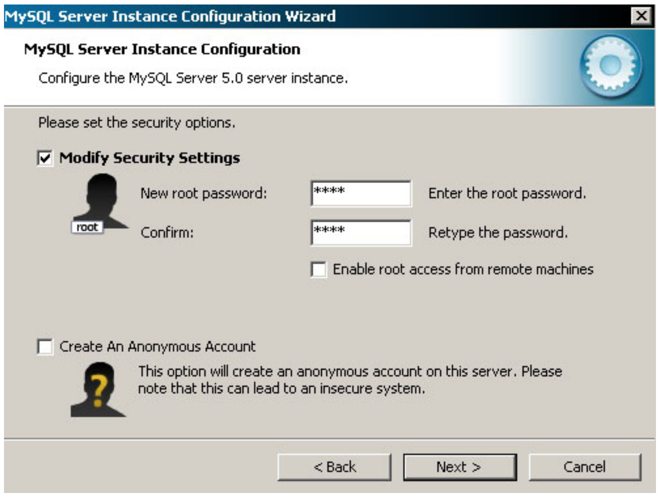
Include Bin Directory in Windows PATH

Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from the command line.

< Back | Next > | Cancel

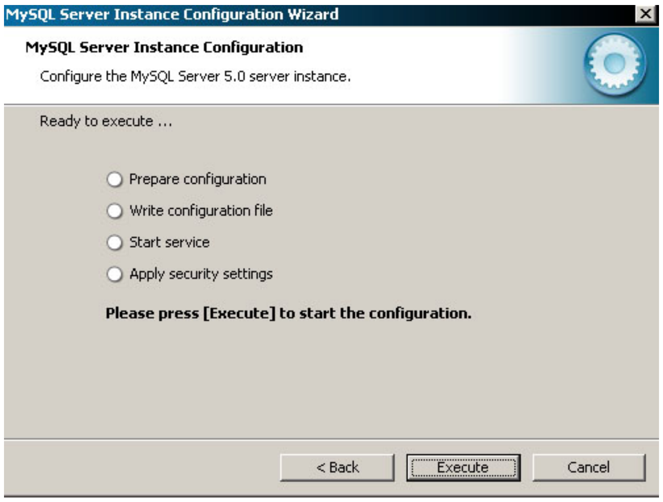
12. After that you come up to the **Security Options Dialog. If you are reconfiguring an existing server then it will ask you to the** Current root password**. Otherwise set**

the root password, enter the desired password into both the New root password **and** Confirm **boxes.**
To prevent root logins from across the network, check the box next to the Root may only connect from localhost **option. This increases the security of your root account. Click the** Next **button.**



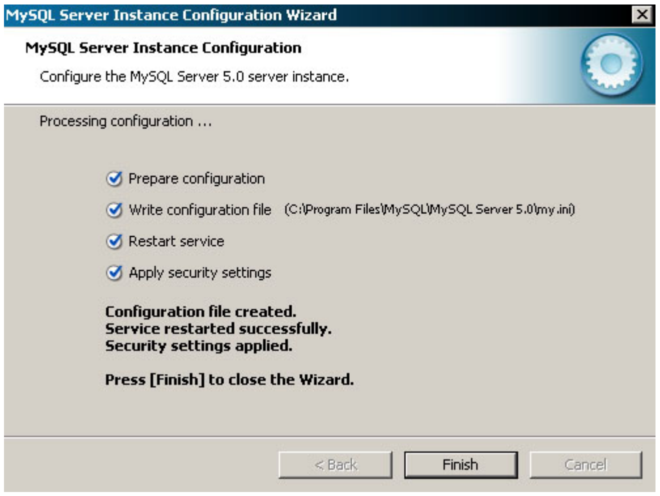
MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Please set the security options.
☑ Modify Security Settings
New root password: **** Enter the root password.
Confirm: **** Retype the password.
☐ Enable root access from remote machines
☐ Create An Anonymous Account
This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

13. The final dialog in the MySQL Server Configuration Wizard is the Confirmation Dialog**. To start the configuration process, click the** Execute **button.**



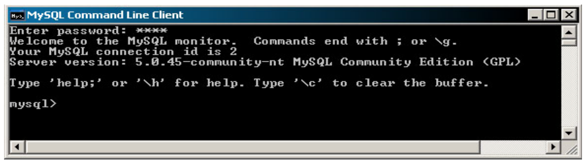
MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Ready to execute ...
○ Prepare configuration
○ Write configuration file
○ Start service
○ Apply security settings
Please press [Execute] to start the configuration.

14. After you click the **Execute** button, the MySQL Server Configuration Wizard performs a series of tasks and displays the progress onscreen as the tasks are performed. If you are reconfiguring a new service, the MySQL Server Configuration Wizard creates and starts the service. If you are reconfiguring an existing service, the MySQL Server Configuration Wizard restarts the service to apply your configuration changes.
After the MySQL Server Configuration Wizard has completed its tasks, it displays a summary. Click the Finish button to exit the MySQL Server Configuration Wizard.

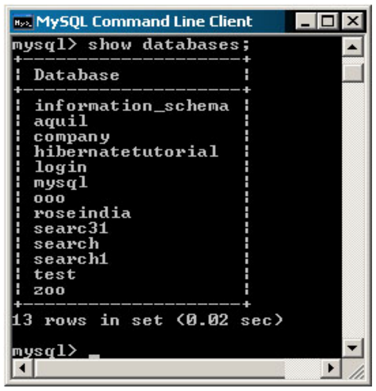


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Processing configuration ...
✓ Prepare configuration
✓ Write configuration file (C:\Program Files\MySQL\MySQL Server 5.0\my.ini)
✓ Restart service
✓ Apply security settings
Configuration file created.
Service restarted successfully.
Security settings applied.
Press [Finish] to close the Wizard.

15. The last step is to confirm on the MySQL Command Line Client prompt that the database is up and running properly. For this, open the **MySQL Command Line Client** under the section of **MySQL Server 5.0**.from the start menu. Then enter the root-password that you gave during configuration for connecting to the Server and start the queries on the command prompt.



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| aquil              |
| company            |
| hibernatetutorial  |
| login              |
| mysql              |
| ooo                |
| roseindia          |
| searc31            |
| search             |
| search1            |
| test               |
| zoo                |
+--------------------+
13 rows in set (0.02 sec)

mysql> _
```

Introduction to Hibernate

When an application is developed, a major portion of that application involves the creation and maintenance of the persistence layer to store and retrieve objects from the database. Whenever any changes are made to the underlying database schema, it can be expensive to spread those changes to the rest of the application. To overcome this problem, a new technology is used called **Hibernate**.

Hibernate is an open-source **object-relational mapping** tool **(ORM)** that lets you develop persistent classes and objects in a relational database using following common **Java idiom** such as - association, inheritance, polymorphism, composition and the Java collections framework. It provides facilities for database connection pooling, data retrieval and update, transaction management, programmatic queries, as well as entity relationship management. In addition, it allows transparent persistence that enables the applications to switch any database.

Hibernate is a better and an effective way to interact with a database. It provides an easy-to-use and powerful object-relational persistence framework for Java applications. It can be used in Java Swing applications, Java Servlet-based applications, or J2EE applications using EJB session beans.

Hibernate also supports persisting objects for collections and object relations providing a rich data query language to retrieve objects from the database that significantly reduce the development time. It maps the Java classes to the database tables. It is most useful with object-oriented domain modes and business logic in the Java-based middle-tier.

Features of Hibernate

Some of the main features of hibernate are listed below:

- Hibernate works with classes and objects instead of queries and result sets. It reduces the development time as it supports inheritance, polymorphism, composition and the Java Collection framework. So it is fully Object Oriented tool with less procedural.
- Hibernate generates SQL expressions for Java. It relieves you from manual JDBC result set handling and object conversion, and keeps your application portable to all SQL databases.
- It handles all **create-read-update-delete (CRUD)** operations using simple API, and generates DDL scripts to create DB schema (tables, constraints, sequences).
- Hibernate is Free under LGPL. It can be used to develop/package and distribute the applications for free.
- Hibernate XML binding enables data to be represented as **XML** and **POJOs** interchangeably.
- Hibernate offers full-featured query options, through which you can write plain SQL, object-oriented HQL (Hibernate Query Language), or create programatic Criteria and Example queries.
- It provides filters for working with temporal (historical), regional or authorized data as well as runtime performance monitoring via **JMX** or local Java API, including a second-level cache browser.
- Hibernate not only supports higher versions of JDK but also runs perfectly with JDK 1.2 supporting automatic generation of primary key.
- **Hibernate** is released under the **GNU Public License**, which can be used in all open source and commercial applications without limitations. It supports for a wide range of databases, including **Oracle** and **DB2**, **Sybase** as well as popular open source databases such as **PostgreSQL** and **MySQL**.

How Hibernate Works?

Hibernate is driven by **XML** configuration files to configure data connectivity and map classes to database tables with which it needs to interact. These XML files contain database connection specifics, connection pooling details, transaction factory settings, as well as references to other XML files that describe tables in the database.

When developer writes code to call API, the called API executes necessary SQL at runtime. Rather than use of byte-code processing or code generation, Hibernate uses runtime reflection to resolve the persistent properties of a class. The persisted objects are defined in a mapping document, which describes the persistent **fields** and **associations**, as well as **subclasses** or **proxies** of the persistent object. The mapping documents are compiled at the time of application startup and provide the framework with necessary information for a class.

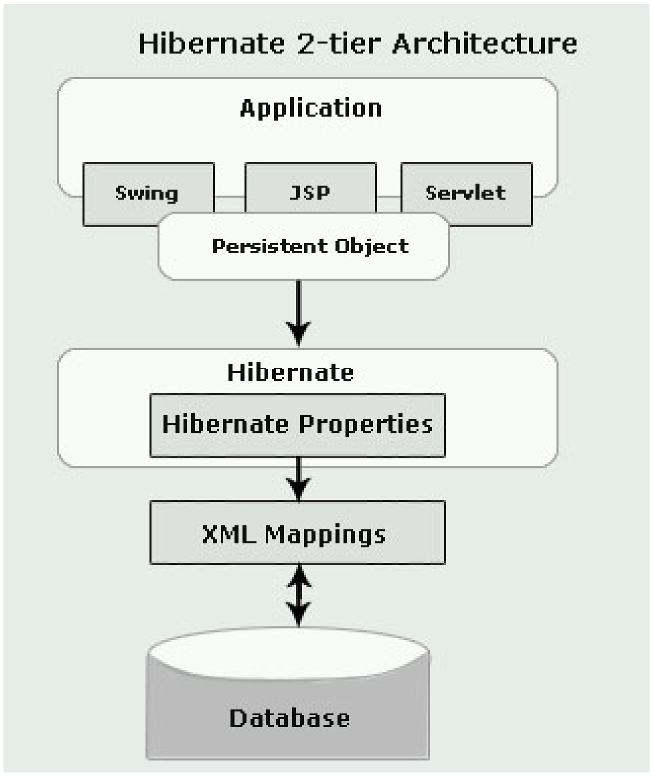
At the compile-time of mapping documents, a **SessionFactory** is also created that provides the mechanism for managing persistent classes, and the Session interface. The **Session** class provides the interface between the persistent stored data and the application. The Session interface wraps a JDBC connection, which can be user-managed or controlled by Hibernate. This interface is intended only used by a single thread-application. That means after completing the session, application is closed and discarded.

Hibernate Architecture

Unlike other technologies, Hibernate provides persistence as a service, rather than as a framework. It integrates flawlessly with various application architectures.

There are two common (recommended) architectures can be seen including Hibernate as a persistence layer.

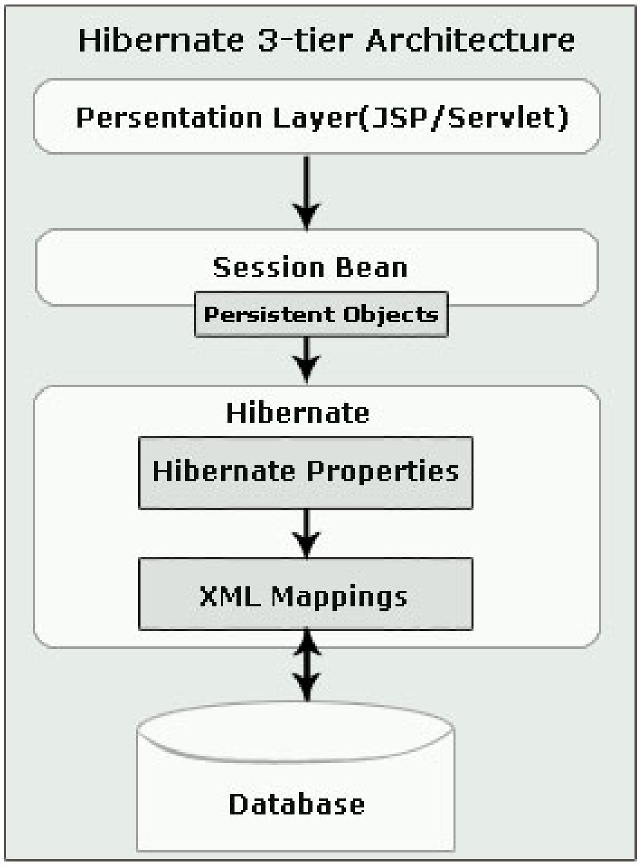
The following diagram describes the Web (two-tiered) Architecture of Hibernate:



The above diagram shows that Hibernate is using **XML mapping** to configure data connectivity to database tables, and map classes for providing persistence services (and persistent objects) to the application.

To use Hibernate, it is required to create Java classes that represent the table in the database and then map the instance variable in the class with the columns in the database. Once the mapping is complete, various operations like **select**, **insert**, **update** and **delete** the records can be performed by the Hibernate on the table of database. Hibernate automatically creates the query to perform these operations. It may also be used to persist **JavaBeans** used by **servlets/JSPs** in Model View Controller (MVC) architecture.

Now, the following diagram describes the Enterprise (three-tiered) Architecture of Hibernate:



The above diagram shows the Enterprise (3-tier) architecture of Hibernate in which, a **Session EJB** that manipulates persistent objects may use Hibernate.

Basically, Hibernate architecture has three main components:

Connection Management:

Hibernate connection management services provide well-organized management of the database connections. Database connection is the most expensive part for interacting with the database, as it requires a lot of resources to open and close the database connection.

Transaction Management:

Through this service, the user can execute more than one database statements at a time.

Object Relational Mapping:

It is a technique of mapping that maps the data representation from an object model to a relational data model. This component is used to select, insert, update and delete the records to and form the underlying table. When we pass an object to a **Session.save()** method, Hibernate reads the state of that object and executes the necessary query.

Hibernate provides a lot of flexibility to the applications interacting with the database. When we use only the object relational mapping component it is called **"Lite"** Hibernate architecture. While all three components (Object Relational mapping, Connection Management and Transaction Management) are used during data-configuration then Hibernate architecture is called **"Full Cream"** architecture.

Creating First Hibernate Application

In this section, we are going to develop first Hibernate application that insert a record into the database using Hibernate. You can run this program from Eclipse or from command prompt as well.

To create a Hibernate application, do the following steps:

- Preparing Database
- Creating persistent java objects
- Mapping the POJO's Objects to the Database table using hibernate mapping document
- Hibernate Configuration File
- Hibernate Sample Code Test (Inserting new record)
- Running the Hibernate application

I. Preparing Database

Let's consider a simple database schema named "**hibernatetutorial**" with a single table as "**CONTACT**". This table has four fields with a primary key field **"ID".**

```
CREATE TABLE contact (
ID int(11) NOT NULL default ‘0’,
FIRSTNAME varchar(20), LASTNAME
varchar(20),
EMAIL varchar(25), PRIMARY KEY (ID));
```

II. Creating persistent java objects

The second step is to create persistent java objects. In Java JDK, all primitive types (like String, char and Date) can be mapped, including classes from the Java collections framework. There is also no need to implement special interface for persistent classes because Hibernate works well with the Plain Old Java Objects programming model for these persistent classes.

Following code sample represents a java object structure that represents the “contact**” table. Generally these domain objects contain only getters and setters methods.**

Here is the code for **Contact.java**:
package roseindia.tutorial.hibernate;

```
public class Contact {

  private String firstName;
  private String lastName;
  private String email;
  private int id;

  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public String getFirstName() {
    return firstName;
  }
  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getLastName() {
    return lastName;
  }
  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

}
```

III. Mapping the POJO’s Objects to the Database table using hibernate mapping document

Each persistent class needs to be mapped with its configuration file. Following code represents Hibernate mapping file **“contact.hbm.xml”** for mapping of Contact’s Objects to the Database **Contact** table.

```
<?xml version=”1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
“-//Hibernate/Hibernate Mapping DTD 3.0//
EN”
“http://hibernate.sourceforge.net/hibernate-
mapping-3.0.dtd”>

<hibernate-mapping>
  <class
name=”roseindia.tutorial.hibernate.Contact”
table=”CONTACT”>
   <id name=”id” type=”int” column=”ID” >
    <generator class=”assigned”/>
  </id>

  <property name=”firstName”>
     <column name=”FIRSTNAME” />
   </property>
  <property name=”lastName”>
    <column name=”LASTNAME”/>
   </property>
  <property name=”email”>
    <column name=”EMAIL”/>
   </property>
 </class>
</hibernate-mapping>
```

Hibernate mapping documents are easy to understand. The **<class>** element maps a table with corresponding class. The **<id>** element represents the primary key column,

and its associated attribute in the domain object. The **<property>** elements represent all other attributes available in the domain object.
The **<generator>** method is used to generate the primary key for the new record. Here is some of the commonly used generators:

- **Increment** - This is used to generate primary keys of type long, short or int that are unique only. It should not be used in the clustered deployment environment.
- **Sequence** - Hibernate can also use the sequences to generate the primary key. It can be used with DB2, PostgreSQL, Oracle, and SAP DB databases.
- **Assigned** - Assigned method is used when application code generates the primary key.

IV. Hibernate Configuration File

Hibernate configuration file contains information that are essential to connect to persistent layer and the linked mapping documents.

In this application, Hibernate provides *connection pooling* and *transaction management* for simplicity. It uses the "**hibernate.cfg.xml"** Hibernate configuration file to create the connection pool and setup required environment. This file required either the **Data Source Name** or JDBC details to hibernate for making JDBC connection to the database. In this file we have given JDBC details.

Here is the code for hibernate.cfg.xml:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//
EN"
"http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">

<hibernate-configuration>
<session-factory>
      <property
name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
      <property
name="hibernate.connection.url">jdbc:mysql:/
/192.168.10.52/hibernatetutorial</property>
      <property
name="hibernate.connection.username">root</
property>
      <property
name="hibernate.connection.password">root</
property>
      <property
name="hibernate.connection.pool_size">10</
property>
     <property name="show_sql">true</
property>
      <property
name="dialect">org.hibernate.dialect.MySQLDialect</
property>
      <property
name="hibernate.hbm2ddl.auto">update</
property>
     <!— Mapping files —>
 <mapping resource="contact.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

In the above configuration file we have specified the database "**hibernatetutorial**" which is running on server and the user of the database is root with its password.

The **dialect** property **"org.hibernate.dialect.MySQLDialect"** tells the Hibernate that we are using **MySQL** Database. Hibernate supports many database. We can use the following databases dialect type property to connect its related database:

- DB2 - org.hibernate.dialect.DB2Dialect
- HypersonicSQL - org.hibernate.dialect.HSQLDialect
- Informix - org.hibernate.dialect.InformixDialect
- Ingres - org.hibernate.dialect.IngresDialect
- Interbase - org.hibernate.dialect.InterbaseDialect
- Pointbase - org.hibernate.dialect.PointbaseDialect

- PostgreSQL - org.hibernate.dialect.PostgreSQLDialect
- Mckoi SQL - org.hibernate.dialect.MckoiDialect
- Microsoft SQL Server - org.hibernate.dialect.SQLServerDialect
- MySQL - org.hibernate.dialect.MySQLDialect
- Oracle (any version) - org.hibernate.dialect.OracleDialect
- Oracle 9 - org.hibernate.dialect.Oracle9Dialect
- Progress - org.hibernate.dialect.ProgressDialect
- FrontBase - org.hibernate.dialect.FrontbaseDialect
- SAP DB - org.hibernate.dialect.SAPDBDialect
- Sybase - org.hibernate.dialect.SybaseDialect
- Sybase Anywhere - org.hibernate.dialect.SybaseAnywhereDialect

The **<Session-Factory>** property provides the mechanism for managing persistent classes. The **<mapping resource="contact.hbm.xml"/>** property refers to the mapping document that contains mapping for domain object and hibernate mapping document.

V. Hibernate Sample Code to Test (Inserting new record)

Now we are ready to write a program to insert the data into database. Firstly we understand about the Hibernate's Session that is the main runtime interface between a Java application and Hibernate. The **hibernate.SessionFactory** allows application to create the Hibernate Sesssion reading the configuration from **hibernate.cfg.xml** file. Then the save method as **session.save(contact)** on session object is used to save the contact information to the database:

Typical Hibernate program begins with configuration. It can be configured in two ways. **Programmatically** and **Configuration file based**.

In **Configuration** file based mode, hibernate looks for configuration file **"hibernate.cfg.xml"** in the claspath. It creates mapping of tables and domain objects based on the provided resource mapping.
While in the **Programmatic** configuration method, the JDBC connection details and resource mapping details etc are supplied in the program using Configuration API.

Following example **HibernateExample.java** shows **Configuration file based** method of hibernate.

```
package roseindia.tutorial.hibernate;

import javax.transaction.Transaction;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateExample {


  public static void main(String[] args) {
    // TODO Auto-generated method stub
     Session session = null;

        try{
         // This step will read hibernate.cfg.xml
and prepare hibernate for use
          SessionFactory sessionFactory = new
Configuration().configure().buildSessionFactory();
           session
=sessionFactory.openSession();
           org.hibernate.Transaction tr =
session.beginTransaction();
           //Create new instance of Contact
and set values in it by reading them from
form object
            System.out.println("Inserting
Record");
           Contact contact = new Contact();
           contact.setId(3);
           contact.setFirstName("Nisha");
           contact.setLastName("Gupta");

     contact.setEmail("nishu_gpt@yahoo.com");
           session.save(contact);
           tr.commit();
           System.out.println("Done");
```

```
      }catch(Exception e){
        System.out.println(e.getMessage());
      }
      finally{
        // Actual contact insertion will happen
at this step
        //session.flush();
        session.close();

        }
  }

}
```

Here is the sample code that shows **Programmatic** configuration of hibernate.

```
Configuration config = new Configuration()
.addResource("roseindia/tutorial/hibernate/
contact.hbm.xml ")
Configuration config = new Configuration()
.addClass(roseindia.tutorial.hibernate.Contact.class)
.setProperty("hibernate.dialect",
"org.hibernate.dialect.MySQLDialect ")
.setProperty("hibernate.connection.driver_class",
" com.mysql.jdbc.Driver ")
. . . SessionFactory sessions =
config.buildSessionFactory();
```

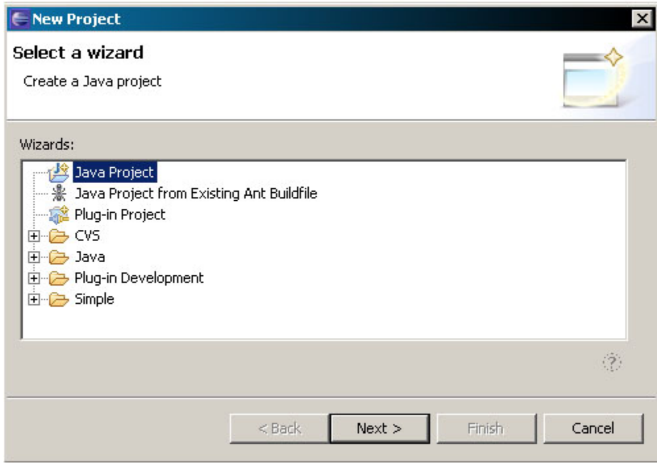
In the next section, you will see how to run and test the Hibernate-application.

Running the Hibernate Application

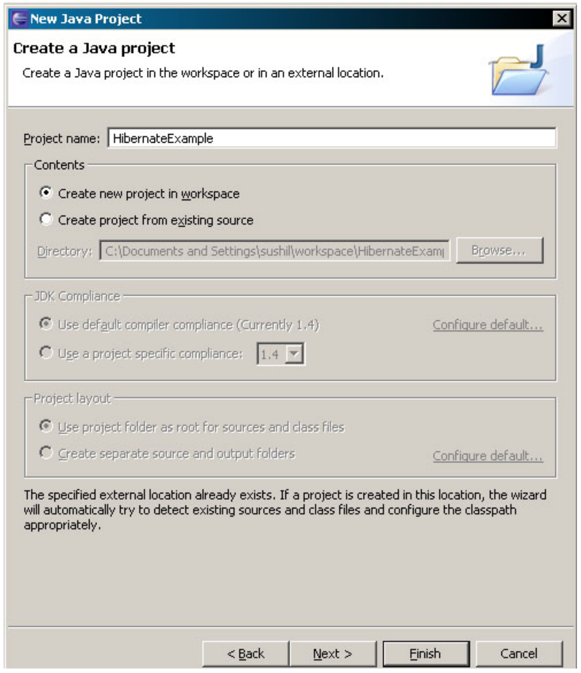
In previous section, we wrote the code for developing a Hibernate application. Now you will see, how this application is to be run and tested.

To run the example you should have the **Eclipse IDE** and **Hibernate** on your machine. Hibernate is free open source software it can be download from http://www.hibernate.org/6.html. Visit the site and download Hibernate 3.0. You can download the Hibernate and install it yourself but we have provided all the necessary files in one zip file. Download the **example-code** and **library** from here and extract the content in your favorite directory say **"C:\hibernateexample".** Start the **Eclipse.exe** project and do the following steps shown below:

1. Once you have opened the Eclipse, open **File** menu, go to **New** and select **Project** option. Then click the **Next** button on the displayed dialog.



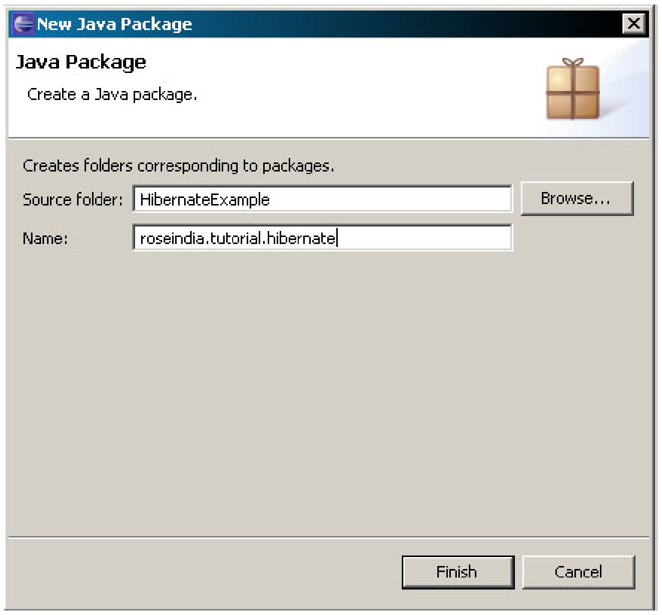
2. In the next dialog, give the project name as **HibernateExample** to the Project name command line, and click the **Finish** button.



Project name: HibernateExample
(●) Create new project in workspace
( ) Create project from existing source
Directory: C:\Documents and Settings\sushil\workspace\HibernateExamp
(●) Use default compiler compliance (Currently 1.4)
( ) Use a project specific compliance: 1.4
(●) Use project folder as root for sources and class files
( ) Create separate source and output folders

The specified external location already exists. If a project is created in this location, the wizard will automatically try to detect existing sources and class files and configure the classpath appropriately.

3. Then after, right click on the main project folder "**HibernateExample"**, go to **New** and select **Package** option. The type the full package name **roseindia.tutorial.hibernate** as we have made it for class files. Click the **Finish** button.



New Java Package

Java Package
Create a Java package.

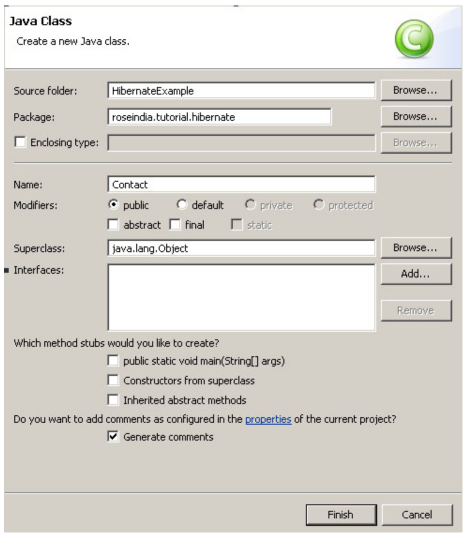
Creates folders corresponding to packages.

Source folder: HibernateExample [Browse...]

Name: roseindia.tutorial.hibernate

[Finish] [Cancel]

4. Now right click on the created package, go to **New** and select **Class** option. Then give class name as **Contact** to the Name command line, and click the **Finish** button. Then write the code for the Contact class to add **getter** & **setter** methods.



Java Class
Create a new Java class.

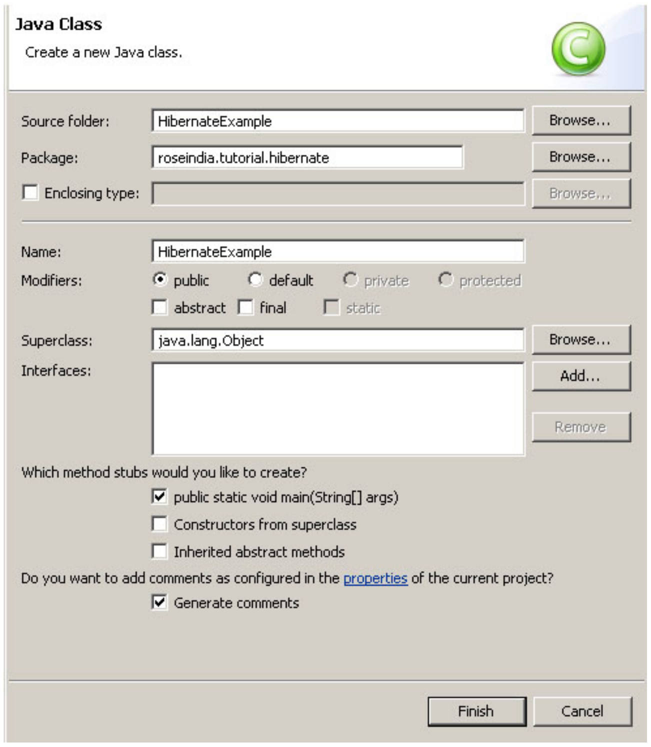
Source folder: HibernateExample [Browse...]
Package: roseindia.tutorial.hibernate [Browse...]
☐ Enclosing type: [Browse...]

Name: Contact
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object [Browse...]
Interfaces: [Add...] [Remove]

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods
Do you want to add comments as configured in the properties of the current project?
☑ Generate comments

[Finish] [Cancel]

5. Repeat the 4th step, and give the class name "**HibernateExample**". Click on the **"public static void main(String []args)"** checkbox, and click the **Finish** button. Then write the code for the **HibernateExample** class to write the main method.



Java Class
Create a new Java class.

Source folder: HibernateExample [Browse...]
Package: roseindia.tutorial.hibernate [Browse...]
☐ Enclosing type: [Browse...]

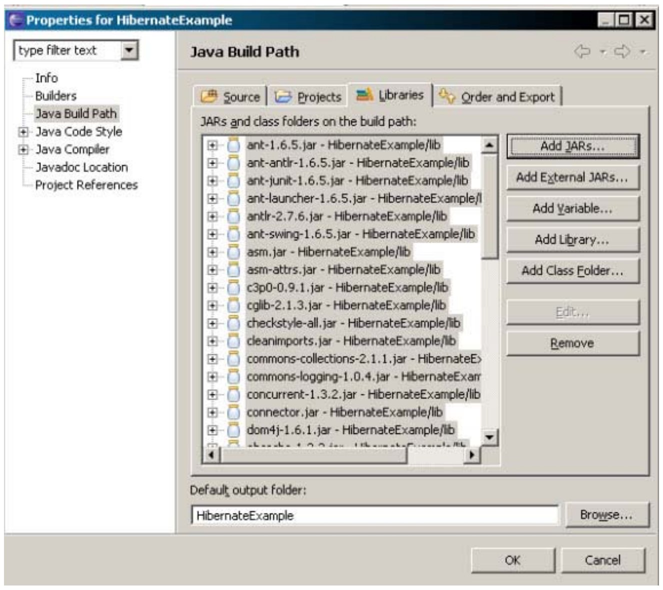
Name: HibernateExample
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object [Browse...]
Interfaces: [Add...] [Remove]

Which method stubs would you like to create?
☑ public static void main(String[] args)
☐ Constructors from superclass
☐ Inherited abstract methods
Do you want to add comments as configured in the properties of the current project?
☑ Generate comments

[Finish] [Cancel]

6. Now go back to the main project "**HibernateExample**", and copy the **lib** folder to that project. Then right click to that project, go to the **Build Path**, and select **Configure Build Path**. Then after select the **JRE System Library** folder, and click the **Add JARs** button.

7. Then after open **lib** folder and select all **jar** files to be configured. Then click **ok** button to close the dialog.

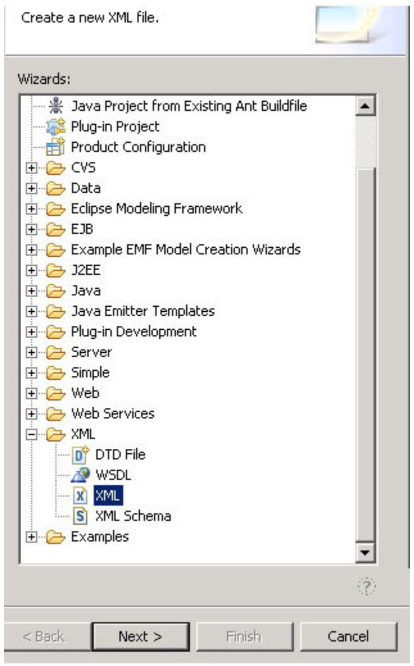


ant-1.6.5.jar - HibernateExample/lib
ant-antlr-1.6.5.jar - HibernateExample/lib
ant-junit-1.6.5.jar - HibernateExample/lib
ant-launcher-1.6.5.jar - HibernateExample/l
antlr-2.7.6.jar - HibernateExample/lib
ant-swing-1.6.5.jar - HibernateExample/lib
asm.jar - HibernateExample/lib
asm-attrs.jar - HibernateExample/lib
c3p0-0.9.1.jar - HibernateExample/lib
cglib-2.1.3.jar - HibernateExample/lib
checkstyle-all.jar - HibernateExample/lib
cleanimports.jar - HibernateExample/lib
commons-collections-2.1.1.jar - HibernateE
commons-logging-1.0.4.jar - HibernateExam
concurrent-1.3.2.jar - HibernateExample/lib
connector.jar - HibernateExample/lib
dom4j-1.6.1.jar - HibernateExample/lib

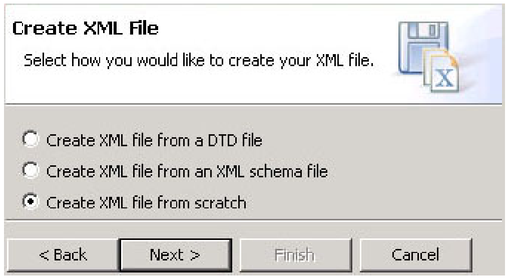
HibernateExample

Finally click **ok** button again.

8. Now to add the both xml files to the main project, right click on "**HibernateExample",** go to **New** and select **Other** option. In the opened dialog, open the **XML** folder, and select file type as **XML** then click the **Next** button.

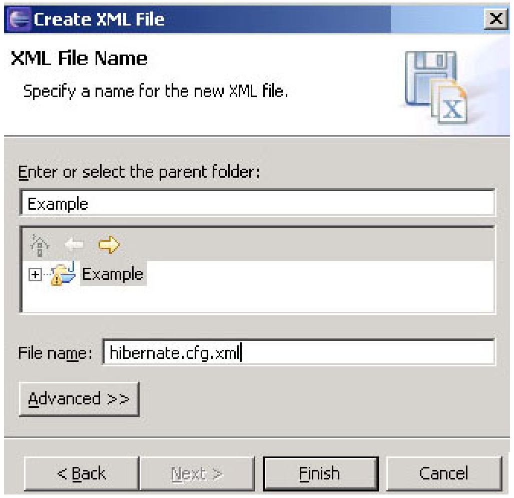


9. Then after make sure to be selected the last one **option** button, and click the **Next** button.



Create XML file from a DTD file
Create XML file from an XML schema file
[✓ Create XML file from scratch]

10. In the next dialog, give the configuration file name as **"hibernate.cfg.xml"** to the File name command line, and click the **Finish** button.



Enter or select the parent folder: Example

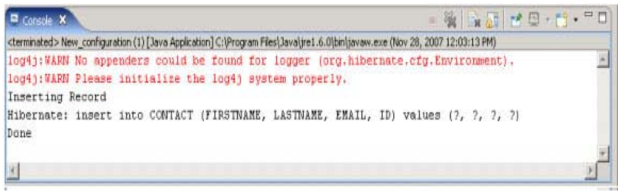
File name: hibernate.cfg.xml

11. Repeat the 10th step to add the **"contact.hbm.xml"** file to the "**HibernateExample"** package.

12. Once all files coded to Hibernate application have been included to the main project folder with the lib folder then save the application, and run it to test.

To run the application, select Run-> Run As -> Java Application from the Run menu bar as shown below.

The application will be run and show the output as:



```
<terminated> New_configuration (1) [Java Application] C:\Program Files\Java\jre1.6.0\bin\javaw.exe (Nov 28, 2007 12:03:13 PM)
log4j:WARN No appenders could be found for logger (org.hibernate.cfg.Environment).
log4j:WARN Please initialize the log4j system properly.
Inserting Record
Hibernate: insert into CONTACT (FIRSTNAME, LASTNAME, EMAIL, ID) values (?, ?, ?, ?)
Done
```

ANT

Initially, it was not an easy task to work on the large java projects but thanks to ant for simplifying the complex details apart from the development concepts that the software developers have to keep in regard while developing the java software.

To understand more clearly, Let’s view a project development scenario without the ant tool where you need to work on a large Java project without any project-building tool. This project consist of multiple java files, classes depending on other classes, stubs or drivers classes situated in multiple directories, additionally the output files are required to save to different locations. If you are coordinating all of this manually then so many of the development hours are spent in changing directories compiling individual files and so on. This consumes a lot of time and increases the development stress and hassle. However, if you have gone for any project-building tool like ant then you would have saved a lot of development time and reduced the stress to a great extent.

Ant (originally an acronym for Another Neat Tool), is a build tool with special support for the Java programming language but can be used for just about everything. Ant is platform-independent like java. Ant is particularly good at automating complicated repetitive tasks and thus is well suited for automating standardized build processes. Ant accepts an instruction in the form of XML documents thus is extensible and easy to maintain.

Apache Ant is a software tool, which is similar to “make” but is written in the Java language, requires the Java platform, and is best suited to building Java projects. Noticeable difference between Ant and make is that Ant uses XML to describe the build process and its dependencies, whereas make has its Makefile format. In ant, the XML file is always named as the build.xml.

Ant is an Apache project. It is open source software and is released under the Apache Software License.
Ant allows the developer to automate the repeated process involved in the development of JEE application. Developers can easily write the script to automate the build process like compilation, archiving and deployment.
Getting Hands-On Ant
For this article I am using jdk1.6.0 and Application Server JBoss 3.2.3 to deploy and test JEE application.

Downloading and Installing Ant

- Before installing Ant make sure you have JDK1.3 or above installed on your machine.
- Download ant from http://jakarta.apache.org/ant/ and unzip the file into your favorite directory.
- Set the class path to the bin directory of the ant.

Let’s assume that Ant has been installed in **c:\ant.**
Put the following code in the autoexec.bat file:

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk1.6
set PATH=%PATH%;%ANT_HOME%\bin
```

Let’s Test The Ant

Go to command prompt and issue the following command.

```
C:\javajazzup>Ant
Buildfile: build.xml does not exist!Build
failed
C:\ javajazzup >
```

If every thing has been installed properly then the Ant will give the message as shown above.

Now its time to do some work with Ant.
Ant always uses the default configuration file called **build.xml** to work. This is the file where you define the process of compiling, building and deploying the application.
Writing build.xml filebuild.xml is a xml file used by ant utility to compile, build and deploy the application.

ANT

Now here is the code of simple **build.xml** file which compiles a java file present in **src** directory and places compiled class file in **build / src** directory.

```
<?xml version=”1.0"?>
<!— Build file for our first application —>
<project name=”Ant test project”
default=”build” basedir=”.”>
<target name=”build” >
<javac srcdir=”src” destdir=”build/src”
debug=”true” includes=”**/*.java” />
</target>
</project>
```

First line of the build.xml file represents the document type declaration. Next line is comment entry. Third line is the project tag. Each build file contains one project tag and all the instruction are written in the project tag. The project tag:

< project name=”Ant test project” default=”build” basedir=”.” >

requires three attributes namely **name, default** and **basedir**.
Here is the description of the attributes:

Attribute	Description
name	Represents the name of the project.
default	Name of the default target to use when no target is supplied.
basedir	Name of the base directory from which all path calculations are done.

All the attributes are required.
One project may contain one or more targets. In this example there is only one target. Which uses task **javac** to compile the java files.

```
<target name=”build” >
<javac srcdir=”src” destdir=”build/src”
debug=”true” includes=”**/*.java”/>
</target>
```

Here is the code of our test.java file, which is to be compiled by the Ant utility.

```
class test{
public static void main (String args[])
{
System.out.println(“This is example 1”);
}
}
```

Download the code of this article and unzip it in c:\. files will be unzipped in anttest directory. To run the Ant utility to compile the file go to **c:\anttest\example1** and issue **ant** command. You will receive the following out put.

```
C:\anttest\example1>antBuildfile: build.xml
build:[javac] Compiling 1 source file to
C:\anttest\example1\build\src
BUILD SUCCESSFUL
Total time: 4 seconds
C:\anttest\example1>
```

Above-mentioned process compiles the file and places in the **build \ src** directory.
Now let’s see how to build a web application and install it on the Jboss 3.2.3 application server.
After the completion of this article you will be able to include jsp, html and servlets in the ear file and deploy on the JBoss 3.2.3 application server. A complete example in this section will provide a strong foundation for the further development.

Let’s write one Hello World Servlet and a JSP file file to retrieve the Hello World Servlet. In order to deploy components we have to build .ear file which is the standard format for the deployment of JEE application.
First of all let’s understand the structure of .ear and .war files.

Enterprise Archive Contents
Enterprise Archive (.ear) component follows the standard directory structure defined in the JEE specification.

Directory Structure of .ear archive
/
.war and .jar files
📁META-INF
application.xml

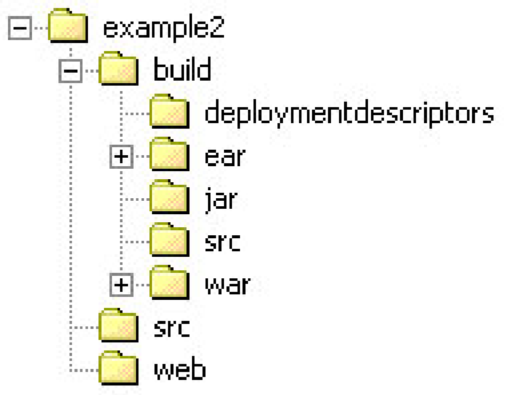
ANT

In the .ear file .war, .jar and application.xml file are packaged in the above format.
Enterprise Archive Contents
Web component follows the standard directory structure defined in the JEE specification.

Directory Structure of Web Component
/
index.htm, JSP, Images etc..
WEB-INF

web.xml
classes
Servlet classes
lib
jar files

Root directory of the web archive ('.war' file) contains all the html, jsp, images files and the additional directories containing these files. In the root directory there is a special directory **'WEB-INF'** which contains the web deployment descriptor (web.xml), classes and the lib directory.
Directory Structure of Example2 directory
After understanding the structure of .ear and .war file let's look at the directory structure of **example2** directory where we have work to develop the deployable .ear file.
Directory structure:



```
example2
  build
    deploymentdescriptors
    ear
    jar
    src
    war
  src
  web
```

Description of Directory and its content:

Directory	**Description**
example2	Base directory which contains **build.xml** and the .ear file generated by Ant utility will be placed here.
build	Various files generated by Ant utility will be placed in different directories under this directory.
build/deploymentdesciptors	Web.xml and application.xml files are placed in this directory.
build/ear	Intermediate files for the assembling of example2.ear ear file are placed here.build/jar Any jar file if required will be placed in this directory.
build/war	Intermediate files for the assembling of example2.war ear file are placed here.
build/src	All the compiled .class files are placed in this directory.
src	All the java source files are placed here.
web	All the html,jsp, images etc. files are placed in this directory.

Here we are creating HelloWorld.java and index.jsp which is in the /src and /web directory respectively.

Source code of HelloWorld.java:

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
```

```
public void service(HttpServletRequest
request, HttpServletResponse response)
throws  IOException, ServletException{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head>");
   out.println("<title>Hello World Servlet!
    </title>");
   out.println("</head>");
   out.println("<body>");
   out.println("<p align=\"center\"><font
    size=\"5\" color=\"#000080\">Hello
    World!  </font></p>");
   out.println("<p align=\"center\"><a
    href=\"javascript:history.back()\">
    Go to Home</a></p>");
   out.println("</body>");
   out.println("</html>");
   }
}
```

Here is the code of **index.jsp** file:

```
<%@page language="java" %>
<html>
<head>
<title>Welcome to JBoss 3.2.3 World</title>
</head>
<body bgcolor="#FFFFCC">
<p align="center"><font size="6"
color="#800000">Welcome to<br>
Jboss 3.2.3 World</font></p>
<p align="center">
<font color="#000080"
size="4">Congratulations you have
successfully installed it </font></p>

<p align="center">
<font color="#000080" size="4">
<a href="servlet/HelloWorld">Click here to</
a> execute Hello World Servlet.
</font>
</p>

</body>

</html>
```

Download all the files for this section from here.

Writing Application and Web deployment descriptor

Since in this section we are developing one servlet and one jsp file so our deployment descriptor is very simple.
web.xml file:

```
<?xml version="1.0" encoding="ISO-8859-
1"?><!DOCTYPE web-appPUBLIC "-//Sun
Microsystems, Inc.//DTD Web Application 2.2/
/EN""http://java.sun.com/j2ee/dtds/web-
app_2_2.dtd"><web-app><servlet>
<servlet-name>HelloWorld</servlet-name>
<servlet-class>HelloWorld</servlet-class> </
servlet> <servlet-mapping>      <url-
pattern>/servlet/HelloWorld</url-pattern>
<servlet-name>HelloWorld</servlet-name></
servlet-mapping></web-app>
```

application.xml file:

```
<?xml version="1.0" encoding="ISO-8859-
1"?>
<application>
<display-name>Example 2 </display-name>
<module>
<web>
<web-uri>example2.war</web-uri>
<context-root>/example2</context-root>
</web>
</module>
</application>
```

Above application.xml file describe the content of example2.ear. Tag **<web-uri>example2.war</web-uri>** describe the name of web module (i.e.. example2.war) packaged in the archive. The context root of this example2.ear is **eample2**.

Writing Ant build xml file
To build example2.ear file, I have written build.xml which compiles source code and builds deployable archive file.
build.xml file:

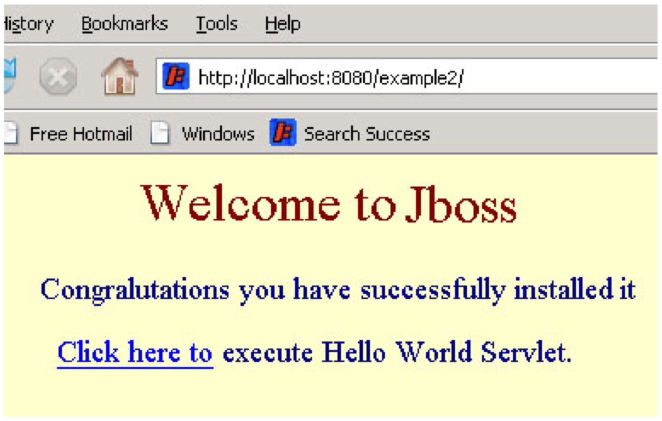
```
<?xml version="1.0"?>
<project name="Jboss 3.0 tutorial series"
default="all" basedir=".">
<target name="init">
<property name="dirs.base"
value="${basedir}"/>
<property name="classdir"
value="${dirs.base}/build/src"/>
<property name="src" value="${dirs.base}/
src"/>
<property name="web" value="${dirs.base}/
web"/>
<property name="deploymentdescription"
value="${dirs.base}/build/
deploymentdescriptors"/>

<property name="warFile"
value="example2.war"/>
<property name="earFile"
value="example2.ear"/>


<property name="earDir"
value="${dirs.base}/build/ear"/>
<property name="warDir"
value="${dirs.base}/build/war"/>


<!— Create Web-inf and classes directories —
>
<mkdir dir="${warDir}/WEB-INF"/>
<mkdir dir="${warDir}/WEB-INF/classes"/>

<!— Create Meta-inf and classes directories —
>
<mkdir dir="${earDir}/META-INF"/>

</target>

<!— Main target —>
<target name="all"
depends="init,build,buildWar,buildEar"/>


<!— Compile Java Files and store in /build/src
directory —>
<target name="build" >
<javac srcdir="${src}" destdir="${classdir}"
debug="true" includes="**/*.java" />
</target>

<!— Create the War File —>
<target name="buildWar" depends="init">
<copy todir="${warDir}/WEB-INF/classes">
<fileset dir="${classdir}" includes="**/
*.class" />
</copy>

<copy todir="${warDir}/WEB-INF">
<fileset dir="${deploymentdescription}"
includes="web.xml" />
</copy>

<copy todir="${warDir}">
<fileset dir="${web}" includes="**/*.*" />
</copy>

<!— Create war file and place in ear directory
—>
<jar jarfile="${earDir}/${warFile}"
basedir="${warDir}" />


</target>


<!— Create the War File —>
<target name="buildEar" depends="init">
<copy todir="${earDir}/META-INF">
<fileset dir="${deploymentdescription}"
includes="application.xml" />
</copy>

<!— Create ear file and place in ear directory
—>
<jar jarfile="${dirs.base}/${earFile}"
basedir="${earDir}" />
</target>

</project>
```

Above **build.xml** file is design to create example2.ear for us in the base directory. Running Ant utility to build example2.ear Now it's time to build example2.ear and deploy on the Jboss 3.2.3 application server. To execute Ant utility go to c:\anttest\example2 directory and issue ant command.

Out put of ant command:

```
C:\anttest\example2>ant
Buildfile: build.xml

init:
[mkdir] Created dir:
C:\anttest\example2\build\war\WEB-INF
[mkdir] Created dir:
C:\anttest\example2\build\war\WEB-
INF\classes
[mkdir] Created dir:
C:\anttest\example2\build\ear\META-INF

build:
[javac] Compiling 1 source file to
C:\anttest\example2\build\src

buildWar:
[copy] Copying 1 file to
C:\anttest\example2\build\war\WEB-
INF\classes
[copy] Copying 1 file to
C:\anttest\example2\build\war\WEB-INF
[copy] Copying 1 file to
C:\anttest\example2\build\war
[jar] Building jar:
C:\anttest\example2\build\ear\example2.war

buildEar:
[copy] Copying 1 file to
C:\anttest\example2\build\ear\META-INF
[jar] Building jar:
C:\anttest\example2\example2.ear

all:

BUILD SUCCESSFUL

Total time: 8 seconds
C:\anttest\example2>|
```

The above process will create example2.ear in c:\anttest\example2 directory.
Deploying and testing J2EE application
Statrt Jboss 3.2.3 and copy example2.ear file into the JBOSS_HOME/server/default/deploy directory. JBoss application server automatically deploys the application. Open web browse and type http://localhost:8080/example2 in the web browser. Browse should show the screen something like this:



Welcome to Jboss

Congralutations you have successfully installed it

Click here to execute Hello World Servlet.

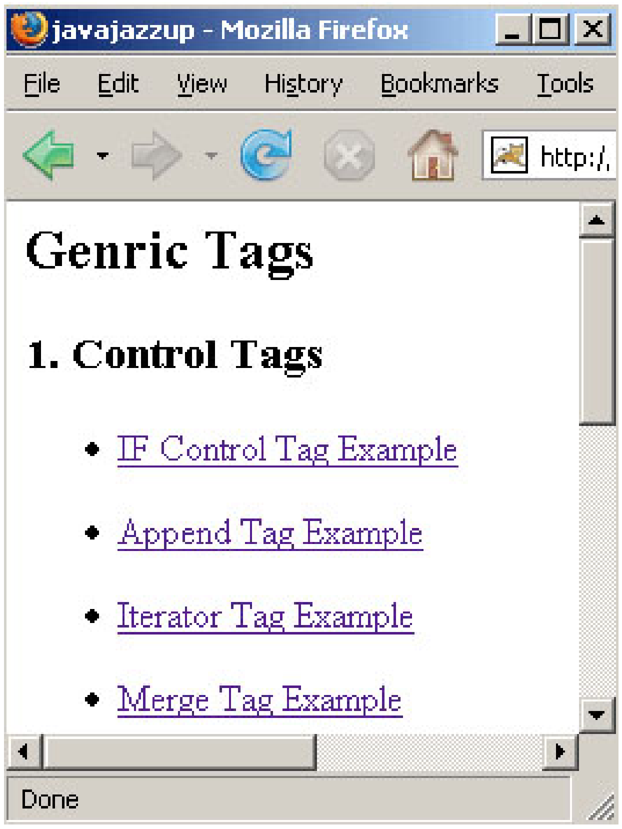
Also try to execute Hello World Servlet by clicking "Click Here to" link on the index.jsp in the browser.

In this lesson you learned how to write build.xml file to automate the process of .ear file creation. Ant utility with help of our build.xml file automatically compiles source code and assembles JEE application for us. Ant utility is very power full and it reduces the development time significantly.

Struts2 Tags

Apache Struts is an open-source framework used to develop Java web applications. Strut1 with all standard Java technologies and packages of Jakarta assists in creating an extensible development environment. However, with the growing demand of web applications, Strut 1 needs to be changed with the increasing demand. This led to the creation of Strut2, which are more users friendly with the features like Ajax, rapid development and extensibility. In this section we will start introducing you with some tags provided with struts 2 framework and the rest will be included in the subsequent issues of the magazine.

Just download the zip file "struts2generictags.zip" from any link given below of each page of this article, unzip it and copy this application to the webapps directory of Tomcat. Start tomcat and write http://localhost:8080/struts2generictags/index.jsp to the address bar. You can examine the result of each tag from thispage.


javajazzup - Mozilla Firefox
File Edit View History Bookmarks Tools
http:/,
Done

Genric Tags

1. Control Tags

- IF Control Tag Example
- Append Tag Example
- Iterator Tag Example
- Merge Tag Example

If you make changes in the application, run the build.xml with ant tool and restart the tomcat server.

Struts 2 tags type:

1. Struts Generic Tags
The struts generic tags are used to control the execution flow when pages are rendered. Another use of struts generic tags is data extraction.

Generic Tags are further classified into:

i) Control **Tags**
The Control Tags are used for flow control, such as if, else and iterate.

ii) Data Tags.
The Data Tags are used for data manipulation or creation, such as bean, push, and i18n.

2. Struts UI tags
Struts UI Tags are mainly designed to use the data from your action/value stack or from Data Tags. These tags are used to display the data on the HTML page. The UI tags are driven by templates and themes.
Struts UI Tags are of two types:

i) Form Tags
ii) Non-Form Tags

This issue starts explaining struts 2 tags from **Control Tags** and subsequent issues will continue further.
Control Tags

1. Control Tags-If / Else If / Else
'If' tag could be used by itself or with 'Else If' Tag and/or single/multiple 'Else' Tag. Create a JSP page IfControlTag.jsp. Set a property 'technologyName' with a value 'Java' as <s:set name="technologyName" value="%{'Java'}"/>. Among if, elseif and else tags only one tag evaluates at a time. Evaluation is based upon the conditions being processed. Evaluated conditions must be of boolean type. This is illustrated in the following Jsp page. If the condition in <s:if > tag evaluates to 'true' then only this tag is evaluated and others are discarded. If the condition in <s:if > tag

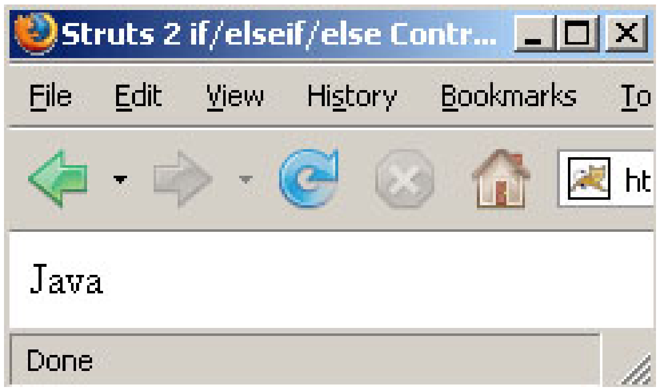
evaluates to 'false' and <s:elseif > tag evaluates to 'true' then the body of the <s:elseif > tag is processed. If the condition in <s:if > tag and <s:elseif > tags evaluates to 'false' then only the <s:else > tag is processed.

IfControlTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title>Struts 2 if/elseif/else Control Tag
Example</title>
  </head>
  <body>
  <s:set name="technologyName"
value="%{'Java'}"/>

      <s:if
test="%{#technologyName=='Java'}">
        <div><s:property
value="%{#technologyName}" /></div>
      </s:if>

      <s:elseif
test="%{#technologyName=='Jav'}">
        <div><s:property
value="%{#technologyName}" /></div>
      </s:elseif>

      <s:else>
        <div>Technology Value is not Java</
div>
      </s:else>

  </body>
</html>
```

In the IfControlTag.jsp only <s:if> tag evaluates to true, we get the output equal to "Java".

Output:


Struts 2 if/elseif/else Contr...
File Edit View History Bookmarks To
ht
Done

```
Java
```

2. Append Tag (Control Tags) Example
The append tag is a generic tag that is used to merge multiple iterators into one iterator. Append Iterator tag is used to append iterators to form an appended iterator through which the entries goes from one iterator to another after each respective iterator is exhausted of entries. Create two lists in the action class and populate them with various items as shown in the "**AppendTag**" class.

AppendTag.java

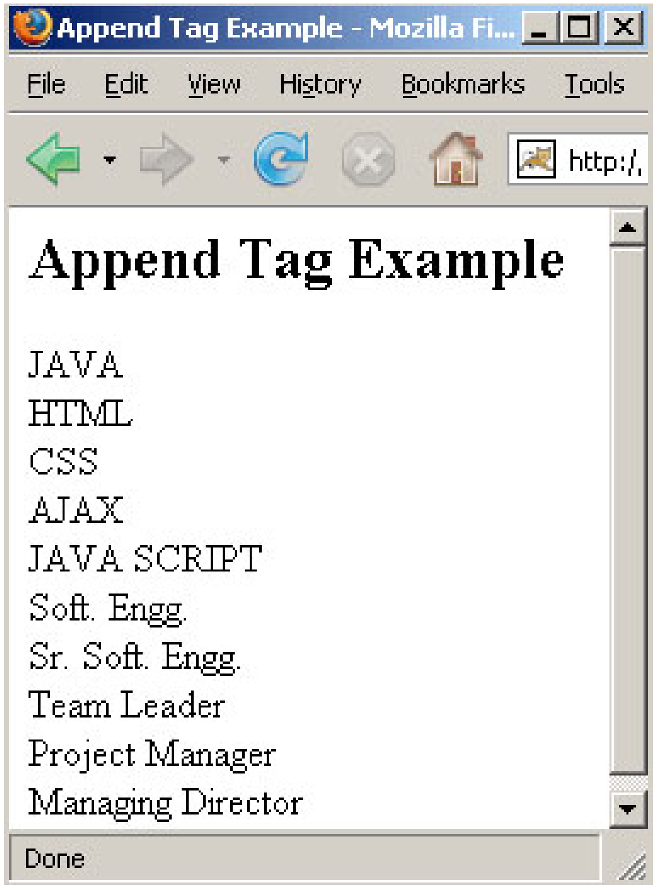
```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class AppendTag extends
ActionSupport{

  private List list1;
  private List list2;

  public String execute()throws Exception{
    list1 = new ArrayList();
    list1.add("JAVA");
    list1.add("HTML");
    list1.add("CSS");
    list1.add("AJAX");
    list1.add("JAVA SCRIPT");

    list2 = new ArrayList();
    list2.add("Soft. Engg.");
    list2.add("Sr. Soft. Engg.");
    list2.add("Team Leader");
    list2.add("Project Manager");
    list2.add("Managing Director");
    return  SUCCESS;
  }

  public List getList1(){
    return list1;
  }

  public List getList2(){
    return list2;
  }
}
```

Now create a jsp page using <s:append> and <s:iterator> tags as shown in the **AppendTag.jsp** page. The append tag is used to merge multiple iterators into one iterator. The "**id**" parameter keeps the resultant appended iterator stored under the stack's context and the **"value"** parameter is used to get the values contained within the resultant iterator.

AppendTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title> Append Tag Example</title>
  </head>
  <body>
  <h2>Append Tag Example</h2>
    <s:append id="myAppendList">
      <s:param value="%{list1}" />
      <s:param value="%{list2}" />
    </s:append>
    <s:iterator value="%{#myAppendList}">
      <s:property /><br>
    </s:iterator>
  </body>
</html>
```

Output:


Append Tag Example - Mozilla Fi...
File Edit View History Bookmarks Tools
http:/,
Done

```
Append Tag Example

JAVA
HTML
CSS
AJAX
JAVA SCRIPT
Soft. Engg.
Sr. Soft. Engg.
Team Leader
Project Manager
Managing Director
```

3. Iterator Tag (Control Tags) Example
Iterator tag is used to iterate over a value. An iterable value can be either of: java.util.Collection, java.util.Iterator.

IteratorTag.java

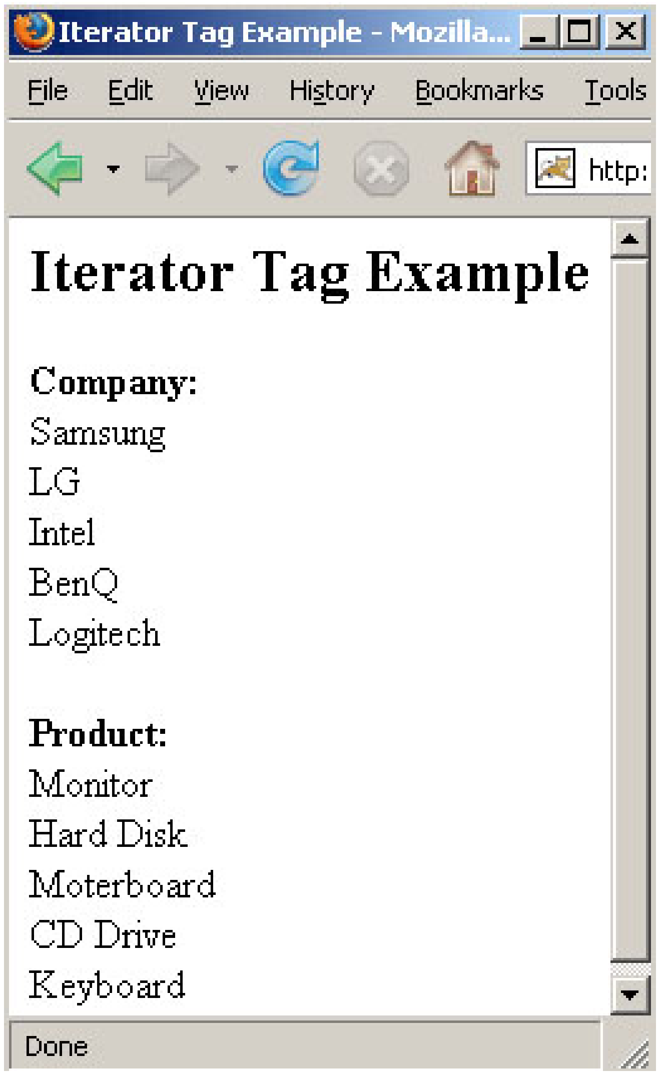
```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class IteratorTag extends
ActionSupport{

  private List list1;
  private List list2;
  public String execute()throws Exception{
    list1 = new ArrayList();
    list1.add("Samsung");
    list1.add("LG");
    list1.add("Intel");
    list1.add("BenQ");
    list1.add("Logitech");

    list2 = new ArrayList();
    list2.add("Monitor");
    list2.add("Hard Disk");
    list2.add("Moterboard");
    list2.add("CD Drive");
    list2.add("Keyboard");
    return SUCCESS;

  }

  public List getList1(){
    return list1;
  }

  public List getList2(){
    return list2;
  }
}
```

The following example retrieves the value of the getMyList() method of the current object on the value stack and uses it to iterate over. The <s:property/> tag prints out the current value of the iterator.

IteratorTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title>Iterator Tag Example</title>
  </head>
  <body>
    <h2>Iterator Tag Example</h2>
    <b>Company:</b><br>
      <s:iterator value="list1" >
      <s:property /><br>
      </s:iterator>
      <br>
      <b>Product:</b><br>
      <s:iterator value="list2">
        <s:property /><br>
      </s:iterator>
  </body>
</html>
```

Output:


Iterator Tag Example - Mozilla...
File Edit View History Bookmarks Tools
http:
Done

```
Iterator Tag Example

Company:
Samsung
LG
Intel
BenQ
Logitech

Product:
Monitor
Hard Disk
Moterboard
CD Drive
Keyboard
```

4. Merge Tag (Control Tags) Example
The merge tag is a generic tag that is used to merge iterators. The successive call to the merge iterator causes each merge iterator to have a chance to expose its element, subsequently next call allows the next iterator to expose its element. Once the last iterator is done exposing its element, the first iterator is allowed to do so again (unless it is exhausted of entries). In the current example, 2 lists being merged where each list have 5 entries.

MergeTag.java

```
package net.javajazzup;
import
```

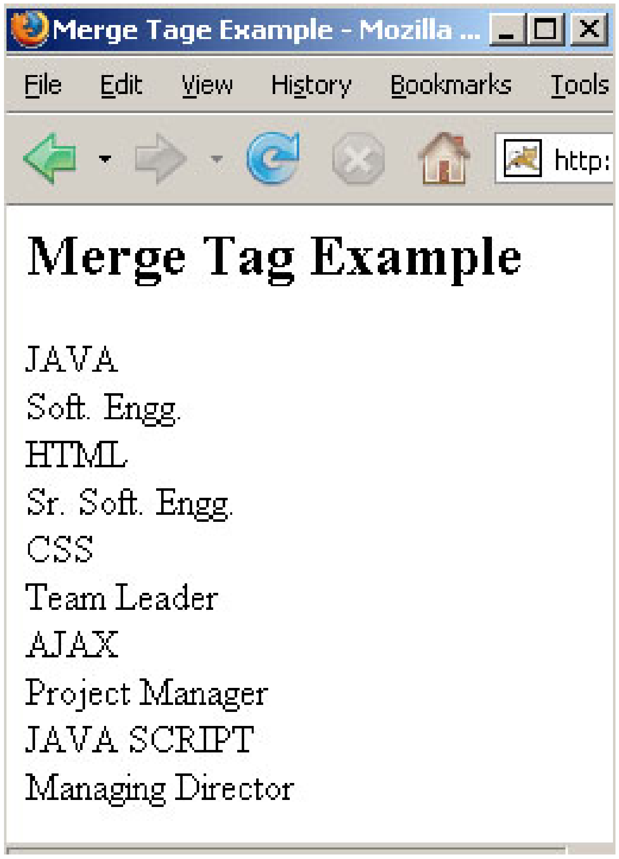
```
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class MergeTag extends ActionSupport
{
  private List list1;
  private List list2;

  public String execute() throws Exception{
    list1 = new ArrayList();
    list1.add("JAVA");
    list1.add("HTML");
    list1.add("CSS");
    list1.add("AJAX");
    list1.add("JAVA SCRIPT");

    list2 = new ArrayList();
    list2.add("Soft. Engg.");
    list2.add("Sr. Soft. Engg.");
    list2.add("Team Leader");
    list2.add("Project Manager");
    list2.add("Managing Director");
    return SUCCESS;
  }

  public List getList1(){
    return list1;
  }

  public List getList2(){
    return list2;
  }
}
```

Now create a jsp page using <s:merge> and <s:param value> tags as shown in the mergeTag.jsp page. The merge tag is used to merge iterators. The "id" parameter keeps the resultant iterator stored under in the stack's context and the "value" parameter in the <s:iterator> is used to get the values contained within the respective iterators.

MergeTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title>Merge Tage Example</title>
  </head>
  <body>
    <h2>Merge Tag Example</h2>
      <s:merge id="mergeId">
        <s:param value="%{list1}" />
        <s:param value="%{list2}" />
      </s:merge>
      <s:iterator value="%{#mergeId}">
        <s:property /><br>
      </s:iterator>
  </body>
</html>
```

Output:


Merge Tage Example - Mozilla ...
File Edit View History Bookmarks Tools
http:

```
Merge Tag Example

JAVA
Soft. Engg.
HTML
Sr. Soft. Engg.
CSS
Team Leader
AJAX
Project Manager
JAVA SCRIPT
Managing Director
```

5. Generator Tag (Control Tags) Example
The generator tag is a generic tag that is used to generate iterators based on different **attributes** passed. Here we will not pass any attribute.

GeneratorTag.java

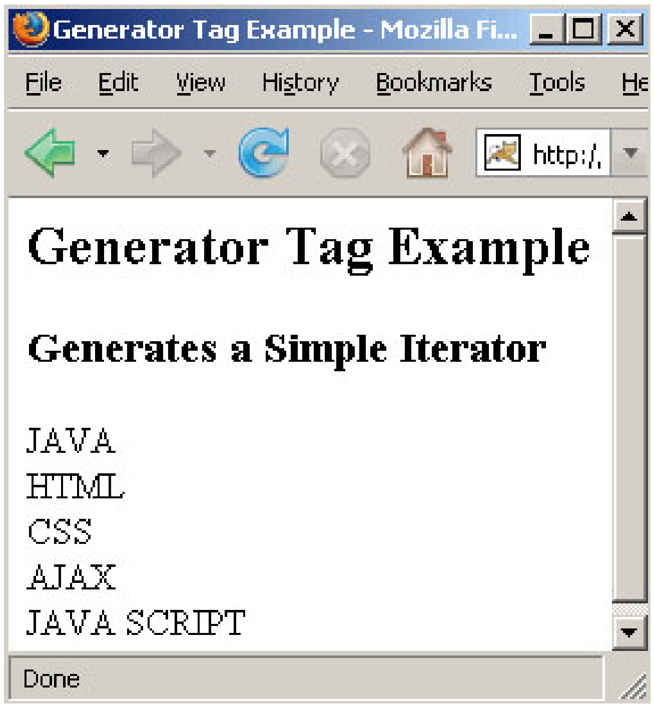
```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import
org.apache.struts2.util.IteratorGenerator.Converter;

public class GeneratorTag extends
ActionSupport {
  public String excute() throws Exception{
    return SUCCESS;
  }
}
```

Create a jsp page where the generator tag <s:generator> generates a simple iterator based on the val attribute supplied and <s:iterator> tag prints it out using the <s:property /> tag. The separator attribute is used to separate the val into entries of the iterator.

GeneratorTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title> Generator Tag Example </title>
  </head>
  <body>
  <h2>Generator Tag Example</h2>
    <h3>Generates a Simple Iterator</h3>
      <s:generator
val="%{'JAVA,HTML,CSS,AJAX,JAVA
SCRIPT'}" separator=",">
        <s:iterator>
          <s:property /><br/>
        </s:iterator>
      </s:generator>
  </body>
</html>
```

Output


Generator Tag Example - Mozilla Fi...
File Edit View History Bookmarks Tools He
http:/,
Done

```
Generator Tag Example

Generates a Simple Iterator

JAVA
HTML
CSS
AJAX
JAVA SCRIPT
```

6. Generator Tag (Control Tags) Using Count Attributes

In this section, we are going to describe the generator tag using the count attribute.

GeneratorTag.java

```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import
org.apache.struts2.util.IteratorGenerator.Converter;

public class GeneratorTag extends
ActionSupport {
  public String excute() throws Exception{
    return SUCCESS;
  }
}
```

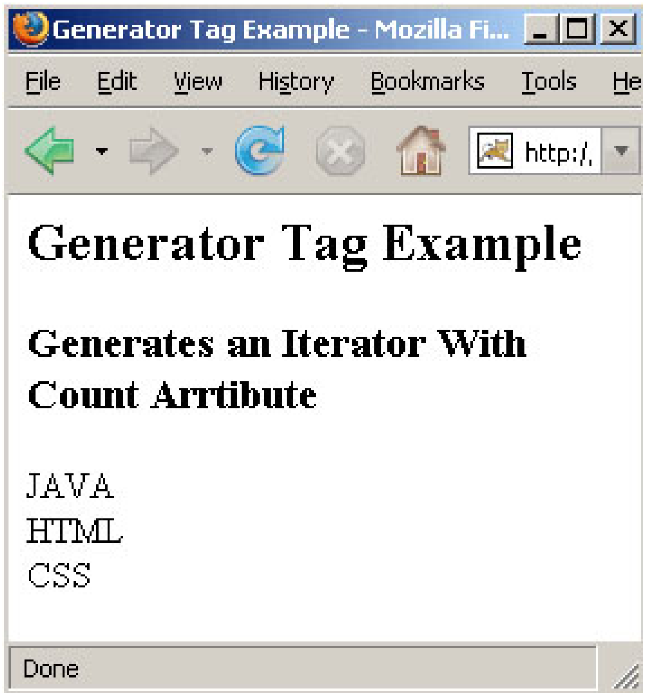
Create a jsp page where the generator tag <s:generator> generates an iterator with "count" attribute and <s:iterator> tag prints it out using the <s:property /> tag. The separator attribute separates the val into entries of the iterator.
This generates an iterator, but only 3 entries will be available in the iterator generated, namely "JAVA", "HTML", "CSS" respectively

because count attribute is set to 3.

GeneratorTagCountAttribute.jsp

```
<%@ taglib prefix=”s” uri=”/struts-tags” %>

<html>
  <head>
    <title> Generator Tag Example </title>
  </head>
  <body>
  <h2>Generator Tag Example</h2>
    <h3> Generates an Iterator With Count
Arrtibute</h3>
      <s:generator
val=”%{‘JAVA,HTML,CSS,AJAX,JAVA
SCRIPT’}” count=”3" separator=”,”>
        <s:iterator>
          <s:property /><br/>
        </s:iterator>
      </s:generator>
  </body>
</html>
```

Output:


Generator Tag Example - Mozilla Fi...
File Edit View History Bookmarks Tools He
http:/,
Done

Generator Tag Example

Generates an Iterator With Count Arrtibute

```
JAVA
HTML
CSS
```

7. Generator Tag (Control Tags) Using an Iterator with Id Attributes

In this section, we are going to describe the generator tag using the **id** attribute.

GeneratorTag.java

```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import
org.apache.struts2.util.IteratorGenerator.Converter;

public class GeneratorTag extends
ActionSupport {
  public String excute() throws Exception{
    return SUCCESS;
  }
}
```

Create a jsp page where the generator tag <s:generator> generates an iterator with “id” attribute and <s:iterator> tag prints it out using the <s:property /> tag. The separator attribute separates the val into entries of the iterator.

The scriplet as shown below generates an iterator and put it in the PageContext under the key as specified by the id attribute.

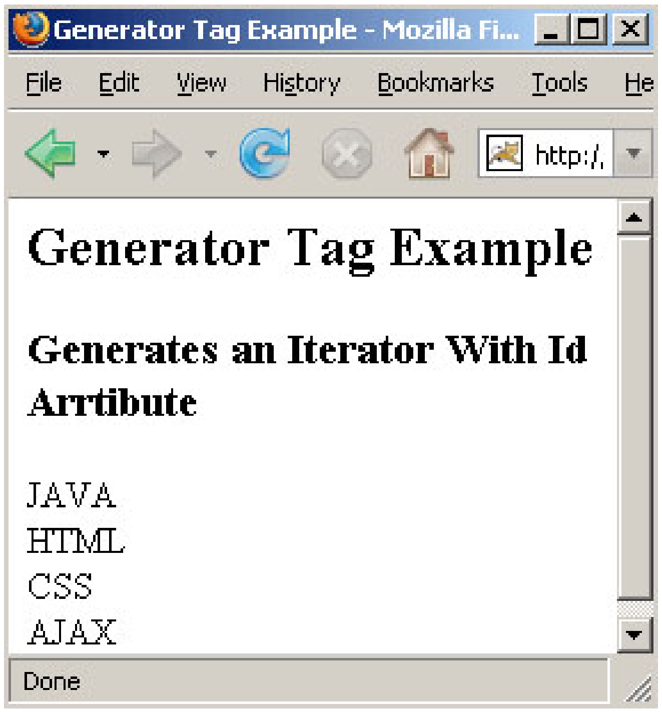
```
<%
      Iterator i = (Iterator)
pageContext.getAttribute(“myAtt”);
      while(i.hasNext()) {
        String s = (String) i.next(); %>
        <%=s%> <br/>
     <%   }
     %>
```

GeneratorTagIdAttribute.jsp

```
<%@ taglib prefix=”s” uri=”/struts-tags” %>
<%@page language=”java”
import=”java.util.*”%>

<html>
  <head>
    <title> Generator Tag Example </title>
  </head>
  <body>
  <h2>Generator Tag Example</h2>
    <h3> Generates an Iterator With Id
```

```
Arrtibute</h3>
      <s:generator
val="%{'JAVA,HTML,CSS,AJAX,JAVA
SCRIPT'}" count="4" separator=","
id="myAtt" />
      <%
        Iterator i = (Iterator)
pageContext.getAttribute("myAtt");
        while(i.hasNext()) {
          String s = (String) i.next(); %>
          <%=s%> <br/>
      <%   }
      %>
  </body>
</html>
```

Output:


Generator Tag Example - Mozilla Fi...
File Edit View History Bookmarks Tools He
http:/,
Done

Generator Tag Example

Generates an Iterator With Id Arrtibute

```
JAVA
HTML
CSS
AJAX
```

8. Subset Tag (Control Tags) Example
The subset tag is a generic tag that takes an iterator and outputs a subset of it. It delegates to org.apache.struts2.util.SubsetIteratorFilter internally to perform the subset functionality. Create a list in the action class and populate it with various items as shown in the "SubsetTag" class.

SubsetTag.java

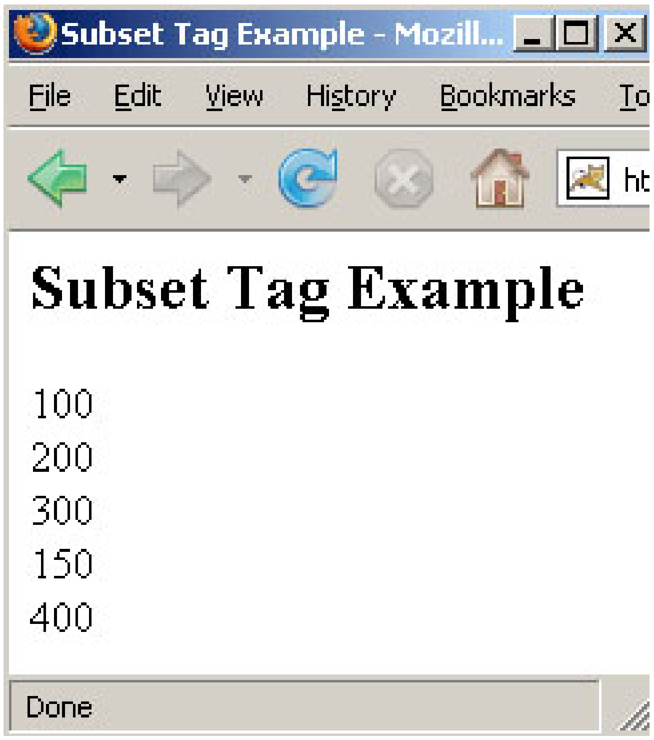
```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class SubsetTag extends
ActionSupport {
  private List list;

  public String execute() throws Exception{
    list = new ArrayList();
    list.add(new Integer(100));
    list.add(new Integer(200));
    list.add(new Integer(300));
    list.add(new Integer(150));
    list.add(new Integer(400));
    return SUCCESS;
  }

  public List getList(){
    return list;
  }
}
```

Now create a jsp page using <s:subset> and <s:iterator> tags as shown in the SubsetTag.jsp page. The subset tag takes an iterator and outputs a subset of it.

SubsetTag.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title>Subset Tag Example</title>
  </head>
  <body>
    <h2>Subset Tag Example</h2>
      <s:subset source="list">
        <s:iterator>
          <s:property /><br>
        </s:iterator>
      </s:subset>
  </body>
</html>
```

Output:


Subset Tag Example - Mozill...
File Edit View History Bookmarks To

Subset Tag Example

```
100
200
300
150
400
```

9. Subset Tag (Control Tags) Example Using Count
In this section, we are going to describe the subset tag using the **count** parameter. The count parameter indicates the number of entries to be set in the resulting subset iterator. Create a list in the action class and populate it with various items as shown in the "SubsetTag" class.

SubsetTag.java

```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class SubsetTag extends ActionSupport
{
  private List list;

  public String execute() throws Exception{
    list = new ArrayList();
    list.add(new Integer(100));
    list.add(new Integer(200));
    list.add(new Integer(300));
    list.add(new Integer(150));
    list.add(new Integer(400));
    return SUCCESS;
  }

  public List getList(){
    return list;
  }
}
```

Now create a jsp page using <s:subset> and <s:iterator> tags as shown in the SubsetTag.jsp page. The subset tag takes an iterator and outputs a subset of it. The parameter count is of integer type and it sets the number of entries to be kept in the resulting subset iterator.

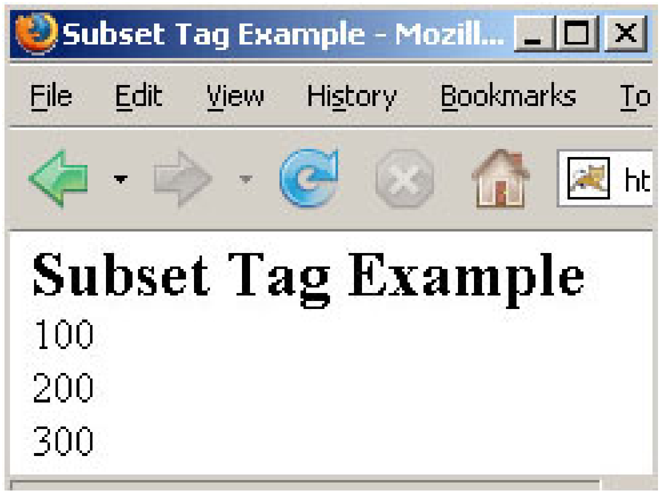
SubsetTagCount.jsp

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<html>
  <head>
    <title>Subset Tag Example</title>
  </head>
  <body>
    <h2>Subset Tag Example</h2>
      <s:subset source="list" count="3">
        <s:iterator>
          <s:property /><br>
        </s:iterator>
      </s:subset>
  </body>
</html>
```

Output:

Output displays only three items because count=3.

10. Subset Tag (Control Tags) Example Using Start
In this section, we are going to describe the subset tag using the **start** parameter. The **start** parameter is of integer type. It indicates the starting index (eg. first entry is 0) of entries in the source (needed to make available as the first entry in the resulting subset iterator). Create a list in the action class and populate it with various items as shown in "SubsetTag" class.


Subset Tag Example - Mozill...
File Edit View History Bookmarks To

```
Subset Tag Example
100
200
300
```

SubsetTag.java

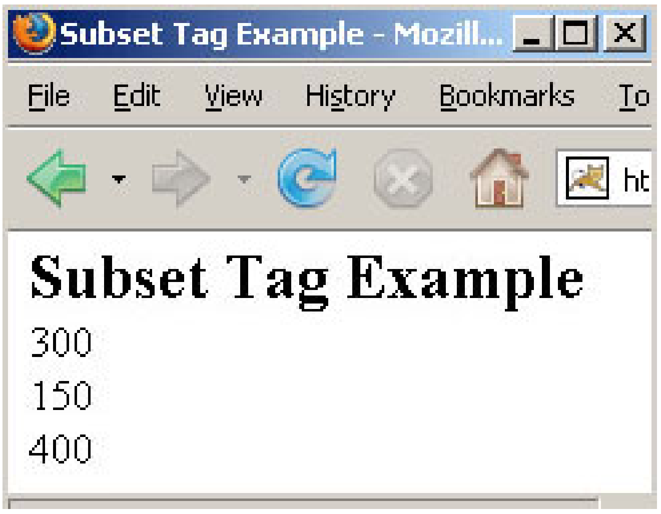
```
package net.javajazzup;
import
com.opensymphony.xwork2.ActionSupport;
import java.util.*;

public class SubsetTag extends ActionSupport
{
  private List list;

  public String execute() throws Exception{
    list = new ArrayList();
    list.add(new Integer(100));
    list.add(new Integer(200));
    list.add(new Integer(300));
    list.add(new Integer(150));
    list.add(new Integer(400));
    return  SUCCESS;
  }

  public List getList(){
    return list;
  }
}
```

Now create a jsp page using <s:subset> and <s:iterator> tags as shown in the SubsetTag.jsp page. The subset tag takes an iterator and outputs a subset of it. The parameter start is of integer type and it indicates the starting index (eg. first entry is 0) of entries in the source (needed to make available as the first entry in the resulting subset iterator).

SubsetTagStartWith.jsp

```
<%@ taglib prefix=”s” uri=”/struts-tags” %>

<html>
  <head>
    <title>Subset Tag Example</title>
  </head>
  <body>
    <h2>Subset Tag Example</h2>
      <s:subset source=”list” count=”3"
start=”2">
        <s:iterator>
          <s:property /><br>
        </s:iterator>
      </s:subset>
  </body>
</html>
```

Output:
Here the items displayed are 300, 150, 400. Items displayed starts from 2nd index of the List.


Subset Tag Example - Mozill...
File Edit View History Bookmarks To

```
Subset Tag Example
300
150
400
```

Integrating Struts and Hibernate

This article explains the integration of Struts and Hibernate, two worldwide-accepted frameworks in the landscape of development of enterprise applications. Completing this article will make you able to use Hibernate in your Struts project. We will be using Hibernate Struts plug-in to write high performance application using Struts and Hibernate.

Hibernate is Object-Oriented mapping tool that maps the object view of data into relational database and provides efficient persistence services such as create, read, update and delete (CRUD).

In this article we will implement small search engine application that shows a search form to the user. User enters search string and presses search button. Struts framework processes the request and passes it to the action for further processing. Action class uses Hibernate to make database calls and retrieves matching records from database and results are displayed to the user.

Full code for the application is provided with the article and you can download the zip file from any link at the bottom of every page of the article and start working on it for your project or to learn Struts and Hibernate Integration. Understanding the application has been divided in 6 sections as below.

1. **Setting up MySQL Database and table**
   This section describes how to setup database and populate it with the data. We are using MySQL Database for this application.

2. **Downloading Struts, Hibernate and Integrate It**
   This section explains you to setup the develop workbench for the application.

3. **Writing Hibernate configuration file, POJO class and Tutorial.hbm.xml (Hibernate mapping class)**
   In this section, we will write required hibernate objects like configuration files and then integrate all the stuffs.

4. **Developing Hibernate Struts Plugin**
   In this section, we will write Hibernate Struts Plugin Java code and integrate it with the Struts.

5. **Writing Web Client to Search the database using Struts Hibernate Plugin**
   In this section, we will write web client to test struts Plugin. We will be developing a search form to search the tutorials from the table.

6. **Build and testing the application**
   In this section, we will build our Struts Hibernate Plugin Application and then test.

1. Setting up MySQL Database and Tables

We are assuming that you have running instance of MySQL Database and you know how to work with the MySQL database. To access the database you should have valid user name and password.

Let's now start by creating the database for our struts- hibernate integration article. Our application is very very simple and it searches for the keywords typed by user in the table. The database will contain one table 'tutorials' for holding the tutorials links and descriptions. Here is the complete sql script for setting up the database.

```
CREATE  DATABASE `struts-hibernate` ;
CREATE TABLE `tutorials` (
`id` INT NOT NULL AUTO_INCREMENT ,
`shortdesc` VARCHAR( 50 ) NOT NULL ,
`longdesc` VARCHAR( 250 ) NOT NULL ,
`pageurl` VARCHAR( 100 ) NOT NULL ,
PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

The detail of each of the fields of the table above is given below:

id: Unique key for the table.

shortdesc: This fields stores the short description of the tutorial.
longdesc: This field stores the full description about the tutorial.
pageurl: This field is stores the url of the tutorial.

Run the following query to populate table with tutorials data:

INSERT INTO `tutorials` VALUES (1, 'JSP Tutorials, Hibernate and struts Tutorials', 'This site contains many quality Java, JSP Tutorials, Hibernate Tutorials, Struts Tutorials, JSF Tutorials, RMI, MySQL Tutorials, Spring Tutorials, source codes and links to other java resources. We have large number of links to the tutorials on java', 'http://roseindia.net/');

INSERT INTO `tutorials` VALUES (2, 'JSP Tutorial', 'Java Server Pages or JSP for short is Sun"s solution for developing dynamic web sites. JSP provide excellent server side scripting support for creating database driven web applications.', 'http://www.roseindia.net/jsp/jsp.shtml');

INSERT INTO `tutorials` VALUES (3, 'Struts Tutorials - Jakarta Struts Tutorial', 'This complete reference of Jakarta Struts shows you how to develop Struts applications using ant and deploy on the JBoss Application Server. Ant script is provided with the example code. Many advance topics like Tiles, Struts Validation Framework, Ja', 'http://www.roseindia.net/struts/index.shtml');

INSERT INTO `tutorials` VALUES (4, 'The Complete Spring Tutorial', 'Spring is grate framework for development of Enterprise grade applications. Spring is a light-weight framework for the development of enterprise-ready applications. Spring can be used to configure declarative transaction management, remote access to', 'http://www.roseindia.net/spring/index.shtml');

INSERT INTO `tutorials` VALUES (5, 'Java Server Faces (JSF) Tutorial', 'JavaServer Faces or JSF is grate technology for the development of user interfaces for web applications. The Java Server Faces specification is defined by JSR 127 of the Java Community Process.', 'http://www.roseindia.net/jsf/');
INSERT INTO `tutorials` VALUES (6, 'Jboss 3.0 Tutorial', ' This lesson shows you how to build you web application and install on the Jboss 3.0 application server. After the completion of this lesson you will be able to compile, assemble and deploy your J2EE application on Jboss 3.0 application server.', 'http://www.roseindia.net/jboss/index.shtml');

In the above steps we have setup our database. In the next section we will write java stuffs to integrate struts and hibernate.

2. Downloading Struts & Hibernate
In this explains downloading Struts & Hibernate and setting up the development environment.

Downloading Hibernate
Hibernate is free open source software it can be download from http://www.hibernate.org/. Visit http://www.hibernate.org/ and then click on the Download link to go to the download page. From the download page download the current latest release of Hibernate Core. For this article, we have downloaded Hibernate hibernate-3.1.1.zip.

Download Struts
The latest release of Struts can be downloaded from http://struts.apache.org/download.cgi. For this article we have downloaded struts-1.2.9-bin.zip. Save downloaded file into your hard disk.

Downloading Ant
Ant is a free tool under GNU License and is freely available at http://jakarta.apache.org/ant/. Ant allows the developer to automate the repeated process involved in the development of J2EE application. Developers can easily write the script to automate the

build process like compilation, archiving and deployment. For this article we are using apache-ant-1.6.5.

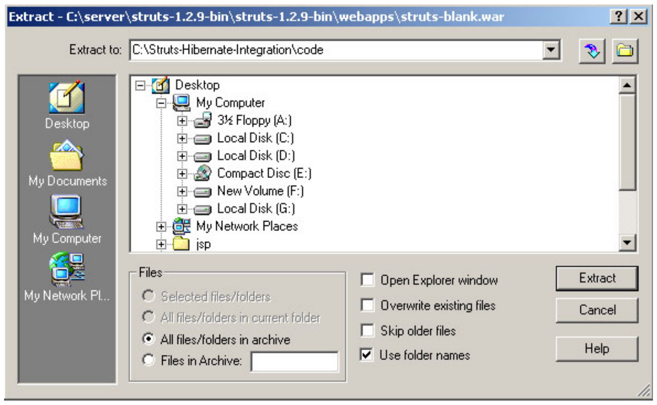
Download MySQL JDBC Driver
Download mysql-connector-java-3.0.16-ga-bin.jar from here mysql-connector-java-3.0.16-ga-bin.jar or you can download and use the latest version of mysql jdbc driver. Copy the JDBC driver file (mysql-connector-java-3.0.16-ga-bin.jar or latest version) to the **jakarta-tomcat-5.5.9\common\lib** directory of your tomcat installation. This will add the MySQL JDBC driver to the tomcat server.

Setting Up Development Environment
First we will create necessary directories and moved the required files to the appropriate directory. Follow the following steps to accomplish this:

1. Create a directory in your C: drive called **Struts-Hibernate-Integration**.

2. Unzip Downloaded file in the directory you have downloaded Struts.

3. Go to the "**struts-1.2.9-bin\webapps**" directory and you will find **struts-blank.war**, **struts-documentation.war**, **struts-examples.war**, **struts-mailreader.war** and **tiles-documentation.war** files in the directory. Open **struts-blank.war** with WinZip and then click on the "Extract" button. WinZip asks for the directory for extracting the file, enter "**C:\Struts-Hibernate-Integration**" and click on Extract button.



Extract - C:\server\struts-1.2.9-bin\struts-1.2.9-bin\webapps\struts-blank.war

Extract to: C:\Struts-Hibernate-Integration\code

Files: All files/folders in archive [✓]

Use folder names [✓]

4. A new directory will be created "**C:\Struts-Hibernate-Integration\code**" and the content of **struts-blank.war** is extracted in the code directory.

5. Now we will add the hibernate code to our development environment. Extract **hibernate-3.1.1.zip** in the directory where you have downloaded.

6. Copy "**hibernate3.jar**" from <your downoaded direvory>**hibernate-3.1** into **C:\Struts-Hibernate-Integration\code\WEB-INF\lib** directory.

7. Copy all the library files from "**hibernate-3.1\lib**" to "**C:\Struts-Hibernate-Integration\code\WEB-INF\lib**" directory.

8. Create a directory **libext** under "**C:\Struts-Hibernate-Integration\code\WEB-INF\**". We will use this directory to put extra jar files. Copy **servlet-api.jar** file your tomcat directory to "**C:\Struts-Hibernate-Integration\code\WEB-INF\libext**" directory.

9. Change in the build.xml file: Open "**C:\Struts-Hibernate-Integration\code\WEB-INF\src\build.xml**" file in your favourite editor and change as instructed below:

a) Find "**<property name="servlet.jar" value="/javasoft/lib/servlet.jar"/>**" in the build.xml file and change it to "**<property name="servlet.jar" value="./libext/servlet-api.jar"/>**"

b) Find "**<property name="distpath.project" value="/projects/lib"/>**" and change it to "**<property name="distpath.project" value="../../dist"/>**"

c) Change "**<property name="jdbc20ext.jar" value="/javasoft/lib/jdbc2_0-stdext.jar"/>**" to "**<property name="jdbc20ext.jar" value="./libext/jdbc2_0-stdext.jar"/>**".

d) Change "**<property name="project.title" value="Jakarta Struts Blank "/>**" to "**<property name="project.title" value="RoseIndia.net Struts Hibernate Integration Tutorial "/>**"

e) Change "**<property name="project.distname" value="blank"/>**" to "**<property name="project.distname" value="strutshibernate"/>**"

f) Change "**<target name="project" depends="clean,prepare,compile,javadoc"/>**" to "**<!—<target name="project" depends="clean,prepare,compile"/>—>**"

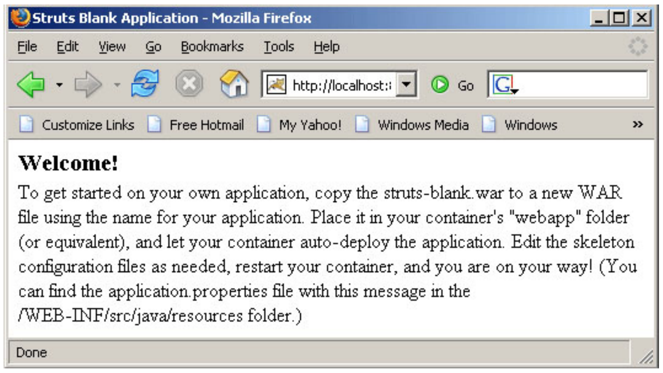
10. Open console and go to the "**C:\Struts-Hibernate-Integration\code\WEB-INF\src**" directory and type **ant** to compile the project. This show the following out put:

```
C:\Struts-Hibernate-Integration\code\WEB-
INF\src>ant
Buildfile: build.xml

clean:
[delete] Deleting directory C:\Struts-
Hibernate-Integration\code\WEB-INF\clas
ses
[mkdir] Created dir: C:\Struts-Hibernate-
Integration\code\WEB-INF\classes

prepare:

resources:
[copy] Copying 1 file to C:\Struts-Hibernate-
Integration\code\WEB-INF\class
es

compile:

project:

dist:
[jar] Building jar: C:\Struts-Hibernate-
Integration\dist\strutshibernate.j
ar
[war] Building war: C:\Struts-Hibernate-
Integration\dist\strutshibernate.war
[war] Warning: selected war files include a
WEB-INF/web.xml which will be
ignored (please use webxml attribute to war
task)

all:

BUILD SUCCESSFUL
Total time: 3 seconds
C:\Struts-Hibernate-Integration\code\WEB-
INF\src>
```

It will create **C:\Struts-Hibernate-Integration\dist\strutshibernate.war** file, which you can deploy on application server to test. *You can ignore the warning generated while running the ant build tool.*

11. To test the application copy the **strutshibernate.war** to your tomcat **webapps** directory and start tomcat.

12. Open browser and type http://localhost:8080/strutshibernate/. You browser page should look like:


Struts Blank Application - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:
Go
Customize Links
Free Hotmail
My Yahoo!
Windows Media
Windows
Done

Welcome!

To get started on your own application, copy the struts-blank.war to a new WAR file using the name for your application. Place it in your container's "webapp" folder (or equivalent), and let your container auto-deploy the application. Edit the skeleton configuration files as needed, restart your container, and you are on your way! (You can find the application.properties file with this message in the /WEB-INF/src/java/resources folder.)

This means you have successfully configured your development environment.

Writing Hibernate Configuration Files

In the previous section we completed the database setup and created required table and populated with the data. In this section we will write required hibernate configuration files. For this application, we need following Hibernate configuration files:

Hibernate Configuration File
Hibernate configuration file (**hibernate.cfg.xml**) is used to provide the information which is necessary for making database connections. The mapping details for mapping the domain objects to the database tables are also a part of Hibernate configuration file.

Here is the code of our Hibernate Configuration File:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//
EN"
"http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">
```

```
<hibernate-configuration>
<session-factory>
<property
name="hibernate.connection.driver_class">
com.mysql.jdbc.Driver</property>
<property
name="hibernate.connection.url">jdbc:mysql:/
/localhost/struts-hibernate</property>
<property
name="hibernate.connection.username">root</
property>
<property
name="hibernate.connection.password"></
property>
<property
name="hibernate.connection.pool_size">10</
property>
<property name="show_sql">true</
property>
<property
name="dialect">org.hibernate.dialect.
MySQLDialect</property>
<property
name="hibernate.hbm2ddl.auto">update</
property>
<!— Mapping files —>
<mapping resource="/roseindia/net/dao/
hibernate/Tutorial.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Place **hibernate.cfg.xml** file in the source directory e.g. **"C:\Struts-Hibernate-Integration\code\src\java"**

The <mapping resource=''> tag is used to specify the mapping file:

```
<mapping resource="/roseindia/net/dao/
hibernate/Tutorial.hbm.xml"/>
```

Code of Tutorial.hbm.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//
Hibernate/Hibernate Mapping DTD//EN"
"http://hibernate.sourceforge.net/hibernate-
mapping-3.0.dtd">

<hibernate-mapping auto-import="true"
default-lazy="false">

<class
name="roseindia.net.dao.hibernate.Tutorial"
table="tutorials"
>

<id
name="id"
type="java.lang.Integer"
column="id"
>
<generator class="increment" />
</id>

<property
name="shortdesc"
type="java.lang.String"
column="shortdesc"
not-null="true"
length="50"
/>
<property
name="longdesc"
type="java.lang.String"
column="longdesc"
not-null="true"
length="250"
/>
<property
name="pageurl"
type="java.lang.String"
column="pageurl"
```

```
not-null="true"
length="100"
/>

</class>
</hibernate-mapping>
```

Place **Tutorial.hbm.xml** file in the source directory e.g. **"C:\Struts-Hibernate-Integration\code\src\java\roseindia\net\dao\hibernate\"**.

POJO Object
Here is the code of Java Bean object (**Tutorial.java**) used to store and retrieve the data from database.

```
package roseindia.net.dao.hibernate;

import java.io.Serializable;


public class Tutorial implements Serializable {

    /** identifier field */
    private Integer id;

    /** persistent field */
    private String shortdesc;

    /** persistent field */
    private String longdesc;

    /** persistent field */
    private String pageurl;

    /** full constructor */
    public Tutorial(Integer id, String shortdesc,
String longdesc, String pageurl) {
        this.id = id;
        this.shortdesc = shortdesc;
        this.longdesc = longdesc;
        this.pageurl = pageurl;
    }

    /** default constructor */
    public Tutorial() {
    }

    public Integer getId() {
        return this.id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getShortdesc() {
        return this.shortdesc;
    }

    public void setShortdesc(String shortdesc)
{
        this.shortdesc = shortdesc;
    }

    public String getLongdesc() {
        return this.longdesc;
    }

    public void setLongdesc(String longdesc) {
        this.longdesc = longdesc;
    }

    public String getPageurl() {
        return this.pageurl;
    }

    public void setPageurl(String pageurl) {
        this.pageurl = pageurl;
    }

}
```

In this section we have created all the Hibernate related stuffs.

4. Developing Struts Hibernate Plugin
In this section, we will develop java code for Struts Hibernate Plugin. Our Hibernate Plugin will create Hibernate Session factory and cache it in the servlet context. This strategy enhances the performance of the application.

Source Code Of Hibernate Struts Plugin:

```
package roseindia.net.plugin;

import java.net.URL;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.apache.commons.logging.Log;
```

```
import
org.apache.commons.logging.LogFactory;
import
org.apache.struts.action.ActionServlet;
import org.apache.struts.action.PlugIn;
import
org.apache.struts.config.ModuleConfig;
import org.hibernate.HibernateException;


public class HibernatePlugIn implements PlugIn
{
   private String _configFilePath = "/
hibernate.cfg.xml";

    /**
     * the key under which the
<code>SessionFactory</code> instance is
stored
     * in the <code>ServletContext</code>.
     */
    public static final String
SESSION_FACTORY_KEY
            = SessionFactory.class.getName();

  private SessionFactory _factory = null;

   public void destroy() {
     try{
       _factory.close();
     }catch(HibernateException e){
      System.out.println("Unable to close
Hibernate Session Factory: " +
e.getMessage());
     }

   }

   public void init(ActionServlet servlet,
ModuleConfig config) throws ServletException
{
   System.out.println("********************
****************");
  System.out.println("**** Initilizing
HibernatePlugIn   **********");
        Configuration configuration = null;
        URL configFileURL = null;
        ServletContext context = null;

   try{
          configFileURL =
HibernatePlugIn.class.getResource
(_configFilePath);
context = servlet.getServletContext();
configuration = (new
Configuration()).configure(configFileURL);
_factory =
configuration.buildSessionFactory();
//Set the factory into session
context.setAttribute(SESSION_FACTORY_KEY,
_factory);

}catch(HibernateException e){
 System.out.println("Error while initializing
hibernate: " + e.getMessage());
   }
   System.out.println("*****************
*******************");

   }

    /**
     * Setter for property configFilePath.
    * @param configFilePath New value of
property configFilePath.
     */
    public void setConfigFilePath(String
configFilePath) {
        if ((configFilePath == null) ||
(configFilePath.trim().length() == 0)) {
throw new IllegalArgumentException(
 "configFilePath cannot be blank or null.");
}

System.out.println("Setting 'configFilePath' to
'"  + configFilePath + "'...");
 _configFilePath = configFilePath;
    }


/*(SessionFactory)
servletContext.getAttribute
(HibernatePlugIn.SESSION_FACTORY_KEY);
*/
}
```

In our plugin class we have define a variable **_configFilePath** to hold the name of Hibernate Configuration file.

```
private String _configFilePath = "/
hibernate.cfg.xml";
```

Following code define the key to store the session factory instance in the Servlet context.

```
public static final String
SESSION_FACTORY_KEY =
SessionFactory.class.getName();
```

The init() is called on the startup of the Struts Application. On startup the session factory is initialized and cached in the Servlet context.

```
configFileURL = HibernatePlugIn.class.
getResource(_configFilePath);
context = servlet.getServletContext();
configuration = (new Configuration()).
configure(configFileURL);
_factory =
configuration.buildSessionFactory();
//Set the factory into session
context.setAttribute(SESSION_FACTORY_KEY,
_factory);
```

Changes to be done in struts-config.xml file

Configuring Hibernate with Struts is very simple work it requires you to have hibernate.cfg.xml in your WEB-INF/classes directory, and to add the following line to the struts-config.xml file.

```
<plug-in
className=”roseindia.net.plugin.HibernatePlugIn”>
</plug-in>
```

Testing the Plugin

Build your application and deploy on the tomcat server. Start tomcat server and observe the console output. It should display following lines:

```
log4j:WARN Please initialize the log4j system
properly.
************************************
**** Initilizing HibernatePlugIn **********
************************************
Nov 27, 2007 10:09:53 AM
org.apache.struts.tiles.TilesPlugin initD
```

This means you have successfully configured your Struts Hibernate Plugin with struts application.

5. Developing Struts Web Module
In this section, we will be creating search interface for enabling the user to search tutorials. This example is a client to test our Struts Hibernate Plugin.

The web component of the application consists of the following files:

1. **Search Tutorial Form (SearchTutorial.jsp):**

This file is used to display the search form to the user. Here is the code of search form:

```
<%@ taglib uri=”/tags/struts-bean”
prefix=”bean” %>

   <%@ taglib uri=”/tags/struts-html”
prefix=”html” %>
  <html:html locale=”true”>
   <head>
<title>
<bean:message key=”welcome.title”/></title>
  <html:base/>
  </head>
  <body bgcolor=”white”>

  <html:form action=”/searchTutorial”>

  <html:errors/>

  <table>

    <tr>
        <td align=”right”>
          Search Tutorial
         </td>
        <td align=”left”>
<html:text property=”keyword” size=”30"
maxlength=”30"/>
         </td>
       </tr>
    <tr>
        <td align=”right”>
 <html:submit>Search</html:submit>
         </td>
      </tr>
    </table>
  </html:form>
  </body>
  </html:html>
```

Save **SearchTutorial.jsp** in to **“C:\Struts-Hibernate-Integration\code\pages”** directory.

2. Search Result Page (SearchResultPage.jsp)
This page is used to display the search result. Here is the code of search result page:

```
<%@page language=”java”
import=”java.util.*”%>
<%@ taglib uri=”/tags/struts-bean”
prefix=”bean” %>
<%@ taglib uri=”/tags/struts-html”
prefix=”html” %>
<p><font size=”4" color=”#800000"
face=”Arial”>Search Results</font></p>

<%
List searchresult = (List)
request.getAttribute(“searchresult”);
 %>
 <%
for (Iterator itr=searchresult.iterator();
itr.hasNext(); )
  {
  roseindia.net.dao.hibernate.Tutorial tutorial
=
(roseindia.net.dao.hibernate.Tutorial)itr.next();
  %>
  <p>
<a href=”<%=tutorial.getPageurl()%>”>
<font face=”Arial” size=”3">
<%=tutorial.getShortdesc()%></font>
</a><br>
<font face=”Arial” size=”2">
<%=tutorial.getLongdesc()%></font></p>

        <%
        }
        %>
<html:link page=”/pages/SearchTutorial.jsp”>
Back to Search Page</html:link>
```

Save **SearchResultPage.jsp** in to **“C:\Struts-Hibernate-Integration\code\pages”** directory.

3. Search Java Form
(SearchTutorialActionForm.java)
This is the Struts action form class. Here is the code of the Action Form:

```
package roseindia.web;

import javax.servlet.http.HttpServletRequest;

import org.apache.struts.action.*;

public class SearchTutorialActionForm
extends  ActionForm
{
  private String keyword=null;

public void setKeyword(String keyword){
    this.keyword=keyword;
  }

  public String getKeyword(){
    return this.keyword;
  }

   public void reset(ActionMapping mapping,
HttpServletRequest request) {
    this.keyword=null;
    }
   public ActionErrors validate(
      ActionMapping mapping,
HttpServletRequest request ) {
      ActionErrors errors = new ActionErrors();

     if( getKeyword() == null ||
getKeyword().length() < 1 ) {
        errors.add(“keyword”,new
ActionMessage(“error.keyword.required”));
     }

      return errors;
  }

}
```

4. Search Action Class
(SearchTutorialAction.java)

This is Struts Action Class of our application. Here is the code of the Action Class:

```
package roseindia.web;

import javax.servlet.http.HttpServletRequest;
```

```
import
javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletContext;

import java.util.List;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import
org.apache.struts.action.ActionForward;
import
org.apache.struts.action.ActionMapping;

import roseindia.net.plugin.HibernatePlugIn;

import roseindia.net.dao.hibernate.Tutorial;

import org.hibernate.SessionFactory;
import org.hibernate.Session;
import org.hibernate.criterion.Restrictions;
import org.hibernate.Criteria;



public class SearchTutorialAction extends
Action
{
  public ActionForward execute(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response) throws
Exception{
  SearchTutorialActionForm formObj =
(SearchTutorialActionForm)form;

   System.out.println("Getting session
factory");
   /*Get the servlet context */
   ServletContext context =
request.getSession().getServletContext();
   /*Retrieve Session Factory */
   SessionFactory _factory = (SessionFactory)
    context.getAttribute
(HibernatePlugIn.SESSION_FACTORY_KEY);
   /*Open Hibernate Session */
   Session  session = _factory.openSession();
    //Criteria Query Example
    Criteria crit =
session.createCriteria(Tutorial.class);
     crit.add(Restrictions.like("shortdesc", "%"
+ formObj.getKeyword() +"%")); //Like
condition

   //Fetch the result from database
   List tutorials= crit.list();
  request.setAttribute("searchresult",tutorials);

    /*Close session */
     session.close();
    System.out.println("Hibernate Session
Closed");
      return mapping.findForward("success");
  }
}
```

5. Entries into struts-config.xml

Add the following lines into your struts-config.xml file.

```
Form Bean:
<form-bean
name="TutorialSearch"
type="roseindia.web.SearchTutorialActionForm">
</form-bean>

Action Entry:
<action
path="/searchTutorial"
type="roseindia.web.SearchTutorialAction"
name="TutorialSearch"
scope="request"
validate="true"
input="/pages/SearchTutorial.jsp">
<forward name="success" path="/pages/
SearchResultPage.jsp"/>
</action>
```

Now we have created all the required stuffs for the web client. In the next section we will test our application.

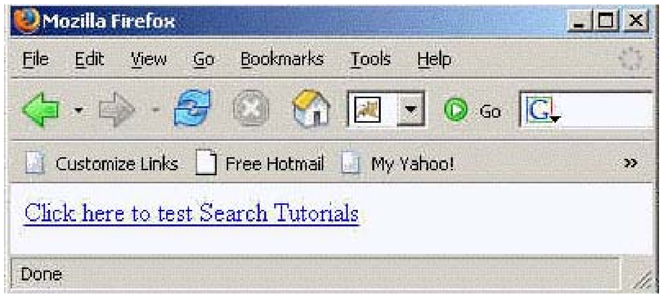
6. Building and Testing Struts Hibernate Plugin Application
In this section we will build and test our Struts Hibernate Integration application.

Compiling and packaging application
Since we are using ant build tool, so the compiling and packaging will done by ant tool. To compile and create war file for deployment, open console and go to **"C:\Struts-**

Integrating Struts and Hibernate

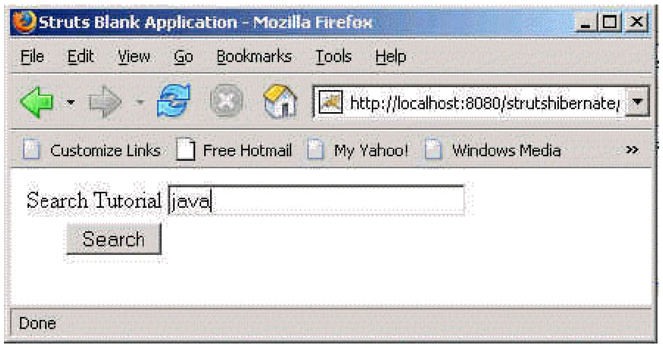
Hibernate-Integration\code\WEB-INF\src" directory. Then just type ant, ant will create **strutshibernate.war** in the **"C:\Struts-Hibernate-Integration\dist"** directory.

Deploying and testing application
Copy **strutshibernate.war** to the **webapps** directory of tomcat and start tomcat server. Now open the browser and type **http://localhost:8080/strutshibernate/** in the browser. You browser should look like:



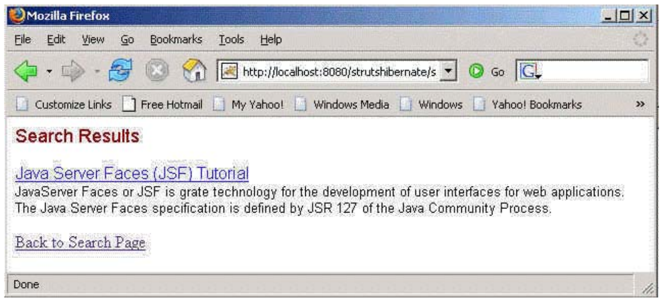
Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
Customize Links Free Hotmail My Yahoo!
Click here to test Search Tutorials
Done

Now click on "Click here to test Search Tutorials" link.



Struts Blank Application - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/strutshibernate/
Customize Links Free Hotmail My Yahoo! Windows Media
Search Tutorial java
Search
Done

Enter some search term say "java" and click on search button. You browser should display the search result as shown below.



Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/strutshibernate/s
Go
Customize Links Free Hotmail My Yahoo! Windows Media Windows Yahoo! Bookmarks
Done

Search Results

Java Server Faces (JSF) Tutorial
JavaServer Faces or JSF is grate technology for the development of user interfaces for web applications.
The Java Server Faces specification is defined by JSR 127 of the Java Community Process.

Back to Search Page

Congratulations now you have successfully integrated your struts application with hibernate using Hibernate Struts Plugin.

Custom Converter Example in JSF

JSF provides a very flexible architecture that not only let the developers use converters provided already by the implementation but also create their own converters according to the requirement of the application. This topic explains about how to create custom converter. When the user enters value to the component, it's simply a string value. Now you may be in the need of using this value as a different object like Boolean, Date etc. Converters can help in this conversion. JSF framework has provided many converters like Boolean Converter, Byte Converter, Number Converter etc. These converters convert values into appropriate type of object and return it also to the page in the appropriate format. JSF flexible architecture provides you freedom to create your own converters. These can be used to check the value in the correct format. For example, In our application user is provided an input box to fill time in "hours:minutes:seconds" format. This String is converted as Object by the converter and also converted back in String when it needs to display in the web page. Now if the user doesn't fill time in correct format then it displays error message showing the conversion could not be successful.

To create custom converter you need to implement **"Converter"** interface of **javax.faces.converter** package in your class.

Steps to follow:

1. Create a class that implements **javax.faces.converter.Converter** interface.
2. Import necessary packages and classes.
3. Implement two abstract classes "**getAsObject()**", "**getAsString()**" provided by Converter interface. **getAsObject()** method converts the String (User Input) to Object and **getAsString()** method converts the Object to String to send back to the page.
4. Configure the configuration file (**faces-config.xml**) adding **<converter>** element. This element has child elements **<converter-id>** (name of the converter to be used while programming) and **<converter-class>** (name of the converter class).
5. Create view page where **<f:converter>** tag is used with attribute **"converterId"** which specifies the name of the converter specified in **<converter-id>** element of **<converter>** element in **"faces-config.xml"** file.
6. Use **<h:message>** tag to display the error message.

The steps above have been implemented in our application "**customconverter**". This will help you to understand the process of creating custom converter. Just go through the following steps:

Step1: Create a class "**TimeConverter**" that implements the **Converter** interface and implements two abstract methods **"getAsObject()"**, **"getAsString()"**. Save this file as **"TimeConverter.java"** in **WEB-INF/classes** directory of your application in Tomcat server.

TimeConverter.java

```
import javax.faces.component.UIComponent;
import  javax.faces.context.FacesContext;
import javax.faces.convert.*;
import javax.faces.application.*;

public class TimeConverter implements
Converter {
   public TimeConverter() {
   }

public Object getAsObject(FacesContext
facesContext, UIComponent uiComponent,
String param){
try {
String hr_mi_se[] = param.split(":");
int seconds =
Integer.parseInt(hr_mi_se[0])*60*60 +
Integer.parseInt(hr_mi_se[1])*60+
Integer.parseInt(hr_mi_se[2]);
Integer sObject= new Integer(seconds);
```

```
          return sObject;
           }
      catch (Exception exception) {
           throw new
ConverterException(exception);
           }
    }

    public String getAsString(FacesContext
facesContext,
                              UIComponent
uiComponent,
                              Object obj) {
      try {
          int total_seconds =
(int)((Integer)obj).intValue();
          int hours=(total_seconds)/(60*60);
          int rem=(total_seconds)%(60*60);
          int minutes=rem/60;
          int seconds=rem%60;
          String str_hours=""+hours;
          String str_minutes=""+minutes;
          String str_seconds=""+seconds;

          if(hours<10){
          str_hours="0"+hours;
           }
          if(minutes<10){
          str_minutes="0"+minutes;
           }
          if(seconds<10){
          str_seconds="0"+seconds;
           }
           return str_hours + ":" + str_minutes
+ ":" + str_seconds;

        }
      catch (Exception exception) {
          throw new
ConverterException(exception);
         }
    }
}
```

In this class **"param"** represents the string provided by the user in the component. This string is passed to the **getAsObject()** method. Now we can use this according to our requirement of manipulation and return the appropriate object. **"obj"** parameter passed in **getAsString()** method represents the converted object in the previous method. This method is called while displaying in the page. So return the appropriate String by manipulating this object. If there is any problem in this process we can handle it by try and catch block. An error message is shown to the current page if conversion is not successful.

Step2: Configure the configuration file (**faces-config.xml**). Open this file and add the following code.

```
<?xml version="1.0"?>
<faces-config>
   <converter>
     <converter-id>TimeConverter</converter-
id>
     <converter-class>TimeConverter</
converter-class>
   </converter>
</faces-config>
```

Here **<converter-id>** gives ID to the converter that will be used in our page and **<converter-class>** specifies the implementing class.

Step3: Now, we can code for the page **"converter.jsp"** where **<f:converter>** tag is used to associate the converter to the component using **converterId** attribute. Value of this attribute is matched with the ID of the converter specified in the configuration file.

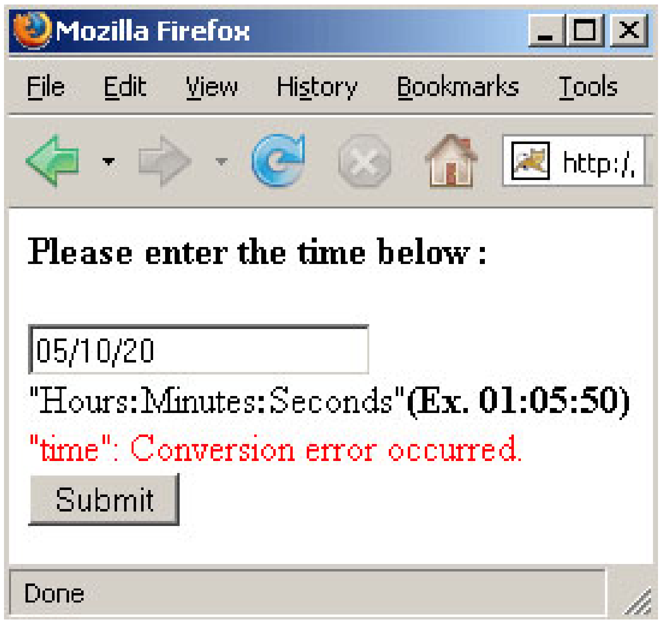
converter.jsp:

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

<f:view>
 <html>
   <body>
   <h:form>
<b>Please enter the time below :</b><br>
</br>
    <h:inputText id="time">
```

JSF Application

```
<f:converter  converterId="TimeConverter"
/>
</h:inputText><br/>"Hours<b>:
</b>Minutes<b>:</b>Seconds"<b>(Ex.
01:05:50)</b></br>

<h:message for="time" style="color:RED"/
></br>
   <h:commandButton value="Submit"/>
    </h:form>
   </body>
 </html>
</f:view>
```

Output: When user fills wrong input, it displays error message in the current page as it is shown below otherwise processes according to the logic of the application.



Please enter the time below :

05/10/20

"Hours:Minutes:Seconds"**(Ex. 01:05:50)**

"time": Conversion error occurred.

Behavioral Patterns

Behavioral patterns are those patterns, which are specifically concerned with communication (interaction) between the objects. The interactions between the objects should be such that they are talking to each other and are still loosely coupled. The loose coupling is the key to **n-tier** architectures. In this, the implementations and the client should be loosely coupled in order to avoid hard coding and dependencies. The behavioral patterns are:

1. Chain of Responsibility Pattern
2. Command Pattern
3. Interpreter Pattern
4. Iterator Pattern
5. Mediator Pattern
6. Momento Pattern
7. Observer Pattern
8. State Pattern
9. Strategy Pattern
10. Template Pattern
11. Visitor Pattern

Some of the important and useful pattern which are frequently used in java are as under:

Chain of Responsibility pattern:

As per concern the Object Oriented Design, it is a design pattern based on chain that consists a source of command objects and a series of processing objects. In this pattern, each processing object holds a set of logic, which describes the kinds of command objects. These Objects are in the form of a chain that forwards the request until one of the objects handles the event.

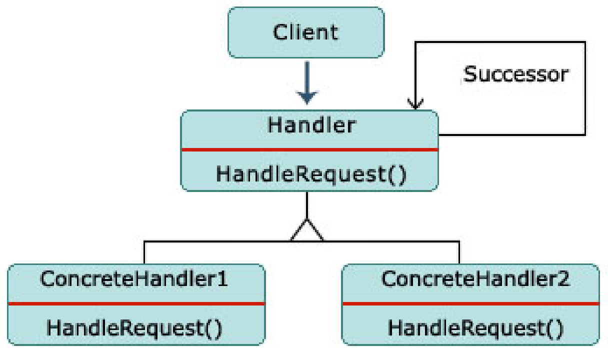
In other words, this pattern decouples the sender who sends the requested data. The receiver or any members of the "chain" pick that data. If the first link of the chain cannot handle the responsibility, it passes the request data to the next level in the chain, i.e. to the next link until one of the objects handles the event. This is similar to concepts Exception Hierarchy in Java. This technique is called "**data-driven**". In most of the behavioral patterns, the data-driven concepts are used to have a loose coupling. There is a mechanism that also exists for including a new processing object to the end of this chain.

The Chain of Responsibility pattern is applicable if:

1. We want to decouple a request's sender and receiver.
2. We don't want to assign handlers explicitly in our code.
3. Multiple objects are nominees to handle a request at runtime.

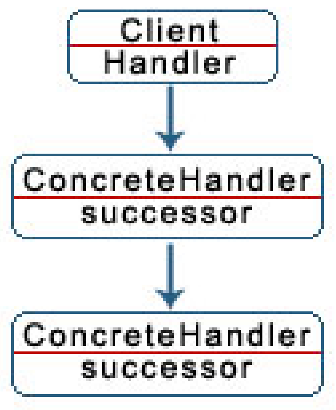
There are some key points to remember when we use the Chain of responsibility pattern like only one object in the chain handles a request. There are also some possibilities for some requests might not get handled.

"Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it."



CoR class diagram

A typical object structure of chain of responsibility is shown as:



CoR object structure

Request handlers are extensions of a base class that maintains a reference to the next handler in the chain. This handler is known as the **successor**. The code of the base class implementing **handleRequest()** method is shown as:

```
  public abstract class BaseHandler {
        ...
        public void
handleRequest(SomeRequestObject sro) {
            if(successor != null)
                  successor.handleRequest(sro);
        }
  }
```

Thus, handlers pass the request to the next handler in the chain by default. A concrete class of **BaseHandler** might be implemented like this:

```
public class SpamFilter extends BaseHandler {
      public void
handleRequest(SomeRequestObject
mailMessage) {
          if(isSpam(mailMessage))   { // If the
message is spam
              // take spam-related action. Do not
forward message.
 }
 else { // Message is not spam.
super.handleRequest(mailMessage);
// Pass message to next filter in the chain.
}
 }  }
```

The class **SpamFilter** handles the request receiving a new email object. If the message is spam then the request goes no further; otherwise, messages are passed to the next handler. Finally, the last filter in the chain might store the message after moving through several filters.

Now we can summarize the characteristics of the Chain of responsibility design pattern. In the CoR pattern multiple handlers are able to handle a request but only one handler actually handles the request because the requester knows the reference of only one handler. The requester also doesn't know that how many handlers are able to handle its request as well as it also doesn't know which handler handled its request. The handler could be specified dynamically. While CoR changing the handlers list will not affect the requester's code.

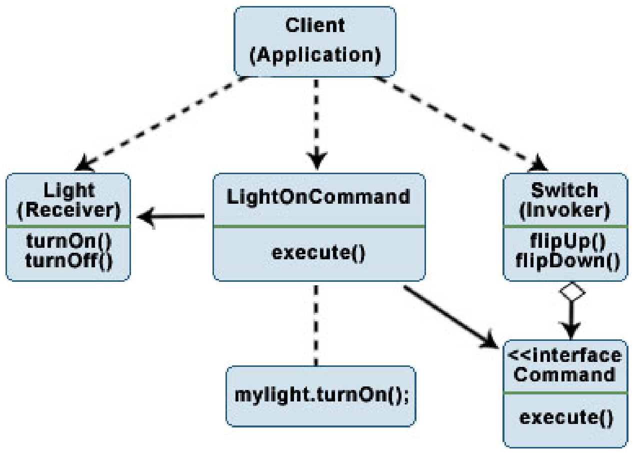
The Command Pattern

In the command pattern the client invokes a particular module with the help of a command. In this type of design pattern, objects are used to represent actions; the client passes a request that gets distributed as a command. The command request maps to particular modules and on behalf of the command the module get invoked.

The command pattern is different from the Chain of Responsibility pattern in a way that, in Chain of Responsibility forwards requests by each of the classes before finding an object that can take the responsibility but the command pattern finds the particular object according to the command and invokes only that one.

It encloses a request for a specific action inside an object and provides it a known public interface. It makes the client able to make requests without knowing anything about the actual action as well as allows us to change that action without affecting the client program in any way.

In a simple way we can say that command pattern is work just like a server that has a lot of services to be given, and on command, it serves to that service for that particular client.



Let us consider an example of a restaurant, which is based on command pattern structure. Where the first step is the order from the customer to the waiter for the food. In the next command or step waiter takes the order and forwards the order to the cook in the kitchen. The cook can make various types of food but now he prepares the ordered item and hands it over to the waiter who in turn serves to the customer.

Let us consider this example in term of java code. As we know that the first step is the order. The order is made of command in which the customer gives the order for the food to the waiter.
Order.java

```
package bahavioral.command;
public class Order {

private String cmd;
public Order(String cmd) {
this.cmd = cmd;
}
}
```

The next step is belonging to the waiter who takes the order and forwards it to the cook. After that waiter calls the prepareFood method of the cook who in turn cooks.

Waiter.java

```
package bahavioral.command;
public class Waiter {

public Food takeOrder(Customer cust, Order
ord) {

Cook cook = new Cook();
Food food = cook.prepareOrder(ord, this);
return food;
}
}
```

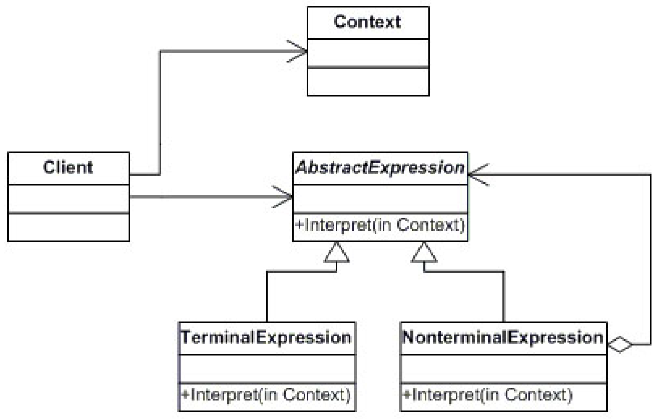
The waiter takes command and wraps it in an order; the order is linked to a specific customer. For, the cook, the order is linked to a cook and also Food is linked to the Order.

Cook.java

```
package bahavioral.command;
public class Cook {

public Food prepareOrder(Order ord, Waiter
wtr) {
Food food = getCookedFood(ord);
return food;
}
public Food getCookedFood(Order ord) {
Food food = new Food(ord);
return food;
}
```

Here, the waiter takes command and wraps it in an order; the order is associated to a particular customer. The order is associated to a cook and also Food is associated to the Order.
The order is an object, which depends on the command. The food item will change as soon as the command changes. This is loose coupling between the client and the implementation.
Interpreter Pattern:
The Interpreter Pattern is a design pattern that defines a grammatical representation for a language along with an interpreter to interpret sentences in the language. The best example of an interpreted language is Java itself, which converts the English-written code to a byte code format, so that all the operating systems can understand it.

The UML class diagram of Interpreter design pattern can be shown as:



In the given diagram, An **abstract** base class specifies the method **interpret()**. Each **concrete** subclass implements interpret() by accepting (as an argument) the current state of the language stream, and adding its contribution to the problem solving process.

To make this thing clear, let's take an example that can take the Sa, Re, Ga, Ma etc and produce the sounds for the frequencies. The "**musical notes**" is an Interpreted Language. The musicians read the notes, interpret them according to "Sa, Re, Ga, Ma…" or "Do, Re, Me… ", etc.

For Sa, the frequency is 256 Hz, similarly, for Re, it is 288Hz and for Ga, it is 320 Hz and so on. In this case, we need these values set somewhere, so that when the system encounters any one of these messages, the related frequency can be sent to the instrument for playing the frequency.

We can have it at one of the two places, one is a constants file, "token=value" and the other one being in a properties file. The properties file can give us more flexibility to change it later if required.

This is how a properties file will look like:

MusicalNotes.properties

```
Sa=256
Re=288
Ga=320
Ma=352
. . . . .
```

After that we make a class NotesInterpreter.java to take an input from the key pressed by user and set those value as a global value. Then we make a method getFrequency for getting the frequency for the note input by the user e.g. if user enter Re, it will return 288.

NotesInterpreter.java

```
package bahavioral.interpreter;
public class NotesInterpreter {

Private Note note;

public void getNoteFromKeys(Note note) {
Frequency freq = getFrequency(note);
sendNote(freq);
}

private Frequency getFrequency(Note note) {

return freq;
}

private void sendNote(Frequency freq) {
NotesProducer producer = new
NotesProducer();
producer.playSound(freq);
}
}
```

Here we need to make another class in which the method produces the sound wave of the frequency it gets.

NotesProducer.java

```
package bahavioral.interpreter;
public class NotesProducer {

Private Frequency freq;

public NotesProducer() {
this.freq = freq;
}
public void playSound(Frequency freq) {
} }
```

The above example is the simplest way to understand the concept of interpreter pattern and know how it works. In case of Interpreter pattern we need to check for grammatical mistakes, which makes it very complex. Also, care should be taken to make the interpreter as flexible as possible, so that the implementation can be changed at later stages without having tight coupling.

Tips & Tricks

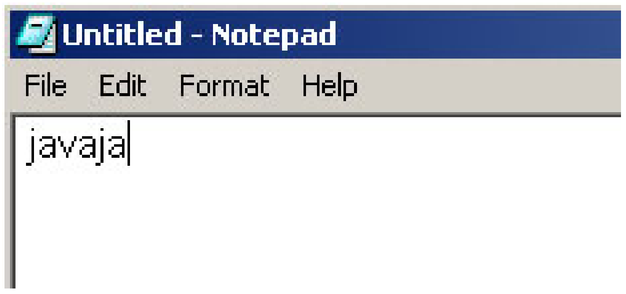
1.Opening, writing, saving notepad automatically with Robot class in Java

Java provides a lot of fun while programming. This article shows you how to use Java.awt.Robot class that is both useful and fun. Robot class is used to take the control of mouse and keyboard, which lets you do any mouse and keyboard related operation through java code for the purposes of test automation, self-running demos.
The sample code given below demonstrates handling of the keyboard events. When this program is run, it does a series of events automatically. It opens notepad and types "javajazzup" in it and saves this file as "robo.txt" automatically.

RobotExample.java

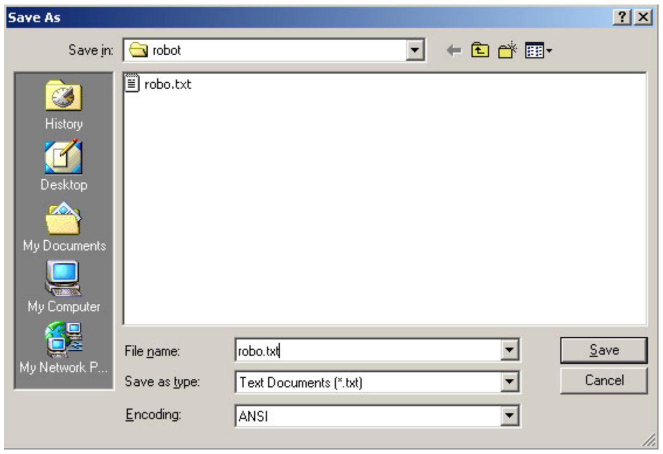
```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;

public class RobotExample {
//Store Keystrokes in an array for the
content in notepad
static int keyInputForNotepadContent[] =
{KeyEvent.VK_J,KeyEvent.VK_A,

KeyEvent.VK_V,KeyEvent.VK_A,KeyEvent.VK_J
,KeyEvent.VK_A,KeyEvent.VK_Z,

KeyEvent.VK_Z,KeyEvent.VK_U,KeyEvent.VK_P};
//Store Keystrokes in an array to give
name of the notepad
static int keyInputForNotepadName[] =
{KeyEvent.VK_R,KeyEvent.VK_O,
KeyEvent.VK_B,KeyEvent.VK_O,
KeyEvent.VK_DECIMAL,KeyEvent.VK_T,
KeyEvent.VK_X,KeyEvent.VK_T};
   public static void main(String[] args) {
    try {
      Robot robot = new Robot();
//Press keys Ctrl+Esc and then key R to
open "run" program.

robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_ESCAPE);

robot.keyRelease(KeyEvent.VK_CONTROL);
robot.keyRelease(KeyEvent.VK_ESCAPE);
robot.keyPress(KeyEvent.VK_R);
//Type NOTEPAD in run program and
press ENTER ket to open a notepad.
      robot.keyPress(KeyEvent.VK_N);
      robot.keyPress(KeyEvent.VK_O);
      robot.keyPress(KeyEvent.VK_T);
      robot.keyPress(KeyEvent.VK_E);
      robot.keyPress(KeyEvent.VK_P);
      robot.keyPress(KeyEvent.VK_A);
      robot.keyPress(KeyEvent.VK_D);
      robot.keyPress(KeyEvent.VK_ENTER);
      robot.delay(1000);
// Creates the delay of 1 sec

//Press keys of characters to write on
the notepad.
for (int i = 0; i <
keyInputForNotepadContent.length; i++){

robot.keyPress(keyInputForNotepadContent[i]);
        robot.delay(500);
      }
      robot.delay(1000);

      // Press keys ALT+F+S to save the file
      robot.keyPress(KeyEvent.VK_ALT);
      robot.keyPress(KeyEvent.VK_F);
      robot.keyPress(KeyEvent.VK_S);
      robot.keyRelease(KeyEvent.VK_ALT);
      robot.delay(500);

//Press keys of characters to write the name
of notepad.
for (int i = 0; i <
keyInputForNotepadName.length; i++){

robot.keyPress(keyInputForNotepadName[i]);
robot.delay(500);
}
robot.keyPress(KeyEvent.VK_ENTER);
// Replace the file if it already exists.
robot.keyPress(KeyEvent.VK_ALT);
robot.keyPress(KeyEvent.VK_Y);
robot.keyRelease(KeyEvent.VK_ALT);
}
catch (AWTException e) {
e.printStackTrace();
       }
    }
}
```

After opening a notepad it starts writing "javajazzup" in it like below:

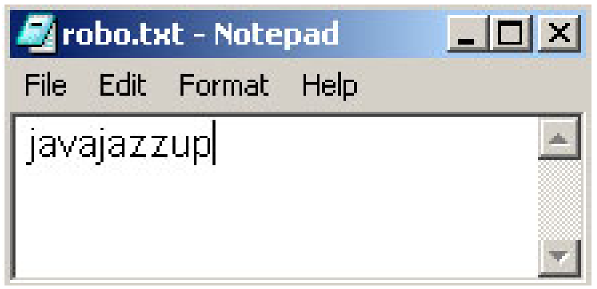


javaja

Now, it opens "Save As" dialog box and writes "robo.txt" in File name and hit enter key.



The file has been saved as "robo.txt" and it can be seen in the figure below.



javajazzup

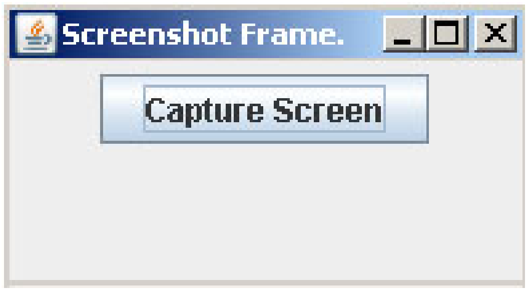
2. Capturing the screen in java:

Capturing the screen is just one line logic in Java, using the java.awt.Robot class. Its createScreenCapture(Rectangle) method takes the location and size of the area to capture and then this screen is stored as a java.awt.image.BufferedImage, which can easily be written to file using the classes from the javax.imageio package.
This sample code demonstrates you how to capture the screen in gif format.

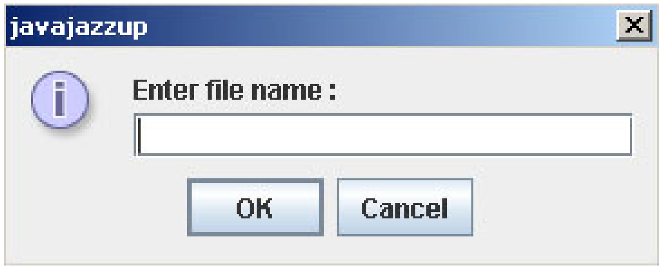
ScreenshotExample.java

```
import javax.swing.*;
import javax.imageio.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;

public class ScreenshotExample {
public ScreenshotExample(){
JFrame frame = new JFrame("Screenshot
Frame.");
frame.setDefaultCloseOperation
(JFrame.EXIT_ON_CLOSE);
    JButton button = new JButton("Capture
Screen");
    button.addActionListener(new
CaptureAction());
   JPanel panel = new JPanel();
    panel.add(button);
    frame.add(panel, BorderLayout.CENTER);
    frame.setSize(200, 200);
    frame.setVisible(true);
  }

  public class CaptureAction implements
ActionListener{
    public void actionPerformed(ActionEvent
ae){
      try{
        //Create Input dialog to take the file
name from the user to save the captured
screen shot.
        String fileName = JOptionPane.
       showInputDialog(null, "Enter file name :
", "javajazzup", 1);
        if
(!fileName.toLowerCase().endsWith(".gif")){
JOptionPane.showMessageDialog
(null, "Error: file name must end with \".gif\".",
"javajazzup", 1);
        }
        else{
          // Create instance of Robot class.
          Robot robot = new Robot();
          // Capture full screen and store it as
BufferedImage.
        BufferedImage image =
robot.createScreenCapture
        (new
Rectangle(Toolkit.getDefaultToolkit().
getScreenSize()));
          // Write the image to file
  ImageIO.write(image, "gif", new
File(fileName));
         // Create the message dialogue
showing the success of the task.
         JOptionPane.showMessageDialog
        (null, "Screen captured successfully.",
"javajazzup", 1);
        }
      }
      catch(Exception e){}
    }
  }

  public static void main(String[] args) throws
Exception {
    ScreenshotExample screen = new
ScreenshotExample();
  }
}
```

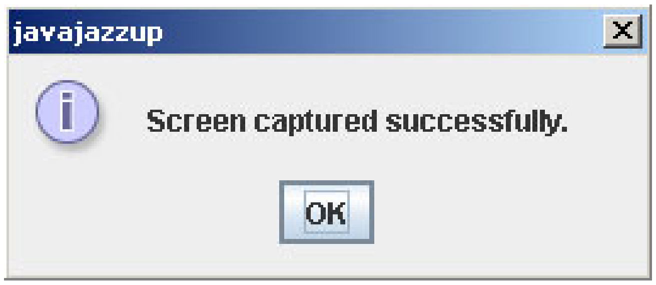
This program shows a frame, which holds a command button labeled by "Capture Screen".



When you click on the button then an input dialog box is open for inputting the file name, which is created after capturing the screen.



And then a message dialog box is opened with message "Screen captured successfully.".



Screen captured successfully.

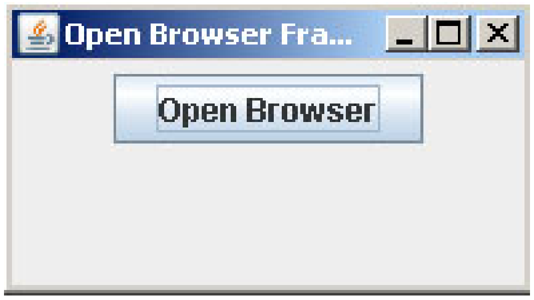
3. Launching the user default browser to show a specified URI.

Using java.awt.Desktop class, a Java application can launch associated applications registered on the current platform to handle a URI or a file. It supports operations like launching the user-default browser to show a specified URI, launching the user-default mail client with an optional mailto URI and launching a registered application to open, edit or print a specified file. This sample code demonstrates how to launch the user default browser to show a specified URI.

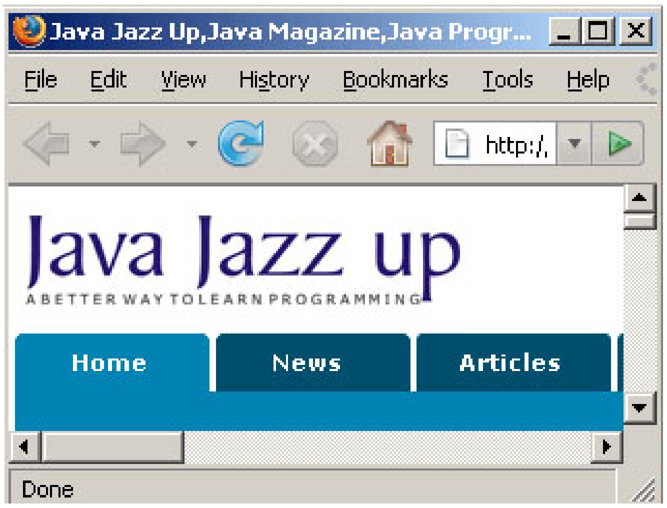
LaunchBrowser.java

```
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.URI;
import java.net.URISyntaxException;

public class LaunchBrowser {
  public LaunchBrowser(){
    JFrame frame = new JFrame("Open
Browser Frame.");
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     JButton button = new JButton("Open
Browser");
     button.addActionListener(new
OpenBrowserAction());
    JPanel panel = new JPanel();
     panel.add(button);
     frame.add(panel, BorderLayout.CENTER);
     frame.setSize(200, 200);
     frame.setVisible(true);
   }
  public class OpenBrowserAction
implements ActionListener{
    public void actionPerformed(ActionEvent
ae){
      try {
        URI uri = new URI("http://
www.javajazzup.com");
        Desktop desktop = null;
                                //Test whether
Desktop class is supported on the current
platform.
        if (Desktop.isDesktopSupported()) {
                                // If it's
supported, retrieve an instance.
          desktop = Desktop.getDesktop();
        }
        if (desktop != null)
                                // Launch
the default browser to display a URI.
          desktop.browse(uri);
          }
        catch (IOException e) {
          e.printStackTrace();
        }
        catch (URISyntaxException e) {
          e.printStackTrace();
        }
    }
  }

  public static void main(String[] args) throws
Exception {
    LaunchBrowser screen = new
LaunchBrowser();
  }
}
```

This program shows a frame, which holds a command button labeled by "Open Browser".



Clicking the button opens "**http://www.javajazzup.com**" in the default browser.



4. Reading XML file:

This program reads the XML file using DOM parser, which loads the XML file into the memory and makes its object model that can be traversed to get its elements. The class **DocumentBuilderFactory** is responsible for obtaining new DOM parsers that produces DOM object trees from XML documents. The java program **ReadMyXMLFile.java** parses **EmpInfo.xml** file and prints its elements to the console in the desired manner.

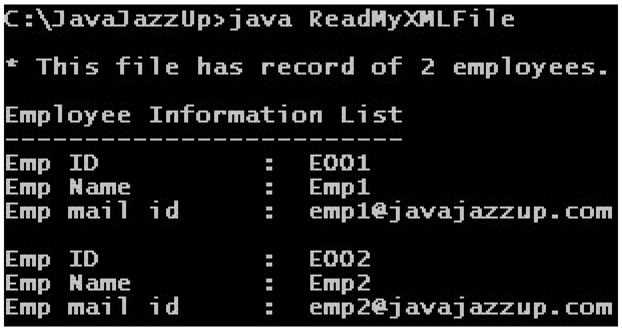
EmpInfo.xml

```
<?xml version = "1.0" ?>
<Employee-Info>

  <Employee>
    <Emp_Id> E001 </Emp_Id>
    <Emp_Name> Emp1 </Emp_Name>
    <Emp_E-mail> emp1@javajazzup.com </
Emp_E-mail>
  </Employee>
  <Employee>
    <Emp_Id> E002 </Emp_Id>
    <Emp_Name> Emp2 </Emp_Name>
    <Emp_E-mail> emp2@javajazzup.com </
Emp_E-mail>
  </Employee>
</Employee-Info>
```

ReadMyXMLFile.java

```
import java.io.File;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class ReadMyXMLFile {
  public static void main(String args[]) {
    try {
      File file = new File("c:\\EmpInfo.xml");
      // Obtain a new instance of a
DocumentBuilderFactory.
      DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
      // Get the DocumentBuilder
      DocumentBuilder db =
dbf.newDocumentBuilder();
      // Parse the content of the given file
as an XML document.
      Document doc = db.parse(file);
      // Get a NodeList of all the Elements
with tag name "Employee".
      NodeList EmpList =
doc.getElementsByTagName("Employee");
      // Get the number of nodes i.e.
"Employee" in the list.
      System.out.println("\n* This file has
record of "+ EmpList.getLength()+ "
employees.\n");
      System.out.println("Employee
Information List");
System.out.println("————————————
");

      for (int index = 0; index <
EmpList.getLength(); index++) {
        // Go to the indexth item
        Node node = EmpList.item(index);
        // Check whether the node is an
Element.
        if (node.getNodeType() ==
Node.ELEMENT_NODE) {
          Element element = (Element) node;

          // Get a NodeList of all
descendant Elements with tag name
"Emp_Id"
          NodeList EIDElmntLst =
element.getElementsByTagName("Emp_Id");
          Element EIDElmnt = (Element)
EIDElmntLst.item(0);
          NodeList EID =
EIDElmnt.getChildNodes();
System.out.println("Emp ID    : "  + ((Node)
EID.item(0)).getNodeValue());
// Get a NodeList of all descendant
Elements with tag name "Emp_Name"
          NodeList ENameElmntLst =
element.getElementsByTagName("Emp_Name");
          Element ENameElmnt = (Element)
ENameElmntLst.item(0);
          NodeList EName =
ENameElmnt.getChildNodes();
          System.out.println("Emp Name  : " +
((Node) EName.item(0)).getNodeValue());

// Get a NodeList of all descendant
Elements with tag name "Emp_E-mail"
NodeList EEmailElmntLst =
element.getElementsByTagName("Emp_E-
mail");
Element EEmailElmnt = (Element)
EEmailElmntLst.item(0);
          NodeList EEmail =
EEmailElmnt.getChildNodes();
          System.out.println("Emp mail id : " +
((Node)
EEmail.item(0)).getNodeValue()+"\n");
        }
      }
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

The output of the above program is given below:



```
C:\JavaJazzUp>java ReadMyXMLFile

* This file has record of 2 employees.

Employee Information List
-------------------------
Emp ID          :  E001
Emp Name        :  Emp1
Emp mail id     :  emp1@javajazzup.com

Emp ID          :  E002
Emp Name        :  Emp2
Emp mail id     :  emp2@javajazzup.com
```

5. Servlet that automatically refreshes

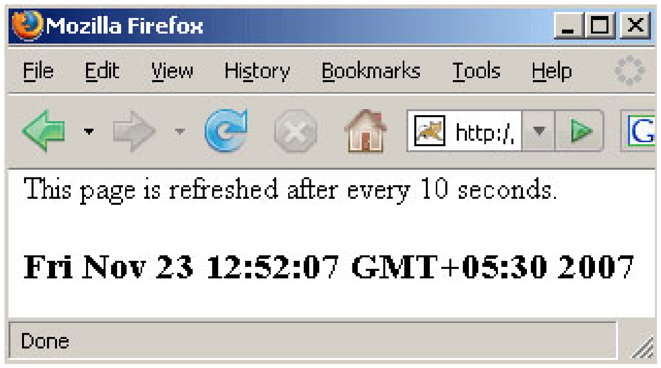
Some times the developer needs to refresh the servlet automatically after some specified time. For example, servlet displaying the scores of the match needs to be refreshed after every little amount of time. This can be done by using the refresh header. Set this header's value to the number of seconds after which you want the servlet to be refreshed periodically. The code sample showing the concept is given below. This servlet is set to be refreshed after each 10 seconds and displaying the current date and time.

```
AutoRefreshServlet.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class AutoRefreshServlet extends
HttpServlet {
  public void doGet(HttpServletRequest req,
HttpServletResponse res) throws
ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
                    // Set the refresh header
to the desired no of seconds.
    res.setHeader("Refresh", "10");
    out.println("<html><body>This page is
refreshed after every 10 seconds.<h3>");
    out.println(new Date().toString());
    out.println("</h3></body></html>");
  }
}
```

Web.xml

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

<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD
Web Application 2.2//EN"
   "http://java.sun.com/j2ee/dtds/web-
app_2_2.dtd">

<web-app>

<servlet>
<servlet-name>AutoRefreshServlet</servlet-
name>
<servlet-class>AutoRefreshServlet</servlet-
class>
</servlet>
<servlet-mapping>
<servlet-name>AutoRefreshServlet</servlet-
name>
<url-pattern>/AutoRefreshServlet</url-
pattern>
</servlet-mapping>

</web-app>
```

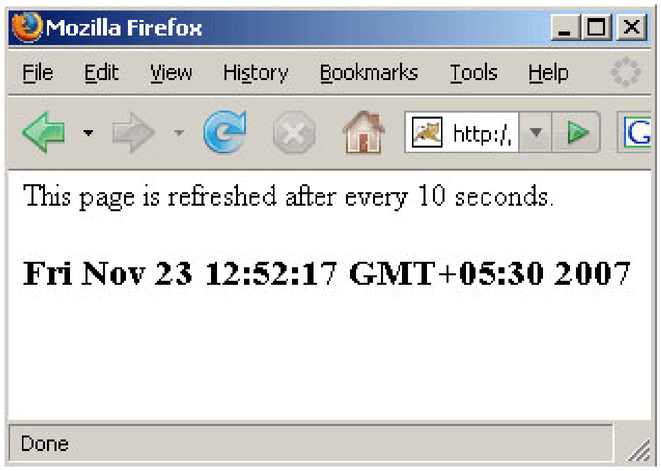
The output can be seen below:



This page is refreshed after every 10 seconds.

Fri Nov 23 12:52:07 GMT+05:30 2007

After 10 seconds the servlet refreshes again and displays the current date and time



This page is refreshed after every 10 seconds.

Fri Nov 23 12:52:17 GMT+05:30 2007

6. Checking password and log information through filter in servlet:

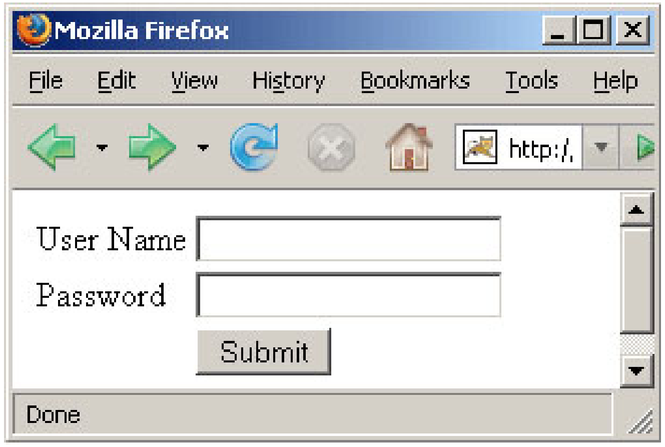
Servlet filters are new component types introduced in Servlet 2.3 specification. Filters are java objects that intercepts requests and responses to transform or use the request and response information. Filters are like preprocessors of the request before sending the request to servlet and postprocessor of the response before sending the response back to the client. Filters allow to take control logic out of servlets and putting this logic in more reusable pieces of code. Request filters can be used to audit or log requests, reformat request headers or perform security checks etc. Response filters can be used to create a different response, compress the response stream, append or alter the response stream etc.

The example given below demonstrates how to use filter to check the password input by the user before passing the request to the requested servlet. If the user passes incorrect password then it displays the page informing the incorrect password. If it is correct, then it passes the control to the desired servlet and does its job, then it returns back to the filter and log some information.

The login.html displays the page to the user to fill the user name and password.

login.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0 Transitional//EN">
<HTML>
  <BODY>
<form action="/javajazzup/WelcomeServlet">
      <table>
        <tr>
          <td>User Name</td>
          <td>
<input type="text" name="username"/>
</td>
        </tr>
        <tr>
          <td>Password</td>
          <td><input type="password"
name="password"/></td>
        </tr>
        <tr>
          <td></td>
          <td><input type="submit"
value="Submit"/></td>
        </tr>
      </table>
    </form>
  </BODY>
</HTML>
```



Create the filter "MyFilter" to be processed before processing the servlet "WelcomeServlet". Like servlets, Filters have their own API. The filter class implements **Filter** interface as servlet implements **Servlet** interface. Filters can get access to the servlet context and can be linked to other filters. Like servlets, filters also have life cycle and has **init()** and **destroy()** methods and like servlet's **doGet()** or **doPost()** methods filters also has **doFilter()** method. Like servlet, Filters are declared in deployment descriptor (web.xml file). A web application can have lots of filters and a given request can cause more than one filter to execute.

MyFilter.java

```
import java.util.*;
import java.io.*;
import javax.servlet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyFilter implements Filter {
  private FilterConfig filterConfig = null;
  public void init(FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
  }
  public void doFilter(ServletRequest req,
ServletResponse res, FilterChain chain)
throws IOException, ServletException {
// Get the password entered by the user.
String password = ((HttpServletRequest)
req).getParameter("password");
// Check whether it matches with the desired
password.
if(password.equals("mypassword")) {
long start = System.currentTimeMillis();
String address =  req.getRemoteAddr();
String file = ((HttpServletRequest)
req).getRequestURI();
// Pass control to the WelcomeServlet's
service() method
chain.doFilter(req, res);

// After returning the control back to the
filter, log user ip, resource uri and time used .
filterConfig.getServletContext().log(" User IP:
" + address + " Resource: " + file + "
Milliseconds used: " +
(System.currentTimeMillis() - start) );
  }
    else {
      // If password doesn't match then send
the page informing incorrect password.
      res.setContentType("text/html");
      PrintWriter pw = res.getWriter();
      pw.println("<html>");
      pw.println("<head><title>Wrong
Password</title></head>");
      pw.println("<body>");
      pw.println("<h3>Sorry, the password
was incorrect.</h3>");
      pw.println("</body>");
      pw.println("</html>");
    }
  }
  public void destroy() { }

}
```

WelcomeServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class WelcomeServlet extends
HttpServlet {
  public void doGet(HttpServletRequest req,
HttpServletResponse res) throws
ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    String username =
req.getParameter("username");
    out.println("<html><body>Welcome :
<b>"+username+"<br/><br/>");
    out.println(new Date().toString());
    out.println("</b></body></html>");
  }
}
```

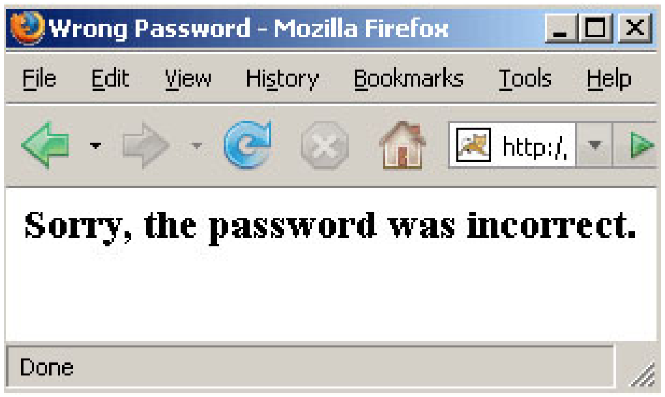
To make the filter work, declare the filter in Deployment Descriptor. If filter is required to be used only when the specific servlet is requested then filter is mapped to that servlet. To create a filter mapping for a specific servlet, map the filter to the name of a servlet by specifying the name of the filter in <filter-name> element and servlet's name in <servlet-name> element.

Web.xml

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

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-
app_2_2.dtd">

<web-app>
  <filter>
    <filter-name>MyFilter</filter-name>
    <filter-class>MyFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>MyFilter</filter-name>
    <servlet-name>WelcomeServlet</servlet-
name>
  </filter-mapping>
  <servlet>
    <servlet-name>WelcomeServlet</servlet-
name>
    <servlet-class>WelcomeServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-
```

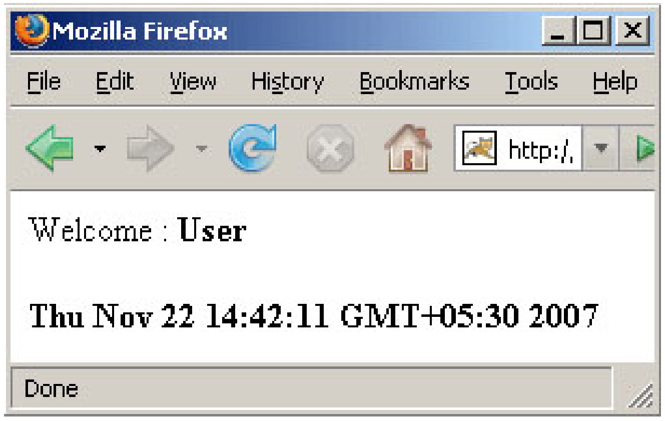
```
name>
    <url-pattern>/WelcomeServlet</url-
pattern>
  </servlet-mapping>
</web-app>
```

If user puts incorrect password then filter sends the page shown below.


Wrong Password - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http:/,
Done

Sorry, the password was incorrect.

If user puts correct password then filter sends control to WelcomeServlet that displays the page given below.



Welcome : **User**

Thu Nov 22 14:42:11 GMT+05:30 2007

After coming the control back to the filter, it logs some information, which can be seen below.

INFO: User IP: 127.0.0.1 Resource: /javajazzup/WelcomeServlet Milliseconds used: 0

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 JavaJazzup Readers Community

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

Contribute to Readers Forum

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

Convene a discussion on a specific subject

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

Sharing Expertise on Java Technologies

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

Show your innovation

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

Hands-on technology demonstrations

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

Present a question, problem, or puzzle

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

Post resourceful URLs

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

Anything else

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

JOY OF WORK

All Cool Jobs
http://www.allcooljobs.com/

NEWSTRACK india
BLUR PRINTING ||||||||||||
http://www.newstrackindia.com