Aug 07 | Volume 1 |Issue 2 | **Free**

Java Jazz up

A BETTER WAY TO LEARN PROGRAMING

Web 2.0

Java Scheduler

Project object Management

Developing Enterprise Applications

Introducing Readers Forum

Interesting Tips & Tricks

SOA Architecture

RoseIndia

Building Enterprise Application

www.roseindia.net

JAVA | SERVLET/JSP | JSF | SPRING | HIBERNATE | more...

Java Jazz Up

"It is not how much you do, but how much love you put in the doing"

Editor	Deepak Kumar
Editor-Technical	Ravi Kant Tamana Agarwal Vinod Kumar
Graphics Designer	Suman Saurabh

© 2007 RoseIndia

Published By

RoseIndia

www.javajazzup.com

Dear Readers,

We are here again with the Java Jazz-up's second issue. This issue reflects our consistent attempts to avail quality technological updates that enforce the reader to appreciate a lot.

Continuing our journey further from the first release, this edition highlights interesting Java articles and tutorials in well described manner developed by Java Jazz-up developer's team.

The current release of Java Jazz-up tries to provide new features orienting around developing enterprise level applications that involves issues like Scheduling, Project Object management, achieving IoC, SOA services, etc. Set of tutorials discussing tools like Maven2, Design patterns, JSF framework, SOA web services, Quartz framework, application servers and various complementary open source technologies are provided in such a manner that a novice learns and implements the concepts in very less time.

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

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

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
Java Jazz up

Contents

Sun patches dangerous Java vulnerabilities

Sun released Java SE 6 Update 2 on 16 July 2007 in the US, which the company said is no longer vulnerable to the flaws, which were highlighted by the Australian Computer Emergency Response Team AusCERT) earlier. Before the release of the atches it was not safe to ***use the Java Runtime Environment*** or ***Java Development Kit*** because some vulnerability were discovered in the Sun Java Runtime Environment by the ***Google's Security team,*** that threatened the security of all platforms, rowsers and even mobile devices. Lot of users and organizations could be affected by it as the bug might destroy devices like PCs, PDAs, mobile phones etc.

Masabi: new mobile security package

A lightweight Java security package has been introduced by UK mobile application developer *Masabi* for mobile phones. Basically the purpose of this launch is to make m-commerce applications more user-friendly and secure.

Acording1 to BT, the only application which validates the implementations of RSA and AES encryption is Masabi's EncryptME Java mobile security component. This technology can run on any cell phone, which is capable to run Java. EncryptME provides friendly interface to the existing messaging applications. The application provides RSA with PKCS Padding for keys up to 4,096 bits, AES with keys up to 256 bits.

Using a single SMS message, or a few bytes of GPRS data, EncryptME can set up a secure session and sign up a new user, a new credit card, and make a transaction. You can easily get this software as it is available in many languages; moreover it aims at applications such as mobile banking and transport.

Important Updates for Adobe Flash, Sun's Java

Adobe and **Sun Microsystems** have issued updates to fix security problems in their **Flash Player** and **Java** applications, respectively. Flash and Java are some of the most widely installed third-party software applications on the planet, so it's a fair bet that criminals could soon be targeting these holes to break into vulnerable systems.

XIC 5.1 Data Access Service: Greater Interoperability

Xcalia, a leading provider of dynamic integration software, launched Xcalia Intermediation Core™ (XIC) version 5.1. XIC 5.1 features the new Data Access Service (DAS). It enables the user to access the heterogeneous data sources based on Java, .NET and web services. Xcalia DAS also leverages the standards specified by the OASIS and Open SOA organizations (i.e. SDO and DAS).

XIC allows the customers to manage their systems towards interoperability in a way that allows them to optimize the information resources and applications they already have, and meanwhile, migrate data and applications at their own pace.

iPhone CPU with Native Java Support

The ARM-based CPU has found special features for accelerating Java code inside the iPhone. Now the chip is likely to be the Samsung-made S3C6400 and will support the Jazelle engine. The Jazelle engine reduces the memory use to a fraction needed for execution of the softwares and it uses a virtual engine that juggles multiple Java programs at once without a major drop in performance.

Now the iPhones will easily load softwares to chat and games without requiring more expensive hardware.

Sun/Intel Partnership Targets Telcos

The Sun/Intel alliances are in a potentially lucrative vertical market. The two companies recently announced the availability of Sun' Solaris operating system on Intel-based telecommunications rack and blade servers. Last month, Sun announced a new line of blade servers available with choice of AMD, Intel or Sun's own Sparc processors.

The systems are carrier grade rack mount servers that are certified Network Equipment-Building System (NEBS) Level 3 and European Telecommunications Standards Institute (ETSI) compliant, and blades that adhere to the Advanced Telecom Computing Architecture (ATCA), spec for next-generation telecommunications equipment.

Updated Releases

ICEFaces 1.6 Released: With Seam and JSF 1.2 compatibility

ICEsoft Technologies Inc. manages ICEfaces.org to provide the developers a wide range of development and support resources regarding ICEfaces. Recently, ICESoft announced the release of ICEFaces 1.6.0, an open source technology which bundles additional JSF components, tools to help integrate with JBoss Seam, compatibility with JSF 1.2 containers, compatibility with Opera, and support for the Liferay Portal, updated IDE support for Eclipse, Netbeans, and JDeveloper.

ICEfaces.org is a place where enterprise Ajax developers can learn, share, and contribute information and ideas to a growing community of ICEfaces enterprise developers. This site provides a wide range of development and support resources to benefit all ICEfaces developers. ICEFaces is an excellent component set for JSF, with the ability to update data on a client as the server determines need and an excellent component set.

JBoss Portal 2.6 Released

Red Hat (Quote) is on a mission to improve the portal experience with the release of its JBoss Portal 2.6, open source portal software. Significant changes have been introduced to the JBoss Portal 2.6. Red Hat aims at making the new release easier to work with the portal. The new release is all about making it easier to work with the portal as Red Hat ramps up its development efforts in the fight for portal supremacy.

Some tremendous improvements have been made to the JBoss 2.6 both at developers' end and to an end user as told by Ram Venkataraman, director of product management JBoss (a division of Red Hat). Features like out of box integration for LDAP, better identity management integration has been added. The Portal is also meant for the end users to interact, it is not alone a developer product.

Azul Systems Released: Next generation Java-based 768 core server

The third-generation Java-based computing appliance has been announced by Azul Systems. This Azul JVM is easily loaded on the conventional server. To execute the compute appliance, the VM bytecode and the application's Java bytecode gets ported over a standard Ethernet network.

However, using a new deployment model called network attached processing the capacity of the compute appliances can be accessed. Moreover, the Azul JVM has replaced the conventional JVM.

Advantages of this approach are:

- One of the most important advantages is that every JVM while executing on the compute appliance can achieve 10X or greater scalability.
- Multiple applications can be executed simultaneously by the compute appliances therefore a lot of virtual compute capacity is made by them. Each application accesses shared capacity of the compute appliances even though it remains isolated on its own dedicated server.

BEA Workshop 10.1 Released: fully integrates Workshop Studio

The integrated Workshop 10.1 is recently launched. This release merges the old BEA Workshop release with the NitroX M7 acquisition, blending the best of both Eclipse extensions into one release. BEA Workshop for WebLogic 10.1 fully integrates all the features from the award winning BEA Workshop Studio, introducing lots of new features.

Tom 2.5 Released: matches pattern in Java

Tom is a text processor that translates classes with some special syntax into Java, and integrates pattern matching and rule based programming facilities. It is mainly suited for programming various transformations on trees/terms and XML based documents.

Tom compiler is mature enough to be used for large scale developments and applications, and integrating Tom in any existing application is straightforward, as Tom produces Java source code.

HDIV 1.3 Released: Struts 2 Security Plugin

HDIV 1.3 has just been released including Struts 2 support. HDIV is an open-source project that extends Struts (Struts 1.x and Struts 2) behavior by adding **web application level Security functionalities** (Integrity, Confidentiality of non editable data and Generic Apache

MyFaces Trinidad 1.2.1, a JavaServer Faces 1.2 component library based on parts of Oracle's ADF Faces. Some of the tags featured: breadcrumbs, navigation panels, panes, and tabbed panels.

Trinidad 1.2.1 is available in both binary and source distributions, and is in the central Maven repository under group id "org.apache.myfaces.trinidad".
Trinidad also supports JSF 1.1, with the older 1.0.1 release.

Maven 2.0.7 Released:

The Apache Maven team has recently launched Maven 2.0.7. This is the third maintenance release within 4 months. The Apache Maven team is rapidly approaching the goal of providing maintenance releases on a monthly basis. The 2.0.8 release is scheduled to provide all relevant information about Maven development in one location so users can easily tell what's happening with Maven at any point in time.

JVM Profiling API (JSR-163) Released:

A new powerful API has been released which has been specified through JSR-163. This new API is JVMTI which has improved profiling interface. It not only includes profiling but monitoring, debugging also and covers the whole range of native in-process tools access.

The implementation has a mechanism for byte code instrumentation and includes Java Programming Language Instrumentation Services (JPLIS). This enables analysis tools to add additional profiling only where it is needed. The advantage of this technique is that it allows more focused analysis and limits the interference of the profiling tools on the running JVM. The instrumentation can even be dynamically generated at runtime, as well as at class loading time, and pre-processed as class files.

Continuous growth of the modern systems always demand the refinements of the respective underlying components. One such requirement for Java applications is of Job scheduling. Java applications depending on the tasks that needs to be performed at specific times (or requires recurring maintenance jobs) are highly benefited with Quartz. Quartz is a fully-featured open source job scheduling framework. It helps to automate scheduling issues within a system and offers an extensive set of multiple job scheduling features.

What is Quartz?

Quartz is a fully featured, an open source job scheduling system that can be integrated with (or used along with) any JSE and JEE applications i.e. from the smallest stand-alone application to the largest e-commerce system. Quartz can be used to create simple as well as the complex schedules (for executing tens, hundreds, or even thousands of jobs, whose tasks can be defined as standard Java components or EJBs). The Quartz Scheduler provides many enterprise-class features, such as JTA transactions and clustering.

Quartz is an open source job scheduling framework that provides simple but powerful mechanisms for job scheduling. It schedules jobs to occur for a specific time or at a specific time interval. It implements many-to-many relationships for jobs and can associate them with different triggers. Quartz can be configured through a property file (in which you can specify a data source for JDBC transactions, global job and/or trigger listeners, plug-ins, thread pools, and more)
Quartz provides a simple Java API, written entirely in Java, that allows you to schedule, monitor, and execute the jobs directly from your upstream application.

Few Features:

- Ability to run in different runtime environments like application servers, web containers, clusters of standalone programs, etc.
- Ability to schedule tasks for a specific time or at a specific time interval. Even, you can block tasks for a specific time.
- Allows to organize triggers as groups in the scheduler.
- Allows to associate multiple jobs with the same trigger.
- Includes event handlers to demarcate start and finish points of a triggered job.
- Allows configuration to automatically re-trigger a job if it fails.
- Allows to configure jobs to participate in JTA transactions.
- Allows to store jobs and triggers in a database, file, or cache them in memory.
- Supports fail-over and load balancing.
- Provides listener support to monitor scheduled jobs from any application.

Who is using Quartz?

Quartz is in use by thousands of people, many of whom have directly embedded Quartz in their own applications, and others who are using products that already have Quartz embedded within them.

Here is a list of just a few of the hundreds of Quartz users:

- **Vodafone Ireland** - uses Quartz for scheduling tests to be carried out on systems in order to generate quality of service information.
- **Covalent Technologies, Inc.** - uses Quartz within their CAM product to handle anything scheduling related in the system, such as: scheduling server actions (start, restart, etc.), metric calculations, data cleanup daemons, etc.
- **PartNET Inc.** - uses Quartz for scheduling application events and driving workflows within many of its products.
- **U.S. Department of Defence** - uses Quartz at various points within a large electronic commerce application, noteably order fulfillment.
- **Level3 Communications** - uses Quartz to drive software builds and deployments.
- **Atlassian** - uses Quartz within their excellent JIRA and Confluence products.
- **Cisco** - uses Quartz in various in-house systems.
- **Apache Jakarta** - Quartz is used within (or as plugins to) several products falling under the Jakarta umbrella.
- **OpenSymphony** - Uses Quartz to drive the OS Workflow product.
- **Spring** - Quartz is used within the Spring Framework.
- **XpoLog** - uses Quartz within XpoLog Center in order to enable automatic technical support.
- **Bloombase Technologies** - has integrating Quartz within their Spitfire EAI Server for XML security processing.
- **Thomson Tax and Accounting** - uses Quartz in its job scheduling framework within it's editorial systems group.
- **The Liferay Portal** - is using the Quartz Scheduler. Just download and check the code. Liferay Inc. Portal.
- **Infoglue CMS** - from infoglue.org
- **Apache Cocoon** - uses Quartz to provide scheduling features and to run application background processes.
- **JBoss** - uses Quartz for the implementation of a number of services within its infrastructure.

Sample uses of job scheduling with Quartz:

- **System Maintenance:** Schedule a job to dump the contents of a database into an XML file every business day (like all weekdays except holidays) at 11:30 PM.
- **Cleaning Resources**: Schedule a job to clean up the unwanted debris left from the running applications. This could be log files, temp tables in a database, clogged connections, or log trails of transactions.
- **Driving Workflow** : As a new order is initially placed, schedule a Job to fire in exactly 2 hours, that will check the status of that order, and trigger a warning notification if an order confirmation message has not yet been received for the order, as well as changing the order's status to 'awaiting intervention'.

Architecture of the Quartz Framework

Quartz framework contains approximately **300 Java Classes** and its multiple interfaces are organized roughly into **12 packages**. Quartz Framework highly depends on the **Quartz Scheduler, Jobs, Triggers and the Job Stores.**

The Quartz Scheduler

The Quartz Scheduler lies at the heart of the Quartz framework. Quartz Scheduler is an engine that manages the runtime environment of the Quartz framework. Scheduler's queue is used to schedule the tasks by queuing them. The framework then allots the threads from its thread pool to execute different tasks (in the queue).

Quartz Framework provides a multi-threaded environment through a pre-defined set of threads configured at startup to manage independent tasks (by multi-tasking them).

When the Scheduler starts up, it initializes the threads and waits for a task to be triggered. At the specified time, the Scheduler assigns a thread to the relevant task and waits. When the task is completed, the thread is returned to the Thread Pool.

Therefore during designing, it's important to take care that there are enough threads available for the Scheduler to run the tasks on time. If there is no thread available when a task is ready to execute, the scheduler will pause until one becomes available. Initializing too many threads can cause degradation of system performance and contention issues.

The easiest way to get a Scheduler working is to use the default Scheduler shipped with Quartz. Call the static **getDefaultScheduler()** method of the **StdSchedulerFactory** class to instantiate the default Scheduler. Now the Scheduler is ready to accept Jobs and Triggers.

Jobs

A Job is simply a Java class that performs a task. This task can be anything that you can code in Java. The only requirement is that you implement the **org.quartz.Job** interface and throw a **JobExecutionException** in the case of a serious error.

Once you implement the Job interface and implement it's **execute()** method, Quartz will invoke the required job when it determines it's time to run. Here are a few examples that can be coded inside a Job:

- Use JavaMail (or another Mail framework like Commons Net) to send emails
- Create a remote interface and invoke a method on an EJB
- Get a Hibernate Session, and query and update data in a relational database
- Use OSWorkflow and invoke a workflow from the Job
- Use FTP and move files around
- Call an Ant build script to kick off scheduled builds

The possibilities are endless and that's what makes the framework so powerful. Quartz provides you with the mechanism to establish a very granular, repeatable schedule, and then you just create Java classes that get called to execute.

Triggers

Quartz's architecture intentionally separates the trigger from the job so that you can make jobs reusable. Multiple jobs can be triggered from a single trigger and multiple triggers can invoke the same job. This facilitates easy code maintenance. Moreover, Quartz provides simple, built-in triggers to do regularly used actions. Job Groups allows you to include the same job in different Job Groups and allows for common actions in the Job Group.

Jobs are only half the equation and requires triggers to get completed. **A Trigger in Quartz is used to tell the Scheduler when a Job should be triggered (or fired)**. The framework comes with a handful of Trigger types, but the two most commonly used are the SimpleTrigger and the CronTrigger.

SimpleTrigger:

The SimpleTrigger is required when you need a simple firing schedule. Typically, if you require a Job to occur at a given time and repeat (n) number of times after specific time-intervals, then the SimpleTrigger is for you. On the other hand, if you need a more complicated schedule for your Job, then the CronTrigger will probably be required.

CronTrigger:

The CronTrigger is based on Calendar-like schedules. The CronTrigger is based on the UNIX cron expression. Let's take you need to execute a job every day at 12:15 a.m., except on Saturdays,Sundays and Mondays then a CronTrigger will work out for you.

The following Quartz cron expression would execute a Job at 12:15 a.m. every day, Tuesday through Friday.
0 15 12 ? * TUE-FRI and this expression 0 15 12 ? * 6L 2005-2008 fires at 10:15 a.m. on the last Friday of every month during the years 2005, 2006, 2007, and 2008.

This is not achievable with the SimpleTrigger. Either can be used for your Jobs.

Job Stores

Quartz allows two types of JobStores : memory-based and a database-based.

The memory-based job store is suitable for a small number of concurrent jobs—information about these jobs is stored in the RAM of the computer. This is the fastest way to access jobs in the Scheduler's queue. However, the flip side is that all these jobs are lost when the machine is restarted—along with their information.

For jobs that need some persistence information, a JDBC-based store is the best solution. Information about these jobs is stored in a database and is accessed by the Scheduler through JDBC.

Hands on Quartz Framework

Installation and Configuration

Quartz is a Java API and thus, can be used in JSE and JEE projects directly as an application module. To begin, just include the required *.jar* files into the project's classpath. All the required *.jar* files come with the Quartz distribution download (check the lib folder of the distribution). The only minimum *.jar* files required to run Quartz are *Quartz.jar* and *commons-logging.jar*. The other files included in the download are supporting libraries supporting additional functions.

Quartz is highly configurable: it allows you to modify the properties of the Scheduler using properties file. The Scheduler first looks for a *quartz.properties* file in the current directory or any folder in the classpath. If it finds one, it starts up with the properties specified in the properties file. Otherwise, it uses the default *quartz.properties* file. The *quartz.properties* file allows you to configure which default Scheduler will be used, the number of threads the Scheduler can start, properties about jobs, properties about triggers, information about JobStores, the priority of the threads being instantiated, etc.

Integration with Other Software

Since Quartz is a Java based API, it can be integrated with any JEE-based API—like JavaMail, JMS, JDBC, etc. It can also be integrated with any third-party API. Quartz can also be embedded into an application server by including the **Quartz .jar** files into the server's classpath.

Now let's develop a simplest program to illustrate the working of the Quartz Framework:

Hello World Quartz Application

In this section, we will develop a simple Quartz Scheduler (“***Hello World***”) that will idisplay “***Hello World Quartz Scheduler : <date & time>***” at the console, after a specific time interval.

To develop a quartz application we need two classes:

1. A Quartz job class implementing the **Job** interface. Here define all types of jobs needed for scheduling.
2. A **scheduler** class where we manage the jobs defined in previous class. The scheduler schedules the jobs and executes them after specified time duration.

Description

To use Quartz, the first thing you need to do is to create a Quartz job that defines a task to schedule.

Create a Quartz job by defining a Java class and implementing the job interface. Override the *execute()* method of the job Interface and add logic to be executed when the job runs.

Next, define a trigger parameter to determine when the job needs to be run and how often. Then, register the Trigger and the job with the scheduler to ensure that the Scheduler can trigger the job at the appropriate time. The Scheduler does not allow you to register a job directly—this needs to be a done through a *JobDetail* class, which allows you to categorize and group similar jobs. The *JobDetail* class also allows you to add a job and define properties like the job's name, the job's Job Group, and the job's Java class. If no Job group is specified, the Scheduler puts the job into a default Job group.

Before using Scheduler you have to instantiate it. Some users may keep an instance of a factory serialized in a **JNDI** store, while others can instantiate it and use a factory instance.

First, create an instance of **SchedulerFactory** using the reference of ***org.Quartz.impl.StdSchedulerFactory*** Class. It invokes the **getScheduler**() method on this instance and instantiates the **Scheduler**.
Now, start the scheduler using the **start()** method. Now instantiate the **JobDetail** and **SimpleTrigger** classes. Finally, pass the objects of the JobDetail and SimpleTrigger classes to the scheduler object using the **scheduleJob()** method.

Code for the Job Class:

```
import org.quartz.Job;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import java.util.Date;
public class HelloJob implements Job {
public void execute(JobExecutionContext arg0) throws
JobExecutionException{
System.out.println("Hello World Quartz Scheduler: " + new
Date()); }}
```

Code for the Scheduler Class:

```
import java.util.Date;
import org.quartz.JobDetail;
import org.quartz.Scheduler;
import org.quartz.SchedulerFactory;
import org.quartz.SimpleTrigger;
import org.quartz.impl.StdSchedulerFactory;
public class HelloSchedule {
public static void main(String args[]){
try{
new HelloSchedule();
}
catch(Exception e){}
}
public HelloSchedule()throws Exception{
SchedulerFactory sf=new StdSchedulerFactory();
Scheduler sched=sf.getScheduler();
sched.start();
JobDetail jd=new
JobDetail("myjob",sched.DEFAULT_GROUP,HelloJob.class);
SimpleTrigger st=new
SimpleTrigger("mytrigger",sched.DEFAULT_GROUP,new
Date(),null,SimpleTrigger.REPEAT_INDEFINITELY,60L*1000L);
sched.scheduleJob(jd, st); }}
```

Before executing this program, set the class path as shown:

```
C:\> cd C:\JavaJazzup\quartz

C:\JavaJazzup\quartz>
setclasspath=%classpath%;C:\quartz-1.6.0\quartz-
1.6.0.jar;C:\quartz-1.6.0\quartz-all-1.6.0.jar;C:\quartz-
1.6.0\quartz-jboss-1.6.0.jar;C:\quartz-1.6.0\quartz-oracle-
1.6.0.jar;
C:\quartz-1.6.0\quartz-weblogic-1.6.0.jar;C:\quartz-.6.0\mysql-
connector-java-3.1.6-bin.jar;
C:\quartz-1.6.0\lib\build\activation.jar;
C:\quartz-1.6.0\lib\build\ejb.jar;C:\quartz-
1.6.0\lib\build\javax.jms.jar;
C:\quartz-1.6.0\lib\build\jdbc2_0-stdext.jar;
C:\quartz-1.6.0\lib\build\jmx.jar;C:\quartz-1.6.0\lib\build\jta.jar;
C:\quartz-1.6.0\lib\build\junit.jar;
C:\quartz-1.6.0\lib\build\mail.jar;
C:\quartz-1.6.0\lib\build\servlet.jar;
C:\quartz-1.6.0\lib\core\commons-collections-3.1.jar;
C:\quartz-1.6.0\lib\core\commons-logging.jar;
C:\quartz-1.6.0\lib\core\commons-logging-api.jar;
C:\quartz-1.6.0\lib\optional\commons-beanutils.jar;
C:\quartz-1.6.0\lib\optional\commons-dbcp-1.2.1.jar;
C:\quartz-1.6.0\lib\optional\commons-digester-1.7.jar;
C:\quartz-1.6.0\lib\optional\commons-modeler-.1.jar;
C:\quartz-1.6.0\lib\optional\commons-pool-1.2.jar;
C:\quartz-1.6.0\lib\optional\commons-validator-1.1.4.jar;
C:\quartz-1.6.0\lib\optional\log4j-1.2.11.jar;
```

Now compile both java files (job and scheduler) and run it.

You will see the output of the application as:

```
C:\JavaJazzup\quartz>javac HelloJob.java

C:\JavaJazzup\quartz>javac HelloSchedule.java

C:\JavaJazzup\quartz>java HelloSchedule log4j: WARN No
appenders could be found for logger
(org.quartz.simpl.SimpleThreadPool).
log4j: WARN Please initialize the log4j system properly.
Hello World Quartz Scheduler:
Fri Jul 13 11:16:35 GMT+05:30 2007
Hello World Quartz Scheduler:
Fri Jul 13 11:17:35
```

JBoss Application Server

JBoss is an open source Java EE-based application server. JBoss is cross-platform, usable on any operating system that supports Java. The core developers were originally employed by "JBoss Inc." founded by Marc Fleury. Red Hat bought JBoss for $420 million in April 2006. The company profits from a service-based business model. As an Open Source project, the project is developed and supported by a wide network of programmers.

JBoss implements the full Java EE suite of services. JBoss pioneered the Professional Open Source business model where the core developers of projects make a living and offer their services. JBoss-related projects include JBoss AS, Hibernate, Tomcat, JBoss ESB, jBPM, JBoss Rules (formerly Drools), JBoss Cache, JGroups, JBoss Portal, SEAM, JBoss Transactions, JBoss Messaging and are marketed under the JBoss Enterprise Middleware Suite (JEMS) brand.

JBoss is purely written in java and is used for hosting the java based business components it is the implementation of JEE that relies on the Enterprise Java Beans for its functionality. It can be used on any java enabled operating system. Due to the open soure it is the one of the most widely used Java Application Server in the market. It is a standard platform for developing and deploying the Web application, Java application and Portals.

EJB 3.0: It is basically designed for fast developing the enterprise Java programming model. It is optional to install the EJB 3.0 capabilities while downloading and running the JBoss as an installer.

JBoss AS 5.0 Beta 1: Now the current version of the next generation of JBoss is available as JBoss AS 5 GA. It is certified for Java EE and includes the following core features:

- Hibernate 3.2: It is JPA certified.
- EJB 3.0: Fully certified to be the part of the Java Enterprise Edition complaint JBoss AS 5.
- JBoss Microcontainer: It is a container based on POJO and removes the dependency on JMX.
- JBoss WebServices 2.0: New custom built JAX-WS complaint WebServices stack.
- JBoss Seam 1.1: Powerful application framework developed for the next generation Web 2.0 applications by unifying and integrating popular service oriented architecture (SOA) technologies.
- JBoss Messaging 1.2: It is HA enabled features next generation platform from JBoss.

Why Choose JBoss Application Server?

Standardized Open Source	JBoss AS is user friendly for business application. It is a J2EE certified open source licensed. JBoss AS is free available and used for embedding and distributing.
Simple not Complex	it helps developers to get started quickly and easily. JBoss provides a bridge that supports for developing the business application based on EJB 3.0.
Best of Breed Technology	It is one of the best available application server in the market. Integration with Apache Tomcat, Hibernate, EJB 3.0 makes it better than others available in the market.
Performance	JBoss AS provides improved performance by providing maximum server utilization than other business applica tion servers.
Service Oriented Architecture	JBoss AS is a service oriented microkernel archi tecture that uses extremely small in footprint to ensure all the service are accessed, managed, and integrated in a unified and consist manner.
Clustering and High Availability	For deploying large scalable enterprise applications JBoss AS provides features like load balancing, clustering, distributed deployment and fail-over.
Professional Support	JBoss Inc. offers for Consult ing, Training and Professional Support required for deploy ing a mission-critical applica tion, rolling out JEMS or testing a proof of concept across the enterprise.
100% Pure Java	Most of the operating sys tems that are capable of running Java Virtual Machine (JVM) like Microsoft Win

dows , Red Hat Enterprise Linux, Sun Solaris, SUSE Linux, HP-UX and many more are compatible with JBoss since JBoss AS is purely written in Java.

Hands on JBoss AS

Install JBoss

To install JBoss first download JBoss 4.2.0. Unzip the files. Note that there is no need to set the path of any variable for JBoss server other than JAVA_HOME.

Set the environment variable JAVA_HOME: Set the environment variable JAVA_HOME for JBoss, failing which prevents the JBoss from compiling the java source code. This variable should contain the path of JDK installation directory. Notice that it should not list the path of bin folder.

For Windows 2000 / NT / XP
Set JAVA_HOME = C:\Program Files\Java\jdk1.5.0_08
or just Go to Start window—>Control Panel—>System—>Advanced—>Environment Variables—>Click "New" button and set the values as

Variable Name: JAVA_HOME

Variable Value: C:\Program Files\Java\jdk1.5.0_08

To start the JBoss Server: To start the JBoss server go to the bin directory of the JBoss server and double click on run.bat or make a shortcut of file at your desktop.

To stop the JBoss Server: To stop the running server press the "**ctrl + c**" keys.

How to deploy and run your application: First give a name to your application directory with the **.war** extension and deploy it in the **jboss-4.2.0.GA/server/default/deploy** directory of the JBoss server.

Start the server, open a browser window and type the url "**http://localhost:8080/application directory name/file name**" and then press enter. If everything is ok your application will run. JBoss server by default runs on the port no 8080.

Note that: write the application directory name in the url without the **.war** extension.

Using Ajax in Apache MyFaces

Ajax, or AJAX, is a web development technique used for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is intended to increase the web page's interactivity, speed, functionality, and usability.

Ajax is a cross-platform technology compatible with different operating systems, computer architectures, and web browsers as it is based on open standards such as JavaScript and XML, together with open source implementations of other required technologies.

Ajax has a big name in the field of web development. One of the biggest advantages of Ajax is that it really improves web application user experience. Ajax is highly compatible with various technologies like HTML, JSP, JSF (Java Server Faces) etc. Traditionally, when the user requests for the page, the entire web page is rendered while in the case of Ajax only a portion of page is rendered.

What is Ajax?

Ajax is a new approach for creating interactive web applications. It is an acronym for "**Asynchronous JavaScript and XML**". Ajax is a bundle of commonly used technologies:

1 HTML (or XHTML) and CSS
2 Document Object Model (DOM)
3 XML
4 JavaScript, XMLHttpRequest Object

Hence, Ajax is not a new technology. An Ajax application looks as if it resideson the user's machine. Data is asynchronously fetched and browser is updated with the gathered information rather than refreshing the entire page i.e. partial refreshing of a page.

Two technologies that will enhance Rich Internet Applications:

AJAX is a perfect match for the JSF presentation tier because of its rich component model. Simply use JSF AJAX components to add AJAX functionality to a JSF application.

JavaServer(tm) Faces (JSF) is an established web application framework standard (JSR127) that accomplishes the MVC paradigm. JSF is comparable to the well-known Struts framework but has features and concepts that are beyond those of Struts. MyFaces Core is a 100% compliant implementation of this standard - it is compliant to the standard yet offers additional value by providing separate extended component sets and many special features which make JSF development easier.

MyFaces has several subprojects, where MyFaces API and MyFaces Impl together comprise the core implementation. Tomahawk (MyFacesComponents), Tobago (Tobago) and Trinidad (Trinidad) are sets of extended components. These components offers you more functionality and flexibility than using the standard components of the core implementation.

Integrating AJAX in Java Server Faces (MyFaces) :
Apache MyFaces has a sub-project named "**Sandbox**" which provides a set of components. Sandbox is a new addition to the Tomahawk project. This helps us to easily integrate Ajax enabled components in MyFaces. To let the sandbox components work, we have to add "**tomahawk-sandbox-1.1.6-SNAPSHOT.jar"** file in lib folder of your application along with complete set of jar files for JSF (MyFaces) environment to run application.

Download "tomahawk-sandbox-1.1.6-SNAPSHOT.jar" file:
"tomahawk-sandbox-1.1.6-SNAPSHOT.jar" file can be downloaded from http://people.apache.org/builds/myfaces/nightly/tomahawk-sandbox-1.1.6-SNAPSHOT-bin.zip address. Just unzip the file, pick the jar file from the lib folder and copy it to the lib folder of your application. Restart the server and start using sandbox components in your application.

Example of using Ajax in MyFaces through Sandbox component

Auto Complete example:

In this example, user can only enter the name of a state (from U.S.). As soon as the user enters any string matching the states name, a drop down list comes in the picture. That helps to select the state from there. Sandbox provides a very nice component **InputSuggestAjax** that is a very good example of using Ajax in MyFaces. We have nothing to do just simply use the tag for the component in the view page and create backing bean to provide functionality for selecting the list of appropriate states matching the string entered by the user in the page. Just follow these steps:

1 Create View page (JSP file)
2 Create JSF managed bean

Creating JSP file:

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

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
<title>JSF Auto Complete Example</title>
<style  type="text/css">
body{
margin-top:30px;
margin-bottom:opx;
margin-left:0px;
margin-right:0px;
}
</style>
</head>
<body>
<center>
<h:panelGrid id="pg1" columns="1" width="500px"
style="background-image:url(images/border2.gif)">
<f:facet name="header">
<h:outputText value="JSF Autocomplet Example"/>
</f:facet>
</h:panelGrid>
<h:form>
</br>
<h:outputText value="Enter U.S State:" style="font-weight:bold"
/>
<s:inputSuggestAjax
suggestedItemsMethod="#{StateBean.getSuggestItems}"
value="#{StateBean.currentValue}"   />
</h:form>
<h:graphicImage id="gi2" value="images/border.gif"
width="500px" height="74">
</h:graphicImage>
</center>
</body>
</html>
</f:view>
```

To use the tag for the component, we have to specify the name of tag library with prefix. We have specified prefix "s" for the sandbox tags. Now we can use the components of sandbox in our JSP file. In this example, we have used only one component "**inputSuggestAjax**". It has "**suggestedItemsMethod"** attribute which has been set to the property "**getSuggestItems"** of backing bean **StateBean**. This bean provides set of states to be displayed when user enters some words of the state. The current value of the component is set to the property "**currentValue**" of the same bean.

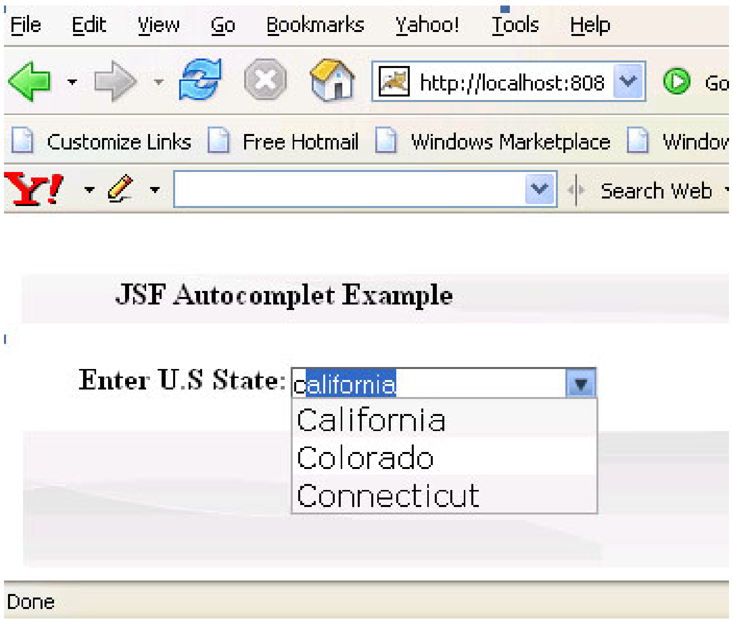
Creating Managed Bean:

```
package net.roseindia.web.ui;
import java.util.*;
public class StateBean{
        String currentValue;
```

```
public String getCurrentValue() {
        return currentValue;
}
public void setCurrentValue(String currentValue)
{
        this.currentValue = currentValue;
}

public List getSuggestItems(String prefix)
{

        List list = new ArrayList();

        list.add("Alabama");
        list.add("Alaska");
        list.add("Arizona");
        list.add("Arkansas");
        list.add("California");
        list.add("Colorado");
        list.add("Connecticut");
        list.add("Delaware");
        list.add("District of Columbia");
        list.add("Florida");
        list.add("Georgia");
        list.add("Hawaii");
        list.add("Idaho");
        list.add("Illinois");
        list.add("Indiana");
        list.add("Iowa");
        list.add("Kansas");
        list.add("Kentucky");
        list.add("Louisiana");
        list.add("Maine");
        list.add("Maryland");
        list.add("Massachusetts");
        list.add("Michigan");
        list.add("Minnesota");
        list.add("Mississippi");
        list.add("Missouri");
        list.add("Montana");
        list.add("Nebraska");
        list.add("Nevada");
        list.add("New Hampshire");
        list.add("New Jersey");
        list.add("New Mexico");
        list.add("New York");
        list.add("North Carolina");
        list.add("North Dakota");
        list.add("Ohio");
        list.add("Oklahoma");
        list.add("Oregon");
        list.add("Pennsylvania");
        list.add("Rhode Island");
        list.add("South Carolina");
        list.add("South Dakota");
        list.add("Tennessee");
        list.add("Texas");
        list.add("Utah");
```

```
                    list.add("Vermont");
                    list.add("Virginia");
                    list.add("Washington");
                    list.add("West Virginia");
                    list.add("Wisconsin");
                    list.add("Wyoming");

                    List itemsFound = new ArrayList();

                    Iterator it = list.iterator();
                    while (it.hasNext())
          {
                              String s = (String) it.next();

if((s.toLowerCase()).startsWith(prefix.toLowerCase()))
          {
                                        itemsFound.add(s);
                              }

                    }
                    return itemsFound;
          }
}
```

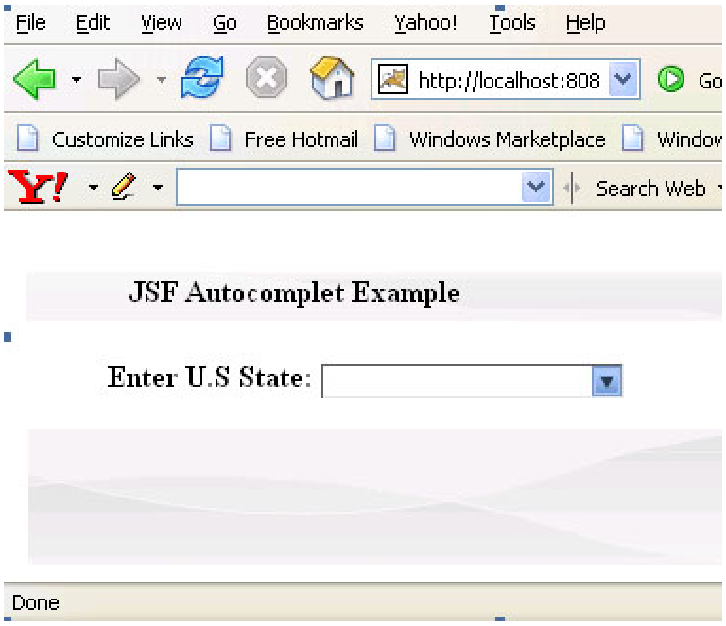
This backing bean has method "**getSuggestItems()"** which takes one String parameter and returns a List object. The string parameter "**prefix**" is the string entered by the user while entering the name of the state in the page. The entered string is checked in the list "**list**" of all states and the selected states are returned as an another list "**itemsFound**". This set of states is shown at the time when the user enters the name of state in the field.



JSF Autocomplet Example

Enter U.S State: california

California
Colorado
Connecticut



JSF Autocomplet Example

Enter U.S State:

Maven 2

Maven is a high-level, intelligent project management, build and deployment tool provided by Apache's software foundation group. Maven deals with application development lifecycle management. Maven was originally developed to manage and to minimize the complexities of building the Jakarta Turbine project. But its powerful capabilities have made it a core entity of the Apache Software Foundation projects. Actually, for a long time there was a need to standardized project development lifecycle management system and Maven has emerged as a perfect option that meets the needs. Maven has become the de- facto build system in many open source initiatives and it is rapidly being adopted by many software development organizations.

Maven was borne of the very practical desire to make several projects at Apache work in a consistence manner. So that developers could freely move between these projects, knowing clearly how they all worked by understanding how one of them worked.

If a developer spent time understanding how one project built it was intended that they would not have to go through this process again when they moved on to the next project. The same idea extends to testing, generating documentation, generating metrics and reports, testing and deploying. All projects share enough of the same characteristics, an understanding of which Maven tries to harness in its general approach to project management.

On a very high level all projects need to be built, tested, packaged, documented and deployed. There occurs infinite variation in each of the above mentioned steps, but these variation still occur within the confines of a well defined path and it is this path that Maven attempts to present to everyone in a clear way. The easiest way to make a path clear is to provide people with a set of patterns that can be shared by anyone involved in a project.

The key benefit of this approach is that developers can follow one consistent build lifecycle management process without having to reinvent such processes again. Ultimately this makes developers more productive, agile, disciplined, and focused on the work at hand rather than spending time and effort doing grunt work understanding, developing, and configuring yet another non-standard build system.

Shifting from Ant to Maven

Maven is entirely a different creature from Ant. Ant is simply a toolbox whereas Maven is about the application of patterns in order to achieve an infrastructure which displays the characteristics of visibility, reusability, maintainability, and comprehensibility. It is wrong to consider Maven as a build tool and just a replacement for Ant.

There is nothing that Maven does that Ant cannot do. Ant gives the ultimate power and flexibility in build and deployment to the developer. But Maven adds a layer of abstraction above Ant (and uses Jelly). Maven can be used to build any Java application. Today JEE build and deployment has become much standardized. Every enterprise has some variations, but in general it is all the same: deploying EARs, WARs, and EJB-JARs. Maven captures this intelligence and lets you achieve the build and deployment in about 5-6 lines of Maven script compared to dozens of lines in an Ant build script.

Ant lets you do any variations you want, but requires a lot of scripting. Maven on the other hand mandates certain directories and file names, but it provides plugins to make life easier.

The restriction imposed by Maven is that only one artifact is generated per project (A project in Maven terminology is a folder with a project.xml file in it). A Maven project can have sub projects. Each sub project can build its own artifact. The topmost project can aggregate the artifacts into a larger one. This is synonymous to jars and wars put together to form an EAR. Maven also provides inheritance in projects.

Maven simplifies build enormously by imposing certain fixed file names and acceptable restrictions like one artifact per project. Artifacts are treated as files on your computer by the build script. Maven hides the fact that everything is a file and forces you to think and script to create a deployable artifact such as an EAR. Artifact has a dependency on a particular version of a third party library residing in a shared remote (or local) enterprise repository, and then publish your library into the repository as well for others to use. Hence there are no more classpath issues. No more mismatch in libraries. It also gives the power to embed even the Ant scripts within Maven scripts if absolutely essential.

Some of the common concerns while building a project are :

I **Project directory structure:** The directory structure of a Web application project is different from that of an EJB application project. Similarly the output of a Web application project is typically a WAR file while that of an EJB application is a JAR file.

II **Directory naming conventions:** For a specific project type, the typical requirements in terms of directory layout and naming conventions are almost the same. Without a unified framework such as Maven, developers mostly spend time in configuring such nitty details like setting up directories for source, resources, test case source, testing time resources, classes, and project dependencies.

III **The build output:** Developers spend a good chunk of time in creating build scripts such as ANT scripts to execute build tasks according to the project layout. This entire endeavor ends up being chaotic and in a large-scale project it can lead to a maintenance nightmare demanding dedicated resources just to focus on such build aspects.

Maven basics

I. Project Object Model

A Project Object Model or POM is the fundamental unit of work in Maven. It is an xml file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.

For examples the build directory, which is “target”; the source directory, which is “src/main/java”;

the test source directory, which is “src/main/test”; and so on.

POM also contains the goals and plugins. So, while executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.
Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

The minimum requirement for a POM are the following:

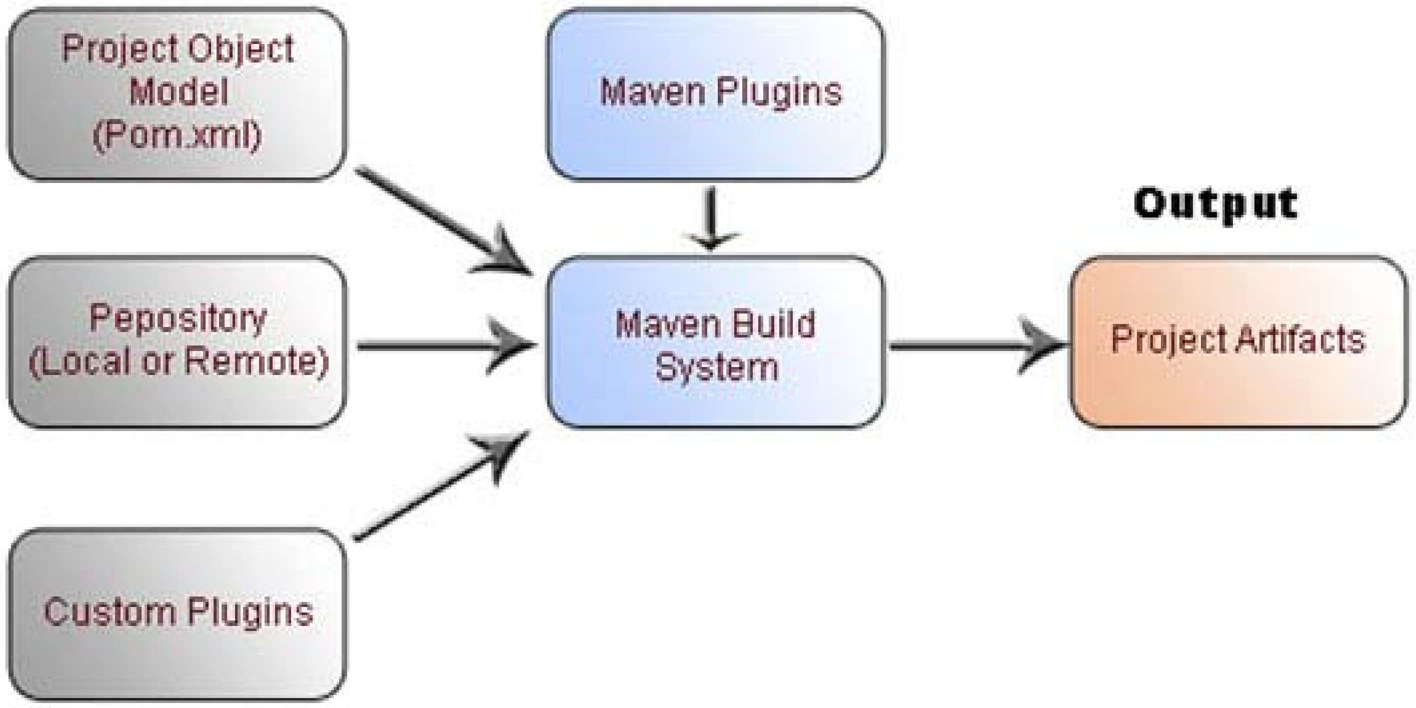
- project root
- modelVersion - should be set to 4.0.0
- groupId - the id of the project’s group.
- artifactId - the id of the artifact (project)
- version - the version of the artifact under the specified group

Here’s an example:

```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>net.roseindia.maven</groupId>
<artifactId>HelloMaven</artifactId>
<version>1</version>
</project>
```

II. Repository

Another concept in Maven is that of a **repository**. The repository holds the artifacts on which your project depends. There are two kinds of repository: local and remote. Both of them can be declaratively set. Unless specified otherwise, the local repository is created in a special directory called *“.m2/repository”*. In Windows, this directory is created in **C:\Documents and Settings\Administrator**. For example, if your project depends on commons-logging version 1.0.4, you can specify the dependency in *pom.xml* and when maven is executed, it will copy the appropriate commons-logging jar file from the remote repository to the local repository and then use it to build your project’s artifact like JARs, WARs, EARs etc.



The maven repository folder has subfolders for each library. For instance, there is a sub folder for commons-logging. Beneath the commons-logging folder there is another Subfolder called jars. This jars folder has the commons logging jar files suffixed by version number.

The role of the repository is immediately obvious. Instead of each project having its own copies of third party libraries, the repository helps developers across projects to share the libraries. Each project can also in turn generate its artifacts and publish it into the remote repository. The process of publishing a jar into the repository is called *"install"* in Maven lingo. This install process helps organizations to share internal artifacts across projects in a standard manner. This also holds the basis for continuous integration among inter-dependent projects.

Figure shows the pom.xml, repository and plugins. The grey colored rectangles are provided by you. The blue colored rectangles are provided by Maven. The orange colored rect-angle shaded is the output - the real deployment artifact obtained for your project. Custom plug-ins are optional. The rest of the inputs are mandatory.

Installing and Getting Hands on Maven

Now that we have sufficient understanding of the *pom.xml* file let us get on try something out.
Download Maven from maven.apache.org, unzip the archive into your local directory.

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

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

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

2. Note create a working directory

```
C:\>md maventest
C:\>cd maventest
```

3. Create your first project. In order to create the sim plest of Maven projects, execute the following from the command line:

```
C:\maventest>mvn archetype:create -
DarchetypeGroupId=org.apache.maven.archetypes -
DgroupId=net.roseindia.maven.quickstart -
DartifactId=HelloMaven
```

Once you have executed this command, you will notice a few things have happened.

First, you will notice that a directory named **HelloMaven** has been created for the new project,and this directory contains a file named pom.xml that should look like this

pom.xml

```
<project xmlns=”http://maven.apache.org/POM/4.0.0"
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-
instance”xsi:schemaLocation=”http://maven.apache.org/POM/
4.0.0 http://maven.apache.org/maven-v4_0_0.xsd”>
<modelVersion>4.0.0</modelVersion>
<groupId>net.roseindia.maven.quickstart</groupId>
<artifactId>HelloMaven</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>HelloMaven</name>
<url>http://maven.apache.org</url>bn
<dependencies><dependency><groupId>junit</groupId>
<artifactId>junit</artifactId><version>3.8.1</version>
<scope>test</scope></dependency></dependencies>
</project>
```

After the archetype generation of your first project you can see, the project created from the archetype has a POM, a target directory, a source tree for your application’s sources and a source tree for your test sources. This is the standard layout for Maven projects.
The application sources reside in ${basedir}/src/main/java and test sources reside in ${basedir}/src/test/java, where {basedir} represents the directory containing pom.xml.

you will notice that the following directory structure has been created:

```
HelloMaven
 |--pom.xml
 |--target
 |--src
     |--main
         |-- java
             |-- net
                 |-- roseindia
                     |-- maven
                         |-- quickstart
                             |-- Calculator.java
     |--test
         |-- java
             |-- net
                 |-- roseindia
                     |-- maven
                         |-- quickstart
                             |-- TestCalculator.java
```

4. Now to compile the application sources. Change to the directory where pom.xml is created and execute the “*mvn compile”* command to compile your application sources:

```
C:\maventest>cd HelloMaven
C:\maventest\HelloMaven>mvn compile
```

Upon executing this command you should see output like the following:

```
C:\maventest\HelloMaven>mvn compile [INFO] Scanning for
projects...[INFO]
---------------------------------------------------------------------------------
[INFO] Building HelloMaven[INFO] task-segment:
[compile][INFO]
---------------------------------------------------------------------------------
[INFO] [resources:resources][INFO] Using default encoding to
copy filtered resources.[INFO] [compiler:compile][INFO]
Nothing to compile - all classes are up to date[INFO]
---------------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL[INFO]
---------------------------------------------------------------------------------
[INFO] Total time: 1 second[INFO] Finished at: Sat Jul 21
17:33:27 GMT+05:30 2007[INFO] Final Memory: 2M/5M[INFO]
---------------------------------------------------------------------------------
```

The first time you execute this command, Maven will download all the plugins and related dependencies it requires to fulfill the command. From a clean installation of Maven this can take quite a while (in the output above, it took almost 4 minutes).

If you execute this command again, Maven will now have what it needs, so it won't need to download anything new and will be able to execute the command much more quickly.

As you can see from the output, the compiled classes were placed in ${basedir}/target/classes, which is another standard convention employed by Maven. So, if you're a keen observer, you'll notice that by using the standard conventions the POM above is very small and you haven't had to tell Maven explicitly where any of your sources are or where the output should go.

5. To compile your test sources and run your unit tests. Now you've got some unit tests that you want to compile and execute.
Execute the following command:

```
C:\maventest\HelloMaven>mvn test
C:\maventest\HelloMaven>mvn test[INFO]
Scanning for projects...[INFO]
---------------------------------------------------------------------------------
[INFO] Building HelloMaven[INFO] task-segment: [test][INFO]
---------------------------------------------------------------------------------
[INFO] [resources:resources][INFO] Using default encoding to
copy filtered resources.[INFO] [compiler:compile][INFO]
Nothing to compile - all classes are up to date[INFO]
[resources:testResources][INFO] Using default encoding to
copy filtered resources.[INFO] [compiler:testCompile][INFO]
Nothing to compile - all classes are up to date[INFO]
[surefire:test][INFO] Surefire report directory:
C:\maventest\HelloMaven\target\surefire-reports
---------------------------------------------------------------------------------
T E S T S
---------------------------------------------------------------------------------
[surefire] Running
net.roseindia.maven.quickstart.TestCalculator[surefire] Tests
run: 1, Failures: 0, Errors: 0, Time elapsed: 0.001 sec Results
:[surefire] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0[INFO]
---------------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL[INFO][INFO] Total time: 2
seconds[INFO] Finished at: Sat Jul 21 11:34:53 GMT+05:30
2007[INFO] Final Memory: 3M/7M[INFO]
---------------------------------------------------------------------------------
```

6. Now you need to create a jar. Making a JAR file is straight forward.Just execute the following command:

```
C:\maventest\HelloMaven>mvn package
```

7. Now to install the artifact you've generated (the JAR file) in your local repository (~/.m2/repository is the default location). If you take a look at the POM for your project you will notice the packaging element is set to jar. This is how Maven knows to produce a JAR file from the above command. You can now take a look in the ${basedir}/target directory and you will see the generated JAR file. Let's move on to installing our artifact. To do so execute the following command:

```
C:\maventest\HelloMaven>mvn install
```

We will dig Maven 2 deeper, in our forth coming issues…

Spring Framework

The **Spring Framework** comes in the form of ZIP file with the necessary jars, examples etc. The Spring framework can be downloaded from **http://www.springframework.org.** There will be two zip files one with dependencies and other without. The spring framework with dependencies is larger and includes all dependent libraries. Download **Spring framework 1.2.9 without dependency** and unzip on the hard disk as **spring1.2.9.**

Inside the folder spring1.2.9 there are 7 folders. The names and the contents of all the folders are given below:

1 **dist:** It contains various Spring distribution jar files.
2 **docs:** It contains general documentation and API javadocs.
3 **mock:** It contains mock JNDI contexts and a set of servlet API mock objects.
4 **samples:** It contains demo applications and skeletons.
5 **src:** It contains the Java source files for the framework.
6 **test:** It contains the Java source files for Spring's test suite.
7 **tiger:** It contains JDK1.5 examples and test suite.

Inside the "dist" directory, we can find all the jar files necessary for compiling and executing the program. The jar files and its contents are listed below:

1 **spring.jar:** It contains the entire spring framework including everything in the other JAR files also.
2 **spring-core.jar**: It contains the core spring container and its utilities.
3 **spring-beans.jar:** It contains the bean spring container and JavaBeans support utilities.
4 **spring-aop.jar:** It contains spring's AOP framework, source-level metadata support, AOP Alliance interfaces etc.,
5 **spring-context.jar:** It contains application context, validation framework, JNDI, templating support and scheduling.
6 **spring-dao.jar:** It contains DAO support and transaction infrastructure.
7 **spring-jdbc.jar:** It contains the JDBC support.
spring-support.jar: It contains JMX support, JCA support, scheduling support, mail support and caching support.
8 **spring-web.jar:** It contains the web application context, multipart resolver, Struts support, JSF support and web utilities.
9 **spring-webmvc.jar:** It contains the framework servlets, web MVC framework, web controllers and web views.
10 **spring-remoting.jar:** It contains remoting support, EJB support and JMS support.
11 **spring-orm.jar:** It contains iBATIS SQL Maps support, Apache OJB support, TopLink support and JDO support
12 **spring-orm.jar:** It contains iBATIS SQL Maps support, Apache OJB support, TopLink support and JDO support.
13 **spring-hibernate.jar:** It contains Hibernate 2.1 support, Hibernate 3.x support.
14 **spring-mock.jar:** It contains JNDI mocks, Servlet API mocks and JUnit support.

org.springframework.beans package

The core of spring is the **org.springframework.beans** package, designed for working with JavaBeans. This package is not used directly by users, but provides much of the spring functionality.
The next higher layer of abstraction is the **bean factory**. A spring bean factory is a generic factory that enables objects to be retrieved by name, and which can manage relationships between objects.

Bean factories support two modes of object:

- **Singleton:** In this case, there's one shared instance of the object with a particular name, which will be retrieved on lookup. This is the default, and most often used, mode. It's ideal for the stateless service objects.

- **Prototype or non-singleton:** In this case, each retrieval will result in the creation of an independent object. For example, this could be used to allow each caller to have its own distinct object reference.

The Spring container manages relationships between objects, so it can add value (where necessary) through services such as transparent pooling for managed POJOs, and support for hot swapping, where the container introduces a level of indirection that allows the target of a reference to be swapped at runtime without affecting callers and without loss of thread safety.

As org.springframework.beans.factory.BeanFactory is a simple interface, it can be implemented in different ways. The **BeanDefinitionReader** interface separates the metadata format from **BeanFactory** implementations themselves, so the generic **BeanFactory** implementations that spring provides can be used with different types of metadata. It is easy to implement your own BeanFactory or BeanDefinitionReader.

The most commonly used BeanFactory definitions are:

XmlBeanFactory: This parses a simple, intuitive XML structure defining the classes and properties of named objects. We provide a DTD to make authoring easier.

DefaultListableBeanFactory: This provides the ability to parse bean definitions in properties files, and create BeanFactories programmatically.

Each bean definition can be a POJO (defined by class name and JavaBean initialisation properties or constructor arguments), or a FactoryBean. The FactoryBean interface adds a level of indirection.

BeanFactories can optionally participate in a hierarchy, "inheriting" definitions from their ancestors. This enables the sharing of common configuration across a whole application, while individual resources such as controller servlets also have their own independent set of objects.

Through its bean factory concept, spring is an **Inversion of Control** container. A Spring BeanFactory is a container that can be created in a single line of code, and requires no special deployment steps. Spring is most closely identified with a flavor of Inversion of Control known as **Dependency Injection**. Now let's develop a Hello example with Spring support.

```
D:\>md springdemo
D:\>cd springdemo
```

As the entire Spring Framework is included in **spring.jar**. We use it to run our examples.

Copy **spring.jar** from **spring1.2.9\dist** folder to the working folder (**springdemo**). Also copy **commons-logging.jar** from **apache-tomcat-6.0.10** to the working directory.

Set path for jdk1.4.2 and above versions only
Now Set the classpath as shown :

```
D:\>springdemo >set classpath=D:\springdemo;
D:\springdemo\spring.jar;
D:\springdemo\commons-logging.jar
```

For a typical Spring Application we need the following files

I An **interface** that defines the functions.
II An **Implementation** that contains properties, its setter and getter methods, functions etc.,
III A XML file called **Spring configuration file.**
IV Client program that uses the function.

Create the following files

1 hello.java
2 helloimpl.java
3 hello.xml
4 helloclient.java

1. D:\springdemo\hello.java

```
public interface hello
{
  public String sayhello(String a);
}
```

2. D:\springdemo\helloimpl.java

```
public class helloimpl implements hello {
private String greeting;
public helloimpl()
{
}
public helloimpl(String a)      {
greeting=a;   }
public String sayhello(String s)   {
return greeting+s;  }
public void setGreeting(String a)
{
greeting=a;
}
}
```

3. D:\springdemo\hello.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//
EN"  "http://www.springframework.org/dtd/spring-
beans.dtd">
<beans>
<bean id="hello" class="helloimpl">
<property name="greeting">
<value>Good Morning!...</value>
</property>
</bean>
 </beans>
```

4. D:\springdemo\helloclient.java

```
import java.io.*;
import org.springframework.beans.factory.*;
import org.springframework.beans.factory.xml.*;
import org.springframework.core.io.*;
public class helloclient  {
public static void main(String args[])  throws Exception{
try {
System.out.println("please Wait.");
Resource  res = new  ClassPathResource("hello.xml");
BeanFactory  factory = new  XmlBeanFactory(res);
hello bean1 = (hello)factory.getBean("hello");
String s = bean1.sayhello(args[0]);
System.out.println(s);
}
catch(Exception e1) {
System.out.println(""+e1);
}
}
}
```

Now, execute the programs:

```
D:\springdemo>javac hello.java
D:\springdemo>javac helloimpl.java
D:\springdemo>javac helloclient.java hello
D:\springdemo>java helloclient AMIT
```

We will get the output as follows:

```
please Wait.Jul 17, 2007 5:40:24 PM
org.springframework.core.CollectionFactory <clinit>INFO:
DK 1.4+ collections availableJul 17, 2007 5:40:24 PM
org.springframework.beans.factory.xml.XmlBeanDefinitionReader
loadBeanDefinitionsINFO: Loading XML bean definitions from
class path resource [hello.xml]Good Morning!...AMIT
```

Thus we have run our first spring program. Here helloimpl implements the hello interface. Although it is not necessary to hide the implementation behind an interface, it is recommended as a way to seperate implementation from interface.

helloimpl class has a single property greeting. This property can be set in two different ways: by property's setter method or by the constructor. The XML file hello.xml declares the instance of helloimpl.java in the spring container and configures its property greeting with the value 'Good Morning!...'

The root of the hello.xml file is the <beans> element, which is the root element of any Spring configuration file. The <bean> element is used to tell the spring container about the class and how it should be configured. Here, the id attribute takes the interface name and the class attribute specifies the bean's fully qualified class name.

Within the <bean> element, the <property> element is used to set the property, in this case the greeting property. By using <property>, we're telling the Spring container to call setGreeting() while setting the property. The value of the greeting is defined within the <value> element. Here we have given **'Good Morning!...'** as the value.

The container instantiates the 'helloimpl' based on the XML definition as,

```
helloimpl hello = new helloimpl();
helloimpl.setGreeting("Good Morning!...");
```

Similarly, greeting property may be set through the single argument constructor of 'helloimpl' as:

```
<bean id="hello"  class="helloimpl"> <constructor-arg>
<value>Good  Morning!...</value>
</constructor-arg> </bean>
```

The container instantiates the 'helloimpl' based on the XML definition as:

helloimpl hello = new helloimpl("Good Morning...");
In the client program, '**BeanFactory'** class is used which is a spring container. Then the 'getBean()' method is called to get the reference to the 'hello'. With this reference, sayhello() method is called.

Creational Design Pattern

Design pattern includes creational, structural, and behavioral patterns. But here we are going to cover few of the creational patterns.

All the creational design patterns define the best possible way in which an object can be instantiated. These describes the best way to CREATE object instances. Creational design pattern tries to reduce the creational complexities (of an object) at its maximum level. It's not only important to understand how to use, but also equally important when to use patterns.

I. Factory Pattern:

One of the goals of object-oriented design is to delegate responsibility among different objects. This kind of partitioning is good since it encourages Encapsulation and Delegation.

A class may need it's subclasses to specify the objects to be created or delegate responsibility to one of several helper subclasses so that knowledge can be localized to specific helper subclasses. Even Sometimes, an Application (or framework) at runtime, is not able to judge properly the class of an object that it must create. The Application (or framework) may know that it has to instantiate classes, but it may only know about abstract classes (or interfaces), which it cannot instantiate. Thus the Application class may only know ***when*** it has to instantiate a new Object of a class, not ***what kind of*** subclass to create.

Creational design pattern, more specifically the Factory Pattern tries to resolve out such issues. Factory Method is a creational pattern. This pattern helps to model an interface for creating an object which at creation time can let its subclasses to decide which class to instantiate. We call this a Factory Pattern since it is responsible for "Manufacturing" an Object. It helps to instantiate the appropriate subclass by creating the right object from a group of related classes.

The Factory Pattern promotes loose coupling by eliminating the need to bind application-specific classes into the code. It is an object oriented design pattern that returns an instance of one of the several possible classes based on the data passed to it. Generally all the classes that it returns have common methods and also have a common parent class. It should be noted that all the subclasses performs the different task and is optimized for different kind of data.

A factory is not needed to make an object. A simple call to **new** will do it for you. However, the use of factories gives the programmer the opportunity to abstract the specific attributes of an object into specific subclasses which create them. The Factory Pattern is all about "Define an interface for creating an object, but let the subclasses decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses"

Benefits: Factory pattern allows the subclasses to choose the type of objects to create. One of the other benefit of factory pattern is that many of the methods created using the factory method technique do not depend on subclasses, this means that it is beneficial to define the methods using the factory method technique that are used to create objects to gain the other benefits. These methods are known as static methods.

Usage: Classes that are parallel in hierarchies usually require objects of one hierarchy for creating appropriate objects from another. Factory methods are mostly used in frameworks and toolkits where library codes required to creating the objects of specific types that may be subclassed by applications with the help of framework.

The Factory patterns can be used in following cases:

1. When a class does not know which class of objects it must create.
2. A class specifies its sub-classes to specify which objects to create.
3. In programmer's language (very raw form), you can use factory pattern where you have to create an object of any one of sub-classes depending on the data provided.

Let's try to understand this pattern with an example.

Example: Let's take an application that ask to enter the name and the Id of a person. If the person is the administrator then it displays welcome message saying Hello Mr. <Name> You are the administrator of the organization otherwise it displays a message saying Hello Mr. <Name> you are not the administrator of the organization.

The skeleton of the code can be given here.

```
public class Employee {
public String name;
private String Id = “123542”;
public String getName() {
return name;
}
public String getId() {
return Id;
}
}
```

This is a simple Employee class that contains the methods for name and Id. Now, we take two sub-classes, Administrator and NotAnAdministrator which will print a message on the screen accordingly.

```
public class Administrator extends Employee {
public Employee(String fullName) {
System.out.println(“Hello Mr. “+ fullName + “You are the
administrator of this organization”);
}
}
```

Also, the Not an Administrator

```
public class NotAnAdministrator extends Person {
public NotanAdministrator(String fullNname) {
System.out.println("Hello Mr. " + fullNname + "you are not the
administrator of this organization");
}
}
```

Now, we will create the Verification class which will return a message depending on the data provided.

```
public class Verification {
public static void main(String args[]) {
Verification verify = new Verification();
verify.getEmployee(args[0], args[1]);
}
public Employee getEmployee(String name, String Id) {
if ((name.equals("Zulfiqar")&&(Id.equals("123542")))
return new Administrator(name);
else
return new NotanAdministrator(name);
}
}
```

This class takes two arguments from the system at runtime and prints the names and a message accordingly.

Running the program:

After compilation, run the code with the arguments Zulfiqar and 123542: java Zulfiqar 123542

The result returned is: "Hello Mr. Zulfiqar You are the administrator of the organization".

II Abstract Factory Pattern:

This pattern is one level of abstraction higher than factory pattern. This means that the abstract factory returns the factory of classes. Like Factory pattern returned one of the several sub-classes, this returns such factory which later will return one of the sub-classes.

It provides a way to encapsulate a group of several related factories. This method is used when to return one of several related classes of objects and each of which have the capability of returning several objects of different types on request. This pattern provides separation from the implementation details of a set of objects from its general usage. This pattern hides the concrete subclass from the client and should be used when the system is independent of how the components are organized.

This pattern allows to interchange the concrete classes without changing the code that they uses even at runtime.

However this pattern incurs the risk of unnecessary complexity.

Benefits: The client does no need to specify the type of the concrete class because the abstract factory creates the actual concrete objects by reading the type of the concrete object from the configuration and returns the abstract pointer of those objects. The client can only access to these objects through their abstract interfaces. It defines a class library of products that provides expose to interface and creates the families of related objects as Kit. It tries to enforce the constraints and includes the related patterns. It provides independency required by the system from how its products are created, composed and support for a system or a family of systems to be extensible.

Usage: It is used to construct the complex objects that are independent of how to make up the objects and how the parts are assembled. The construction process must allow the constructed object to represent differently. It is used to make the concrete classes isolate from their super classes. The client code have no need to add the header files, class declarations and also no need to know about the concrete class. The abstract factory class creates the objects of the concrete class these objects are accessed by the client's code.

Let's take an example to clearly understand this pattern.

Suppose we need the configuration of a Computer. RAM, Hard disk and Processor are the different parts of computer and workstation Server and PC are the different types of computers.
Therefore we are taking the Computer as the abstract base class.

```
package creational.abstractfactory;

public abstract class Computer {
public abstract Parts getHarddisk();
public abstract Parts getRAM();
public abstract Parts getProcessor();
}

package creational.abstractfactory;

public class Parts {
public String configuration;
public Parts(String configuration) {
this.configuration = configuration;
}
public String getConfiguration() {
return configuration;
}

package creational.abstractfactory;

public class PC extends Computer {
```

```
public Parts getRAM() {
return new Parts("256 MB");
}
public Parts getProcessor() {
return new Parts("Pentium3");
}
public Parts getHarddisk() {
return new Parts("40GB");
}
}

package creational.abstractfactory;

public class Workstation extends Computer {
public Parts getRAM() {
return new Parts("1 GB");
}
public Parts getProcessor() {
return new Parts("Pentium4");
}
public Parts getHarddisk() {
return new Parts("80GB");
}
}

package creational.abstractfactory;

public class Server extends Computer{
public Parts getRAM() {
return new Parts("2 GB");
}
public Parts getProcessor() {
return new Parts("DualCore");
}
public Parts getHarddisk() {
return new Parts("160GB");
}
}

package creational.abstractfactory;

public class CatagoryType {
private Computer comp;
public static void main(String[] args) {
CatagoryType type = new CatagoryrType();
Computer computer = type.getComputer("Server");
System.out.println("Harddisk:
"+computer.getHarddisk().getConfiguration());
System.out.println("RAM:
"+computer.getRAM().getConfiguration());
System.out.println("Processor:
"+computer.getProcessor().getConfiguration());
}
public Computer getComputer(String catagoryType) {
if (catagoryType.equals("PC"))
comp = new PC();
else if(catogoryType.equals("Workstation"))
comp = new Workstation();
else if(catagoryType.equals("Server"))
comp = new Server();
return comp;
}
}
```

The above class gives the output like this:

```
Harddisk: 160GB
RAM: 2 GB
Processor: DualCore.
```

Service Oriented Architecture

Service Oriented Architecture or SOA for short is a new architecture for the development of loosely coupled distributed applications. In fact service-oriented architecture is collection of many services in the network. These services communicate with each other and the communications involves data exchange & even service coordination. Earlier SOA was based on the DCOM or Object Request Brokers (ORBs). Nowadays SOA is based on the Web Services. Service Oriented Architecture is basically an evolution of distributed computing. SOA provides a modularity of business logic, which can be presented as service for clients (client as in client-server architecture). These services are loosely coupled in nature, in the sense the 'User Interface' can remain completely independent of the service layer.

Loose Coupling: It is more efficient to reuse the functionality of the existing system rather than building a complete new system. It is not the task the of Tom Dick and Harry to reuse the things easily because they (building blocks) are heavily dependent on each other. There are two types of dependency. The first one is the real dependency and the second one is the artificial dependency. A real dependency is one in which one system depends on the functionality of the other system. The artificial dependency is the set of factors due to which a system accepts the features or services provided by the other system. But the problem is that we create the real dependency along with the artificial dependency. This is known as loose coupling.

Let's take an example that includes loose coupling. Suppose you are going overseas for business and you are carrying your power adapter. This is real dependency that your need power. But your adapter's plug is not getting fit into the outlet, this is artificial dependency. While looking at the power adapter you will notice that some of them are varying in shapes and sizes, while the others are big and bulky in size being used in different country.

The conclusion here is that we can not almost remove the artificial dependency from the real dependency but can be reduce up to its minimum. If we are able to reduce the artificial dependency among the systems, it means we have achieved the loose coupling. Artificial dependencies can be reduced to its minimum but the real dependency can not be altered.

Service Oriented Architecture is nothing but the architectural style to achieve loose coupling among various interactive systems. In the present time most of the organizations are implementing IT systems across their various departments. It is not a big deal, but the challenge is to find the solution that must be flexible, extensible and best fit for the existing legacy system. Its not a good idea to replace a legacy system with the new architecture and it is not only costly but also risky. For such type of problems Service Oriented Architecture (SOA) provides the best solution that is more cost effective and relatively cheap. Although it is not a new concept but SOA needs more attention with the advent of platform-neutral data models and platform-independent programs.

Characteristics of Service Oriented Architecture:

- SOA services communicates with messages by using the XML Schema. Communications among various consumers, providers and services takes place in heterogeneous environment with little or no knowledge regarding to the provider.
- SOA services act as a directory listing and are maintained by a registry within the enterprise. Applications invoke the service by looking up the service in the registry.
- Every SOA service contains the quality of service (QoS) like security requirements such as reliable messaging, authentication, authorization and policies associated with it.
- WSDL is a standard, used for describing the services as SOA services that includes the platform independent XML document containing self-describing interfaces.

Why SOA?

SOA is most widely used in the market as it links computational resources and promotes their reuse. SOA is helpful as it responds more quickly and makes cost-effective changes according to the market situations. It also simplifies the usage of the existing IT (legacy) assets and interconnection among the assets.

SOA isolates the user from the service implementation in order to run the services on various distributed platform and make it accessible across the network.

SOA architecture enables seamless Enterprise Information Integration. Here are some of the Benefits of the **Service Oriented Architecture**:

- Due to its platform independence, it allows companies to use the software and hardware of their choice.
- There is no threat of vendor lock-in.
- SOA enables incremental development, deployment, and maintenance.
- Companies can use the existing software (investments) and use SOA to build applications without replacing the existing applications.
- The training costs are low, so the available labor pool can be used for running the applications.

Techniques used by SOA to achieve loose coupling among interactive software agents:

- Distribute a small set of simple interfaces among all the participating software agents. The interface contains only generic semantics and should be available globally to all providers and consumers.

- Interfaces are used to deliver descriptive messages defined through extensible schemas (i.e. a schema defines the vocabulary and structure of messages). An extensible schema allows new versions of services to be introduced without breaking existing services. Messages are not intended to describe the system behavior.

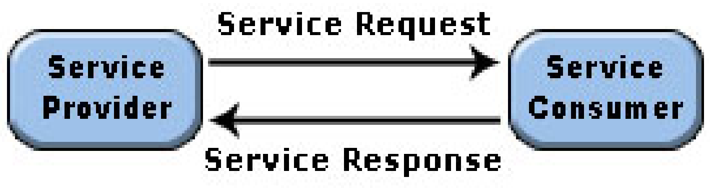
The above example illustrates that interfaces play an impor tant roll in any system. For a distributed application system interfacing is less error-prone and comparably costlier. It is quite difficult to implement without knowing the system behavior across different platforms and languages. It is better to reuse the few generic interfaces for all the applications. We can send any kind of messages over the interfaces just Service Oriented Architectur by following the few kinds of rules like:

- Since the service provider is responsible to solve any kind of problem therefore the message must be descriptive rather than instructive. It's just like going to a restaurant and place an order without bothering the cooking steps.
- Service provider will not be able to understand if the message is not properly formatted by using the standard rules. Providing the limitations to a message is the basic need to make the communication most effective. It is easier to understand as long as the message is restricted, although it is much expensive to reduce extensibility.

Broadly SOA can be classified into two terms: Services and **Connections**

Services: A service is a function or some processing logic or business processing that is well-defined, self-contained, and does not depend on the context or state of other services. Example of Services are **Loan Processing Services**, which can be self-contained unit for process the Loan Applications. Other example may be **Weather Services**, which can be used to get the weather information. Any application on the network can use the service of the Weather Service to get the weather information.

Connections: Connections means the link connecting these self-contained distributed services with each other, it enable client to Services communications. In case of Web services SOAP over HTTP is used to communicate the between ervices.
The following figure is a simple illustration of the service-oriented architecture. It shows how a service consumer sends a service request to a service provider. After accepting the request, service provider send a message to the service consumer. In this case a service provider can also be a service consumer.



Different Technologies Used:

SOA is much different from point-to-point architectures. SOA comprise of loosely coupled, highly interoperable application services. These services can be developed with different development technologies (such as Java, .NET, C++, PERL, PHP), the software components become reusable i.e. the same C# (C Sharp) service may be used by a Java application (and/or any other programming language).

SOA Terminologies

Term	**Definition**
Service (Ideally)	a self-contained, stateless business func tion which accepts one or more requests and returns one or more responses through a well-defined, standard interface. Services can also perform discrete units of work such as editing and processing a transaction. Services should not depend on the state of other functions or processes. The technol ogy used to provide the service, such as a programming language, does not form part of this definition.
Orchestration	Sequencing services and providing addi tional logic to process data. Does not include data presentation.
Stateless	Not depending on any pre-existing condi tion. In a SOA, services should not depend on the condition of any other service. They receive all information needed to provide a response from the request. Given the statelessness of services, service consum ers can sequence (orchestrate) them into numerous flows (sometimes referred to as pipelines) to perform application logic.
Provider	The function which performs a service in response to a request from a consumer.
Consumer	The function which consumes the result of a service supplied by a provider.
Discovery	Service oriented architecture relies on the ability to identify services and their capabili ties. Therefore, a SOA depends on a directory which describes the services available in its domain.
Binding	The relationship between a service provider and consumer is dynamic; it is established at runtime by a binding mechanism.

Web 2.0

Introduction

Web 2.0, a phrase is a cluster term for the new phase of World Wide Web, which was coined by O'Reilly and Media live International in 2003 and popularized by the first Web 2.0 conference in 2004. There is no certain definition of Web 2.0, even though; it stands for the transformation of the web into a full-fledged computing platform.

Web 2.0 is not a modified version of World Wide Web, but it is a different way to utilize Internet into web platform like weblogs, social book marking, wikis, podcasts, RSS feeds (and other forms of many-to-many publishing), social networking web, Web APIs, Web standards and online service provider. It is like open sourcing and genuine interactivity in which user can upload anything, download anything and can use the content according to its own wish. There is no restriction of more or less measure of content, uploading and downloading. All these are absolutely free.

According to 'O'Reilly, the inventor of Web 2.0, "Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as platform, and an attempt to understand the rules for success on that new platform". So Web 2.0 is a new way of business via Internet. It's really a new business tactic that is being used on the mass level across the world. The success of 'YouTube', 'Orkut', 'MySpace', 'Google', 'live', 'Wikipedia' and many more websites are the biggest examples of Web 2.0.

Definitions and Components

As we have already mentioned that Web 2.0 has not any specific definition. Many users have defined its in their own way. According to Wikipedia, "Web 2.0 is a term often applied to a perceived ongoing transition of the World Wide Web from a collection of websites to a full-fledged computing platform serving web applications to end users. Ultimately Web 2.0 services are expected to replace desktop computing applications for many purposes."

On the other hand, according to Wall Street Technology powered by CMP 'United Business Media', the co-inventor of Web 2.0, "Web 2.0 refers to Rich Internet Applications (RIAs) that use the Internet as a platform to create interactive user interfaces that resemble PC-based applications. Typically, RIAs emphasize online collaboration among users."

Several supporters of Web 2.0 have defined it according to their uses, observations and experiences, but in brief, we can say that:

- Web 2.0 is a conversion of websites from unique information structure having the sources of content and functionality. That's why being a computing platforms it serves web applications to end-users.
- Web 2.0 is a tendency of generating and distributing web content itself, without claiming any sort of authority on the basis of Wikis and freedom to share and re-use according to user's requirement including marketing.
- Web 2.0 is a new way of organizing and categorizing of the content, audio, video, pictures and movies highly stressing to the growth of the economic value of the Web.
- Tim O'Reilly, the father of Web 2.0 along with his colleague John Battelle summarized the key principles Web 2.0 applications in 2005. According to them:
  - The web as a platform
  - Data as the driving force
- Network effects created by an architecture of participation
- Innovation in assembly of systems and sites composed by pulling together features from distributed, independent developers (a kind of "open source" development)
- Lightweight business models enabled by content and service syndication
- The end of the software adoption cycle ("the perpetual beta")
- Software above the level of a single device, leveraging the power of the "Long Tail"
- Ease of picking-up by early adopters

Web 2.0 includes two major model move, one is 'user generated content' and other is 'thin client computing'.

User Generated Content

User generated contents refer to those content which user can upload it on the Web 2.0 based software especially social networking sites in the form of text, audio, video, pictures, movies and many more on the low level or the mass level itself. The advantage of this move is the content can spread very rapidly on the mass level and truly talented authors, artists, musicians and moviemakers can gain an audience quickly and easily that was not so easy in the past. 'Orkut', 'YouTube', 'Wikipedia' and blogs are the best examples of User generated Content Paradigm.

Thin Client Computing

Data and applications are stored on Web servers, and a user can access these from any computer through a Web browser. This is known as thin client computing. Though, it is not a new concept for the Internet, but in Web 2.0 user can access any data from the massive server through Browsers. Browsers interpret scripts in such a way, that the data are accessed extremely quick no matter which hardware or software environment they reside in. 'Google', 'Live', 'Yahoo' and 'msn' is the best examples of thin client computing.

Origin of Web 2.0

Before the origin of Web 2.0, Web 1.0 was known as a term 'Web' that was like warehouse of information and static content. Then, as time passes, with the advancement of technology and software, a huge amount of data and content became dynamic and returning custom results to users. With the evolution of new century, the Web became much more interactive. It allowed the users to play, stop, rewind and fast-forward through audio and video content. It was Web 1.5. But gradually, Web-based applications act like local applications, but on a world-wide level with the social illusion just before since last two or three years. This is known as Web 2.0

The concept of "Web 2.0" began with a conference brainstorming session between O'Reilly and MediaLive International in 2003. Dale Dougherty, web pioneer and O'Reilly VP, noted that the dotcom companies were being crashed very rapidly despite of having quality and right marketing strategy. It is assumed that something is common in all the dotcom companies that were not being distinguished by the client and being the causes of crashing. Though they observed that Web applications have a lot more than it had been used so far. They decided to do something different with web application, thus the concept of Web 2.0 has been evolved. O'Reilly had presented the feature of Web 2.0 in 2004 in a conference claiming the new version of Web. It began to popular since then.

Google, Live, Orkut, YouTube etc are the best examples of Web 2.0. In the model of Web 2.0 O'Reilly had presented that the Web 2.0 based software can do better business and are more efficient. This causes the revolution and many more web applications replacing the most common and popular Web applications e.g. Google AdSense replaced DoubleClick, Flickr replaced Ofoto, Napster replaced mp3.com, Wikipedia replaced Britannica Online, Weblogs replaced personal websites, Search Engine Optimization replaced domain name speculation, Wikis replaced Content Management System while folksonomy replaced taxonomy. There is a long list of Web 2.0 that is accepted widely. But the controversy is still going on about the definition of Web 2.0. Some people are criticizing it saying that it is a meaningless marketing buzzword while many people heartily accept it and enjoy it too.

Characteristics of Web 2.0

Though there is a controversy still going on over the definition of Web 2.0, yet it has some basic common characteristics. These include:

- Web 2.0 use network as a platform as it deliver or receive applications thoroughly via a browser.
- Users gets, manipulates and controlled the data on the site.
- Participatory architecture in which user can add or edit value to the application according to their requirement.
- A rich, interactive, user-friendly interface based on Ajax or similar frameworks.
- Some social-networking aspects.
- Enhanced graphical interfaces such as gradients and rounded corners (absent in the so-called Web 1.0 era).

Usage of Web 2.0

After emerging of Web 2.0, it is being vastly used because of its wide range of variety and very attractive features. Descriptive list of Web 2.0 tools are endless even though we can say that the new generation of Internet approximately uses its tools. Web 2.0 tools include Weblogging, Wikis, Social networking, Podcasts, Feeds, Social bookmarking, and Cascading Style Sheet.

The Approach behind using Web 2.0 is different. Some uses it accidentally as for browsing purpose. Some uses it to fulfill theirs' job because they need it. Some uses it by curiosity as they want to check it and some uses it by default as they have no knowledge about it. Overall, many people and companies use it but they don't know why? The reason may vary, but its utility is still undoubted.

Technical Overview

Web 2.0 has a complex and growing technology that includes server-software, content-syndication, messaging-protocols, standards-based browsers with plugins and extensions, and various client-applications. All these differ in functions and approaches but provide all the requirements beyond the expectation such as information-storage, creation, and dissemination capabilities.

- A web 2.0 website may usually feature a number of following techniques:
- Rich Internet application techniques, optionally Ajax-based
- Cascading Style Sheet, CSS
- Semantically valid XHTML markup and the use of Microformats
- Organization and collection of data in RSS/Atom
- Clean and meaningful URLs
- Excessive use of folksonomies (in the form of tags or tagclouds)
- Use of wiki software either completely or partially (where partial use may grow to become the complete platform for the site)

- Use of Open source software either completely or partially, e.g. the LAMP solution stack
- XACML over SOAP for access control between organizations and domains
- Blog publishing
- Mashups (A mix up of content and Audio usually from different musical style)
- REST or XML Webservice APIs.

Innovations associated with "Web 2.0"

Web-based applications and desktops

Ajax, the rich internet application technique has prompted the development of web-sites that copy personal computer applications like (M.S. Office package) word processing, the spreadsheet, and slide-show presentation while some wiki sites replicate many features of PC authoring applications. Some sites perform collaboration and project management functions. Web 2.0 also innovated various browser based operating system that works like an application platform not merely operating system as it copy the user experience of desktop operating systems having similar features and function like a PC environment. They have as their distinctive characteristic to run within any modern browser.

Rich Internet applications

The new feature included in the Web 2.0 based application in which user does not need to refresh the page, the whole page or a portion of page get refreshed automatically like in some real time web page. E.g. Cricket websites, Share Market etc. Some of the rich-internet application techniques are Ajax, Adobe Flash, Flex, Nexaweb, OpenLaszlo and Silverlight and many more.

Server-side software

Web 2.0 application server functions on existing web server architecture but strongly depend on back-end software. The weaving of software varies only nominally due to methods of publishing via using dynamic content management but web services usually need highly vigorous database and workflow support. It has analogues to traditional intranet functionality of an application server. Vendor moves towards to date fall either under a universal server approach or under a web-server plugin approach. (A universal server refers to a common server that bundles most of the necessary functionality in a single server platform while under a plugin refers to standard publishing tools enhanced with API interfaces and other tools.)

Client-Side Software

Web 2.0 provides several extra functions that a user can use according to its own ability and requirements. It can be accessed in various forms like an HTML page, Javascript, Flash, Silverlight or Java. All these methods reduce the server workload and increase the accessibility of the application.

XML and RSS

Web 2.0 supporters consider the syndication of site content as a Web 2.0 feature includes because it standardized protocols that allows users to implement data for other purpose like for using another website, a browser plugin or a separate desktop application. XML based protocols like RSS, RDF and atom allow syndication. As the popularity of these technologies increase by name of Web feed because of its high usability the RSS icon replaced by more user-friendly icons.

Specialized protocols

Social networking sites uses the specialized protocols like FOAF (Friend of A Friend) and XFN (XHTML Friends Network), which enhance the functionality of the site by allowing end users to interact directly without centralized website.

Web protocols

Web communication protocols support the Web 2.0 infrastructure. Major Web protocols are:

- REST (Representational State Transfer) provides a way to access and manipulates data on a server using the HTTP verbs GET, POST, PUT, and DELETE.
- SOAP (Simple Object Access Protocol) includes POSTing XML messages and requests to a server to follow the quite complex but pre-defined instructions.

Usually servers use proprietary APIs, even though standard web-service APIs have also been used vastly. Web service communications mostly involve some form of XML.

Besides above protocols, WSDL (Web Services Description Language) is also used for web services. The composition of WSDL with UDDI is expected to promote the use of Web services worldwide.

Web 2.0 and Language Learning Technologies

Web 2.0 technologies are new and evolving techniques for learning language, but new added features like video, file sharing, blogs, wikis, podcastingin and many more included features in Web 1.0 have made Web 2.0 very popular among the scholars, educators and students. The user of these technologies have appreciated the social networking and wikis aspect quating it as a natural helper for a constructivist learning methodology.

Here are some basic implementations of java language, which you would like to know.

Have a look at the given program below. Found anything unusual?? You got it right! The program below runs without the main method. Here we have used the static block that executes the class directly as soon as it gets loaded without creating any object for it. At the run time, JVM looks for the main method to execute but fails to get it hence terminates the execution. However before exiting from the program it will throw an exception. So use System.exit(0) to terminate the program at the end of the static block itself.

```
public class JavaProgramWithoutMainMethod{
static{
System.out.println("This java program runs without the main
method");
System.exit(0);
}
}

Output:

C:\javac>javac
JavaProgramWithoutMainMethod.javaC:\javac>java
JavaProgramWithoutMainMethodThis java program runs
without the main methodC:\javac>
```

Know to read text from Standard I/O Console

Let's see an easy way to use standard input console to read the user input through the standard I/O classes for reading text from a file or a keyboard.

In the example given below, the BufferedReader class has used its readLine() method. The BufferedReader class is the subclass of the FilterReader class. To input text from a character-input stream, the BufferedReader class uses read() and readLine() methods as shown in the example. The instance variable of the BufferedReader class reads a single line of text from the input stream as shown.

```
import java.io.*;
public class ReadIt{
public static void main(String[] args) throws IOException{
BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter text : ");
String str = in.readLine();
System.out.println("You entered a String : ");
System.out.println(str);
}
}
```

Output: Enter a string

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

Enter text :  Hello Amit
You entered a String : Hello Amit
```

Know to capture a screen shot

Hey this is very interesting, as we take a screenshot by pushing the print screen button on the keyboard, same way we can do it through java programming. For this we need to use few methods and APIs as shown in the program below. The methods and APIs, which have been used to accomplish this task, are:

createScreenCapture(): An image is created which is read and displayed at the display screen with the help of this method. This is the method of Robot class. The area is created by getDefaultToolkit().getScreenSize() method of the Toolkit class.

write(): This is the method of ImageIO class. The captured image is stored in the created image buffer and the image file for the captured image is made with the help of this method. The three arguments taken by this method are:

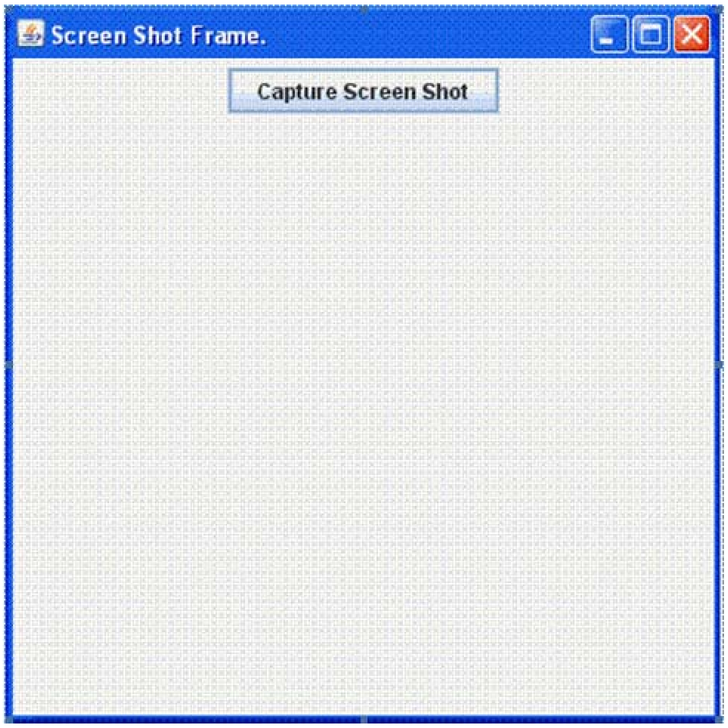
- First is the rendered image.
- Second is the file format.
- And last is the output file name.

ImageIO: This is the class of javax.imageio.* package. This class is used to read an image or can be used to write an image.

The program below displays how to capture a screen and make a "jpg" file. After the execution of this program a frame appears on the screen, which holds a command button labeled as "Capture Screen Shot". An input dialog box comes up as the "Capture Screen Shot" button is pressed. This input dialog box asks the user to type a file name, which has to be created. Then a message dialog box gets opened with message "Screen captured successfully."

```
import javax.swing.*;
import javax.imageio.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
public class Screenshot {
public static void main(String[] args) throws Exception {
Screenshot ss = new Screenshot();
 }
 public Screenshot(){
JFrame frame = new JFrame("Screen Shot Frame.");
JButton button = new JButton("Capture Screen Shot");
button.addActionListener(new MyAction());
JPanel panel = new JPanel();
panel.add(button);
frame.add(panel, BorderLayout.CENTER);
frame.setSize(400, 400);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true); }
public class MyAction implements ActionListener{
public void actionPerformed(ActionEvent ae){
 try{
String fileName = JOptionPane.showInputDialog(null, "Enter file
name : ", "Roseindia.net", 1);
if (!fileName.toLowerCase().endsWith(".gif")){
JOptionPane.showMessageDialog(null, "Error: file name must
end with \".gif\".", "Roseindia.net", 1);
}
else{
Robot robot = new Robot();
BufferedImage image = robot.createScreenCapture(new
Rectangle(Toolkit.getDefaultToolkit().getScreenSize()));
ImageIO.write(image, "gif", new File(fileName));
JOptionPane.showMessageDialog(null, "Screen captured
successfully.", "Roseindia.net", 1);
}
}
catch(Exception e){}
}
}}
```

Output:



Know to add a Node to the JTree Component

Here is a program that demonstrates how to insert a node to the JTree.

Description of the code:

getModel(): This method returns a data model. A list of items that needs to be displayed by the JList component is stored in this data model.

getNextMatch(String prefix, int startIndex, Position.Bias bias): This method returns the index of the next list element started with a given prefix otherwise would start with '1'. It takes the following arguments:
prefix: This is the string to test for matching the JTree component.
startIndex: This is an index to start the search.
bias: This indicates the search direction that can be either Position.Bias.Forward or Position.Bias.Backward.
Forward: This field indicates to bias toward the next character in model.
Backward: This field indicates to bias toward the previous character in model.
getLastPathComponent(): This method returns the last component of tree path.

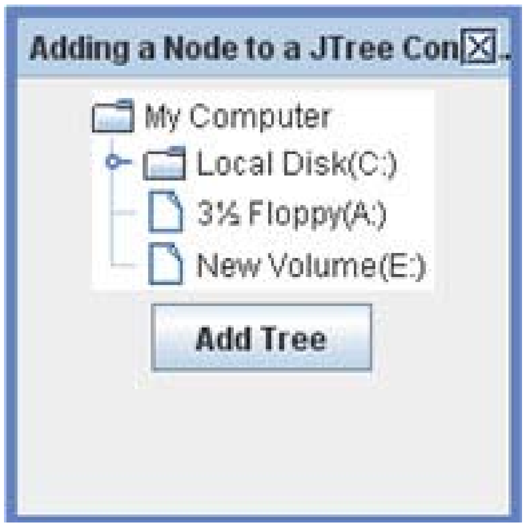
insertNodeInto(MutableTreeNode nNode, MutableTreeNode node, int index):This method is used to insert a new child at specified location.

A graphical layout appears on the screen once you run this program. It displays a tree with root node, child of root node and one command button (Add Tree). After clicking the button an input box comes up that asks for a node name to be inserted in the JTree. However if the input box is empty or blank then click the "OK" button. None of the nodes is added after clicking on the "OK" button and a message will be displayed as "Node is not added in the tree" in message box. But then if you again click on the "Add Tree" command button then an input box comes up again that asks a node name to be inserted. Hence the given node is added to the JTree when you click the "OK" button. And it displays a message "Node is added in the tree".

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.text.*;
public class AddNodes{
public static DefaultTreeModel model;
public static TreePath path;  public static JTree tree;
public static DefaultMutableTreeNode nNode;
public static MutableTreeNode node;
String nodeName;
public static void main(String[] args) {
```

```
JFrame frame = new JFrame("Adding a Node to a JTree
Component!");
JPanel panel = new JPanel();
DefaultMutableTreeNode myComputer = new
DefaultMutableTreeNode("My Computer");
DefaultMutableTreeNode c = new
DefaultMutableTreeNode("Local Disk(C:)");
DefaultMutableTreeNode vinod = new
DefaultMutableTreeNode("Vinod");
DefaultMutableTreeNode swing = new
DefaultMutableTreeNode("Swing");
DefaultMutableTreeNode tr = new
DefaultMutableTreeNode("Tree");
DefaultMutableTreeNode a = new
DefaultMutableTreeNode("3½ Floppy(A:)");
DefaultMutableTreeNode e = new
DefaultMutableTreeNode("New Volume(E:)");
c.add(vinod);   vinod.add(swing);   swing.add(tr);
myComputer.add(c);
myComputer.add(a);
myComputer.add(e);
tree = new JTree(myComputer);
JButton button = new JButton("Add Tree");
button.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent ae){
model = (DefaultTreeModel)tree.getModel();
String nodeName = JOptionPane.showInputDialog(null, "Enter
the node name:");
if(nodeName.equals("")){
JOptionPane.showMessageDialog(null, "Node is not added in
the tree!");
}
else{
//    create a new node
nNode = new DefaultMutableTreeNode(nodeName);
path = tree.getNextMatch("M", 0, Position.Bias.Forward);
node = (MutableTreeNode)path.getLastPathComponent();
model.insertNodeInto(nNode, node, node.getChildCount());
JOptionPane.showMessageDialog(null, "Node is added in the
tree!");       }      }    });
panel.add(tree);
panel.add(button);
frame.add(panel);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setUndecorated(true);
frame.getRootPane().setWindow
DecorationStyle(JRootPane.PLAIN_DIALOG);
frame.setSize(200,200);
frame.setVisible(true);
}
}
```

Output:



Know to Make a Non Resizable Frame in Java

The setResizable(): This method is also a part of JFrame class. This method is used to disable or enable the size of the frame. It takes the Boolean value i.e. true or false. If you pass the value false then the frame or window will be non-resizable and on the contrary if you pass true then it will be resizable.

```
import javax.swing.*;
public class SwingFrameNonResizable{
public static void main(String[] args){
JFrame frame = new JFrame("Non Resizable Frame");
frame.setResizable(false);
frame.setSize(400, 400);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  }}
```

Here is the Output

Know to play an Audio Clip in the Java Applet

One of the most interesting features of Java is playing the sound file. Let's see how to play an audio clip in Java Applet Viewer or within a web browser.

First create an applet Class named as "PlaySoundApplet". We have added two buttons to it, one is play button used to play the sound in a loop and the other one is stop button used to stop the sound. The play() method of AudioClip object is used to play the sound and the stop() method is used to stop the running audio clip, in between.

AudioClip class: Create an AudioClip class, which is an abstract class that can't be instantiated directly. Therefore we have used getAudioClip() method to create an object of AudioClip. This method is of Applet class. There are two versions of getAudioClip() function:

1. public AudioClip getAudioClip(URL url)
2. public AudioClip getAudioClip(URL url, String name)

The other method, which we have used here :

AudioClip = getAudioClip(getCodeBase(), "TestSnd.wav")

This method has the following versions:

1. public abstract void play() – used to play the sound only once.
2. public abstract void loop() – used to play the sound in loop.
3. public abstract void stop() – used to stop the playing sound.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class PlaySoundApplet extends Applet implements
ActionListener{
Button play,stop;
AudioClip audioClip;
public void init(){
play = new Button("  Play in Loop  ");
add(play);
play.addActionListener(this);
stop = new Button("  Stop  ");
add(stop);
stop.addActionListener(this);
audioClip = getAudioClip(getCodeBase(), "TestSnd.wav");
}
public void actionPerformed(ActionEvent ae){ ]
Button source = (Button)ae.getSource();
if (source.getLabel() == "  Play in Loop  ")
{
audioClip.play();
}
else if(source.getLabel() == "  Stop  "){
audioClip.stop();
}  }}
```

The HTML code is given below:

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

Advertise With Us

Advertise with JavaJazzUp Network

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 × 80px	5,00,000	USD 2,000
Box Banner	125 × 125px	5,00,000	USD 800
Banner	460 × 60px	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.

Contact at: deepak@roseindia.net

Readers Forum: Get Involved

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 Reader's Forum

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

Convene a discussion on a specific subject

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

Sharing Expertise on Java Technologies

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

Show your innovation

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

Hands-on technology demonstrations

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

Present a question, problem, or puzzle

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

Post resourceful URLs

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

Anything else

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

RoseIndia

Please Send Your Query, Problems & Valued Suggestions at :
editor@javajazzup.com

log on to http://www.allcooljobs.com/

NEWSTRACK india

Freedom to Express

http://www.newstrackindia.com