

Oct 07 | Volume 1 | Issue 4 | Free

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING

Java ME

Jboss Seam

James Gosling: Father of Java

JavaFX



JAXB

Maven2

Tomahawk Tags

**Rich Internet
Application**

Advertise With Us



A global leader in Services...

Software Solutions, Web Development, Web Designing, Search Engine Optimization, E-Commerce Solutions, Customer Relationship Management, Web Redesigning, Web Promotion, Search Engine Re-submission, Web Hosting, Linux Hosting, Windows Hosting, E-mail Hosting, Outsourcing, Content Development, Technical Documentation, Online Free Education.

Visit us at <http://www.roseindia.net/services/>

" Just don't do because others are doing, Give it a thought....
Still circumstances want you to do the same
Just explore a smarter way to get the deeds in perfection "

Published by

RoseIndia

JavaJazzUp Team

Editor-in-Chief

Deepak Kumar

Editor-Technical

Ravi Kant
Noor-En-Ahmed

Sr. Graphics Designer

Suman Saurabh

Jr. Graphics Designer

Amardeep Patel
Santosh Kumar

**Register with JavaJazzUp
and grab your monthly issue
"Free"**

Editorial

Dear Readers,

We are here again with the fourth issue of Java Jazz-up. The current edition highlights the interesting Java technologies in form of articles developed by the Java Jazz-up Developer's Team. This issue reflects our consistent attempts to avail the quality technological updates that enforce the readers to appreciate a lot and be a part of its Readers Community.

With this issue, we have begun few of the new sections like Editorial Choice, Java ME and Know The Champions. The Editorial Choice section highlights the editor's viewpoint orienting around the java innovations in diverse spheres of human interest. This section talks high of JavaFX Technology and tries to avail its features which aims to provide a consistent user experience across a wide variety of devices including desktops (as applets and stand-alone clients), set-top boxes, mobile devices and Blu-Ray players.

Java ME section highlights the role of java in the development of softwares for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.

Next, newer section is "Know The Champions" which highlights the "Java Champions program" sponsored by Sun Microsystems. This program is an effort to recognize leaders in the Java Community and invite them to participate in the development of the Java platform in collaboration with Sun engineers and Java Luminaries.

Set of articles discussing technologies like Maven2, Design patterns, JSF Tags, web services, spring framework etc. are provided in such a manner that even a novice learns and implements the concepts in a easy manner.

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

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

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

Editor-in-Chief

Deepak Kumar
Java Jazz up

Content

- 05 [Java News](#) |** Java Around the Globe- Sun adds features to the Java platform's existing security by announcing two new Java SE security features.
- 07 [New Releases](#) |****[Hibernate Search 3.0 available:](#)** Now the developers can take advantage of advanced search capabilities like Google without extra infrastructure coding with the help of Hibernate Search 3.0.
- 09 [JavaFX](#) |** The Java revolution, which started more than a decade ago, gains even more momentum with the arrival of JavaFX.
- 14 [Java Platform Micro Edition](#) |** It is a natural phenomenon to think about learning and adopting new technologies while there exists some well-established and popular ones.
- 22 [Know the Java Champions: James Gosling](#)|**The Java Champions program was launched in June 2005 at the JavaOne Conference in San Francisco.
- 26 [JBoss Seam: Web 2.0 Applications](#) |** JBoss Seam is a powerful new application framework developed by JBoss, a division of Red Hat for building next generation Web 2.0 applications.
- 31 [Maven 2 Eclipse Plug-in](#) |** Plugins are great in simplifying the life of programmers; it actually reduces the repetitive tasks involved in the programming.
- 35 [Tomahawk Tags](#) |**Tomahawk tags are collection of standard components with extended functionality and many more extra set of components with rich set of functionality.
- 41 [Remoting with Spring](#) |** Spring features remoting support using various technologies. Remoting support eases the development of remote-enabled services, implemented with usual (Spring) POJOs.
- 46 [Java Architecture for XML Binding](#) |** Today, XML has emerged as the standard for exchanging data across disparate systems, and Java technology provides a platform for building portable applications
- 50 [Structural Design Patterns](#) |** Structural Design Pattern establishes a relationship between the two unrelated interfaces such that they work together.
- 55 [Develop - JSF Application](#)|** This section is very useful for any beginner in the field of JSF (Java Server Faces) framework of Java. This example covers all you need to develop the application, for example, using JSF tags, creating properties files and managed beans, modifying configuration files like faces-config.xml and web.xml, directory structure of the application etc.
- 62 [Rich Internet Application](#)|** The term RIA (Rich Internet Applications) refers to web applications that have the features and functionality of traditional desktop applications, it means Rich Internet Applications are a cross between web applications and traditional desktop applications that shift some of the essential processing among the bulk of the data for the user interface to the Web client while rest of some remain on application server.
- 68 [Tips & Tricks](#)|** You must have worked with Notepad to write programs. Now its turn to create notepad by own with the help of java language.
- 76 [Advertise with Us](#) |** We are the top most providers of technology stuffs to the java community.

JAVA AROUND THE GLOBE

Sun Advances Security for the Java SE Platform

Sun adds features to the Java platform's existing security by announcing two new Java SE security features. These features strengthen Java as the most widely used and secure software platform. These two new features are "synchronized release of Java SE security fixes" and "advance customer notification of those releases". Sun will begin delivering synchronized security releases of current and legacy versions of Java. Although user is recommended to use latest release of Java to get the latest features but now for those who works on older versions of Java platform have the opportunity to install synchronized updates of security fixes that are same as in the latest release. Sun also planned to provide advance notification of its Java SE security updates on the Sun Security Blog up to a week before its release. It will contain summarized information of the release and the expected time to release.

Microsoft and Sun Expand Strategic Alliance

Now Sun Microsystems and Microsoft, two big names in the field of computer world, have decided to work together on a support process for customers who are using the virtualization solutions. Both together will ensure that Solaris will run well as a guest operating system in Microsoft virtualization technologies and the same for Windows Server on Sun virtualization technologies. "Sun is now a single source for today's leading operating systems - Solaris and Windows - on the industry's most innovative x64 systems and storage products. Customers can now take advantage of the virtualization benefits of Windows and Solaris on Sun's energy efficient x64 systems," said John Fowler, executive vice president, Systems Group, Sun Microsystems. "Today's announcement is another example of Microsoft's commitment to 64-bit computing," said Bob Muglia, senior vice president, Server and Tools Division at Microsoft. "The Sun hardware platform is an excellent foundation for Windows-based enterprise solutions, such as Microsoft Virtual Server, Microsoft SQL Server, Microsoft

Exchange Server and Microsoft Internet Protocol Television (IPTV) Edition. Our customers will have an additional choice of Windows Server OEM partners with Sun". Both jointly announced this news during a press conference on September 12, 2007.

Linux, Java entertain in flight

Now in-flight entertainment system used in corporate jets will use Java software on the Aonix PERC Ultra virtual machine on top of MontaVista Software's Mobilinx operating system and the Freescale i.MX31 multimedia applications processor. Aonix is the provider of the PERC product line for embedded and real-time Java developers. Aonix announced in the "Embedded Systems Conference", Boston on 18 September that PERC has been selected to be used in-flight systems.

PERC Ultra is a virtual machine and toolset for embedded and real-time systems supporting J2SE. It supports J2SE efficiently and maintains integrity, performance or real-time behavior. Support for Java 5 SE was an important factor to choose PERC Ultra as best Java solution candidate because it offers required capabilities for OpenGL and OSGi. PERC offers AOT and JIT compilation, remote debug support, deterministic garbage collection, standard graphics and extended commercial RTOS support.

Freescale and Sun Microsystem exploring new embedded market

Freescale Semiconductor Inc., a global leader in the design and manufacture of embedded semiconductors for the automotive, consumer, industrial, networking and wireless markets is planning **to drive Java® Technology deeper into embedded markets** with Sun Microsystem. Their joint effort will provide implementation of Java SE for embedded that is optimized to leverage the outstanding hardware acceleration, gigahertz performance, and ultra-low power capabilities of the PowerQUICC III processor family. This implementation is designed for high-

JAVA AROUND THE GLOBE


performance embedded applications where PowerQUICC processors are needed like robotic systems, military avionics systems, multi-function printers (MFPs) and medical imaging equipment.

Microsoft Patch for Java VM Applet Vulnerability


Now the Internet Explorer is free from security hole that affects the JVM in version 4x and 5x in the Windows 95, 98, NT and 2000 environment. This patch will not allow the hacker to gather personal login information through Java applet on a bobby-trapped site.

FBI looks to Java to streamline wiretap requests

Now it's the turn of US investigating giant FBI on a massive shift of the existing system National Security Letter (NSL) built with Microsoft Access database management software. Now the same functionality is being implemented with Java Enterprise Edition application server using Oracle software targeted to track National Security Letter (NSL) wiretap cases.



*We are the best in
Shaping the
Java Technology Stuff*



<http://www.roseindia.net>

Grow your business with our

SERVICES

Software Solutions

E-Commerce Solutions

Website Development

Web Promotion

Web Hosting

Content Development

 **RoseIndia**
<http://www.roseindia.net/services/>

New Releases

Hibernate Search 3.0 available: provides full-text search

Now the developers can take advantage of advanced search capabilities like Google without extra infrastructure coding with the help of Hibernate Search 3.0. This hibernate search is using Apache Lucene internally and provides full text search capabilities to Hibernate-based applications. Hibernate Search can be integrated in applications with advanced features like query filter and index sharing. It solves the problems of structural mismatch, duplication mismatch, the API mismatch. It works well in clustered and non-clustered mode, synchronous and asynchronous index updates.

Workflow and scheduler Flux adds embedding capability for web Apps

Flux Corporation has announced the release of Flux 7.5. Flux is an embeddable Java job scheduler, file transfer, workflow and BPM (Business Process Management) engine that can be embedded in web applications using a JavaScript widget.

"By using Flux's embeddable designer, Java developers can meld rich, interactive job, workflow, and BPM design capabilities seamlessly into their Web applications," said David Sims, President of Flux.

RSF 0.7.2 released

RSF (Reasonable Server Faces) version 0.7.2 has been released. RSF is an open source Java web framework that is built on spring framework. There are number of existing Java frameworks managing the stateful components in complex way but RSF promotes zero server state designs and promotes minimal and clean designs in compare to other Java frameworks. RSF 0.7.2 features complete integration with Spring Web Flow, porting SWF flow to a pure-HTML and pure-Spring environment, combining SWF's robust state management with RSF's strong markup focus.

HDIV 2.0, Web Application Security Framework, support Spring MVC

HDIV 2.0 is currently released version of an open source Java web application security framework. Previously HDIV supported Struts 1.x and Struts 2.x, however this release offers its availability for frameworks like Spring MVC 2.0 and JSTL 1.1 also by extending the Spring and JSTL tags. HDIV prevents web application vulnerabilities like SQL injection, cross-site scripting, and parameter tampering.

Netbeans 6.0 beta and Glassfish V2 released

Sun has announced the release of GlassFish V2, the next version of open source Java EE 5 application server and the availability of NetBeans 6.0 beta IDE. GlassFish V2 provides features required for scalable and mission critical deployments. This release adds enterprise features like clustering, centralized administration, project metro, open ESB, NetBeans IDE integration. NetBeans 6.0 Beta provides improved productivity by an integrated and configurable IDE. It supports for dynamic language like Ruby, JavaScript. It adds multiple language support, much faster editor, local history of file changes, and integrated support for Subversion, and a built-in profiler. "Together, GlassFish and NetBeans offer a feature-rich, low-cost platform for developing and delivering applications that meet the demanding needs of the enterprise," said Karen Tegan Padir, vice president, Software Infrastructure, Sun.

Jerry Messenger Server 1.01 released - XMPP server

Jerry Messenger Server is now available with its new version 1.01. It's a combination of XMPP/ Jabber server and a web server, which allows live chatting for the web site on the server as a standalone application. You can be online with site visitors every time through mobile phone also. This messenger supports multiple accounts where each may have several operators. Accounts can be created for different

New Releases

departments of your organization and for each of your web sites. It's fully customizable messenger. Some of its features can be listed as: XMPP based messaging protocol, Any Jabber compatible operator software, multiple account support, private and secured live chat messaging, customizable operator loading, message log, online presence management, multilingual, fully customizable, extensible and cross platform.

Open Source Business Library (OSBL) 1.0 released

Open Source Business Library (OSBL) 1.0 has been released. It covers everything required to build sophisticated master data management applications and process based applications with services, relational persistence, a web UI, authentication and authorization, identity management and more. The goal of OSBL is to provide a solution that efficiently supports work processes in a business environment. Several open source high quality components are available in the market but they individually fail to achieve a solution mainly when integrating into a whole. OSBL reaches to the aim of achieving this integration.

Profiler YourKit 7.0 released

YourKit 7.0, a profiler for Java and .NET, has been released with set of new features. Some of them are: automatic memory inspections, object generations that is very helpful in finding memory leaks, ability to compare snapshots from different runs, monitor profiling, Better detection of local profiled applications, internal improvements and optimizations, improved reliability and performance.



A vertical banner with a green background and a blue border. The text 'Get reliable SERVICES' is at the top. Below it is a list of services with arrows pointing to the right. At the bottom is the RoseIndia logo and website URL.

Get
reliable
SERVICES

- Software Solutions
- E-Commerce Solutions
- Website Development
- Web Promotion
- Web Hosting
- Content Development

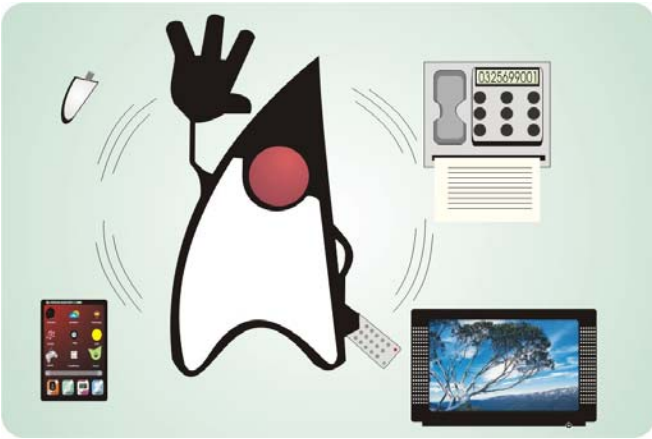
 **RoseIndia**
<http://www.roseindia.net/services/>

Easy to learn Java
Basic codes and simple Language
Updated new examples

Java Jazz up
A BETTER WAY TO LEARN PROGRAMMING

JavaFX

JavaFX: New Paradigm in Rich Internet Applications



“Most scripting languages are oriented at banging out Web pages. This is oriented around interfaces that are highly animated.”

—James Gosling, ^[1]

“There are parts of the world where a person’s desktop computer is their cell phone, and that’s the kind of end point that we’re going to get to.” —James Gosling, ^[2]

It is a natural phenomenon to think about learning and adopting new technologies while there exists some well-established and popular ones, it is the scenario prevailing with the rich internet application (RIA) development landscape. There has been a constant demand for RIAs to provide interactive content applications and services that would run on a variety of clients with new features and capabilities. RIAs are basically the web applications that have the features and functionality of traditional desktop applications. They typically transfer the processing necessary for the user interface to the web client but keep the bulk of the data (i.e., maintaining the state of the program, the data etc) back on the application server.

To simplify and speed up the creation and deployment of high-impact content for a wide range of devices, Sun Microsystems announced JavaFX, a family of products based on Java

technology to create Rich Internet applications (RIAs).

JavaFX: Sun’s New Product Family and Technologies

The Java revolution, which started more than a decade ago, gains even more momentum with the arrival of JavaFX. It is a new innovation targeting the billions of consumer devices and computers powered by Java technology.

JavaFX comprises a comprehensive set of runtime environments, widgets, development tools, and scripting environments. It aims to provide a consistent user experience across a wide variety of devices including desktops, (as applets and stand-alone clients) set-top boxes, mobile devices and Blu-Ray players.

Sun Microsystems first announced JavaFX at the JavaOne developer’s conference in May 2007. The JavaFX products are intended to create Rich Internet applications (RIAs). Currently JavaFX consists of JavaFX Script and JavaFX Mobile (an OS for mobile devices), although further JavaFX products are planned. Sun plans to release JavaFX Script as an open source project, but JavaFX Mobile will be a commercial product available through an OEM license to carriers and handset manufacturers.

Now, JavaFX is going to compete with Adobe AIR and Microsoft’s Silverlight technologies to occupy space in the current RIA market.

Emergence of JavaFX

JavaFX began as a project by Chris Oliver called F3 which stands for “Form follows function”, and its purpose was to explore making GUI programming easier in general.

F3 attempted to demonstrate that we’re not exploiting the full capabilities of the Java platform for GUI development. Taking together the supporting tools like F3, Java platform is highly competitive with or superior to competing GUI development platforms such as Macromedia Flash/Flex/Open Laszlo, Adobe Apollo, Microsoft WPF/XAML, Mozilla XUL, AJAX/DHMTL.

JavaFX

At the 2007 JavaOne Conference, Sun introduced two products in the JavaFX family: JavaFX Script and JavaFX Mobile.

JavaFX: A big picture



Fig: JavaFX Video Player

From smart cards to mobile phones to enterprise applications and supercomputers, Java technology has become one of the world’s most significant and pervasive platforms. Java technology truly is everywhere. The JavaFX family will make it easier than ever to build and quickly deploy rich Internet applications and interactive content on clients ranging from the browser to devices.

The JavaFX product family leverages the Java platform’s write-once-run-anywhere portability, application security model, ubiquitous distribution and enterprise connectivity

Today’s Internet offers a world of possibility for those who can quickly develop and deploy rich internet applications (RIAs). But only the Java platform is pervasive enough on mobile devices and browsers to effectively marry client- and browser-based technologies with RIAs enabling applications to run on multiple platforms virtually unchanged. JavaFX is Sun’s new product family that addresses this market. JavaFX Script will enable developers to more quickly and easily develop RIAs and next-generation services that can be proliferated across virtually any device — from desktop browsers and mobile devices, to set-top boxes and Blu-ray Discs — securely and without local installation. JavaFX Mobile software makes these types of applications a reality for the mobile world.



JavaFX

I. JavaFX Mobile

JavaFX Mobile is a complete mobile operating and application environment built around Java and Linux open source technologies.

It is a complete, fully integrated Java software system for advanced mobile devices designed to enable developers to author rich, high-impact content and network-based services. Built around open and standards-based technologies, JavaFX Mobile enables control and flexibility for the mobile ecosystem.

Being centralized around Java technologies, JavaFX Mobile software system provides a greater scalability and portability, speed time-to-market, and enhances the consistency of applications and services. Additionally, it also provides a support for Java ME applications and other standard Java APIs which enables a broad range of new and existing Java applications.

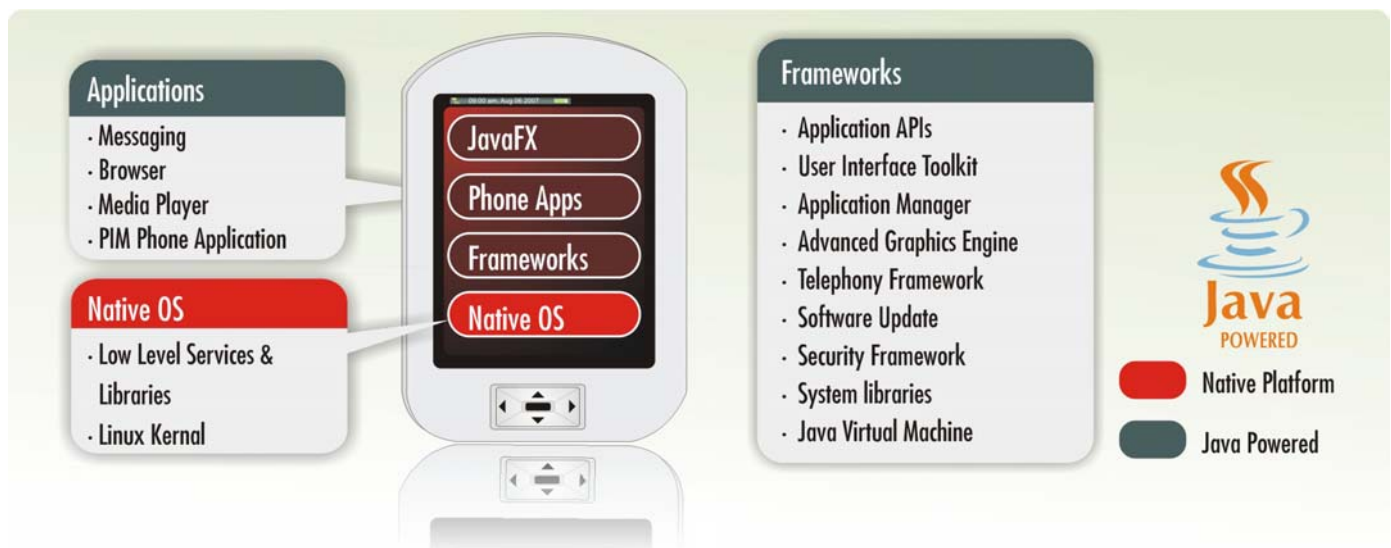


It is like the networking in your hand. It is based on open standards. JavaFX Mobile runs on all mobile phones with Java support e.g. Nokia N800.

JavaFX Mobile, Sun's software system for mobile devices, is available via OEM (i.e. Original equipment manufacturers) license to carriers, handset manufacturers and others seeking a branded relationship with consumers. Through OEM multiple companies can simplify and accelerate the development of powerful standardized software systems to leverage the power across a wide range of consumer devices.

Additionally, It allows content creators to create rich media content without relying on developers, allowing the drag and drop of desktop and mobile content, which is not possible with any other RIA.

Get Ready for Dynamic Interactive Content on Any Device: It is unpredictable to guess what devices or platforms the end user is using; however with JavaFX, you don't have to worry about the things. JavaFX Mobile uses industry standard technologies—this means that applications built with JavaFX can run on a wide range of Java-enabled devices, and content providers can add them to their devices quickly. JavaFX Mobile includes the latest standards, like the Mobile Services Architecture, set of device APIs, which allows developers to have a rich set of highly portable capabilities.



JavaFX

End users always look for an exciting and dynamic content on the web and expect a better interactive experience with web services. JavaFX helps in delivering visually compelling applications, such as maps and mashups, video, audio, and pictures, that is standardize across cell phones, TVs, and more.

Reduced Integration Costs with Expanded Opportunities: Implementation of the majority of the solution in Java, including middleware and resident applications, JavaFX Mobile reduces integration costs, improves device software consistency, and enables device manufactures to provide new offerings with substantially faster time-to-market.

JavaFX Mobile Architecture:

JavaFX Mobile streamlines the environment and reduces reliance on underlying technologies by providing a complete middleware and application stack implemented in Java.

II. JavaFX Scripting Language

JavaFX Script is specifically designed to optimize the creative process of building rich and compelling UIs leveraging Java Swing, Java 2D and Java 3D for developers and content authors.

In layman style - JavaFX lets you enjoy a consistence user experience irrespective of whatever device you are currently online with, whether you are sitting in front of your desktop, whiling away commuting time with your PDAs, or relaxing at home.

JavaFX Script is a highly productive scripting language for content developers to create rich media and interactive content for deployment on Java environments. Since JavaFX Script is statically typed, it has the same code structuring, reuse, and encapsulation features that make it possible to create and maintain very large programs using Java technology. It gives Java developers the power to quickly create content-rich applications for the widest variety of clients, including mobile devices, set-top boxes, desktops, even Blu-ray discs. Content creators now have a simple way to develop content for any Java Powered consumer device.

With JavaFX, the start-ups, enterprises and developers are free from issues like local installation and performance degradation, they can now quickly develop and deploy new secure services for a variety of clients. This is going to simplify the development of RIAs running across a range of platforms.

JavaFX technologies being built around open standards, offer consistency for apps and services across different platforms.

Applications written with JavaFX Script have WORA (write-once-run-anywhere) features and application security support with enterprise connectivity.

JavaFX Script is easier to understand and maintain, above all the structure of the written code closely matches the actual layout of the GUI. JavaFX Script enables rapid development of rich 2D interfaces in an easy fashion.

JavaFX Script offers an advantage of the Java security model so that the consumers can have a secure access to the assets (e.g., pictures, music files, word documents) on their desktop.

The write once, run anywhere portability of Java technology has helped to make it the world's most widely deployed application platform.

Features:

- JavaFX Script is going to work with all major IDEs, including NetBeans.
- JavaFX Script is capable of supporting GUIs of any size or complexity
- JavaFX Script makes it easier to use Swing, one of the best GUI development toolkits of its kind.
- JavaFX Script uses a declarative syntax for specifying GUI components, so a developer's code closely matches the actual layout of the GUI.
- Through declarative data-binding and incremental evaluation, JavaFX Script enables developers to easily create and configure individual components by automatically synchronizing application data and GUI components.


JavaFX

Benefits with JavaFX Script

- Increases developer productivity
- Zero loss of functionality across devices
- Requires less code
- Enables faster development cycles
- Offers an intuitive language design

Download the JavaFX Code, Join the Community

Go to openjfx.org to join the JavaFX community and download the JavaFX Script alpha code. Contribute to it and participate in Sun's ongoing enhancement of the new family of Java products



All Cool Jobs

The Best job Search Engine Site

Free Membership
Log on to search now!

<http://www.allcooljobs.com/>

All Cool Jobs

This advertisement features a woman sitting on the floor using a laptop. The text is arranged vertically, with the company name at the top, followed by a tagline, a call to action for free membership, a website URL, and the company name again at the bottom.



Java News...

- Java Around the Globe
- Updated Releases
- API Updates
- and much more...

Java Jazz up
A BETTER WAY TO LEARN PROGRAMING

This advertisement is split into two sections. The top section, titled 'Java News...', lists various topics covered by the news service. The bottom section, titled 'Java Jazz up', promotes a learning resource with the tagline 'A BETTER WAY TO LEARN PROGRAMING'. The top image shows a newspaper titled 'Java News' with a Java logo above it.

Java Platform Micro Edition

It is a natural phenomenon to think about learning and adopting new technologies while there exists some well-established and popular ones. It is the scenario prevailing with the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.

Today, Java interferes every sphere of life with its incredible variety of platforms and APIs like Java SE, Java EE, Java ME, Java FX, internet TV, Telephony, embedded Systems and a lot more. Java started small, aimed at television set top boxes and other interactive devices. As soon as it aimed toward web browsers and applets, it got the wings to explore the unlimited horizons. As a result, the platform got all kinds of amazing features like Swing, Java 2D, Java 3D, JDBC, EJB, and so on. And in a very short span of time, Java with its diverse specifications start accommodating the wide variety of device capabilities and features.

From smart cards to mobile phones to enterprise applications and supercomputers, Java technology has become one of the world's most significant and pervasive platforms. Java technology is truly everywhere.

Emergence of Java ME

As time and technology moved on, Sun recognized the need to collect the device-oriented platforms under one umbrella. At JavaOne in 1999, Sun introduced the Java 2 Micro Edition. J2ME (now Java ME) is not a specific virtual machine, API, or specification. Instead, J2ME provides a modular, scalable architecture to support a flexible deployment of Java technology to devices with diverse features and functions.

In computing, the Java Platform, Micro Edition or Java ME is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.

Java ME was designed by Sun Microsystems and is a replacement for a similar technology,

PersonalJava. Originally developed under the Java Community Process as JSR 68, the different flavors of Java ME have evolved in separate JSRs. Sun provides a reference implementation of the specification, but has tended not to provide free binary implementations of its Java ME runtime environment for mobile devices, rather relying on third parties to provide their own. As of 22 December 2006, the Java ME source code is licensed under the GNU General Public License, and is released under the project name phoneME.

Java ME has become a popular option for creating games for cell phones, as they can be emulated on a PC during the development stage and easily uploaded to the phone. This contrasts with the difficulty of developing, testing, and loading games for other special gaming platforms such as those made by Nintendo, Sony, Microsoft, and others, as expensive system-specific hardware and kits are required.

Java ME: Usage

Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. Applications based on Java ME are portable across many devices, yet leverage each device's native capabilities.

Java ME devices implement a profile, the most common of these are the **Mobile Information Device Profile** aimed at mobile devices, such as cell phones, and the **Personal Profile** aimed at consumer products and embedded devices like Set-top boxes and PDAs. Profiles are subsets of configurations, of which there are currently two: **the Connected Limited Device Configuration** and **the Connected Device Configuration**. A Java ME "configuration" targets devices with a specific range of capabilities. A "profile" selects a configuration and a set of APIs targets a specific domain of applications. Selection of the best configuration and profile enables a vendor to produce a wide range of flexible applications. The lightweight appliances do not need to support the entire Java platform which promotes the use of

Java ME

modular extensions of the platform allowing the vendors to differentiate themselves by producing innovative applications and incorporating value-added features.

Java ME Platform, technology alone provides the true open solution for building secure mobile applications for the industry. It allows the portability of applications between the platforms and simultaneously keeps the investments to a minimum through the possibility of reuse. The increasing demands for capabilities and performance in the industry drives the continuous java platform evolution which is assured through the definition of the platform components and APIs in the Java Community Process. The community of developers creating applications for the Java ME platform is large and increasing at a good pace because it is open for everyone to use.

Quick Glance: Java ME Technology Ecosystem

Being open to all , Java ME assures the continuous improvement and availability of applications for different platforms which in turn drives business for everybody involved in the eco-system. This ecosystem is evolving around a number of different players in the industry, all of them participating in, and influencing, the continuous improvement of the technology and platform.

The **end users** are the drivers of the system who constantly demand new features and capabilities to enhance the prevailing services. This forces the content developers, OEMs and Carriers to work in sync and generate the best result to meet the demands.

The **content developers adopt** the user requirements and creates new appealing services with new capabilities.

The **OEMs** (i.e. Original equipment manufacturers) creates new capable devices to host the new services and features and also creates new demands by presenting new capabilities to the end users.

Carriers create the mobile environment to host and deploy services on and also drives the exploration of new business-driving services to the end users. This constant evolution of demands and capabilities is the single most important reason for the success of the Java platform and ensures it will continue to evolve into the future needs of everyone involved in the eco-system.

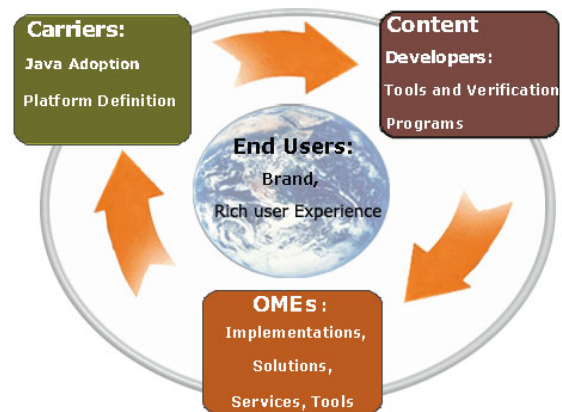


Fig: Java ME Technology Ecosystem

Java ME platform consists of a number of specified components. It has been defined, and are continuously improved and developed, by the industry through the Java Community Process. This ensures the ongoing evolution and adoption of the platform by inviting all players in the industry to participate in the definition of the platform and its capabilities.

The **Mobile & Embedded** community site establishes a central location for the collaborative development of open source Java ME technologies and applications. Deployed in over 1.5 billion mobile and embedded devices, Java ME represents the ideal development platform for the creation and deployment of mobile data services. By open sourcing implementations of Java ME, Sun will enable the community to accelerate platform innovation, reduce development costs through the Java ME ecosystem, and, ultimately, drive a more consistent application platform.

Getting Hands On to Java ME

Writing a Java ME application is different from

Java ME

writing the classical java application. Though it uses the same basics programming constructs as used with Java SE applications. Java ME platform provides a better space to develop games and applications for small devices like PDA's and handheld devices.

Basically there are two types of configurations involved in Java ME application development which are:

- CLDC (Connected Limited Device Configuration)
- CDC (Connected Device Configuration)

Architecture

Java ME architecture consists of layers highly compatible with the native environment (i.e. OS) of the device. These layers are collectively known as the Connected Limited Device Configurations (CLDC). The CLDC installed on the OS forms a fine environment for small computing device. The Java ME Architecture comprises of three software layers:

- The first layer is the **configuration layer** that includes the JVM, which directly interacts with the native OS. The Configuration layer also handles the interaction between the profile and the JVM.
- The second layer is the **profile layer** which consists of the minimum set of application programming interface (API) for the small devices.
- The third layer is the **Mobile Information Device profile** (MIDP) layer. The MIDP layer contains java APIs for user network connections, persistence storage, and the user interface. It also has access to CLDC libraries and MIDP libraries.

Connected Limited Device

Configuration (CLDC): This Configuration is much popular among the developers community to build the Java ME applications. CLDC defines

the core set of API and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream personal digital assistants.. There are two versions of CLDC: First one is CLDC1.0 which was released in 2000 and very soon it was termed as Java Specification Request (JSR) 30. Second one is CLDC1.1 or more specifically as JSR 139 However 1.0 is much popular in use. When CLDC is coupled with a profile such as the Mobile Information Device Profile (MIDP), it provides a solid Java platform for developing applications to run on devices with limited memory, processing power, and graphical capabilities.

Connected Device Configuration (CDC):

Developed under the Java Community Process (JCP), it is a standard framework of Java technology used for building and delivering application that can be shared over a wide range of networks and devices ranging from pagers, mobile phones, set top boxes and other PDA devices. It comes in two flavors: First one is JSR 36 (CDC 1.0) and second one is JSR 218 (CDC 1.1).

Mobile Information Device Profile (MIDP):

The MIDP specification was defined through the Java Community Process (JCP) by an expert group of more than 50 companies, including leading device manufacturers, wireless carriers, and vendors of mobile software. MIDP provides a standard Java runtime environment for various mobile devices. It defines a platform for dynamically and securely deploying optimized, graphical, networked applications.

MIDP, when combined with the CLDC, provides the Java runtime environment for the compact mobile information devices, such as cell phones and mainstream PDAs. Developers can develop application once and then redistribute them into various mobile information devices in a very small period of time with help of MIDP. Its principal functions include to provide the user interface, network connectivity data storage and overall application process management. The Mobile Information Device Profile (MIDP) is a key element of the Java Platform, ME.

There are two versions of MIDP: First one MIDP 2.0 or JSR 118 and second one MIDP

Java ME

1.0 or JSR 37.

The Java ME Application Development

I. System Requirements - Hardware

Minimum hardware requirements are:

- 100 MB hard disk space
- 128 MB system RAM
- 800 MHz Pentium III CPU

II. Minimal Software Requirement

- IDE – Sun ONE Studio 4, Mobile Edition, (formerly Forte for Java)
- GUI – Sun Java ME Wireless Toolkit 2.5.1 (WTK 2.5.1) for CLDC

For Windows:

Download the Sun Java Wireless Toolkit for CLDC from <http://java.sun.com/products/sjwtoolkit/download.html>. Ensure that you have installed an appropriate Java SE environment.

Run the installer, sun_java_wireless_toolkit-2_5_1-windows.exe. Follow the instructions provided by the installer.

For Linux:

Download the Sun Java Wireless Toolkit for CLDC from <http://java.sun.com/products/sjwtoolkit/download.html>. Ensure that you have installed an appropriate Java SE environment.

Run the installer, sun_java_wireless_toolkit-2_5_1-linux.exe. Follow the instructions provided by the installer.

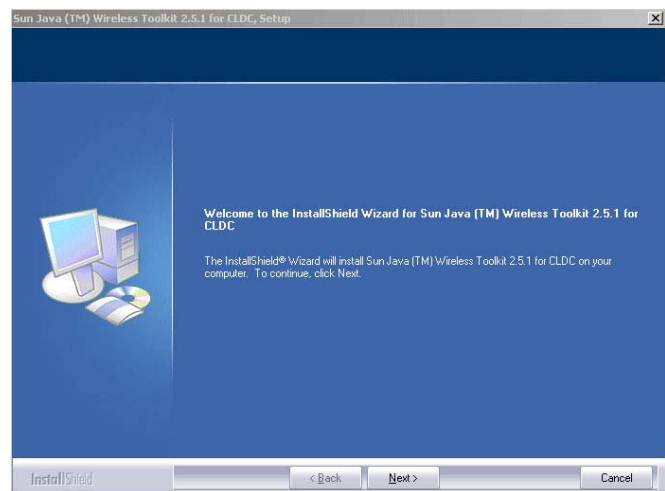
III. Steps to install Sun Java ME Wireless Toolkit 2.5.1 (WTK 2.5.1) on the Windows platform.

- Download the installer file i.e. netbeans_mobility-5_5_1-win.exe
- Double Click the icon of downloaded exe.



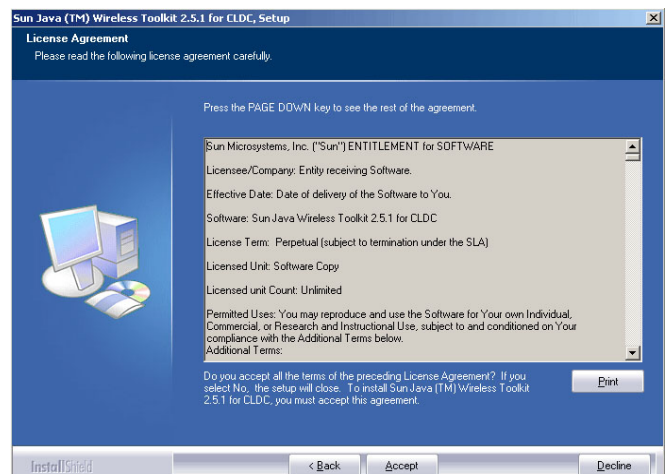
You will see a similar window as shown below.

Step 2: Now a welcome page of the install Shield wizard for Sun Java(TM) Wireless Toolkit 2.5.1



for CLDC appears, click the button with label "Next". It is shown below

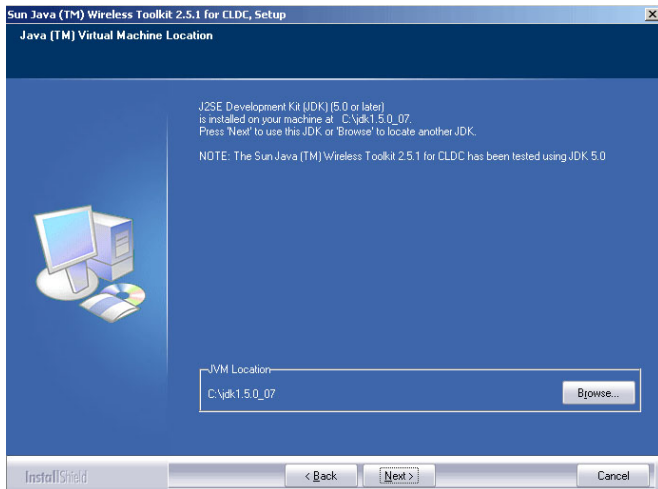
Step 3: Now a "License Agreement" window opens. Just read the agreement and click



Java ME

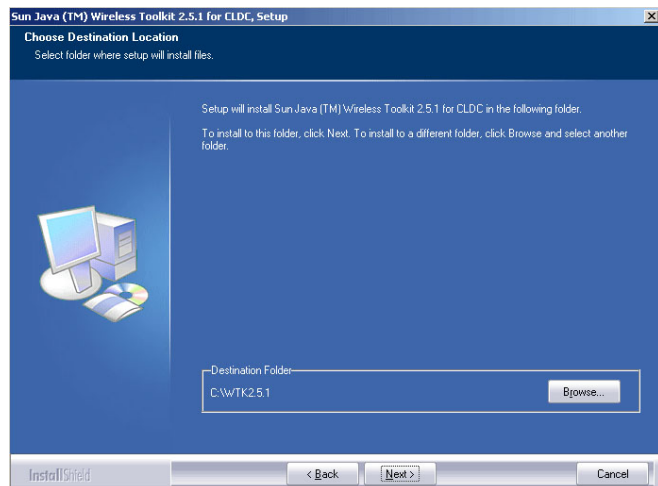
"Accept" button to accept terms and conditions. Then click "Next"

Step 4: Click on "Browse" button and choose your installation directory. Here it is chosen as "C:\jdk1.5.0_07". Now click on "Next" button. Now the installation wizard searches for



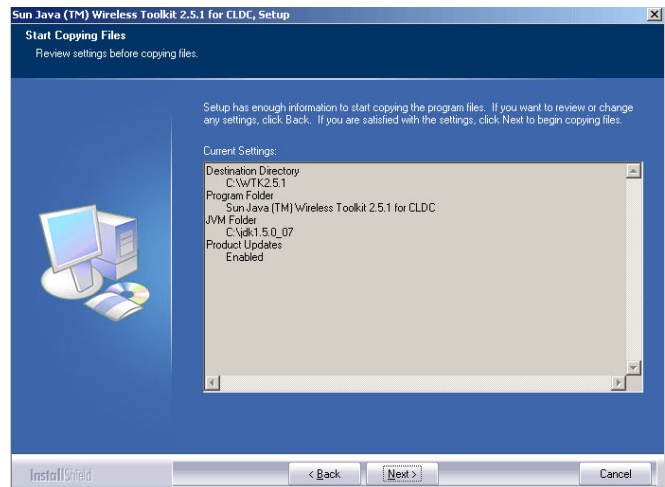
compatible JDK installations at the specified location.

Step 5: Choose the Destination Folder for the

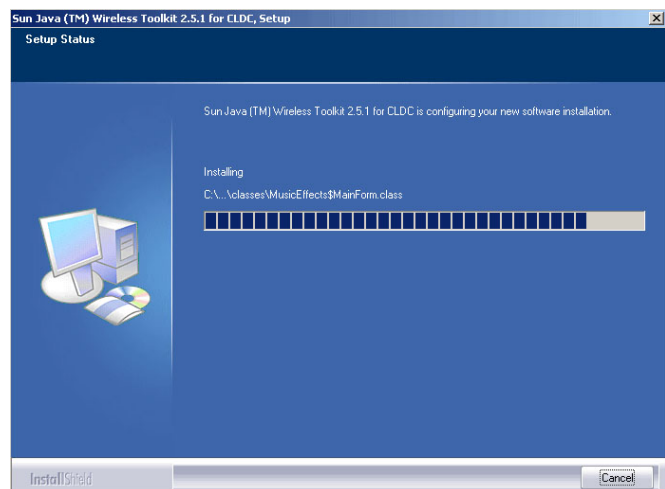


installation of the Sun Java(TM) Wireless Toolkit 2.5.1 for CLDC.

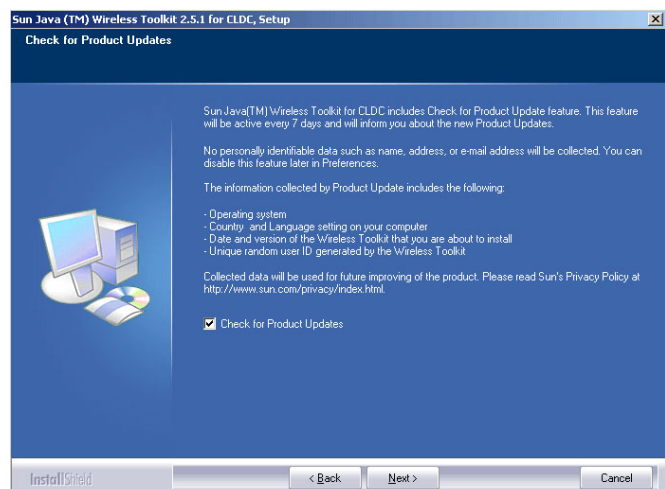
Step 6: Click the "Next" button to choose the installation directory of Runtime Environment. We prefer not to change it. So click "Next"



button. When installation is complete, click "Finish" to exit the installation wizard process.



Step 7: Click "Next" button to start the installation process.



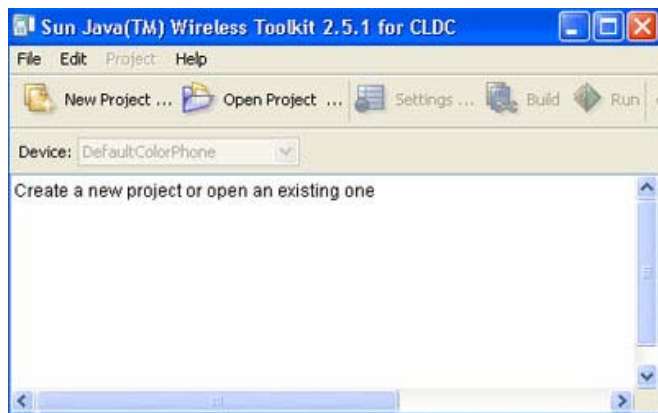
Java ME

Step 8: Click the check button to activate the “Product Updates” features.

Step 9: Ultimately, A window appears indicating the successfully completion of installation of WTK2.5.1. Click “Finish” button to exit the installation process.

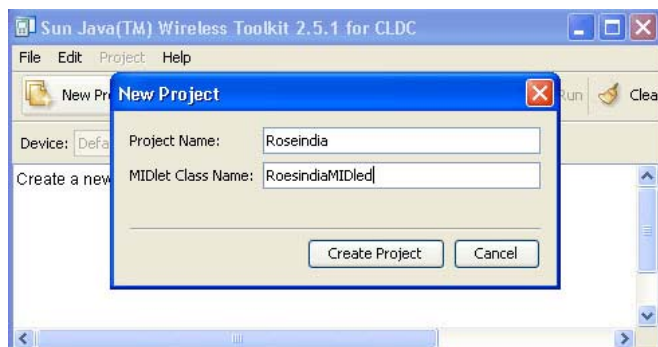
IV. Now we are ready to create an application with Java Platform ME. Lets create a new project with the following steps:

Step 1: Go to Windows start panel and choose



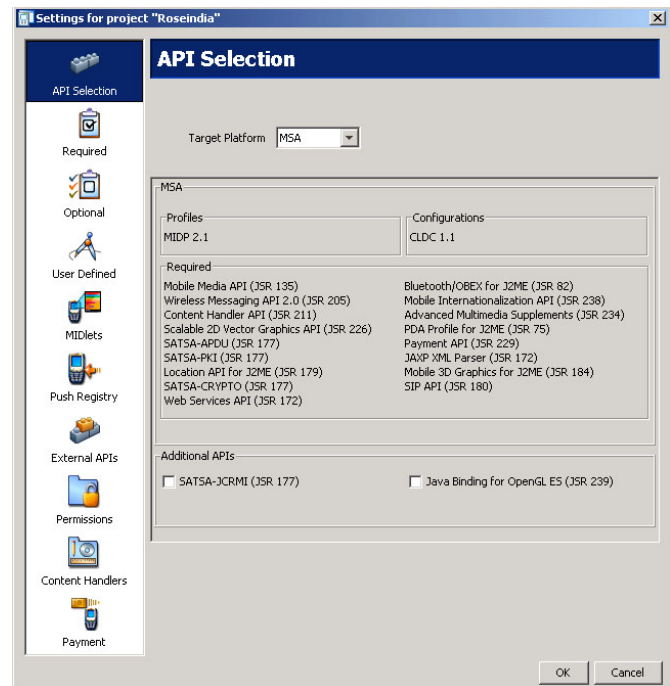
“Wireless Toolkit 2.5.1” as:
Start > Programs > Sun Java Wireless Toolkit 2.5.1 for CLDC > Wireless Toolkit 2.5.1.
The console window appears like this.

Step 2: Now, Click the “New Project” on the

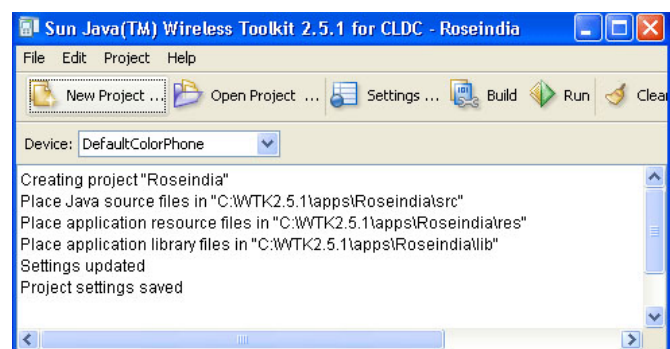


toolkit menu bar, then a new project box opens. Fill the appropriate Project name and MIDlet class Name of your choice. After that, click **Create Project** button.

Step 3: Then a “Settings for project” window appears. For default settings, click OK. It is



better to choose the default settings for a beginner. However you can choose your own settings too, to affect the build environment



for the project.

Step 4: Next appears a window indicating the updated project settings saved in the Console.

Step 5: Now we need to develop a simplest

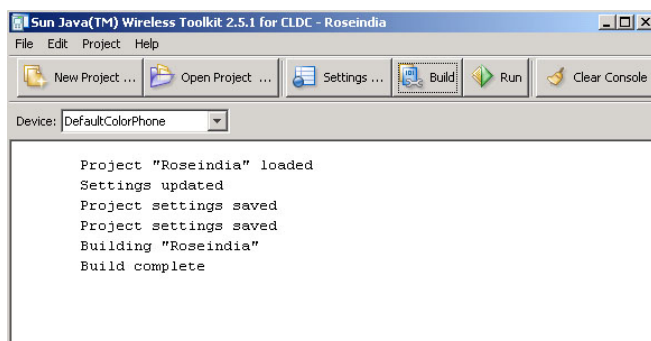
Java ME

program like "RoseindiaMIDlet.java" and save it in the location

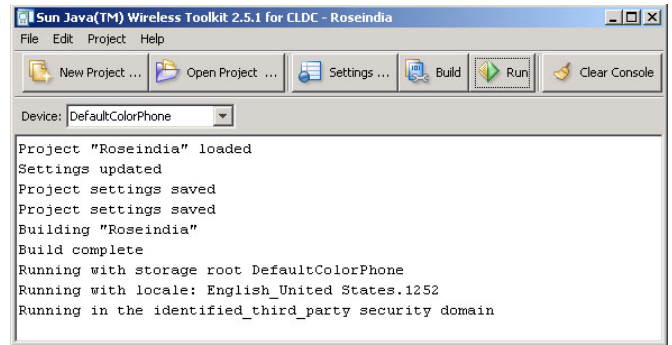
```
"C:\WTK2.5.1\apps\Roseindia\src\".
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;
public class RoseindiaMIDlet extends MIDlet
implements CommandListener {
public void startApp() {
Display display = Display.getDisplay(this);
```

```
Form mainForm = new
Form("RoseindiaMIDlet");
mainForm.append("Welcome to
www.Roseindia.net");
Command exitCommand = new
Command("Exit", Command.EXIT, 0);
mainForm.addCommand(exitCommand);
mainForm.setCommandListener(this);
display.setCurrent(mainForm);
}
public void pauseApp()
{
}
public void destroyApp(boolean
unconditional) {}
public void commandAction(Command c,
Displayable s)
{
if (c.getCommandType() == Command.EXIT)
notifyDestroyed();
}
}
```

Step 6: Next click the "Build" button from the

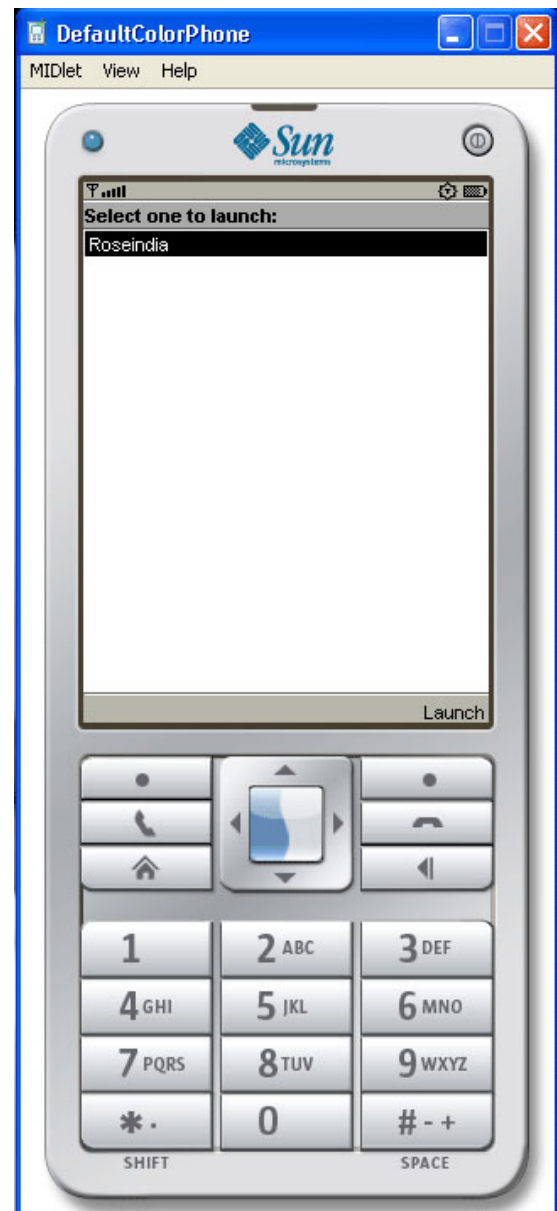


toolkit menu bar. This causes the Sun Java Wireless Toolkit for CLDC to compile and preverify the Java source files. The whole build process is shown below.



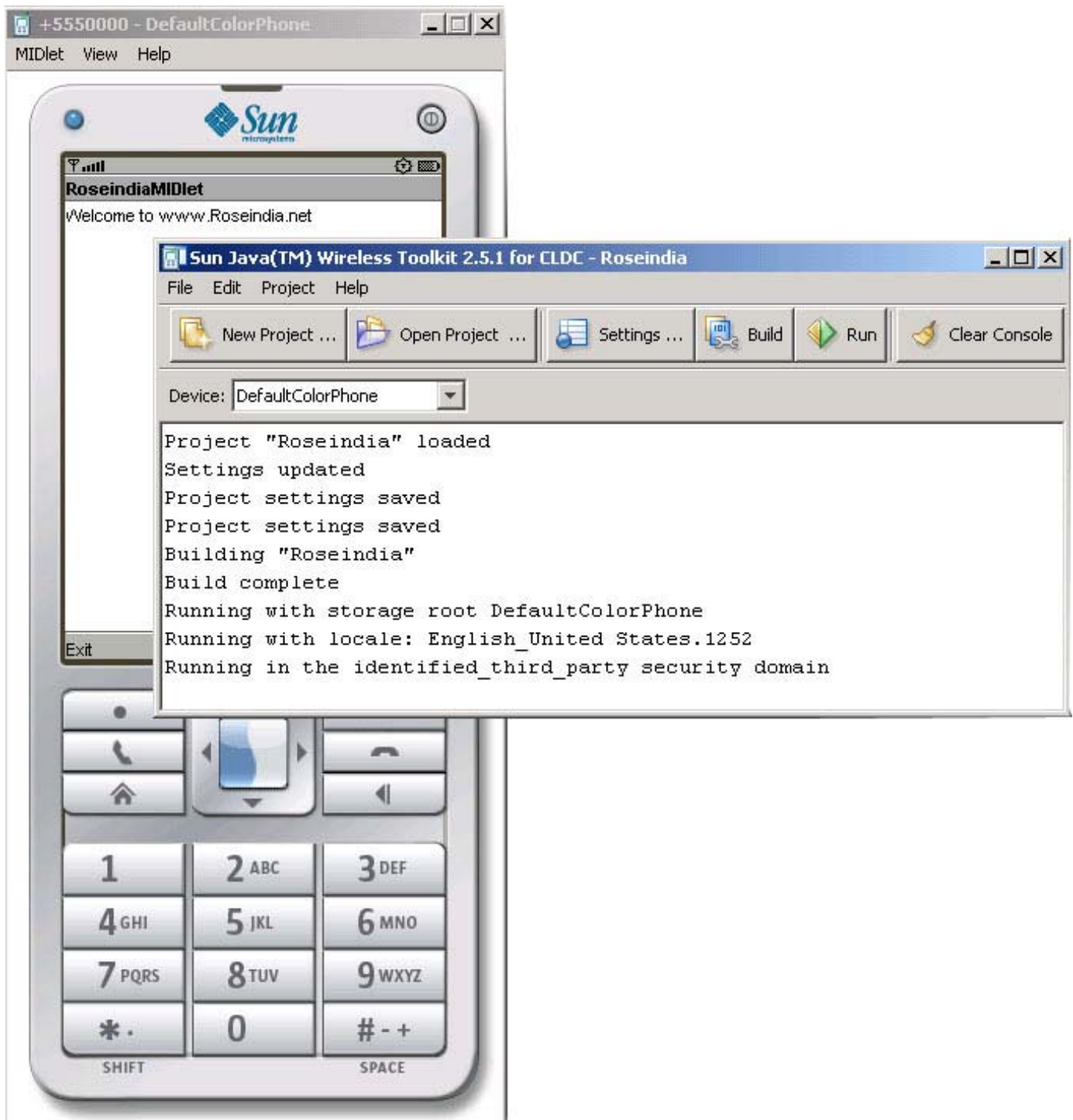
Step 7: Next click the "Run" button from the toolkit menu bar. This executes the compiled Java class files on the emulator.

Step 8: Execution of the compiled Java class



Java ME

files on the emulator gives the following customized output. This output window have



proper event handlers attached with each button.

Step 9: When you press any non-numeric key, an output with a message flashes on the screen.

We will dig the things deeper in the forth coming issue.

Know the Java Champions

Java Champions Program

The Java Champions program was launched in June 2005 at the JavaOne Conference in San Francisco.

Sun recognized that there are leaders in the Java Developer Community. The Java Champions program is an effort to strengthen and encourage this community of leaders. As a part of the Sun Community Champions program, the title is created to recognize the aspirants in the various aspects of the Java technology ecosystem—including authors, trainers, professors, researchers, developers, and JUG leaders.

The Java community itself nominates most candidates, and selects new Java Champions through a peer review process. Out of a pool of nearly 6,000,000 Java developers worldwide, there are more than 85 Java Champions - so it is an elite group.

When excited individuals form into committed groups that develop new and interesting ways to promote the Java Platform, then communities of passionate “Champions” is born. Though it’s possible to compete with individual products however it’s tough to compete against the entire communities.

The Java Champions program is sponsored by Sun Microsystems and is an effort to recognize leaders in the Java Community and invite them to participate in the development of the Java platform in collaboration with Sun engineers and Java Luminaries.”

Java Champions Community aims to build an informal group of Java technology proponents outside of Sun Microsystems to which the Java Development team and engineers could have meaningful discussions with. Champions from within Java Eco-System includes:

- Java Luminaries: senior developers; architects; consultants; conference speakers, etc
- Academics/University Professors
- Authors of Java-related content (online & print)

- Java User Group (JUG) Leaders and the Leaders of online Java portals

Sun wants to engage and recognize the segments within the Java Eco-System through the Java Champions program. Sun wants to give these leaders the chance to provide feedback, ideas, and direction to grow the Java Platform. This interchange may be in the form of technical discussions and/or community-building activities with Sun’s Java Development and Technology Outreach teams. Sun shares a common goal with the Java developer community i.e. the technology adoption of the Java Platform by software developers world-wide.

Five Principles to Become a Java Champion Candidate:

A Java Champions selection committee (*over 50 members*) has been formed to help out the selection of new champions to this online community.

Here are some of the guidelines for selecting the new champions and they are being used by the committee (one or several principles may be applied during the selection process):

Principle 1: Java Champions are Leaders...If a candidate is leading a Java related project, a JUG community, or an online Java portal they will be considered.

Principle 2: - Java Champions are Luminaries in their field.... So the candidate should be a Java engineer or an architect who is relatively a senior and have lots of experiences.

Principle 3: - Java Champions should have Credibility. The candidate can be neutral, for, or against Sun. Also, a Champion can still write anything or publish any material that may be pro, neutral or con towards Sun; unless there is a litigious issue.

Principle 4: - Java Champions are involved with some “really cool” applications of Java Technology or are involved with some sort of humanitarian or educational effort. The application must be openly available to the Java

Know the Java Champions

Community (online or in print) vice a company proprietary or government classified project.

Principle 5: - Java Champions are able to Evangelize or influence other developers through their own professional activities (Example: Consulting, University Professors, Book Authorship, etc.)

Nomination Process: How to be a part of Champions Team *Participation is limited* and only interested java.net members are needed to apply to this java.net Project.

HOW THE JAVA CHAMPIONS ARE NOMINATED BY THE JAVA CHAMPIONS SELECTION COMMITTEE : When started the process (Jun '06), Initially they looked around within the Java Community for leaders in the various parts of the Java Eco-System (like JUG leaders, authors, trainers, professors, researchers, etc.). From there the program grew into a "**community nomination**" process, which is unique, and have a real strength. The community itself nominates most nominees and selects the new Java Champions through a peer review process. The criteria used in the peer review have been listed above as the (5) Principles.

COMMUNITY SIZE? - Currently, there are about 85 Java Champions around the world, with another 20 or so in the selection process queue. Once they get up to 100 or more then they will evaluate the issue like how many more would be appropriate for the program. The idea is always to build a community of Java Champions that reflects the top echelon of contributors to the Java Community.

HOW DO THEY NOMINATE A JAVA CHAMPION? - If someone influential (in their Java Community and/or is an advocate for the Java platform or Java tools) is known to you then please nominate him by sending an email to the **Java Champions selection committee** with the following information:

Note: The candidate's contributions to the Java Community must be available to its members either online or in readily available print media. Proprietary or government classified projects

while innovative are not available to general public. Nomination **MUST** cite specific examples (URLS, book names, presentations, etc) of the

**candidate's direct contributions

** in order for them to be considered by the selection committee.

(NOMINATION FORMAT in email)

Name:

email address:

Category of advocate:

[Java Engineer/Architect,
Author, Professor/Instructor,
Consultant, Trainer,
Other community member]

Reason for nomination

: Please see Java Champion (5) Principles. Also, please include any URL links to the person's online bio (if they have one), blog, published material, JUG group- they belong to, etc. and 1-2 sentences on why this person would be a good addition to our community)

Nominations can be sent via email

: Java Champions "at" sun.com

Sun's goal for the Java Champions project is to build a community of representative Java leaders with whom Sun could have conversations with about the state of the Java Platform. Sun Microsystems sponsors this project through certain administrative functions, but the community is free to elect its own members. Sun's engineers are also participating in dialogues with the Java Champions through the project's private mailing lists.

James Gosling

Let's talk about our #1 Honorary Champion: James Gosling



"James Arthur Gosling" - Father of Java

James A. Gosling, O.C., Ph.D (born May 19, 1955 near Calgary, Alberta, Canada) is a famous software developer, best known as the father of the Java programming language.

Education and career :

In 1977, James Gosling received a B.Sc in Computer Science from the University of Calgary. In 1983, he earned a Ph.D in Computer Science from Carnegie Mellon University, and his doctoral thesis was titled "The Algebraic Manipulation of Constraints". While working towards his doctorate, he wrote a version of emacs (gosmacs), and before joining Sun Microsystems he built a multi-processor version of Unix[1] while at Carnegie Mellon University, as well as several compilers and mail systems.

Since 1984, Gosling has been with Sun Microsystems, and is generally known best as the founder of the Java programming language.

Contributions :

He is generally credited as the inventor of the Java programming language in 1994. He did the original design of Java and implemented its original compiler and virtual machine. For this achievement he was elected to the United States National Academy of Engineering. He has also made major contributions to several other software systems, such as NeWS and Gosling Emacs. He also cowrote the "bundle" program, a utility thoroughly detailed in Brian Kernighan and Rob Pike's book The Unix Programming Environment.

He also built a WYSIWYG text editor, a

constraint based drawing editor and a text editor called 'Emacs' for Unix systems.

Over the years he has built satellite data acquisition systems, a multiprocessor version of Unix, several compilers, mail systems and window managers, as well as text and drawing editors.

At Carnegie-Mellon University in Philadelphia, he did his doctorate where he developed a text editor called "Emacs," which became the most widely used Unix text editor. After completing his doctorate in computer science in 1983, Dr. Gosling worked briefly as a researcher for IBM and then, in September 1984, accepted an invitation to join a small startup company in California - Sun Microsystems. There he pursued his interest in networking techniques and products. In 1990 he became part of a team called the Green project that was developing new networking tools. The rise of the World Wide Web enabled him to conjure up a system where "applets" of applications move through the Internet and provide multimedia capabilities on any computer. Launched in 1995, Java has freed programmers from the confines of proprietary systems. Applications can run on computers across the Internet regardless of the operating system they use.

At Sun his early activity was as lead engineer of the NeWS window system. He did the original design of the Java programming language and implemented its original compiler and virtual machine.

Currently, he is acting as Vice President and Sun Fellow with Sun Microsystems Inc. and actively contributing in sun's new innovations and products. He is still exploring the new java horizons.

Honours :

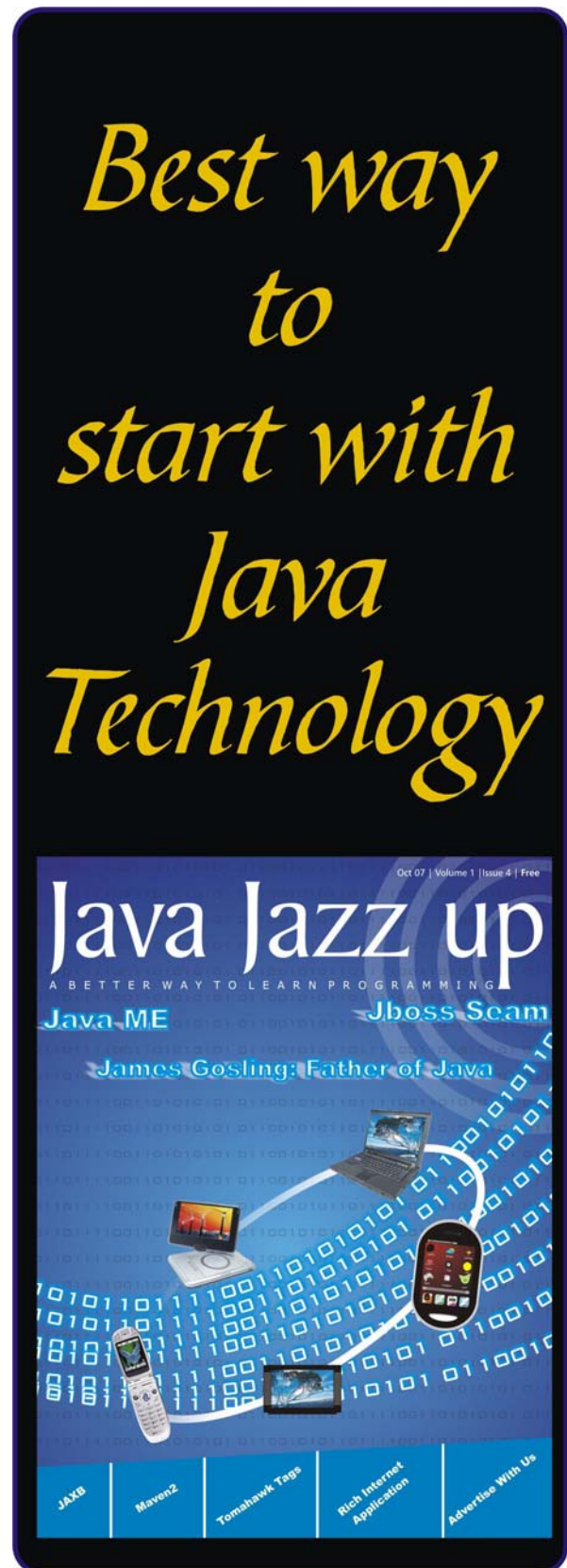
In Feb 2007, he was appointed as an Officer of the Order of Canada. The Order is Canada's highest civilian honour. Officers are the second highest grade.

Personal Corner: He enjoys being an amateur chef and living in Redwood City, halfway between San Francisco and San Jose, with his wife Judy and daughters Kate and Kelsey.

James Gosling

Books: He is also making a great contribution in providing the technical stuff. Here is a listing of his great contributions:

- James Gosling, Bill Joy, Guy L. Steele Jr., Gilad Bracha, *The Java Language Specification, Third Edition*, Addison-Wesley Professional, 2005, ISBN 0-321-24678-0
- Ken Arnold, James Gosling, David Holmes, *The Java Programming Language, Third Edition*, Addison-Wesley Professional, 2000, ISBN 0-201-70433-1
- James Gosling, Bill Joy, Guy L. Steele Jr., Gilad Bracha, *The Java Language Specification, Second Edition*, Addison-Wesley, 2000, ISBN 0-201-31008-2
- Gregory Bollella (Editor), Benjamin Brosgol, James Gosling, Peter Dibble, Steve Furr, David Hardin, Mark Turnbull, *The Real-Time Specification for Java*, Addison Wesley Longman, 2000, ISBN 0-201-70323-8
- Ken Arnold, James Gosling, *The Java programming language Second Edition*, Addison-Wesley, 1997, ISBN 0-201-31006-6
- Ken Arnold, James Gosling, *The Java programming language*, Addison-Wesley, 1996, ISBN 0-201-63455-4
- James Gosling, Bill Joy, Guy L. Steele Jr., *The Java Language Specification*, Addison Wesley Publishing Company, 1996, ISBN 0-201-63451-1
- James Gosling, Frank Yellin, The Java Team, *The Java Application Programming Interface, Volume 2: Window Toolkit and Applets*, Addison-Wesley, 1996, ISBN 0-201-63459-7
- James Gosling, Frank Yellin, The Java Team, *The Java Application Programming Interface, Volume 1: Core Packages*, Addison-Wesley, 1996, ISBN 0-201-63453-8
- James Gosling, Henry McGilton, *The Java language Environment: A white paper*, Sun Microsystems, 1996



JBoss Seam: Web 2.0 Applications



JBoss Seam is a powerful new application framework developed by JBoss, a division of Red Hat for building next generation Web 2.0 applications. It unifies and integrates technologies such as Asynchronous JavaScript and XML (AJAX), Java Server Faces (JSF), Enterprise Java Beans (EJB3), Java Portlets and Business Process Management (BPM) into a single unit i.e. Seam. Seam is designed from the ground, up to eliminate the complexities at the architecture and the API level. It enables developers to assemble complex web applications with simple annotated Plain Old Java Objects (POJOs), componentized UI widgets and very little XML. The simplicity of Seam 1.0 will enable easy integration with the JBoss Enterprise Service Bus (ESB) and Java Business Integration (JBI) in the future.

Previously, creating J2EE 1.4 applications, especially those involving web-based user interface and back-end EJBs, required a lot of tedious coding. Now JEE 5 and EJB 3, with their focus on lightweight Java support, have planned to reduce the code bloat for the developers. However the specifications for JEE 5 along with EJB3 are still not yet finalized, there is already a framework available i.e. JBoss Seam, this framework is built on top of JEE 5 and EJB3. It aims to further reduce the code required for building a functional application. Seam being, a component framework promises to deliver full-featured JEE 5 applications in a lightweight code base i.e. requiring only a fraction of the code of regular JEE applications.

JBoss Seam is a flexible web application framework introduced for Java EE 5. It eliminates the traditional tedious coding using clever architecture. Seam provides a consistent

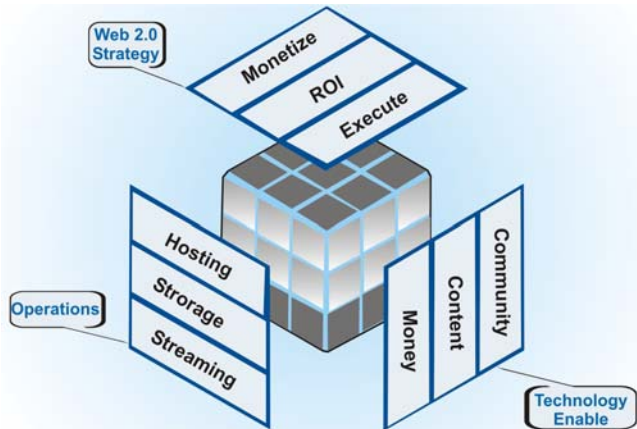
and easy to understand streamlined programming model for developing web based applications. Seam enables to access any back-end EJB 3.0 component from the front-end simply by addressing it with the Seam component name.

JBoss Seam is compatible with most of the available application servers which support EJB 3.0 public draft and JSF 1.1 implementation. Seam is also usable with JBoss Micro- container and Hibernate due to its unique approach to maintain state management while not plugging into Java EE 5 and EJB 3.

Seam framework provides a command line tool seam-gen that can automatically generate a CRUD (create-read-update-delete) web application from the existing database. Seam provides the concept of declarative application state management for POJO component that means Seam extends annotations defined in EJB 3.0 having a new set of annotations for declarative state management, declarative context demarcation, state validation and eliminating XML required by plain JSF.

Every Seam component exists within a context as seam components are stateful and contextual having a well defined container-managed lifecycle. This approach helps to fix the bugs and performance problems results in plague web applications having non-linear or multi-window navigation. Seam integrates with the JBoss jBPM deeply to make the business process management, a first class construct. Seam also simplifies testing Java EE 5 applications in unit test framework by enforcing the JBoss Embedded EJB3 container. In other words, we can say that Seam is completely for enhancing developer's productivity and application scalability.

JBoss Seam



Exploring the Strengths of Seam

1. The first application framework for EJB 3.0

EJB 3.0 has totally changed the notion of EJB components as coarse-grained, heavy-weight objects to EJBs as lightweight POJOs with fine-grained annotations. Seam eliminates the distinction between the presentation tier components and the business logic components and brings a uniform component model to the EE platform where any class may be an EJB.

Seam is providing a backward compatibility with JEE i.e. it is not limited to Java EE 5.0 servers that support EJB 3.0. Seam may be used in any JEE environment, or even in plain Tomcat.

2. Integrate and Enhance Java EE Frameworks

The core frameworks included in Java EE 5.0 are JSF 1.2 and EJB3.0. EJB 3.0 is nothing but the light weight middle tier framework based on POJO (Plain Old Java Objects). It is used for data persistence and business services. JSF is the component framework based on MVC architecture meant for web applications development. Most of the web applications based on Java EE 5.0 do not have both EJB3 and JSF modules for business logic and front end respectively. EJB3 configures services by using annotations while JSF uses XML files.

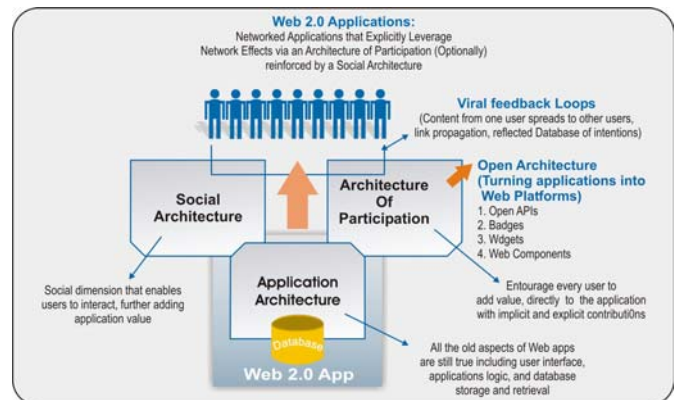
Furthermore at the framework level EJB 3.0 and JSF components are not aware of each other. Artificial facade objects enable the EJB 3 and JSF components to work together by tying

up the business components to web pages and boilerplate code so that the method calls can be made across the framework boundaries. Seam is responsible for gluing these technologies together.

Seam breaks the artificial layer between EJB3 and JSF and also integrates EJB3 and JSF by using consistent and annotation-based approach. Seam lets the developer to use the "same kind of stuff", annotated POJOs for all the components of any application. Seam applications are quite simple and includes less significant code (for both Java as well as XML code) while having the same functionality.

3. Web 2.0 Ready

Seam is mainly designed for the web applications of the Web 2.0 style. Seam supports AJAX (Asynchronous JavaScript and XML) in various ways such that the Seam components can directly access the custom JavaScript library from the browser like a JavaScript object. Seam supports an advanced concurrency model that internally manages multiple AJAX requests of a single user.

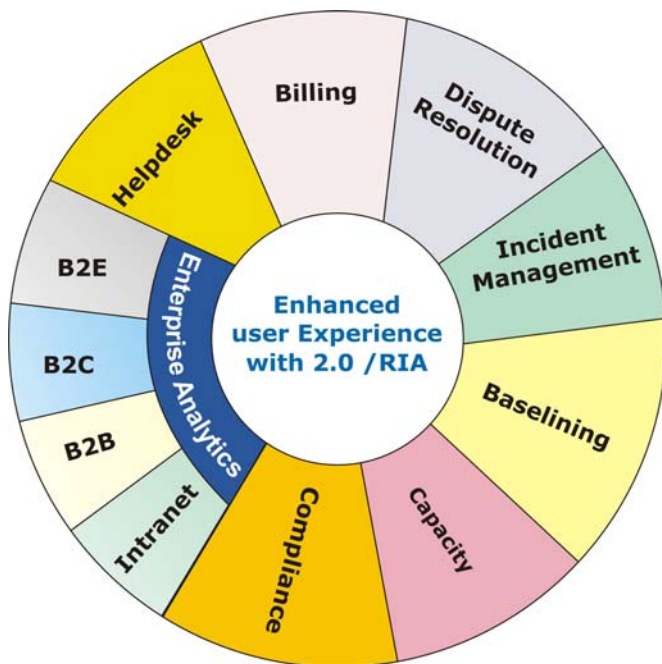


AJAX applications simplifies the frequent requests to the server as compared to non-AJAX applications. However the frequent requests make it challenging to handle the increase in database load. The database can not handle the load if the database serves all the request. To combat this challenging job Seam provides an in-memory cache, which provides a stateful persistence context. This in-memory cache is capable to hold the information throughout the session and hence,

JBoss Seam

leads to the reduction of the database round trips.

Web 2.0 applications also manages complex relational models for its data. Social networking sites mainly manages and represents the relationships among the users. Today, Seam is the sole technology that supports lazy loading for web application in the right direction.



4. A Web Framework that Understands ORM

Today's most of the applications use the Object Relational Mapping (ORM) solutions. However most of these applications use those business and web frameworks which are not designed for ORM. Such framework does not even handle the persistence context throughout the interaction lifecycle starting with a request to render a response.

Initially Seam was designed to promote as a ORM best practices. In case of Seam, no more DAOs are required to write; only lazy loading can work. Since the extended persistence context behaves like a natural cache and reduces database round trips therefore the ORM performance greatly improves. Since Seam

integrates the ORM layer with the business and presentation layer, therefore ORM objects can be observed working directly.

5. POJO Services via Dependency Bijection

Seam is a "lightweight framework" as it uses POJO (Plain Old Java Objects) as its services components. No framework dependent interfaces or abstract classes exist in Seam which does not allow the components to get hooked into the applications. This generates a curiosity to know the things like how do the POJOs interact with each other to develop an application and How do they interact with the services of the container.

Seam uses the most commonly used designed pattern known as "dependency injection" (DI) to connect to the POJO components with each other. Seam maintains the life cycle of all of its component with the implementation of the DI design pattern. Whenever a component uses another component then it shows this dependency to Seam with the help of annotations. Seam decides the injection place of this component according to the application's current state.

Let's take a seam component say P which needs to create another component say Q. It does that simply by expanding the dependency and "outjects" the component Q back to Seam for another components say R to later use it. This type of bi-directional dependency management is known as dependency bijection.

6. Avoid XML Abuse

Java annotations play an important role to express and manage the Seam configuration metadata. In the beginning of J2EE, XML was supposed as the better option for configuration management. Framework designers put all the configuration related information (such as Java classes and method names) into the XML files. However, it was not an optimised solution because XML configuration files are repetitive in nature i.e. already existing information in the code is repeated just to connect the configuration with the code and this makes an

application prone to errors. JEE developers refer this problem as the “XML hell”. This way of using the XML files is popular among the whole Java community as “the XML abuse”. Java community made the successful attempts to replace these XML files with the introduction of the annotations to the Java source code.

To promote the use of annotations at enterprise level officials of the Java standardization body introduced EJB3. EJB3 provides the options to use the annotations in place of the XML file and expands the sphere of annotation-based programming model for the web application. It doesn't mean that XML is fully useless, XML best suits for specification of web application page flows and can also define the work flow of the business process. XML files enable a user to centrally manage the work flow of the entire web application rather than to scatter the information around the Java source file.

7. Designed for Easy integration Testing

Seam is mainly designed to simplify the testing process. Unit testing is easier with seam as all the Seam components are nothing but the annotated POJO. Simply one need to create the POJO instances which can be tested with different testing frameworks such as TestNG and JUnit. If there requires any interaction between the Seam components then just create the individual instances of those components and manually establish the relationship between them.

Testing of the entire Seam application is a little bit complex because it needs entire application to run inside the Seam container. Seam comes along with an embedded lightweight container to handle the integrated testing. One can also programmatically load the Seam container to run the test into the test framework.

8. Designed for Stateful Web Applications

Seam also designs stateful web applications which are inherently multi-user. Today, most of the business applications are stateful and transactional. In case of Seam, all the

components of an application are stateful therefore it is quite easy to use seam for managing states as compared to HTTP session. Seam applications does not require a developer to write the state management code, but simply it, itself annotates the scope of the components, lifecycle methods and other stateful properties required. Stateful components of Seam also manages the user states in a better way in comparison to conventional HTTP sessions.

Seam automatically ties up the transactions and the database caches with the application state. It temporarily holds database updates in memory and commits updates to the database while ending the conversation. For complex stateful applications, in-memory cache reduces the database load of the application.

Seam takes state management in web applications a big step further by supporting integration with the Open Source JBoss jBPM business process engine. One can now specify the work flows of different people in the organization (i.e., customers, managers, technical support etc.) and use the work flow to drive the application, instead of relying on the UI event handlers and databases.

9. Great Tools Support

Seam simplifies to accomplish difficult tasks with JSF. Generally, to bookmark a JSF web page and get it via HTTP GET is hard. Generating a bookmarkable resultful web page with Seam is quite simple. Seam makes the JSF applications efficient by providing a number of JSF tags and use of annotations increases the “web friendliness”.

Simultaneously, Seam expands the web tier to the business components and also expands the EJB3 component model to POJOs. Even though Seam also integrates a number of commonly used other open source framework like JBoss Portal, jBPM, JBossMicrocontainer, JBoss Rules etc.

Although Seam is integrated with Java EE 5.0, it doesn't mean that it is limited to Java EE 5.0 servers. Seam applications can also be deployed

JBoss Seam

in the J2EE 1.4 application servers as well as in plain Tomcat servers.

Seam not only integrates various frameworks into the one framework but also provides its own managed stateful context allowing the frameworks to deeply integrate with others through annotations, EL (Expression Language) expressions etc.

Seam provides a great tools support intended to enhance the developers productivity. Seam also comes with the command line application generator which is known as the Seam Gen. It generates complete CRUD applications from the database. Additional support for tools with seam increases the developers turn around time by providing a better testing support and supports for features like edit / save /reload browser actions etc.


But more importantly, Seam Gen generated projects work out-of-the-box with leading Java IDEs such as Eclipse and NetBeans. With Seam Gen, one can get started with Seam in no time!



*We are the best in
Shaping the
Java Technology Stuff*




<http://www.roseindia.net>



$8 \times 3 = 24\text{hours}$

It's always for **24h**



NEWSTRACK india

Maven 2 Eclipse Plug-in

Plugins are great in simplifying the life of programmers; it actually reduces the repetitive tasks involved in the programming. In this article our experts will show you the steps required to download and install the Maven Plugin with your eclipse IDE.

Why Maven with Eclipse

Eclipse is an industry leader in IDE market, it is used very extensively in developing projects all around the world. Similarly, 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-Eclipse Integration makes the development, testing, packaging and deployment process easy and fast. Maven Integration for Eclipse provides a tight integration for Maven into the IDE and avails the following features:

- It helps to launch Maven builds from within Eclipse
- It avails the dependency management for Eclipse build path based on Maven's pom.xml
- It resolves Maven dependencies from the Eclipse workspace without installing to local Maven repository
- It avails an automatic downloading of the required dependencies from the remote Maven repositories
- It provides wizards for creating new Maven projects, pom.xml or to enable Maven support on plain Java project
- It helps to search quickly for dependencies in Maven remote repositories
- It quickly fixes in the Java editor for looking up required dependencies/jars by the class or package name.

What do you need?

1. Get the Eclipse Development

Environment

In this tutorial we are using the eclipse-SDK-3.3-win32, which can be downloaded from <http://www.eclipse.org/downloads/>

2. Get Maven-eclipse-plugin-plugin

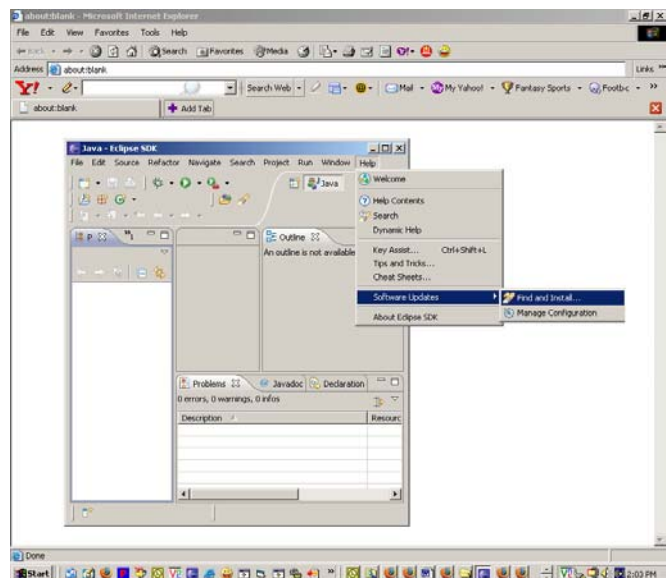
It is available at <http://mevenide.codehaus.org/maven-eclipse-plugin-plugin/>

Download and Install Eclipse

First download and install the eclipse plugin on your development machine then proceed with the installation process of the eclipse-maven plugin.

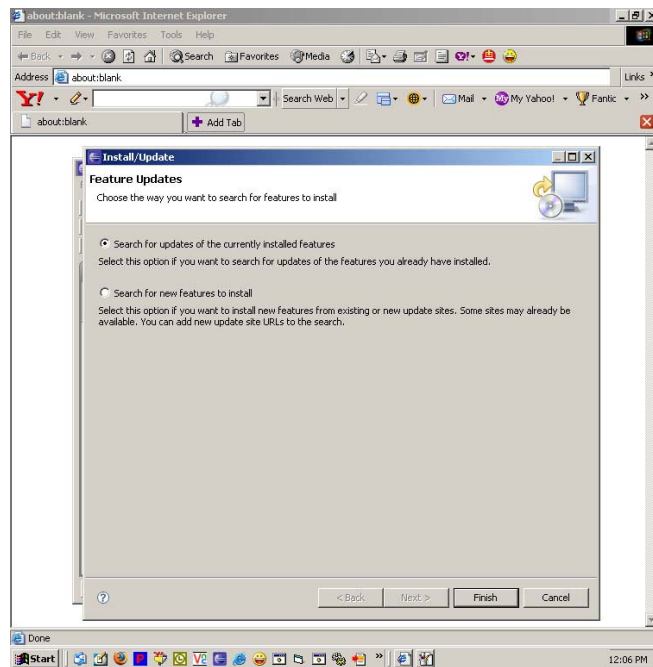
Steps to Install the eclipse-maven plugin

1. Open eclipse IDE and go to **Help->Software Updates-> Find and Install** as shown in Figure 1 below:



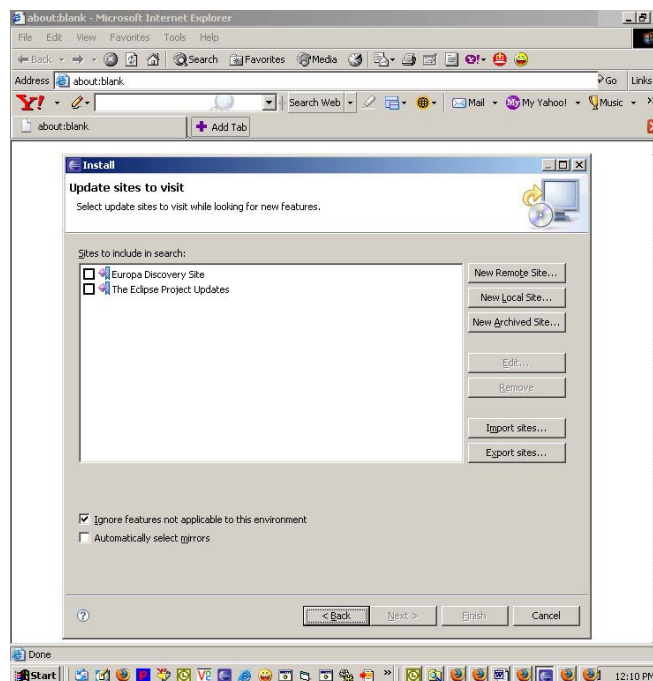
Maven 2

2. Choose the way you want to search for the features to install in the **Install/update window**. It is show in the figure 2:



Now, Select the option **“Search for new features to install”** and click on the **“Next”** button.

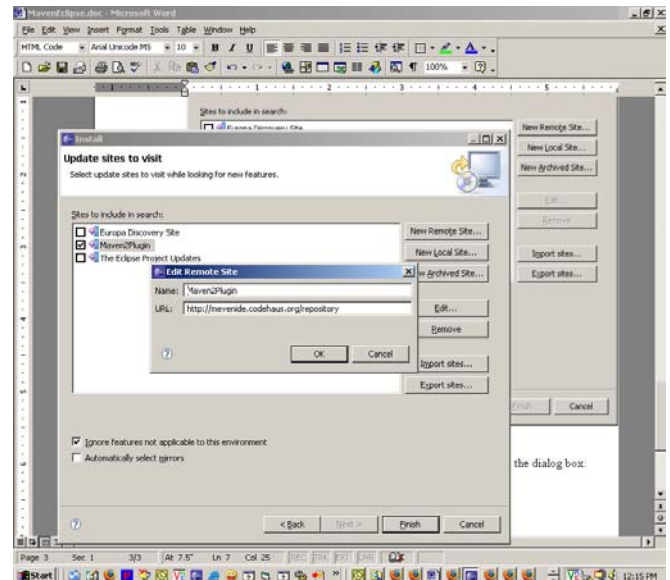
3. An Install wizard appears as shown in Figure3



Now click on **“New Remote Site...”** and enter the following details in the dialog box:

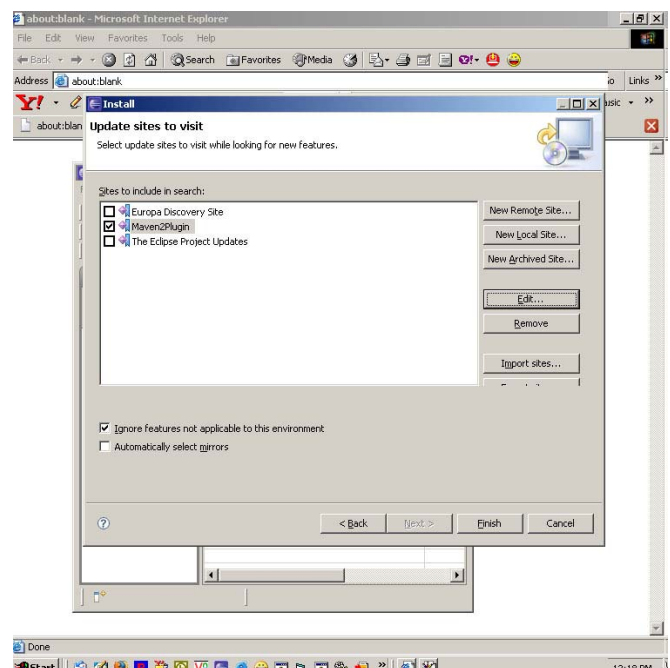
Name: **Maven2Plugin**

URL: **http://mevenide.codehaus.org/repository** As shown in figure 4



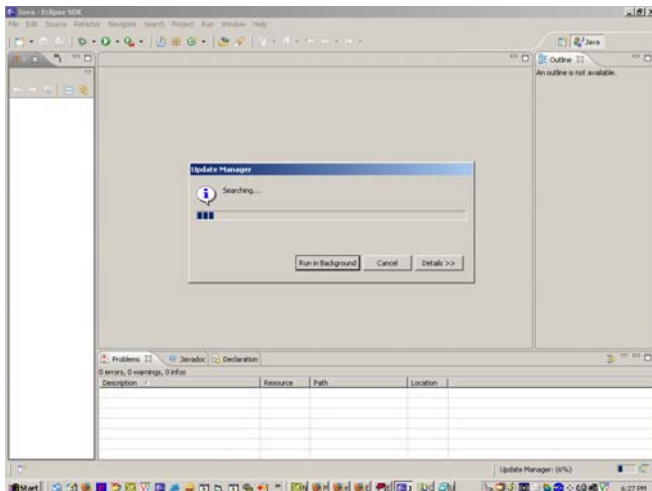
Now click on **“OK”** button.

4. Then update the information of the sites to visit for instance we have selected the **Maven2plugin** site in the Eclipse update list as shown in following figure (Figure 5)

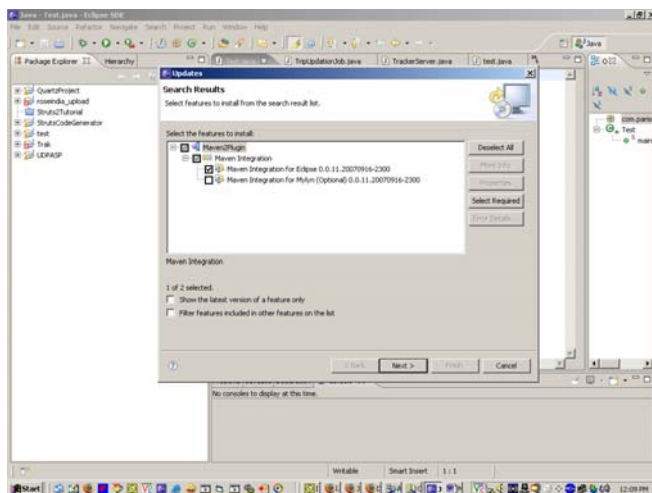


Maven 2

5. Click on the **"Finish"** button. Update manager starts searching for the updates as shown in Figure 6

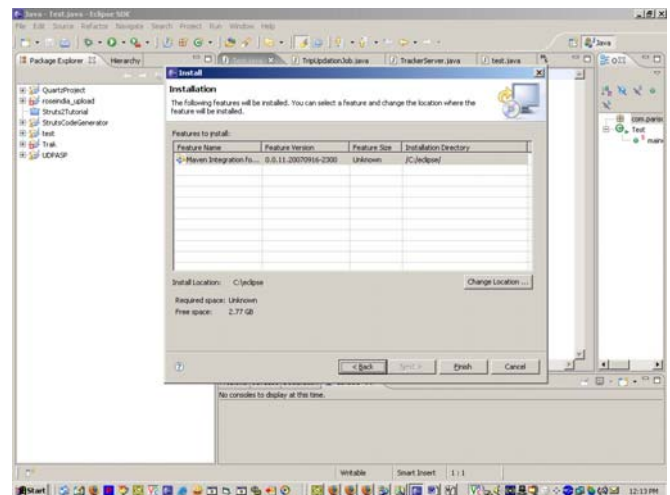


6. Then an Eclipse update window appears, select **"Maven2Plugin"** check box as show below:



Click on the **"Next"** button and then Accept the terms and conditions in the next window and click the next button.

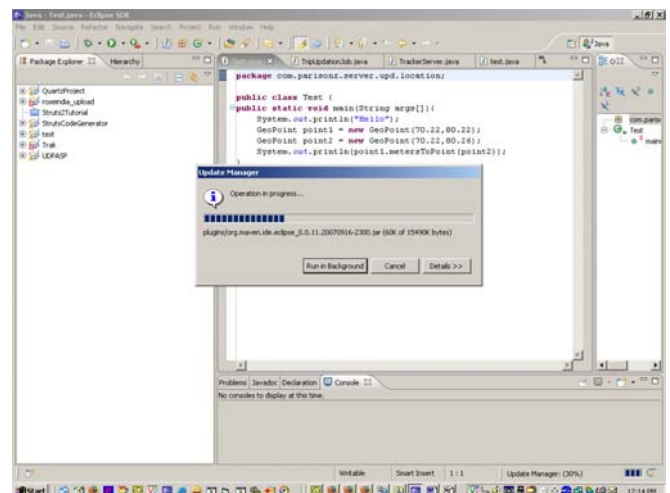
7. Then, An Installation confirmation window



appears as shown below:

Click on **"Finish"** button to complete the installation process.

8. **Update manager** will download the files and

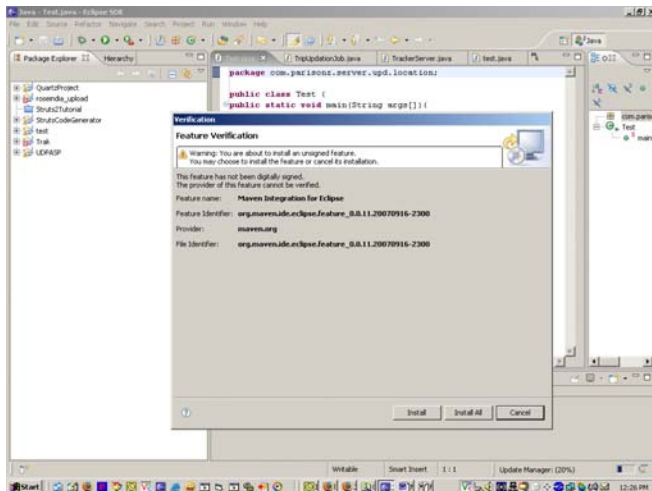


install the Maven eclipse plugin for you.

9. Then the installer displays the **Feature Verification** window as shown below:

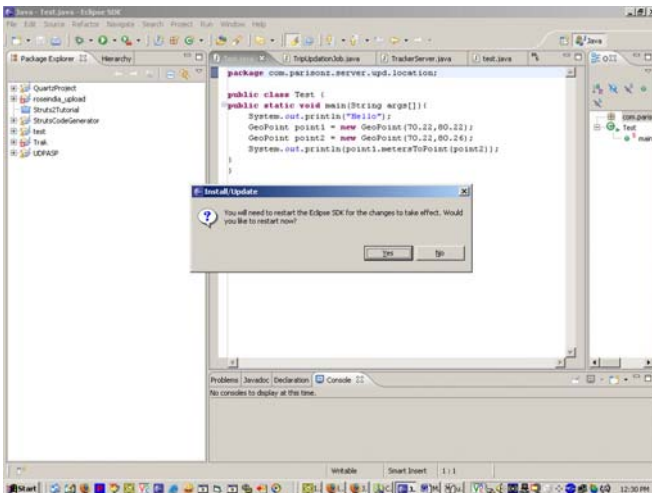
Maven 2

Click on **"Install All"** button.



10. Finally the installer will display the following message:

Now the Eclipse-Maven plugin is ready for use.



Change Your Life

AllcoolJobs
Just log on

<http://www.allcooljobs.com>



A BETTER WAY TO LEARN

JSP | EJB | JDBC | Java Servlets | WAP | Free JSP Hosting | Spring Framework | Web Services | BioInformatics |
Java Server Faces | Jboss 3.0 tutorial | Hibernate | XML

www.roseindia.net

JSF Tags: Tomahawk Tags

Tomahawk tags are collection of standard components with extended functionality and many more extra set of components with rich set of functionality.

1-Tomahawk `inputTextHelp` tag

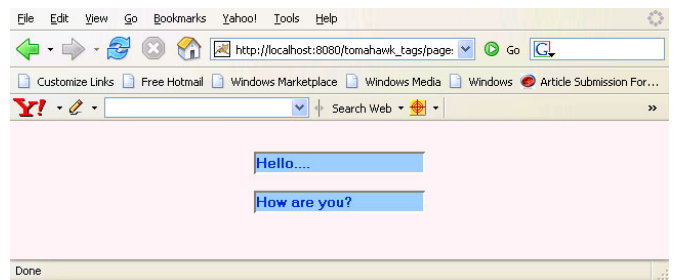
This tag creates an input text box but it has extra ability to select or removing the text displayed. This text is used to help the user and is set by "helpText" attribute. The capability of disappearing or selecting the text is provided by "selectText" attribute of the tag. This tag has additional feature of displaying value only, not the widget of the box. This component also has ability to be visible or not visible according to the role of the user. In the same way, it also has the ability to be enabled or disabled according to the user role. Normally the naming system of JSF renders the id for the component with some additional text, typically with id of the form, as prefix. But this component has an attribute `forceId`, which forces the component to render the same id mentioned in the id attribute.

Code Description :

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:inputTextHelp example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
.inputstyle{
background-color: #99CCFF;
}
-->
</style>
</head>
```

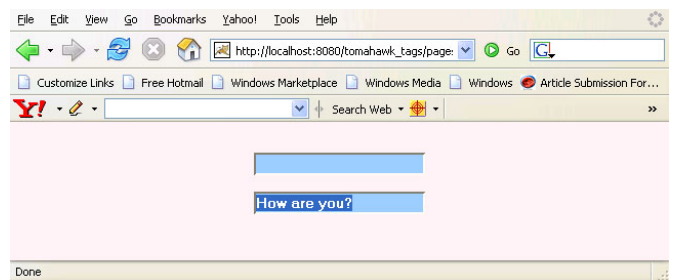
```
<body ><center>
```

```
<t:inputTextHelp id="ith1" value=""
helpText="Hello...."
style="color: #0033CC; font-weight:bold"
styleClass="inputstyle" title="inputTextHelp
example 1" /><p>
<t:inputTextHelp id="ith2" value=""
helpText="How are you?" selectText="true"
style="color: #0033CC; font-weight:bold"
styleClass="inputstyle" title="inputTextHelp
example 2" />
</center></body>
</html>
</f:view>
```



Rendered Output:

This is the output of the page. In this Hello.... and How are you? texts are the texts specified in the helpText attribute of the tag.



If we take focus on the first component then text in the component disappears while in the other whole text is selected because of setting the selectText attribute to true in the second component.

Tomahawk Tags

2-Tomahawk selectOneCountry tag

This tag is used to create the component, which displays the list of countries in selection box according to the locale.

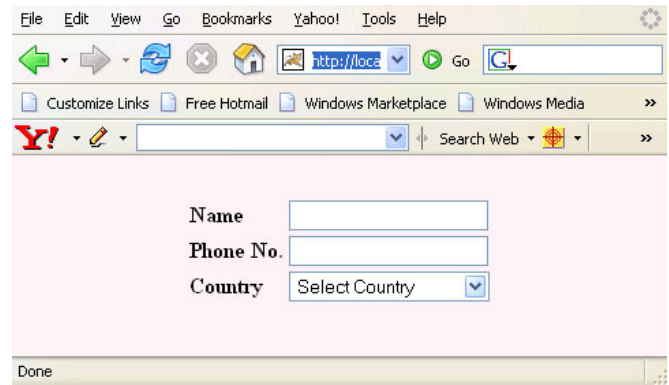
Code Description:

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

<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:selectOneCountry example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form><center>
<t:panelGrid columns="2" style="font-weight:bold;">
<t:outputLabel for="name" value="Name" />
<t:inputText id="name"/>
<t:outputLabel for="phone" value="Phone No." />
<t:inputText id="phone"/>
<t:outputLabel for="country" value="Country" />
<t:selectOneCountry id="country"
maxLength="16"
emptySelection="Select Country"/>
</t:panelGrid>

</center>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output:



3-Tomahawk selectOneLanguage tag

This tag is used to create the component, which displays the list of languages in selection box according to the locale.

Code Description :

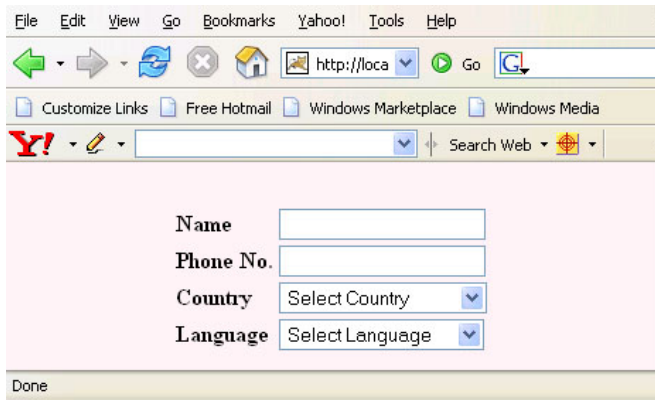
```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>

<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:selectOneLanguage example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form><center>
<t:panelGrid columns="2" style="font-weight:bold;">
<t:outputLabel for="name" value="Name" />
<t:inputText id="name"/>
<t:outputLabel for="phone" value="Phone
```


Tomahawk Tags

```
No." />
<t:inputText id="phone"/>
<t:outputLabel for="country"
value="Country" />
<t:selectOneCountry id="country"
maxLength="16"
emptySelection="Select Country"/>
<t:outputLabel for="language"
value="Language" />
<t:selectOneLanguage id="language"
maxLength="16"
emptySelection="Select Language"/>
</t:panelGrid>
</center>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output:



4-Tomahawk validateEmail tag

This tag validates the email address entered in the field. We can render the validation message using message, detailMessage and summaryMessage attributes.

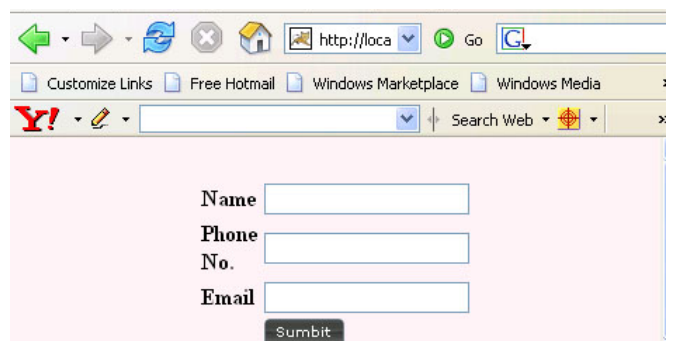
Code Description:

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

<f:view>
```

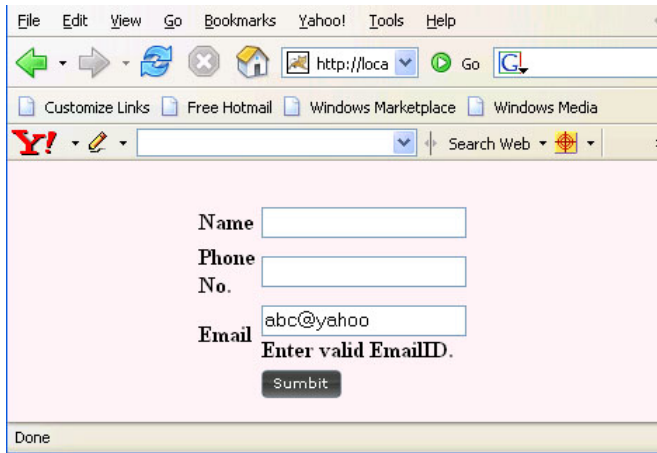
```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:validateEmail example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form><center>
<t:panelGrid columns="2" style="font-weight:bold;" width="40%">
<t:outputText value="Name" />
<t:inputText id="name"/>
<t:outputText value="Phone No." />
<t:inputText id="phone"/>
<t:outputText value="Email" />
<t:panelGroup>
<t:inputText id="email" required="true">
<t:validateEmail message="Enter valid EmailID."/>
</t:inputText>
<f:verbatim></br></f:verbatim>
<t:message for="email"/>
</t:panelGroup>
<t:outputText value=" " />
<t:commandButton id="cb" image="images/submit_button.gif" action="welcome"/>
</t:panelGrid>
</center>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output:



Tomahawk Tags

If we enter wrong email id then the message written is message tag is displayed like is the figure below:



5-Tomahawk validateRegExpr tag

This tag is used to validate a string entered by the user. If we want the user to enter a specific pattern of string then we can set the pattern for that component. For example, we have an input field and we want the user to enter a number that consists of any number with "1" in the beginning but only one "2" at the last. So for this, Tomahawk provides a tag `validateRegExpr`, which has "pattern" attribute that is used to specify the pattern to be followed by the user while inputting the string in the box. If the entered string is not following the pattern then a message can be displayed using "message" attribute.

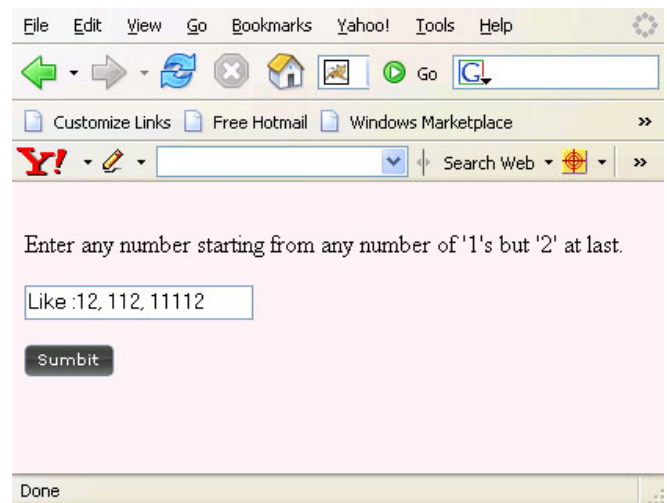
Code Description:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t"%>
<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t.validateRegExpr example</title>
<style type="text/css">
```

```
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form>
<t:outputText value="Enter any number
starting from any number of '1's but '2' at
last."/></p>
<t:inputTextHelp id="regExprValue"
helpText="Like :12, 112, 11112"
required="true">
<t:validateRegExpr pattern="1*2"
message="Type correct Number."/>
</t:inputTextHelp>
<t:message for="regExprValue"/></p>
<t:commandButton id="cb" image="images/
submit_button.gif" action="welcome"/>
</h:form>
</body>
</html>
</f:view>
```

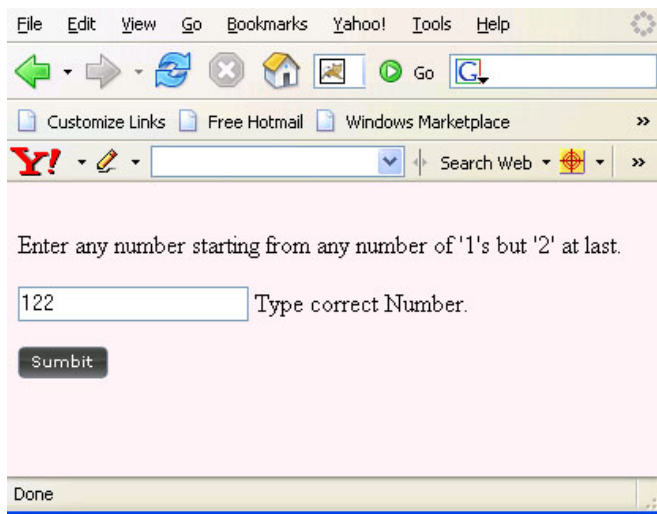
Rendered Output:

This is the output of the above code:



Tomahawk Tags

If the user enters a wrong input then the message is displayed like below:



6-Tomahawk validateEqual tag

This tag is used to validate the value against the other component. In the for attribute we specify the id of the other component whose value is compared to the value of the component for which the validation is performed. If both are same then no error otherwise it displays the validation message. We can render the validation message by the use of message, detailMessage and summaryMessage attributes.

Code Description:

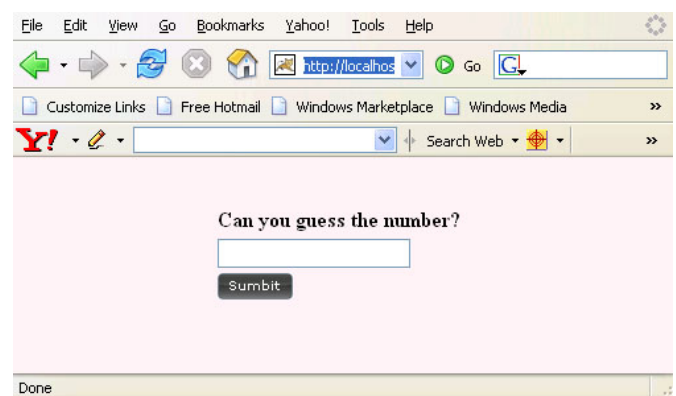
In this example, we have taken a hidden field with value "999" and one more input component where the user is asked to guess the number. If both matches then next page are rendered otherwise validation message "Try again..." is displayed in the same page.

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

```
<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
```

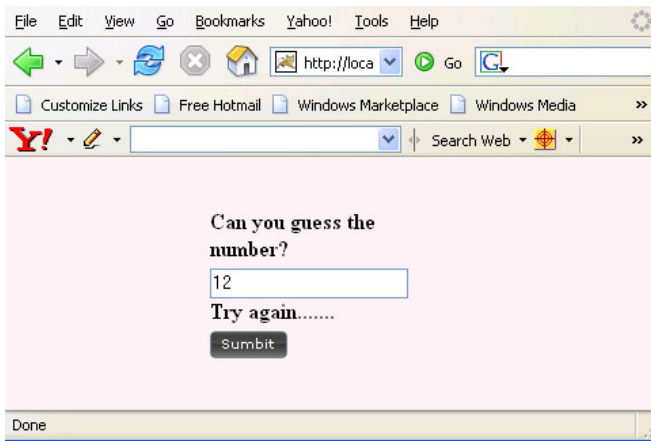
```
content="text/html;
harset=iso-8859-1">
<title>t:validateEqual example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form>
<center>
<t:panelGrid columns="1" style="font-weight:bold;" width="40%">
<t:inputHidden id="ih" value="999" />
<t:outputText value="Can you guess the number?" />
<t:panelGroup>
<t:inputText id="number" required="true">
<t:validateEqual for="ih"
message="Try again....."/>
</t:inputText>
<f:verbatim></br></f:verbatim>
<t:message for="number"/>
</t:panelGroup>
<t:commandButton id="cb"
image="images/submit_button.gif"
action="welcome"/>
</t:panelGrid>
</center>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output:



Tomahawk Tags

The output below is the result of bad guessing of the number by the user.



7-Tomahawk validateCreditCard tag

This tag is used to validate a Credit Card Number entered by the user. If we want the user to enter a valid Credit Card Number then validateCreditCard tag can do this validation. If we don't want to allow any of the "American Express","Visa","mastercard" or "Discover" then we can set value false for the attributes "ameex","visa","mastercard" and "discover" respectively. If we don't want to allow anyone then we can set value true for the attribute "none". If the entered value is not a valid then we can display the message of our interest by using "message" attribute.

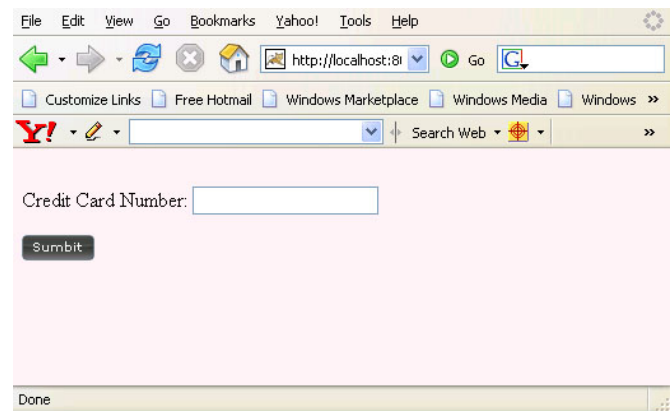
Code Description:

```
<%@ taglib uri="http://java.sun.com/jsp/
html" prefix="h"%>
<%@ taglib uri="http://java.sun.com/jsp/
core" prefix="f"%>
<%@ taglib uri="http://myfaces.apache.org/
tomahawk" prefix="t"%>
<f:view>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>t:validateCreditCard example</title>
<style type="text/css">
<!--
body{
background-color: #fff2f2;
```

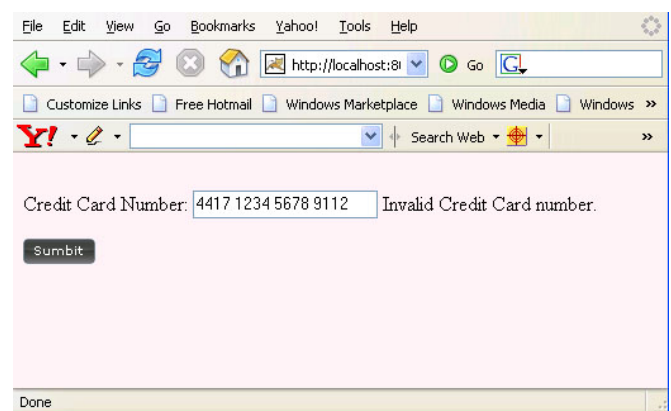
```
margin-top:30;
}
-->
</style>
</head>
<body >
<h:form>
<t:outputText value="Credit Card Number:"/>
<h:inputText id="creditCardNumber"
required="true">
<t:validateCreditCard message="Invalid Credit
Card number."/>
</h:inputText>
<t:message for="creditCardNumber"/> </p>
<t:commandButton id="cb" image="images/
submit_button.gif" action="welcome"/>
</h:form>
</body>
</html>
</f:view>
```

Rendered Output:

This is the output of the above code:



If the number entered by the user is not valid then error message is displayed like below:



Remoting with Spring

Spring features remoting support using various technologies. Remoting support eases the development of remote-enabled services, implemented with usual (Spring) POJOs. Currently, Spring supports different remoting technologies few of them are:

- **Remote Method Invocation (RMI):** Through the use of the `RmiProxyFactoryBean` and the `RmiServiceExporter`, Spring supports both traditional RMI (with `java.rmi.Remote` interfaces and `java.rmi.RemoteException`) and transparent remoting via RMI invokers (with any Java interface).
- **Spring's HTTP invoker:** Spring provides a special remoting strategy which allows Java serialization via HTTP, supporting any Java interface (just like the RMI invoker). The corresponding support classes are `HttpInvokerProxyFactoryBean` and `HttpInvokerServiceExporter`.
- **Hessian:** By using the `HessianProxyFactoryBean` and the `HessianServiceExporter`, one can transparently expose his services using the lightweight binary HTTP-based protocol provided by Caucho.
- **Burlap:** Burlap is Caucho's XML-based alternative for Hessian. Spring provides support classes such as `BurlapProxyFactoryBean` and `BurlapServiceExporter`.
- **JAX RPC:** Spring provides remoting support for web services via JAX-RPC.
- **JMS:** Remoting using JMS as the underlying protocol is supported via the `JmsInvokerServiceExporter` and `JmsInvokerProxyFactoryBean` classes.

In all these models, services are configured into the application through the spring configuration file known as **spring managed**

beans. This is accomplished by using a proxy factory bean that enables to wire remote services into the properties of our beans as if they were local objects.

Spring provides '**RmiProxyFactoryBean**' to use the RMI service and '**RmiServiceExporter**' to export any spring managed bean as a RMI service.

For wiring a Hessian based service to Spring client, Spring's '**HessianProxyFactoryBean**' is used. To export a Hessian Service '**HessianServiceExporter**' is used and similarly for wiring a Burlap service '**BurlapProxyFactoryBean**' is used and '**BurlapServiceExporter**' is used to export a burlap service. Similarly, for exporting beans as HTTP invoker services, '**HttpInvokerServiceExporter**' is used. To access an HTTP invoker service '**HttpInvokerProxyFactoryBean**' can be used. Spring provides two proxy factory beans to access the Enterprise Java Beans. '**LocalStatelessSessionProxyFactoryBean**' is used to access the EJB in the same container(local) and another one is '**SimpleRemoteStatelessSessionProxyFactoryBean**' which is used to access the remote EJBs.

For all the above models spring provides service exporter classes that exports Java Beans as remote service.

Spring does not provide any EJB Service Exporter and it provides four abstract support classes to make the development of Spring enabled EJB. They are

1. **AbstractMessageDrivenBean** to develop MDBs that accept sources other than JMS.
 2. **AbstractJmsMessageDrivenBean** to develop MDBs that accept messages from JMS sources.
 3. **AbstractStatelessSessionBean** to develop stateless session bean.
 4. **AbstractStstefulSessionBean** to develop stateful session bean.
- '**JaxRpcPostProxyFactoryBean**' is used to wire a web service into the spring application.

Spring

The client makes calls to the proxy to provide the service and the proxy calls the remote service on behalf of the client. Now let's see - how to wire other RMI services into spring application and how to export our own services using the RMI model

Remote Method Invocation (RMI) Model:

RMI was first introduced in JDK 1.1. But developing and accessing RMI services involves various steps and also have lookups which makes the code hard to test. Spring simplifies the RMI by providing a '**proxy factory bean**' that enables us to wire the RMI services into spring application as if they are local beans. Spring also provides a remote exporter that converts our 'spring managed beans' into RMI services. Spring's '**RmiProxyFactoryBean**' is a factory bean that creates a proxy to RMI service. It is declared in the spring configuration file under the <bean> tag as follows,

```
<bean
id="service1" class="org.springframework.remoting.rmi.
RmiProxyFactoryBean>
<property name="serviceUrl">
<value>rmi://${hostname}/service1</
value>
</property>
<property name="serviceInterface">
<value>service1</value>
</property>
</bean>
```

The url of the RMI service is set through the '**serviceUrl**' property. The 'serviceInterface' property specifies the interface that the service implements and only through that the client invokes methods on the service. For using the service the implementation code is wired to the RMI using the following code,

```
<bean id="serviceimpl" class="serviceimpl">
<property name="service1">
<ref bean="service1"/>
</property>
</bean>
```

I. Lets Set up the Environment Variables and start the things:

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

- 1 Copy spring.jar from spring1.2.9\dist folder to the working folder (say D:\springdemo), also copy commons-logging.jar from apache tomcat- 6.0.10 to the working directory.
- 2 Set path for **jdk1.4.2** and above versions.
- 3 Now set the classpath as shown:
D:\springdemo\>set
classpath=D:\springdemo;
D:\springdemo\spring.jar;
D:\springdemo\commons-logging.jar;
- 4 For a typical Spring Application we need the following files:
 - i. An interface that defines the functions.
 - ii. An Implementation that contains properties, its setter and getter methods, functions etc.
 - iii. A XML file called Spring configuration file.
 - iv. Client program that uses the function.

II. Create the following files

1. rmserver.java
2. rmserverimpl.java
3. rmserver.xml
4. rmspring.java

1. D:\springdemo\rmserver.java

```
import java.rmi.*;
public interface rmserver extends Remote
{
String getResult(String s) throws
RemoteException;
}
```

Spring

2. D:\springdemo\rmserverimpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class rmserverimpl extends
UnicastRemoteObject implements rmserver {
public static void main(String args[]) {
try {
rmserverimpl ob = new rmserverimpl();
Naming.rebind("rmserver",ob);
System.out.println("ready");
}
catch(Exception e1) {
System.out.println(""+e1);
}
}
public rmserverimpl() throws
RemoteException {
System.out.println("constructor ok");
}
public String getResult(String a) throws
RemoteException
{
return "Hai..." + a;
}
}
```

3. D:\springdemo\rmserver.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="rmserver"
class="org.springframework.remoting.rmi.
RmiProxyFactoryBean">
<property name="serviceUrl">
<value>rmi://localhost/rmserver</value>
</property>
<property name="serviceInterface">
<value>rmserver</value>
</property>
</bean>
<bean
id="rmserverimpl" class="rmserverimpl">
<property name="rmserver">
<ref bean="rmserver"/>
</property>
</bean>
</beans>
```

4. D:\springdemo\rmspring.java

```
import java.rmi.*;
import org.springframework.beans.factory.*;

import
org.springframework.beans.factory.xml.*;
import org.springframework.core.io.*;
public class rmspring {
public static void main(String args[]) {
try {
System.out.println("Wait..");
Resource res = new
ClassPathResource("rmi.xml");
BeanFactory factory = new
XmlBeanFactory(res);
rmserver bean1 = (rmserver)
factory.getBean("rmserver");
String r=bean1.getResult(args[0]);
System.out.println(r);
}
catch(Exception e1) {
System.out.println(""+ e1);
}
}
}
```

III Compile and run the classes as shown:

```
D:\springdemo>javac rmserver.java
D:\springdemo>javac rmserverimpl.java
D:\springdemo>rmic rmserverimpl (To create
stub and skeleton)
D:\springdemo>javac rmspring.java
D:\springdemo>start rmiregistry (a blank
window will appear)
D:\springdemo>java rmserverimpl
Open another Window and run the client
code by giving the argument
D:\springdemo>java rmspring "Amit"
```

We will get the output as:

```
Wait...
Sep 28, 2007 3:53:10 PM
org.springframework.core.CollectionFactory
<clinit>
INFO: JDK 1.4+ collections available
Sep 28, 2007 3:53:10 PM
org.springframework.beans.factory.xml.
XmlBeanDefinitionReader loadBeanDefinitions
```

Spring

```
INFO: Loading XML bean definitions from
class path resource [rmi.xml]point3
Sep 28, 2007 3:53:10 PM
org.springframework.remoting.rmi.
RmiClientInterceptor prepare
INFO: Using service interface [rmserver] for
RMI stub [rmi://localhost/rmserver] - directly
implemented Sep 28, 2007 3:53:10 PM
org.springframework.aop.framework.
DefaultAopProxyFactory <clinit>
INFO: CGLIB2 not available: proxyTargetClass
feature disabled
Hai Amit
```

Here we have removed the 'lookup' code in the client side.

The Server side of RMI

Spring also supports the server side of RMI. Here the service itself is written with spring and it is exposed as an RMI service. Here the bean is written as a simple JavaBean. Also we need not to generate the stub and skeleton using 'rmic' command and manually add it to RMI registry. Instead of these traditional procedure '**RmiServiceExporter**' is used to export any Spring managed bean as an RMI service. It wraps the bean in an adapter class. The adapter class is then bound to RMI registry and the proxies request the service.

```
<bean
class="org.springframework.remoting.rmi.
RmiServiceExporter">
<property name="service1">
<ref bean="service1"/>
</property>

<property name="serviceName">
<value>service1</value>
</property>
<property name="serviceInterface">
<value>service1</value>
</property>
</bean>
```

The 'serviceName property' indicates the name of service and 'serviceInterface' specifies the interface implemented by the service. There is no need of 'serviceUrl' here. First set the path and classpath as before.

Next edit the service i.e. rmervice.xml

1. D:\springdemo\rmervice.java

```
public interface rmervice {
String getResult(String s);
}
```

2. D:\springdemo\rmerviceimpl.java

```
public class rmerviceimpl implements
rmervice {
    public static void main(String args[]) {
        System.out.println("ready");
    }
    public rmerviceimpl() {
        System.out.println("constructor ok");
    }
    public String getResult(String a) {
        return "Hai"+a;
    }
}
```

3. D:\springdemo\rmervice.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
class="org.springframework.remoting.rmi.
RmiServiceExporter">
<property name="service">
<value>rmervice</value>
</property>
<property name="serviceName">
<value>service1</value>
</property>
<property name="serviceInterface">
<value>rmervice</value>
</property>
</bean>
<bean id="rmervice" class="rmerviceimpl">
</bean>
/beans>
```


Spring

4. D:\springdemo\rmServiceImpl.java

```
import java.io.*;
import org.springframework.beans.factory.*;
import org.springframework.beans.factory.xml.*;
import org.springframework.core.io.*;

class rmServiceImpl {
    public static void main(String args[]) {
        try {
            System.out.println("Wait..");
            Resource res = new
            ClassPathResource("rmService.xml");

            System.out.println("wait..");
            BeanFactory factory = new
            XmlBeanFactory(res);
            System.out.println("factory created");
            rmService bean1 =
            (rmService)factory.getBean("rmService");
            String s = bean1.getResult(args[0]);
            System.out.println(s);
        }
        catch(Exception e1) {
            System.out.println(""+e1);
        }
    }
}
```

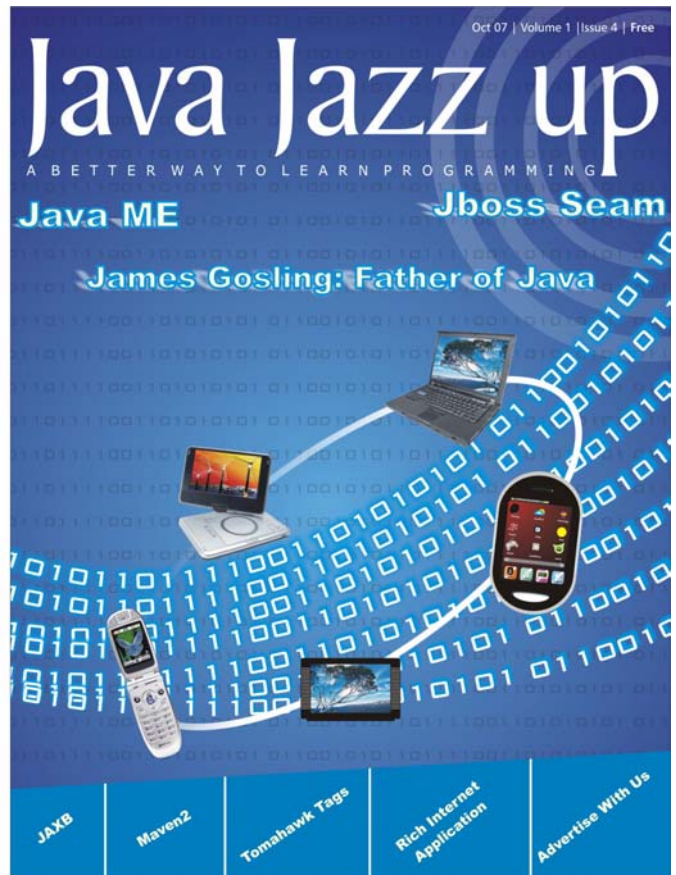
To run:

```
D:\springdemo>javac rmService.java
D:\springdemo>javac rmServiceImpl.java
D:\springdemo>javac rmServiceClient.java
D:\springdemo>java rmsserviceclient "Amit"
```

We will get Output as:

```
wait.
wait...
Sep 28, 2007 3:16:22 PM
org.springframework.core.CollectionFactory
<clinit>
INFO: JDK 1.4+ collections available
Sep 28, 2007 3:16:22 PM
org.springframework.beans.factory.xml.
XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from
class path resource [rmService.xml]
here comes....
constructor ok
Hai Amit
```

Here the service interface doesn't extend the 'java.rmi.Remote' method and 'RemoteException' is not thrown by the methods. There is no binding in the implementation code. Also we can directly run the client. Now there is no need to run the 'rmServerImpl' class. Also there is no need to run the RMI registry.



Java Architecture for XML Binding

Today, XML has emerged as the standard for exchanging data across disparate systems, and Java technology provides a platform for building portable applications. They partners naturally in helping developers to exchange data and programs across the Internet.

Together, they are the most ideal building blocks to develop Web services and Applications accessing web services. This partnership is particularly important for implementing Web services, which provides the users and the application developers, the program functionality on demand from anywhere to anywhere on the Web.

But how do you couple these technologies together in practice? More specifically, what matters much is the issue like how to access and use an XML document (that is, a file containing XML-tagged data) through the Java programming language.

One way to do this is through parsers that conform to the Simple API for XML (SAX) or the Document Object Model (DOM), which is perhaps the most typical way. Java API for XML Processing (JAXP) provides both of these parsers. Java developers invoke a SAX or DOM parser in an application through the JAXP API to parse an XML document — that is, scan the document and logically break it up into discrete pieces. The parsed content is then made available to the application.

Now developers have another Java API which makes it easier to access XML documents: **Java Architecture for XML Binding (JAXB).**

A Reference Implementation of the API is now available in the Java Web Services Developer Pack 2.0 which makes it easier to access XML documents from applications written with Java.

Java Architecture for XML Binding (JAXB) allows java developer for mapping between the Java classes and the XML representations. JAXB allow us for marshalling of java objects into XML and vice-versa i.e. unmarshalling of XML documents back into the java objects. Or we can say JAXB enables us to store and retrieve the data in any XML format into the memory.

Storing and retrieving data from the memory does not require any implementation to a specific set of XML. JAXB is useful in situations where specification is complex and changing. Regularly changing XML schema definitions can be time consuming and error prone as they keep the XML schema definitions synchronized with the java definitions.

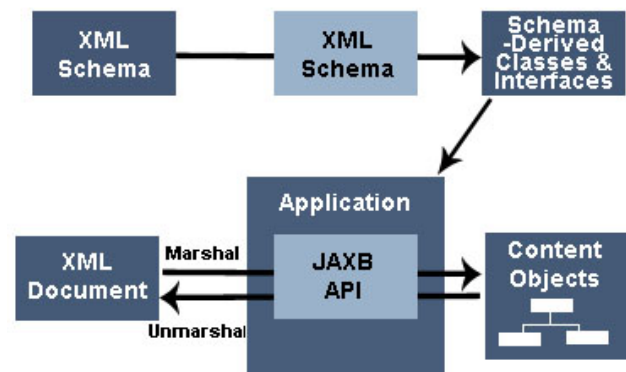
Suppose one need to develop a Java application that accesses and displays data in XML documents.

One approach could be to use the SAX or DOM parsers to access an XML document and then display the data.

In that case, the user would need to:

- Write a program that creates a SAX parser and then uses that parser to parse the XML document. The SAX parser starts at the beginning of the document. When it encounters something significant (in SAX terms, an “event”) such as the start of an XML tag, or the text inside of a tag, it makes that data available to the calling application.
- Create a content handler that defines the methods to be notified by the parser when it encounters an event. These methods, known as callback methods, take the appropriate action on the data they receive.

Now let’s look at how you use JAXB to access an XML document



Java Architecture for XML Binding

The general steps to use the JAXB API are:

1. Bind the schema

Bind the schema for the XML document requires two steps:

Generate classes. An XML schema is used as input to the JAXB binding compiler to generate JAXB classes based on that schema.

Compile classes. All of the generated source files and the application code must be compiled.

2 Unmarshal

XML documents written according to the constraints in the source schema are unmarshalled by the JAXB binding framework i.e. the XML documents are unmarshalled into the Java content objects. Unmarshalling is a wider process, it includes the following steps to follow:

Generate content tree. The unmarshalling process generates a content tree of data objects instantiated from the generated JAXB classes; this content tree represents the structure and content of the source XML documents which are now directly available to the developers program. Now developer can access and process XML data without having to know XML or XML processing.

Validate (optional) the unmarshalling process optionally involves validation of the source XML documents before generating the content tree.

Process the content. The client application can modify the XML data represented by the Java content tree by means of interfaces generated by the binding compiler.

3. Marshal.

The processed content tree is marshalled out to one or more XML output documents. The content may be validated before marshalling.

Now its Time to Dig the Things Deeper

I. Bind the Schema



JAXB simplifies access to an XML document from a Java program by presenting the XML document to the program in a Java format. The first step in this process requires to bind the schema for the XML document into a set of Java classes that represents the schema. Where a schema is an XML specification that governs the allowable components of an XML document and the relationships between the components.

Binding: Binding a schema means generating a set of Java classes that represents the schema. All JAXB implementations provide a tool called a **binding compiler** to bind a schema (the way the binding compiler is invoked can be implementation-specific). For example, the JAXB Reference Implementation provides a binding compiler that one can invoke through scripts i.e. you may bind a schema with a binding compiler provided by the JAXB Reference Implementation

The binding compiler generates a set of interfaces and a set of classes implementing the interfaces.

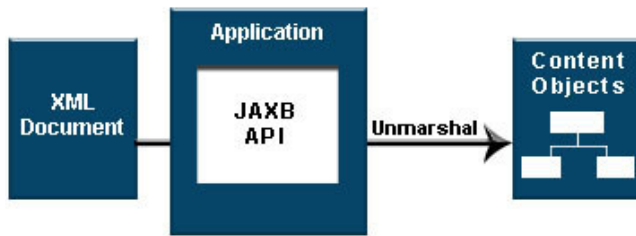
Then this binding compiler compiles and packages the generated interfaces and classes into a package.

II. Unmarshal the Document

Unmarshalling an XML document means creating a tree of content objects representing the content and the organization of the document. The content tree is not a DOM-based tree. In fact, content trees produced through JAXB are more efficient in terms of memory use than DOM-based trees.

Java Architecture for XML Binding

The content objects created are instances of the classes produced by the binding compiler.



Apart from a binding compiler, a JAXB

implementation must provide runtime APIs for JAXB-related operations such as marshalling. The APIs are provided as a part of the binding framework. The binding framework comprises of a main package, `javax.xml.bind`. This package contains classes and interfaces for performing operations such as unmarshalling, marshalling, and validation (marshalling and validation will be covered later).

Validating the Source Data: You can validate source data against an associated schema as part of the unmarshalling operation. JAXB providers have a lot of flexibility here. The JAXB specification mandates that all provider implementations report validation errors when the errors are encountered, but the implementation does not have to stop processing the data. It is possible for a JAXB implementation to successfully unmarshal an invalid XML document, and build a Java content tree. However, the result won't be valid. The main requirement is that all JAXB implementations must be able to unmarshal valid documents.

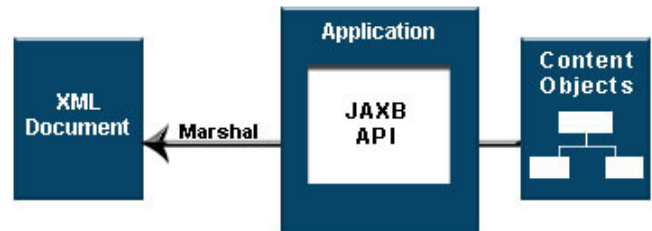
Additionally, you also have the flexibility of turning the validation switch off if you don't want to incur the additional validation processing overhead.

III. Marshal the Content Tree:

Marshalling is the opposite of unmarshalling. Marshalling involves transforming the content tree into a XML document.

Features of JAXB 2.0

JAXB is one of the APIs included with the Java EE platform and also a part of Java Web Services Development Pack (JWSDP). In this



article we are discussing the features of JAXB 2.0; However JAXB 2.1 is available for downloading.

JAXB 2.0 includes some new features:

- It supports for all of the XML Schema constructs.
- It includes parameterized types also.
- It allows us to bind Java-to-XML by using the annotations.
- In this version `setValidation()` method of the `Unmarshaller` interface has been replaced with the JAXP 1.3 validation.
- The new version of JAXB (JAXB 2.0) requires smaller runtime libraries that require lesser runtime memory.
- JAXB 2.0 generates a value class instead of an interface and its implementation class. For each top-level element, JAXB 2.0 generates a Factory class method instead of an interface and an implementation class.
- It generates significantly fewer Java classes from a schema.

How JAXB 2.0 works

To understand how to process XML documents in Java with JAXB 2.0, one need to give a closer look at the two main JAXB components:

- I The **binding compiler** (which binds a

Java Architecture for XML Binding

given XML schema to a set of generated Java classes).

II The binding runtime framework (which provides unmarshalling, marshalling, and validation functionalities).

Binding compiler:

This compiler binds the XML schema with the generated set of java classes. The JAXB binding compiler (or xjc) lets the developers generate Java classes from a chosen XML schema. The JAXB binding compiler transforms an XML schema into a collection of Java classes which matches the structure described in the XML schema. These classes are annotated with special JAXB annotations, which provide the runtime framework with the mappings it needs to process the corresponding XML documents.

Marshaller and unmarshaller together as a technology lets the Java developers easily manipulate XML data in the form of Java objects — without any need to know the details of the processing mechanisms of the parsers like SAX or DOM.

Binding runtime framework:

This framework provides support for marshalling, unmarshalling and the validation functionalities. Marshalling is the mechanism of placing data items before moving it over the communication channel. To unmarshall an XML document, you need to create an unmarshaller from the context. The unmarshaller processes the XML data from a wide variety of data sources, including files, input streams, URLs, DOM objects, SAX parsers, and more.

Marshalling involves transforming your Java classes into XML format. In JAXB 2.0, it's simple to create and manipulate these Java classes. One can treat XML as an ordinary class (according to the corresponding XML schema).

Getting the things ready to go with JAXB 2.0:

- 1 Download the Java Web Services Developer Pack 2.0. It includes an implementation of JAXB 2.0.
- 2 **Install** JWSDP 2.0 at the location say C:\Sun\jwsdp-2.0 (which is the default installation directory). JAXB 2.0 will get installed in the directory C:\Sun\jwsdp-2.0\jaxb.
- 3 Include the path of the bin directory (i.e. C:\Sun\jwsdp-2.0\jaxb\bin) to the PATH variable.
- 4 Since JAXB 2.0 the parameterized types features of JDK5.0, so first install the JDK 5.0 and then set the environment variables for the JAVA_HOME, JAXB_HOME and JAVA.

The following table lists the jar files required by JAXB 2.0.

JAR File Description	
C:/Sun/jwsdp-2.0/jaxb/lib/jaxb-api.jar	JAXB 2.0 API classes
C:/Sun/jwsdp-2.0/jaxb/lib/jaxb-impl.jar	JAXB 2.0 implementation classes
C:/Sun/jwsdp-2.0/jaxb/lib/jaxb-xjc.jar	JAXB 2.0 Compiler classes
C:/Sun/jwsdp-2.0/jwsdp-shared/lib/activation.jar	javax.activation package classes.
C:/Sun/jwsdp-2.0/sjsxp/lib/jsr173_api.jar	StAX API classes
C:/Sun/jwsdp-2.0/sjsxp/lib/sjsxp.jar	StAX classes

JAXB prerequisites: To get started with JAXB 2.0 we require

- Java Platform and Standard Edition 5: JAXB 2.0 mainly based on the features of Java SE 5, like generics and annotations.
- Implementation of JAXB 2.0

We will implement JAXB 2.0 in the next issue.

Structural Design Patterns

I. Adapter Pattern:

Structural Design Patterns This pattern establishes a relationship between the two unrelated interfaces such that they work together. This is similar to the conversion of one interface of one class to the interface expected by the client. To understand clearly let's take an example:

Suppose there are several sockets of different sizes, and shapes in a house. Due to variation in sizes, all the sockets are not capable of plug-in a mobile charger. So we use an Adapter to make it portable across different sockets. Here Adapter works as a connector. This connector connects them together and meets the purpose of the client.

Benefits: It helps the developers to relate the unrelated class such that they may work together. It provides compatibility between the classes and increases the transparency. Additionally it provides a pluggable kit, delegates objects, makes the classes highly reusable and achieves the goal through inheritance or composition.

It also tries to match the interfaces such as WindowAdapter, KeyAdapter, MouseAdapter, ContainerAdapter, ComponentAdapter, FocusAdapter and MouseMotionAdapter.

Usage: The adapter pattern is meant to use an existing class to fulfill the client's class requirements. For example, suppose a client specifies his requirement in an interface, and then to meet the purpose usually a class is created that implements the required interface and inherits the existing class into the subclasses. This approach creates a class known as the adapter class which translates the client's calls to the existing class methods. Adapter pattern also helps to make improvements in the new code and to make the code more manageable. It improves the performance while designing the system.

Now think all such things technically and try to resolve the problem. There are two ways of implementing the Adapter Pattern, either use the Inheritance or use the composition.

Let's do it with the approach of Inheritance
First develop a socket interface as:

Switch.java

```
public interface Switch {  
  
    public void switchOn();  
    public void switchOff();  
  
}
```

Implement Switch.java in a class say Fan class. The class is given below:

Fan.java

```
public class Fan implements Switch{  
  
    public void switchOn() {  
        System.out.println("FAN Switched ON");  
    }  
    public void switchOff() {  
        System.out.println("FAN Switched OFF");  
    }  
    public static void main(String arg[]){  
        Fan f = new Fan();  
        f.switchOn();  
        f.switchOff();  
    }  
}
```

Here is the output of the above Fan.java program:

```
C:\>javac Fan.java  
C:\>java Fan  
Fan Switched ON  
Fan Switched OFF
```

Finally, there will be an adapter class. This will inherit the Switch and give output for Fan.

Structural Design Patterns

Bulb.java:

```
public class Bulb implements Switch {
    public void switchOn() {
        System.out.println("BULB Switched ON");
    }
    public void switchOff() {
        System.out.println("BULB Switched OFF");
    }
}
```

Here is the output of the above Bulb.java program:

```
C:\>javac Bulb.java
C:\>java Bulb
Bulb Switched ON
Bulb Switched OFF
```

Similarly, let's consider the Association and Composition of objects by which Adapter can be implemented.

The above example shows how Adapter pattern works. When one interface cannot be changed and needed to suit again a cannot-be-changed client, then an adapter is used so that both the interfaces work together.

II. Bridge Pattern:

Builder pattern provides independence to the interface from its implementation. It provides flexibility to both to vary independently. Suppose we have a database containing multiple questions and we want to display the questions on the basis of the user selections. Then such type of problems can be solved with the Bridge Design Pattern. It does that simply by decoupling the relationship among the objects.

Benefits: It separates the abstraction from the implementation details. Inheritance tightly couples the abstraction with the implementations at the compile time, However the Bridge pattern hides the implementation details, improves the extensibility, shares an implementation among multiple objects. It reduces the number of subclasses, sometimes use of pure inheritance increases the number of subclasses. It also improves the extensibility

by extending independency between the abstraction and the implementation.

Usage: It is used in such situation if you do not want the permanent binding between an abstraction and its implementation. It is frequently used in those places where changes made in the implementation does not effects the clients. It can be used to fulfill such requirements where the abstractions and the implementations needs to be extensible.

Now lets take an example:

First develop a **Question.java** interface:

Question.java:

```
interface Question {

    public void nextQuestion();
    public void priorQuestion();
    public void newQuestion(String q);
    public void deleteQuestion(String q);
    public void displayQuestion();
    public void displayAllQuestions();

}
```

Develop another class **QuestionManager** implementing the Question.java interface:

```
class QuestionManager {

    protected Question questDB;
    public String catalog;

    public QuestionManager(String catalog) {
        this.catalog = catalog;
    }

    public void next() {
        questDB.nextQuestion();
    }

    public void prior() {
        questDB.priorQuestion();
    }

    public void newOne(String quest) {
        questDB.newQuestion(quest);
    }

}
```

Structural Design Patterns

```
public void delete(String quest) {
questDB.deleteQuestion(quest);
}

public void display() {
questDB.displayQuestion();
}

public void displayAll() {
System.out.println("Question Catalog: " + catalog);
questDB.displayAllQuestions();
}

}
Develop another class QuestionFormat
extending the QuestionManager class
QuestionFormat.java
class QuestionFormat extends QuestionManager {

public QuestionFormat(String catalog){
super(catalog);
}

public void displayAll() {

System.out.println("\n~~~~~");
super.displayAll();
System.out.println("~~~~~");
}
}
```

Develop another class **JavaQuestions** implementing the "Question" interface:

JavaQuestions.java:

```
import java.util.*;
class JavaQuestions implements Question {

private List <String> questions =
new ArrayList<String>();

private int current = 0;

public JavaQuestions() {
questions.add("What is Java? ");
questions.add("What is marker interface? ");
questions.add("What is cross-platform? ");
questions.add("How multiple polymorphism is achieved in java? ");
questions.add("How many types of exception handling are there in java? ");
}
```

```
questions.add("Define the keyword final for variable, method, and class in java? ");
questions.add("What is multi-tasking? ");
questions.add("What is multi-threading? ");

}

public void nextQuestion() {
if( current <= questions.size() - 1 )
current++;
}

public void priorQuestion() {
if( current > 0 )
current--;
}

public void newQuestion(String quest) {
questions.add(quest);
}

public void deleteQuestion(String quest) {
questions.remove(quest);
}

public void displayQuestion() {
System.out.println( questions.get(current) );
}

public void displayAllQuestions() {
for (String quest : questions) {
System.out.println(quest);
}
}

}
```

Develop another class TestBridge

TestBridge.java

```
class TestBridge {

public static void main(String[] args) {

QuestionFormat questions =
new QuestionFormat("Java Language");

questions.questDB = new JavaQuestions();
}
```


Structural Design Patterns

```
//
questions.questDB = new CsharpQuestions();
//
questions.questDB = new CplusplusQuestions();

questions.display();
questions.next();

questions.newOne("What is polymorphism? ");
questions.newOne("How many types of
polymorphism are there in java?");
questions.displayAll();
}
}
```

Here is the output of the above program:

```
C:\ Command Prompt
C:\> javac TestBridge.java
C:\> java TestBridge
What is Java?
~~~~~
Question Catalog: Java Language
What is Java?
What is marker interface?
What is cross-platform?
How multiple polymorphism is achieved in
java?
How many types of exception handling are
there in java?
Define the keyword final for variable, method,
and class in java?
What is multi-tasking?
What is multi-threading?
What is polymorphism?
How many types of polymorphism are there in
java?
~~~~~
```

The above example tries to show how the Bridge pattern decouples the interface from its implementation. One can easily notice that the class JavaQuestion can be launched independently working as an independent system.

III. Composite Pattern:

Individual objects as well as the composite objects can be represented with the Composite Design Pattern. Composite pattern represents

these objects with a tree structure. Suppose, within a company, there is an employee hierarchy where a manager has its subordinates, Project Leader also has the subordinates, while the developer has no subordinates.

Benefits: It is more flexible as compared to the static inheritance. It simplifies coding by implementing each feature in a class. It enhances the capability of an object as the new classes are created to add new features and make some changes.

Usage: This design pattern is used when the responsibilities are needed to be added dynamically to the individual objects without affecting other objects. Where an object's responsibilities may vary from time to time. Now, let's try to solve the above problem technically.

Develop a class "Employee" having the getters and setters for the attributes empname, empsal and emp subordinates.

Employee.java

```
class Employee {
String Empname;
double Empsalary;
Employee(String n, double s){
Empname = n;
Empsalary = s;
}
String getName() {
return Empname;
}
double getSalary() {
return Empsalary;
}
public String toString() {
return "Employee" + Empname;
}
}
```

For example, General Manager may have several employee and some of them are Managers, further these managers have several employees. To illustrate all these things, let's design a simple Manager class.

Structural Design Patterns

Manager.java:

```
class Manager {
    Manager manager;
    Employee[] emply;
    String dept;
    Manager(Manager mgr, Employee[] e, String d) {
        this(e, d);
        this.manager = mgr;
    }

    Manager(Employee[] e, String d) {
        emply = e;
        dept = d;
    }
    String getDept() {
        return dept;
    }
    Manager getManager() {
        return manager;
    }
    Employee[] getEmployee() {
        return emply;
    }
    public String toString() {
        return dept + " manager";
    }
}
```

Here is the "Test" class that shows the information about the "Manager" class.

Test.java:

```
class Test {
    public static void main(String[] args) {
        Employee[] e1 = {new Employee("Zulfiqar", 90),
            new Employee("Amit", 80)};
        Manager m1 = new Manager(e1, "Accounting");
```

```
        Employee[] e2 = {new Employee("Aquil", 70),
            new Employee("Ravi", 60),
            new Employee("Vinod", 40)};
```

```
        Manager m2 = new Manager(m1, e2, "Production");
```

```
        System.out.println(m2);
        Employee[] emp = m2.getEmployee();
        if (emp != null)
            for (int k = 0; k < emp.length; k++)
                System.out.println
                    (" "+emp[k]+" Salary: $" + emp[k].getSalary());
        Manager m = m2.getManager();
        System.out.println(" " + m);
        if (m != null) {
            Employee[] emps = m.getEmployee();
            if (emps != null)
                for (int k = 0; k < emps.length; k++)
                    System.out.println
                        (" " + emps[k]+" Salary: $" + emps[k].getSalary());
        }
    }
}
```

Here is the output of the program:

```
C:\> java Test
Production manager
Employee Zulfiqar Salary: $90.0
Employee Amit Salary: $90.0
Employee Aquil Salary: $80.0
Accounting manager
Employee Ravi Salary: $70.0
Employee Vinod Salary: $50.0
```

The above example concludes that the **composite pattern** allows us to create a tree like structure for both simple and complex objects.



Develop-JSF Application

This section is very useful for any beginner in the field of JSF (Java Server Faces) framework of Java. This example covers all you need to develop the application, for example, using JSF tags, creating properties files and managed beans, modifying configuration files like faces-config.xml and web.xml, directory structure of the application etc. Detailed explanation of this example will definitely support you to develop JSF applications with rich set of functionality of JSF.

In this application, the first page that the user experiences is displayed with an input text box and a command button component. User enters name in the box and presses the button then user is welcomed in the next page.

Steps Followed:

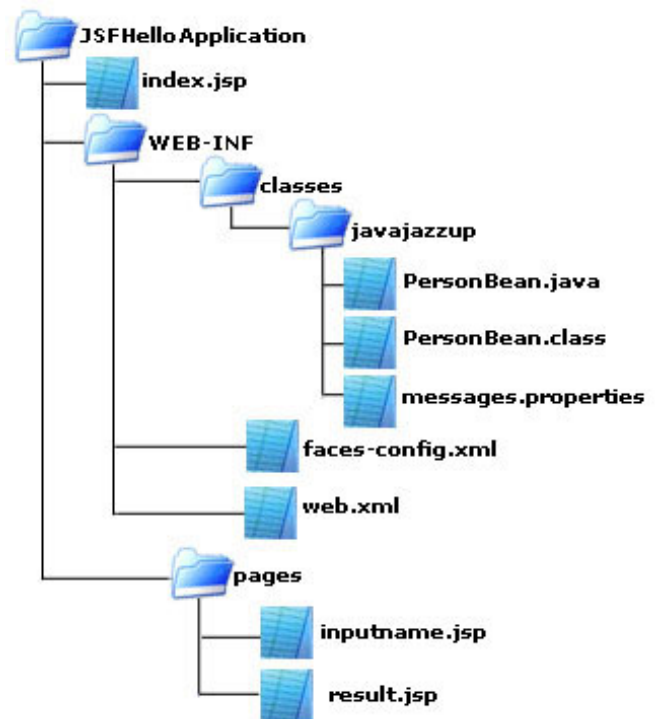
We will follow the steps below to create this application:

- 1 Create development directory structure (root directory and sub directories)
- 2 Create and place configuration files in appropriate place
- 3 Create JSP pages
- 4 Create a properties file
- 5 Create a managed bean
- 6 Register managed bean in configuration file
- 7 Define a navigation rule in configuration file
- 8 Run the application

To understand clearly where to place which file, directory structure of this application will help you a lot. So have a look on it below:

Directory structure of this application:

The pictorial directory structure given below is very useful to understand the application. This structure shows where to put which file or directory.



Create and place directories, configuration files:

We used JDK 1.6.0 and TOMCAT 5.5.23 to deploy and run this application. So you are expected to install and configure TOMCAT for JSF.

Directories:

In tomcat, web applications are placed within webapps folder. Now we are going to start creating "JSFHelloApplication" application so the first step is to create a folder in web apps with the name "JSFHelloApplication". This is the root directory of the application. Now create WEB-INF folder in root directory and place configuration files web.xml and faces-config.xml file.

Configuration files:

1. web.xml: You can get web.xml file from WEB-INF folder of any other application available in TOMCAT by default or you can create yourself with the same name and extension of the file i.e. "web.xml". If you are creating this file then take care of mentioning version of xml. For ex. `<?xml version="1.0"?>` at the top of file and after that all elements will be written within

JSF Application

<web-app> element. So the initial format of this file will be like this:

```
<?xml version="1.0"?>
  <web-app>
    .....
    .....
    .....
  </web-app>
```

If you want to create this file then write above code in notepad and save it with name "web.xml" in the WEB-INF folder of your application. After creating and placing this file to the appropriate position, we have to add child elements within web-app element. We will write those elements will be described later.

2. faces-config.xml: Now we come to the second file "faces-config.xml" that will be in the same place where web.xml is i.e. within WEB-INF folder. Here also you have to take care of mentioning version of xml as we did in web.xml file. All sub elements will be written within <faces-config> element. So initial format of this file will be like this:

```
<?xml version="1.0"?>
  <faces-config>
    .....
    .....
    .....
  </faces-config>
```

You can create this file also by your own or copy from other JSF Application. If you want to create this file then you can write the above code in notepad and save it with the name "faces-config.xml" in WEB-INF folder of your application. After creating and placing this file to the appropriate position, we have to add some elements within faces-config element. How we will write those elements will be described later.

So now there will be two xml files web.xml and faces-config.xml in WEB-INF directory.

This JSF application contains:

- 1 Three JSP pages for viewing purpose
- 2 JavaBean to hold model data
- 3 Configuration files specifying managed bean, navigation rules, controller servlet.

Now our first step is to create view part of the application. For this we have created three JSP files given below:

1. index.jsp
2. inputname.jsp
3. result.jsp

Creating JSP pages:

1-index.jsp:

The index page is stored in root directory "JSFHelloApplication". The code for "index.jsp" is:

```
<html>
  <body>
    <jsp:forward page="/pages/
inputname.jsf" />
  </body>
</html>
```

Description:

As you can see in code above, this page simply forwards the user to the page "inputname.jsp" through <jsp:forward page="/pages/inputname.jsf" /> line of code. Index page doesn't display anything to the user so you can leave creating this page but it has one benefit that you can start application simply mentioning the application name and not specifying any file name at the end of URL i.e. we can simply write http://localhost:8080/JSFHelloApplication in the URL and see output of the application.

So the first page that appears to the user is "inputname.jsp" not "index.jsp". The code for "inputname.jsp" is:

JSF Application

2-inputname.jsp:

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f" %>
<f:loadBundle
basename="javajazzup.messages"
var="message"/>

<f:view>
<html>
  <head><title>enter your name page</
title></head>

  <body>
    <h:form>
      <h1><h:outputText
value="#{message.inputname_header}"/></
h1>
      <h:outputText
value="#{message.prompt}"/>
      <h:inputText
value="#{StoreNameBean.personName}" />
      <h:commandButton
action="result"
value="#{message.button_text}" />
    </h:form>
  </body>
</html>
</f:view>
```

Description:

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

"taglib" directive is used to include JSF tag libraries. First line tells where to find JSF html tags that defines html elements and second line tells where to find JSF core tags. A page that contains JSF tags is represented by a tree of components. The root of this tree is UIViewRoot. This root is represented by view tag. So it is necessary to include all component tags (tags representing UI components) within view tag.

```
<f:loadBundle basename="
javajazzup.messages" var="message"/>
```

This line loads our properties file (resource bundle) that holds messages to be displayed in the JSP page. Actually this file is a collection of "param=value" pair. The name of this file is "messages.properties" in this application which is saved in /WEB-INF/classes/ javajazzup folder. We will explain more about this in subsequent section.

<h:form> tag is used to create html <form> using JSF tag. Typically JSP page includes a form, which is submitted when a button is clicked. Form components must be used within <h:form> and </h:form>.

```
<h:outputText
value="#{message.inputname_header}"/>
```

This tag displays text on the page defined in the value attribute by looking into the resource bundle i.e. properties file (messages.properties). It looks the value for inputname_header parameter in "message.properties" file and set the value of it to value attribute. Finally this tag prints this value. So in this example this line prints "Java Jazz Up".

```
<h:outputText value="#{message.prompt}"/
>
```

In this line the value of "prompt" param is looked in "messages.properties" file and this tag prints this value. So in this example this line prints "Enter Your Name:"

```
<h:inputText
value="#{StoreNameBean.personName}"/>
```

inputText tag is used to create input text box component. The value attribute is connected with the managed bean attribute. Here StoreNameBean is the name of Bean and personName is the name of attribute of bean. After pressing the submit button, bean gets the value filled in the input text box. This bean is nothing but a Java Bean that contains attributes and setter and getter methods to

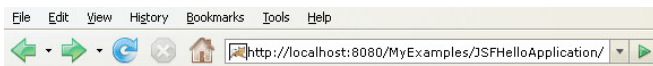
JSF Application

set and get those attributes. We will explain more about Managed Bean later in this section.

```
<h:commandButton action="result"
value="#{message.button_text}" />
```

commandButton tag represents command button component. Here again the value attribute gets value of button_text param from "messages.properties" file. So in this example this line prints "Submit" on button component. The action attribute is used to see which page will be displayed next when we will press this button. This "result" value is matched in faces-config.xml file in WEB-INF folder where navigation rules are defined. How to define navigation rules in faces-config.xml file will be described later in this section. Here in this application the next page is "result.jsp" when submit button is pressed.

The collective output of tags used in inputname.jsp will give rise to the first page appeared in front of the user: Output of the page is given below:



Java Jazz Up

Enter Your Name:

Done

When above page appears to the user, user enters name to the input text field and submits the button, a new page "result.jsp" is generated that welcomes the user with the user name. The code for "result.jsp" is:

3-result.jsp:

```
<%@ taglib uri="http://java.sun.com/jsf/
html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/
core" prefix="f" %>
<f:loadBundle
basename="javajazzup.messages"
var="message"/>
```

```
<html>
  <head><title>greeting page</title></
head>

  <body>
    <f:view>
      <h3><h:outputText
value="Hi,#{StoreNameBean.personName}!"
/>
      <br/><h:outputText
value="#{message.greeting_text}" /></h3>
    </f:view>
  </body>
</html>
```

Description:

First three lines are same as in "inputname.jsp" file.

```
<h:outputText value="Hi,
#{StoreNameBean.personName}!" />
```

The above line is used to access the value of personName attribute from Java Bean named "StoreNameBean" and prints this value (i.e. person's name) on the page.

```
<h:outputText
value="#{message.greeting_text}" />
```

This line looks the value of greeting_text in "message.properties" file and prints this value to the page. Here this line prints "Welcome In Java Jazz Up".

Output of result.jsp:

So output will be like this (if "rose" is entered in the text field of "inputname.jsp" page):



JSF Application

Now we come to those topics that have been left unexplained above.

Creating properties file (resource bundle):

In above JSP files we have used "message.properties" file. A properties file is a collection of "param=value" pairs. We can use there param names in our JSP file and get its value from here as we did previously. The benefit of using properties file is that we can modify these values easily in one place and there is no need to change JSP files. In this application we have created "messages.properties" file in javajazzup folder in WEB-INF/classes folder. The code for this file is:

messages.properties:

```
inputname_header=Java Jazz Up  
prompt=Enter Your Name:  
greeting_text=Welcome In Java Jazz Up  
button_text=Submit
```

Creating Managed Bean:

In the above JSP files we have used Managed Bean named "StoreNameBean". For this we have created "PersonBean.java" file. This Managed Bean is nothing but a Java file that contains attributes and setter and getter methods to set and get those attributes. Here in this example, there is only one attribute named "personName" and so only one setter method setPersonName() and one getter method getPersonName() is created. This bean is used to set the value to the bean attribute from the page and get the value from bean to the page. Make sure the attribute in this class must be same as the field name in JSP. In this example this file is created in package javajazzup. So compile this file and place its class file i.e. PersonBean.class in javajazzup folder in WEB-INF/classes folder. The code for this class is:

```
PersonBean.java:  
package javajazzup;  
  
public class PersonBean {  
    String personName;  
    public String getPersonName() {
```

```
        return personName;  
    }  
  
    public void setPersonName(String name) {  
        personName = name;  
    }  
}
```

If you want to access the bean classes in your JSP files, you have to register the bean classes in faces-config.xml.

Registering managed bean:

We have already created faces-config.xml file with empty <faces-config> element. This configuration file is used to register managed beans, specifying navigation rules etc. We will add <managed-bean> element within <faces-config> and </faces-config> tag to register Managed Bean.

```
<managed-bean>  
<managed-bean-name>StoreNameBean</  
managed-bean-name>  
<managed-bean-class>  
javajazzup.PersonBean</managed-bean-  
class>  
<managed-bean-scope>request</managed-  
bean-scope>  
</managed-bean>
```

Bean's name is given in <managed-bean-name> tag. This name is used in JSP files to represent the bean. The class name that corresponds to this bean is given in <managed-bean-class> tag. <managed-bean-scope> defines the scope for the bean. In this Application, name of the bean that will be used in JSP files is "StoreNameBean".

Defining navigation rule:

Now we will understand how navigation from one page to the next page is performed as in our application inputname.jsp page navigates to result.jsp page when user presses submit button after filling text in input text field. To understand this we come back to the line of code used in "inputname.jsp":

JSF Application

```
<h:commandButton action="result"
value="#{message.button_text}" />
```

Here action attribute is set to "result". When user presses the command button then which page will be displayed is determined by the navigation rule defined in faces-config.xml configuration file. This rule has been defined like this for our application:

```
<navigation-rule>
  <from-view-id>/pages/inputname.jsp</from-view-id>
  <navigation-case>
    <from-outcome>result</from-outcome>
    <to-view-id>result.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

<navigation-rule> defines navigation rule. <from-view-id> is used to specify the jsp file for which navigation rule is to be defined. Here in our application it is inputname.jsp that is in pages folder. <navigation-case> specifies the value which is matched with the value specified in action attribute of commandButton tag. If it matches then the page specified within <to-view-id> tag is displayed. Here in our application it is "result.jsp".

So after editing faces-config.xml file, it will look like following:

```
<?xml version="1.0"?>
<!DOCTYPE faces-config PUBLIC
"-//Sun Microsystems, Inc.//DTD JavaServer
Faces Config 1.1//EN"
"http://java.sun.com/dtd/web-
facesconfig_1_1.dtd">

<faces-config>
  <managed-bean>
    <managed-bean-
name>StoreNameBean</managed-bean-
name>
    <managed-bean-class>
javajazzup.PersonBean</managed-bean-
class>
    <managed-bean-scope>request</
```

```
managed-bean-scope>
  </managed-bean>
  <navigation-rule>
    <from-view-id>/pages/
inputname.jsp</from-view-id>
    <navigation-case>
      <from-outcome>result</from-
outcome>
      <to-view-id>result.jsp</to-view-
id>
    </navigation-case>
  </navigation-rule>
</faces-config>
```

Editing web.xml:

The "FacesServlet" servlet handles JSF applications. So as we are using JSF framework in our web application, we will edit the deployment descriptor file web.xml to define "FacesServlet" and its mapping in web.xml file.

```
<servlet>
<servlet-name>Faces Servlet</servlet-name>
<servlet-
class>javax.faces.webapp.FacesServlet</
servlet-class>
<load-on-startup> 1 </load-on-startup>
</servlet>

<!-- Faces Servlet Mapping -->
<servlet-mapping>
<servlet-name>Faces Servlet</servlet-name>
<url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

<servlet> element maps the "javax.faces.webapp.FacesServlet" servlet class to a symbolic name i.e. Faces Servlet is an alias for "javax.faces.webapp.FacesServlet" servlet. <servlet-mapping> element is used to map any request of pattern like .jsf in the URL must be passed to the Faces servlet.

The FacesServlet servlet works as an engine for all JSF applications(handling of all JSF related requests, building component tree of the JSP page, accessing all JSP pages in the application, creating an Event object and passing it to any registered listener). So all requests that need

JSF Application

FacesServlet processing must be directed to this servlet. So if we want to invoke this servlet with every request we have to do mapping in <servlet-mapping> element. This is done to map a particular URL pattern with the Faces servlet. URL of every request must contain <file name>.jsf pattern because we have mentioned this pattern in <url-pattern> tag.

If we look in "index.jsp" file, we will find that there is .jsf file suffix not .jsp in the path for the forward.

```
<jsp:forward page="/pages/inputname.jsf" />
```

This is because we have used *.jsf in the URL pattern in the web.xml file for the application. This is used to signal that the forwarded page should be handled by the FacesServlet servlet within Tomcat.

Running the application:

This application is now complete. To run this application start Tomcat server and type "http://localhost:8080/JSFHelloApplication" URL in the address bar of your browser and hit enter.

Liberty from tough books

Java codes

Java News

Maven 2

JSF tags

Web 3.0

Java Jazz up

A BETTER WAY TO LEARN PROGRAMMING



<http://www.javajazzup.com>

**Get
More and More
Tutorials**

FIND QUALITY

STUFF ON

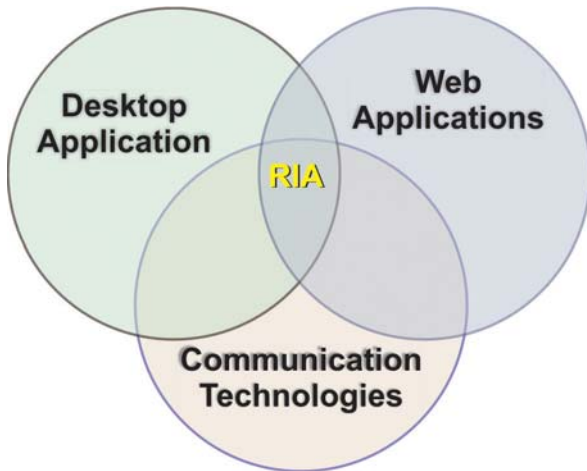
JAVA TECHNOLOGY

AT



<http://www.roseindia.net>

Rich Internet Application



The term RIA (Rich Internet Applications) refers to web applications that have the features and functionality of traditional desktop applications, it means Rich Internet Applications are a cross between web applications and traditional desktop applications that shift some of the essential processing among the bulk of the data for the user interface to the Web client while rest of some remain on application server.

In the Rich Internet Application, the term 'Rich' stands for the broad range of media includes multiple fonts, vector and bitmap graphic files, animations, online conferencing, audio and video. It soothes to the eyes and the materials are more catchy due to its richness medium. e.g. any picture in the flash is more attractive than ordinary still picture.

Usually a Rich Internet Application runs in a Web browser. It does not need the installed software. It runs locally in a sandbox that provides a secure environment.

Origin of RIAs

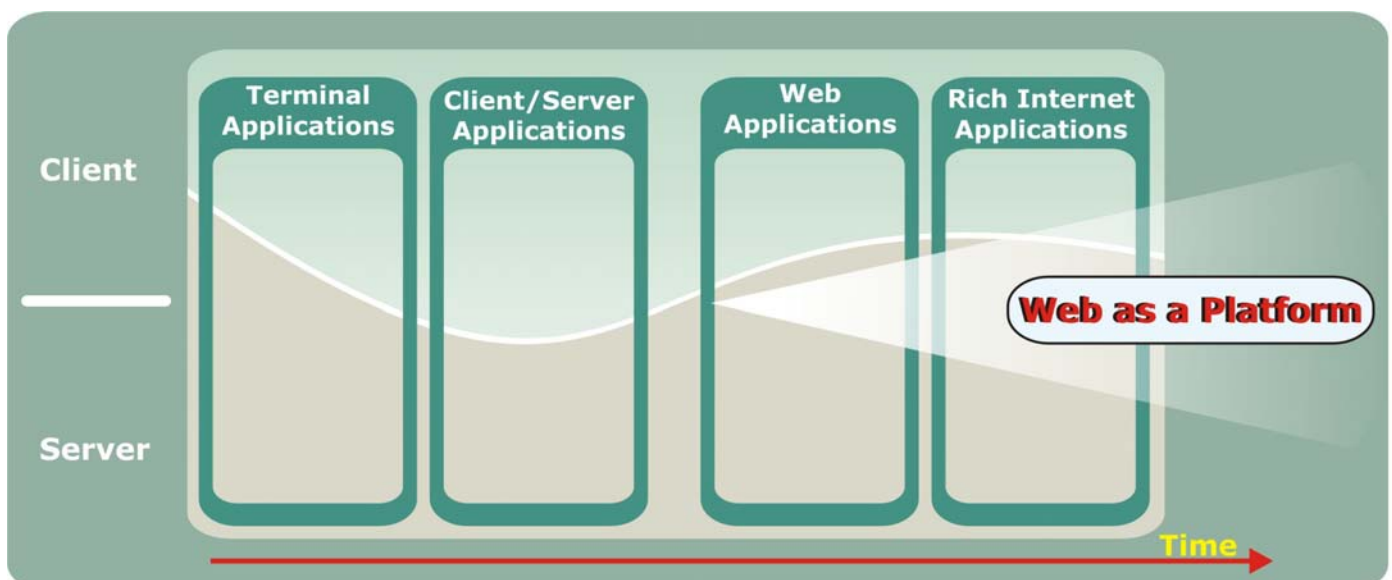
Before transforming in its in original name RIA had been exist in several names and forms. For example: Remote Scripting, X Internet, Rich Web clients and Rich Web Application.

Microsoft exposed the initial name of Remote Scripting in 1998, while Forrester Research recalled it with 'X Internet' in October 2000. Later it is also known as Rich Web client and Rich Web Application.

The term 'Rich Internet Application' was first introduced in March 2002 in a Macromedia white paper.

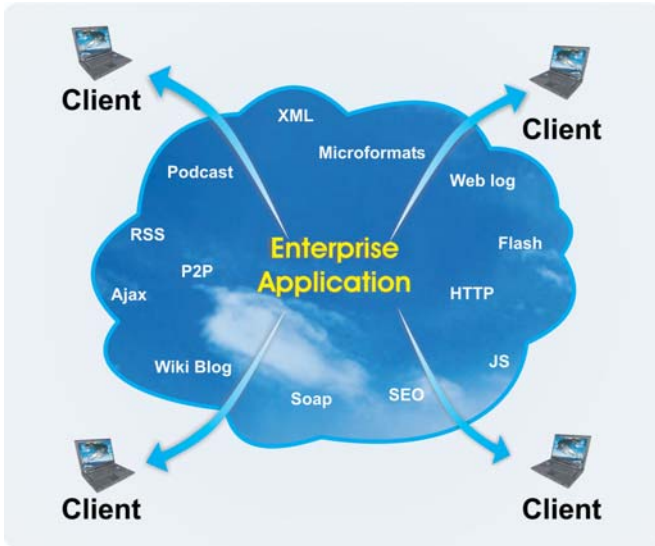
Rich Internet Application Vs. Standard Web Applications

Traditional web applications processed all activity around client-server architecture and a thin client. All the processing held on the server, while client only displays static content in case of being HTML content. Passing all the interaction through the server that is required to reload the data is the biggest drawback of this system because all interactions with the application



Rich Internet Application

have to pass through the server that is unnecessary and lowers the processing speed. On the other hand, RIAs can dodge this slow and synchronous loop for many user interactions. This difference is rather similar to the difference between “terminal and mainframe”



and Client-server/Fat client approaches.

As the development of Internet standards have held gradually but continuously according to turning time, it is very difficult to draw a strict line between the composition materials of an RIA.

But one characteristic is similar in all RIAs as they introduce an intermediary layer of code between the user and the server that is often known as a client engine. It acts as an extension of the browser and generally downloaded at the beginning of the application that can be supplemented by further code downloads as the application progresses. The client engine usually takes over responsibility for rendering the application's user interface and for server communication. In most RIAs the client engine performs additional asynchronous communications with servers.

Benefits

There are several benefits of RIAs over Traditional Web Application; these are:

- RIAs are usually richer in functionality as they offer user-interface behaviors using only the HTML widgets that can include any technology being used by the client side, including drag and drop, using a slider to change data, calculations performed only by the client and not need to be sent back to the server.
- The interface behaviors of RIAs are usually much more responsive in the comparison of interface behaviors of a standard Web browser while it can also generate other performance benefits when it uses a client engines. These are:

- 1 It is able to make better balance between Client and Server that frees server resources allowing the same server hardware to handle more client sessions concurrently.
2. It is also be able to make asynchronous communication without waiting for the user to perform an interface action like clicking on a button or link. So the RIA designers feel free to move data between the client and the server without making the user wait. Besides this prefetching is the most common application, in which an application anticipate a future need for certain data, and downloads it to the client before the user requests it, because of running at high speed and getting up a consequent response. Google Maps uses this technology with efficiently and on the massive scale to move adjacent map segments to the client before the user scrolls their view.

There is another benefit of RIA in terms of network efficiency as the huge traffic significantly reduces in Rich Internet Applications because an application-specific client engine is more intelligent than a standard Web browser when deciding what data needs to be exchanged with servers. This boosts the individual requests or responses due to less data transferring in each interaction. Thus the overall network load becomes reduced.

Rich Internet Application

Meanwhile, use of asynchronous prefetching techniques can either neutralize or can reverse this potential benefit, as many times the code cannot anticipate exactly what every user will do next, it is common for such techniques to download extra data, not all of which is actually needed, to many or all clients.

Shortcomings and restrictions

Despite of having flair advantage over the standard web application, RIAs have several shortcomings and restrictions too. These are:

1- Sandbox

As RIAs run within a sandbox, so the correct operation of sandbox is necessary to run RIA successfully. If assumptions about access to resources are incorrect, RIAs may fail to operate correctly that restricts access to system resources.

2- Disabled scripting

For running RIAs, any scripting language includes JavaScript is usually essential. In case of disabled active scripting in the browser, there is no functioning in RIA.

3- Client processing speed

Some Rich Internet Applications uses client-side scripts written in interpreted language that moderate the performance speed, while compiled client language in traditional application have no relation with speed.

4- Script download time

If user downloads any document, it has to be transferred at least once from the website to the system's cache memory. Though the downloaded document does not need to be installed on the system, yet it sometimes takes unexpected long time. RIA developers can reduce the delay time by compressing the scripts, and by staging their delivery over multiple pages of an application.

5- Loss of visibility to search engines

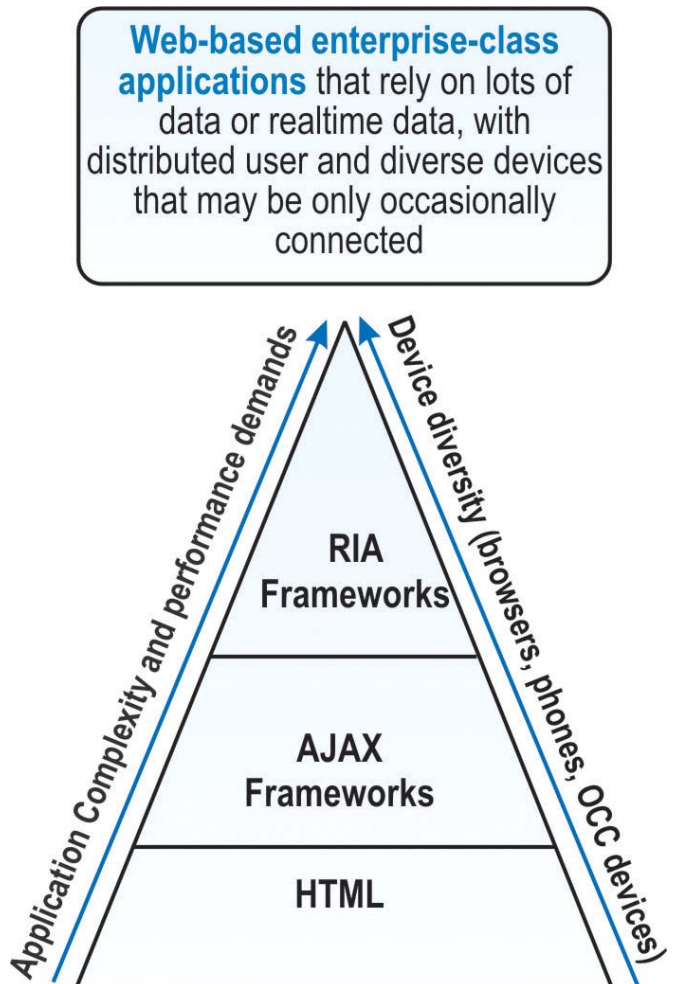
Search engines may not be able to index the text content of the application.

6- Dependence on an Internet connection

Like traditional Web Application, RIA also needs internet connectivity; the speed of RIA operation also depends upon the network connection. An ideal network connection is usually suitable for running RIA smoothly, otherwise it may cause of headache.

7- Management complications

Traditional Web applications are simpler because of having only standard HTML built-in format while the initiation of RIA technologies had make it more complex and difficult to handle. The



Rich Internet Application

additional complexity of RIA makes them harder to design, test, measure, and support. These complications elongate the software development process, despite of the particular methodology or process being employed. Due to its sluggish processing, it becomes difficult to test the applications and incomplete testing lowers the application's quality and its reliability while using.

RIA architecture provides a new Web page paradigm

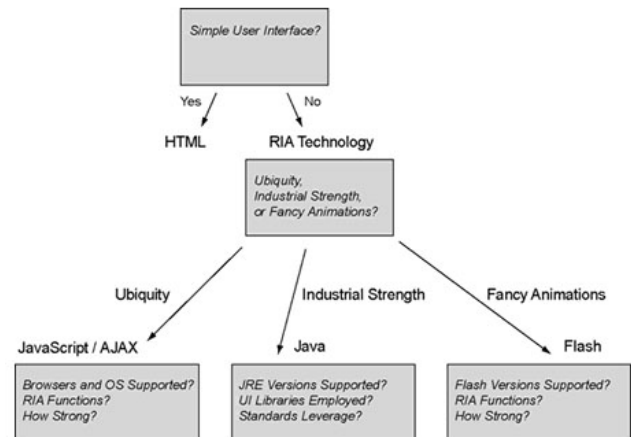
Traditional Web applications displays in a series of Web pages that needs a distinct download for each page, this is called web page paradigm. On the other hand RIA takes no longer time in downloading the page because the client engine may be prefetching some of the downloaded content for future use. New measurement techniques have been formulated RIA that reflects user's experience initiated by an HTTP GET request to permit reporting of response time. RIA developers must instrument their application code to produce the measurement data needed for SLM.

The current status of RIA development and adoption

At present RIAs are still in the early stages of development and user adoption that still have a number of restrictions and requirements remaining in it. These are:

- **Browser approval:** Many RIAs need modern web browsers for running that include Advanced JavaScript engines that uses the techniques like XML HTTP Request for client-server communication and DOM Scripting and advanced CSS techniques to enable the rich user interface.
- **Web standards:** Different versions create difficulties in writing RIA that cannot run in all platforms. After evaluation of Java 1.1, it becomes simpler to write in Java applets that run on all platforms.
- **Development tools:** To build RIA, some

essential products require including some Ajax Frameworks and products like Curl, Adobe Flex and Microsoft Silverlight to provide an integrated environment.



- **Accessibility apprehension:** Additional interactivity needs technical approaches that limit applications' accessibility.
- **User's acceptance:** Users expecting standard web applications may find that some unexpected browser functionality e.g. "Back" button.

Methods and techniques

JavaScript

It is the first major client side language technology that has the ability to run code and installed on several major of web clients. Earlier its uses were relatively limited but the development in DHTML makes possible to piece together an RIA system without using unified client-side solution. Ajax, the advance tool of Java Script becomes more prominent technique to develop RIA. Google is using this tool on the mass scale to develop its one of the most popular software Gmail and Google maps. Despite of this it is not so easy to create a large application in this framework. Several other different technologies have to include with efficiency. For making process easier several open source Ajax Frameworks have been developed along with commercial frameworks.

Rich Internet Application

Adobe Flash

This is another major tool to develop RIA as this technology is cross-platform and quite powerful to create an application user interface. Flash user interface can be compiled by MXML (a XML based interface description language), through using Adobe Flex tool. Adobe is currently working on providing a more powerful platform with the product Adobe AIR, a combo technology of HTMLs (including Ajax applications) Flash player based applications and PDFs.

OpenLaszlo

OpenLaszlo is an open source rich Internet application framework, which was developed by Laszlo Systems Inc.. The OpenLaszlo server



compiles programs from LZX language (a mixture of XML tags and JavaScript) into either DHTML (commonly known as AJAX now) or Adobe Flash bytecode. At present it is supporting Flash7 and Flash8. OpenLaszlo is the only rich Internet application platform, which can compile two different runtimes of same code base.

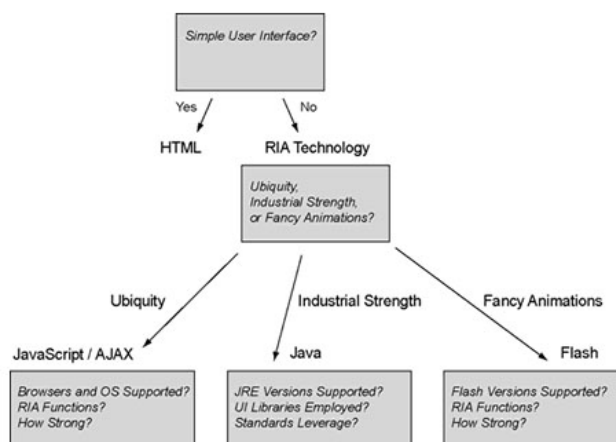
Windows Presentation Foundation and Silverlight

Microsoft has launched Windows Presentation Foundation (WPF) with .NET 3.0 Framework that provides a way to build single-platform

applications having some similarities to RIAs that uses XAML and programming languages like C# and Visual Basic. Besides this, Microsoft has announced to launch Silverlight that will provide a subset of WPF functionality on devices and other platforms.

ActiveX Controls

Inclusion of ActiveX controls into HTML is a very powerful mean to develop rich Internet applications, but they have no guarantee of running it on Internet Explorer properly. Many times they can break the sandbox model. This



makes it a suitable tool to target to manufacture computer viruses and malware.

Curl 5.0, Rebol 2.6 and Seaside for Smalltalk

There are three alternatives of Java and the JVM are available in the RIA developing world. These are: Curl, Rebol and Smalltalk. All these are abstract machines that are available in various forms. Curl facilitates Client-side persistent data while Rebol does not need a browser and Seaside uses a minor extension to Smalltalk to provide a much richer web experience for Smalltalk. All three alternatives are far more mature than more familiar options and are as old or older than Java and the JVM.

Rich Internet Application

Java applications

Java based Rich Internet Applications can be launched from within the browser or as free Standing applications through Java Web Start. Java RIAs can perform full-fledged functionality of Java include 2D & 3D graphics, and off-line capabilities, but at the cost of delayed startup. Some useful frameworks for Java RIAs are XUI, Swixml, or Canoo's thin Swing-approach, UltraLightClient.

User Interface languages

New user interface markup language can be used in RIAs as an alternative to HTML/XHTML. RIA's user interfaces may also be richer via using scriptable Scalable Vector Graphics (SVL) as well as SMIL Synchronized Multimedia Integration Language (SMIL). Though all browsers do not support native SVG rendering yet.

Java Jazz up
A BETTER WAY TO LEARN PROGRAMMING

Java News

JavaFX

JSF Tags

Web Services

Maven 2



<http://www.javajazzup.com>

Enjoy with
Java Jazz Up

Share your
opinions with Us

Java Jazz up
A BETTER WAY TO LEARN PROGRAMMING
<http://www.javajazzup.com/feedback.shtml>

Tips & Tricks

1- Create your own Notepad in Java

You must have worked with Notepad to write programs. Now its turn to create notepad by own with the help of java language. This section explains some basic functionalities of notepad which will help creating full fleshed notepad application. So just go through the example and see how it works.

First, a class extending JFrame and implementing ActionListener is created. JFrame is the main container for swing-based application. This program sets look and feel of the working platform by the statement

```
UIManager.setLookAndFeel(
    UIManager.getSystemLookAndFeelClassName());
```

When we add components to a JFrame we don't directly add them to the JFrame but we have to specify the pane of the JFrame's JRootPane In this example components are added to the contentPane which can be got by calling getContentPane(). The default behavior, when the user attempts to close the window, is to simply hide the JFrame. To change the default behavior, invoke setDefaultCloseOperation (JFrame.DISPOSE_ON_CLOSE) method which Disposes the frame automatically when it closes. Within the class, a menu bar, a menu and some of its menu items are created and added to the appropriate component. Action listeners are added to the menu items to listen the action when they are clicked and appropriate functions are implemented that functions according to the item clicked. For example, when user clicks on open menu item then listener listens the action and open() method is called that results in displaying the content of the file.

Notepad.class:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.datatransfer.*;
import java.io.*;
import javax.swing.text.*;

public class Notepad extends JFrame
implements ActionListener {
```

```
JMenuBar menuBar;
JMenu menu;
JMenuItem open,copy,cut,paste,save,quit;
JTextArea field;
FileFilterClass javaFileFilter = new
FileFilterClass ();
File file = new File ("file.java");
public Notepad1(String title){
    super(title);
    try {
        //Set Look and Feel of the current
platform.
        UIManager.setLookAndFeel
        l(UIManager.getSystemLookAndFeelClassName());
    }
    catch (Exception ex) {
        System.err.println("Error loading System
specific Look and Feel. " + ex);
    }
    Container contentpane = getContentPane
();
    contentpane.setLayout ( new
BorderLayout () );
    this.setSize(300, 300);
    // Dispose the frame automatically when it
closes.
    this.setDefaultCloseOperation
(JFrame.DISPOSE_ON_CLOSE);
    field = new JTextArea();
    field.setDragEnabled(true);
    contentpane.add ( field, "Center");
    //Create Menu Bar.
    menuBar = new JMenuBar();
    setJMenuBar(menuBar);
    //Create File menu.
    menu = new JMenu("File");
    //Activate the keyboard shortcut for File
menu.
    menu.setMnemonic(KeyEvent.VK_F);//It
underlines the character 'F' passed into the
setMnemonic( ) method.
    //Create menu items.
    open = new JMenuItem("Open");
    copy = new JMenuItem("Copy");
    cut = new JMenuItem("Cut");
    paste = new JMenuItem("Paste");
    save = new JMenuItem("Save");
    quit = new JMenuItem("Quit");
    //Add items to the menu.
    menu.add(open);
    menu.add(copy);
```


Tips & Tricks

```
menu.add(cut);
menu.add(paste);
menu.add(save);
menu.add(quit);
menuBar.add(menu);
//Add listeners to the items.
open.addActionListener(this);
copy.addActionListener(this);
cut.addActionListener(this);
paste.addActionListener(this);
save.addActionListener(this);
quit.addActionListener(this);
}

public static void main(String[] args){
    new Notepad("Notepad").setVisible(true);
}

public void actionPerformed(ActionEvent ae)
{
    String cmd =
    (String)ae.getActionCommand();
    if (cmd.equals("Open")) open();
    else if (cmd.equals("Copy")) copy();
    else if (cmd.equals("Cut")) cut();
    else if (cmd.equals("Paste")) paste();
    else if (cmd.equals("Save")) save();
    else if (cmd.equals("Quit")) quit();
}

public void open() {
    JFileChooser fileChooser = new
JFileChooser();
    fileChooser.setFileFilter
(javaFileFilter);
    int returnVal =
    fileChooser.showOpenDialog(this);
    if(returnVal ==
    JFileChooser.APPROVE_OPTION){
        file = fileChooser.getSelectedFile();
        String fileContent = readFile(file);
        field.setText(fileContent);
    }
}

public String readFile (File file) {
    StringBuffer strBuffer;
    String fileContent=null;
    String lineString;
    try {
        FileReader fr = new FileReader(file);
        BufferedReader br = new
```

```
BufferedReader(fr);
strBuffer = new StringBuffer() ;
while ((lineString = br.readLine ()) != null) {
    strBuffer.append (lineString + "\n");
}
fr.close();
fileContent = strBuffer.toString();
String name = file.getName();
if(name != null) {
int extensionIndex = name.lastIndexOf('.');

setTitle(name.substring(0,extensionIndex));
}
}
catch (IOException e ) {
    return null;
}
return fileContent;
}

public void copy() {
    String s = field.getSelectedText();
    int start=field.getSelectionStart();
    int end=field.getSelectionEnd();
    StringSelection ss = new
StringSelection(s);
    // Set the content to the clipboard.
    this.getToolkit().getSystemClipboard().
    setContents(ss, ss);
}

public void cut() {
    String s = field.getSelectedText();
    int start=field.getSelectionStart();
    int end=field.getSelectionEnd();
    field.replaceRange("",start,end);
    StringSelection ss = new
StringSelection(s);
// Set the content to the clipboard.

this.getToolkit().getSystemClipboard().setContents(ss,
ss);
}

public void paste() {
    Clipboard cb =
this.getToolkit().getSystemClipboard();
    Transferable tr = cb.getContents(this);
    try {
// Get the content from the clipboard.
        String s = (String)
```

Tips & Tricks

```
tr.getTransferData(DataFlavor.stringFlavor);
    int start=field.getSelectionStart();
    int end=field.getSelectionEnd();
    field.replaceRange(s,start,end);
}
catch (Exception e) {
    return;
}
}

public boolean save() {
JFileChooser fileChooser = new
JFileChooser();
fileChooser.setFileFilter (javaFileFilter);
int returnVal =
fileChooser.showSaveDialog(this);
if(returnVal ==
JFileChooser.APPROVE_OPTION){
file = fileChooser.getSelectedFile ();
if (file.exists()) {
int response =
JOptionPane.showConfirmDialog (null,"File
already exists. Do you want to
continue?","Overwrite
Confirmation",JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE);
    if (response ==
JOptionPane.CANCEL_OPTION) return false;
    }
    String fileName = file.getName();
    if(fileName != null) {
        int extensionIndex =
fileName.lastIndexOf('.');

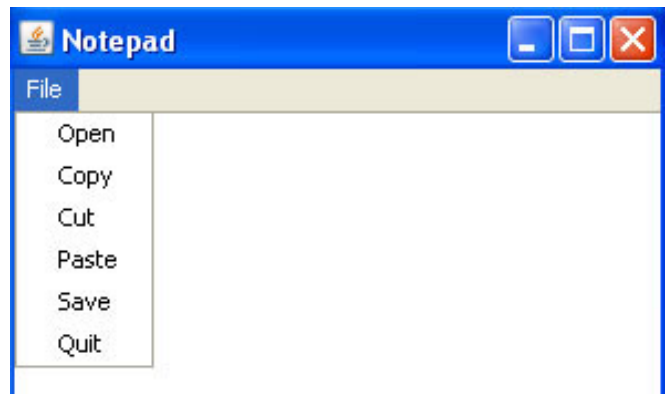
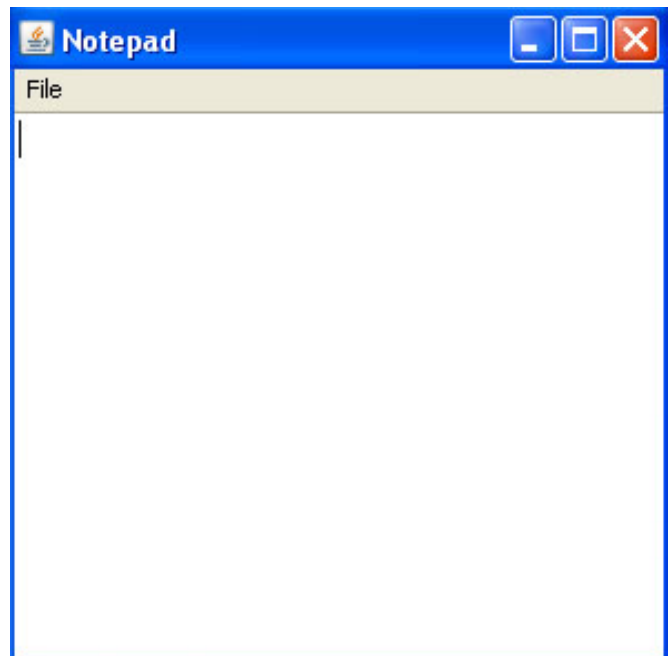
setTitle(fileName.substring(0,extensionIndex));
    }
    return writeFile (file, field.getText());
}
return false;
}

public static boolean writeFile (File file, String
content) {
    try {
        PrintWriter pw = new PrintWriter (new
BufferedWriter (new FileWriter (file)));
        pw.print (content);
        pw.flush ();
        pw.close ();
    }
    catch (IOException e) {
        return false;
    }
}
```

```
return true;
}

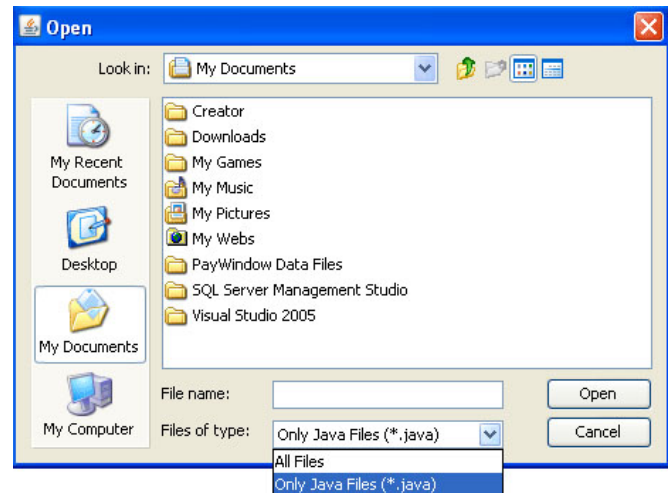
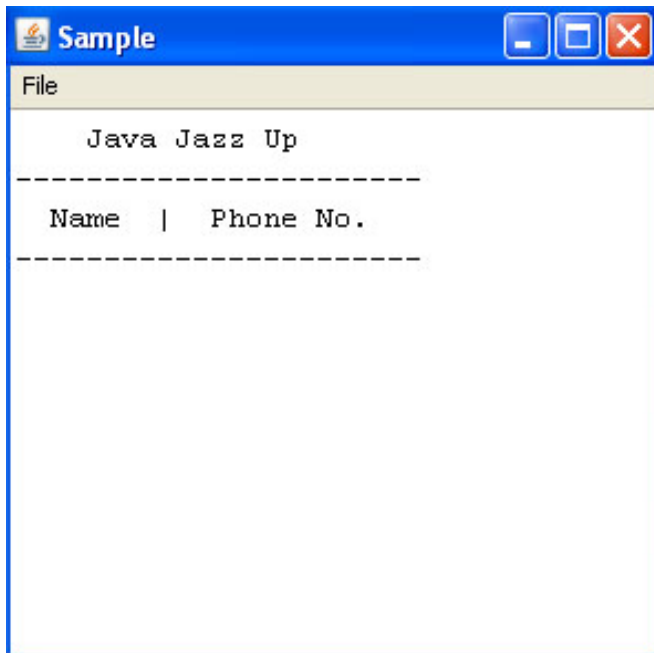
public void quit() {
    System.exit(0);
}
}
```

The result of the above class can be seen below:



Clicking the open menu item results in opening the selected file and file name appears at the top showing the current file opened.

Tips & Tricks



Now you can check rest of the functionalities like copy, cut, paste, save and quit work properly and also add more new functions to make it more modern.

FileFilterClass

This class is responsible to filter files. When file chooser opens a dialog box to open or save the file, file filter gives an option to choose only filtered files. In this example, **FileFilterClass** class has been created to filter java files.

```
import java.io.*;
import javax.swing.filechooser.FileFilter;

//Class responsible to filter java files.
public class FileFilterClass extends FileFilter {
    public boolean accept (File file) {
        return file.getName ().toLowerCase
().endsWith (".java")|| file.isDirectory ();
    }
    public String getDescription () {
        return "Only Java Files (*.java)";
    }
}
```

The result of creating this class can be seen in the figure below. When you click open menu, an open dialog is opened like below:

2- Create PDF file with Java Program

Sometimes, you may need creating a PDF (Portable Document Format). iText is a java library that contains classes to generate documents in PDF, XML, HTML, and RTF. For this you need to place iText.jar file to lib folder of your JDK and set classpath for it. You can download this jar file from the link <http://www.lowagie.com/iText/download.html> The program below will create PDF for you. Just have a glance over the program below:

```
import java.io.*;
import com.lowagie.text.*;
import com.lowagie.text.pdf.*;
import com.lowagie.text.Font;
import com.lowagie.text.FontFactory;

public class JavaPDF{
    public static void main(String arg[])throws
Exception{
        //Create a document-object
        Document document=new Document();
        String p2 = "JavaJazzUp is a new wing of
Roseindia.net that is initiated to provide the
full-fledged and in-depth information to a wide
range of Java lovers. A magazine has a
characteristics of having a vast segments
related to a wide range of public choice and
must contain at least a portion of every
categories choice. So RoseIndia.net has decided
```

Tips & Tricks

to provide a wide range of Java news, information and invention with latest update and technology to Java lovers. This is a small effort of Rose India organization to provide the latest happening in the Java to its online users every month.”;

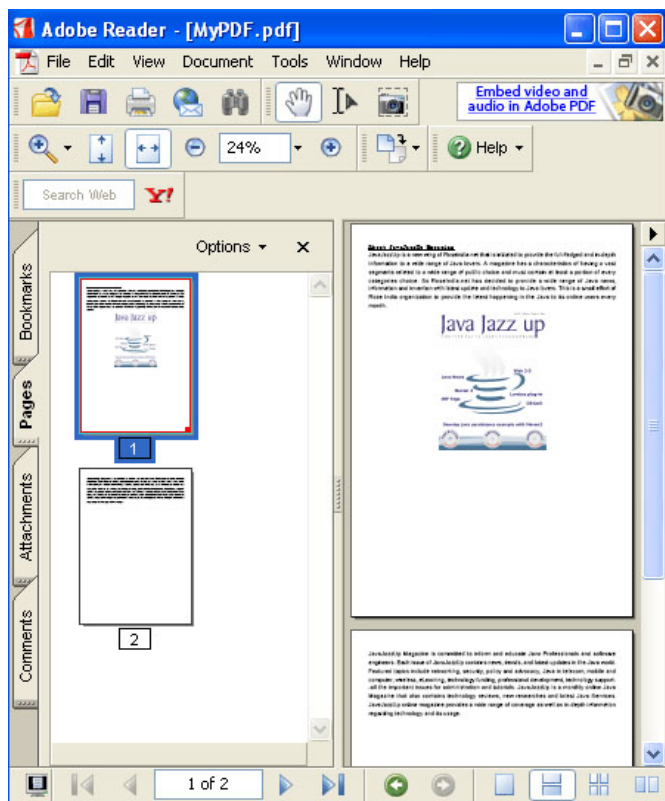
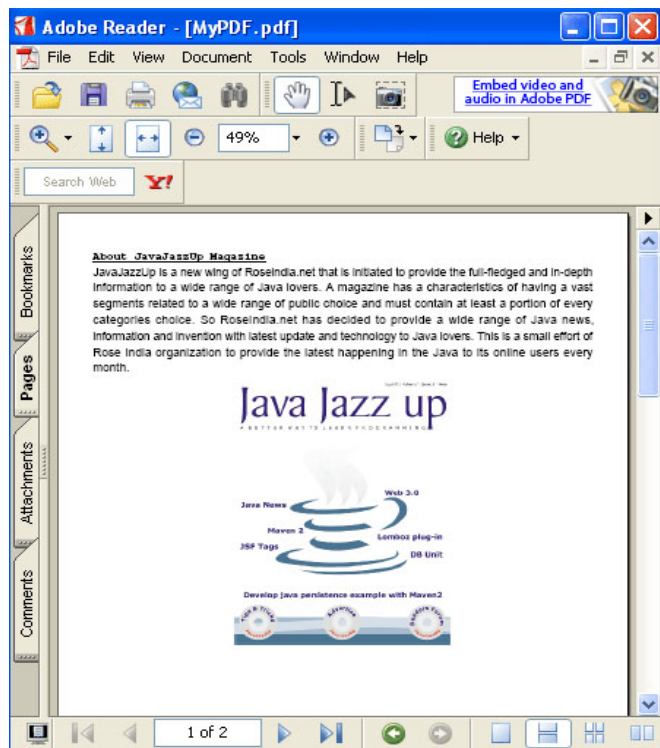
String p4 = “JavaJazzUp Magazine is committed to inform and educate Java Professionals and software engineers. Each issue of JavaJazzUp contains news, trends, and latest updates in the Java world. Featured topics include networking, security, policy and advocacy, Java in telecom, mobile and computer, wireless, eLearning, technology funding, professional development, technology support—all the important issues for administration and tutorials. JavaJazzUp is a monthly online Java Magazine that also contains technology reviews, new researches and latest Java Services. JavaJazzUp online magazine provides a wide range of coverage as well as in-depth information regarding technology and its usage.”;

```
try {
    Chapter chapter1,chapter2;
    Paragraph
    paragraph1,paragraph2,paragraph3,
    paragraph4,paragraph5;
    Chunk chunk;
    Font font;
    Image image;
// Create a PDF document writer.
    PdfWriter.getInstance(document,new
    FileOutputStream("MyPDF.pdf"));
//Open the document
    document.open();
    paragraph1 = new Paragraph();
    font = new Font(Font.COURIER,
    Font.DEFAULTSIZE, Font.BOLD);
// Set the font for the chunk.
    chunk = new Chunk("About JavaJazzUp
    Magazine", font);
// Set underline for the content of
    chunk.
    chunk.setUnderline (0.2f, -2f);
    paragraph1.add(chunk);
    paragraph2=new Paragraph(p2);
// Set the alignment for the content of
    the paragraph.
```

```
    paragraph2.setAlignment
    (Element.ALIGN_JUSTIFIED);
    paragraph3=new Paragraph();
    image = Image.getInstance
    ("coversept.gif");
// Set alignment for the image to be
    pasted.
    image.setAlignment(Image.MIDDLE);
// Set the size of the image.
    image.scalePercent(75);
// Add image to the paragraph.
    paragraph3.add(image);
    paragraph4=new Paragraph(p4);
    paragraph4.setAlignment
    (Element.ALIGN_JUSTIFIED);
// Create chapters/pages of the pdf
    file.
    chapter1=new Chapter(paragraph1,1);
    chapter2=new Chapter(paragraph4,2);
// Hide page number from chapters.
    chapter1.setNumberDepth(0);
    chapter2.setNumberDepth(0);
    chapter1.add(paragraph2);
    chapter1.add(paragraph3);
    document.add(chapter1);
    document.add(chapter2);
}
catch(DocumentException de) {
    System.err.println(de.getMessage());
}
catch(IOException ioe) {
    System.err.println(ioe.getMessage());
}
//Close the document
document.close();
}
```

The above program creates “**MyPDF.pdf**” file as in the figure below:

Tips & Tricks



3- Count number of active users through servlet

A session is just like a temporary unique connection between the client (browser) and server which helps the server to keep track of the specific user. Session creation and destruction events can help to count the number of active session. We can create listener object that will be called every time a session is created or destroyed by the server.

In this section, SessionCounter is a listener class which will be registered in the web.xml file. Create SessionCounter class and implement HttpSessionListener interface. Implement its two methods `sessionCreated()` and `sessionDestroyed()` passing `HttpSessionEvent` as an argument. In the first method `sessionCreated()`, increase the value of the variable responsible for counting the number of active sessions because one new session has been created and in the second method `sessionDestroyed()` decrease the value of the same by one because one session has been ended. Save this file to the WEB-INF/classes folder of Tomcat server.

SessionCounter.java

```
import
javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;

public class SessionCounter implements
HttpSessionListener {

    private static int activeSessions = 0;
    private static int destroyedSessions = 0;
    private static String aSID = null;
    private static String iaSID = null;

    public void sessionCreated(HttpSessionEvent
se) {
        aSID = se.getSession().getId();
        activeSessions++;
    }

    public void
sessionDestroyed(HttpSessionEvent se) {
        iaSID = se.getSession().getId();
```

Tips & Tricks

```
if(activeSessions > 0){
    activeSessions--;
    destroyedSessions++;
}

public static int getActiveSessions() {
    return activeSessions;
}
public static int getInactiveSessions() {
    return destroyedSessions;
}
public static String getActiveSessionID() {
    return aSID;
}
public static String getInactiveSessionID() {
    return iaSID;
}
}
```

Now create a servlet that will use above listener's variables to show the current status of session.

ServletSessionCounter.java

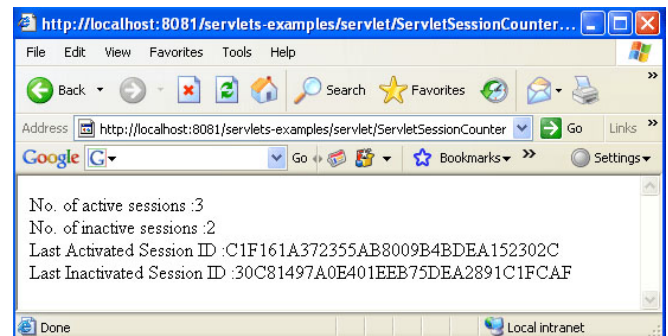
```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class ServletSessionCounter extends
HttpServlet{
    public void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException{
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter();
        HttpSession session=
        request.getSession();
        pw.println("No. of active sessions      :\" +
        SessionCounter.getActiveSessions()+"<br>");
        pw.println("No. of inactive sessions   :\" +
        SessionCounter.getInactiveSessions()+"<br>");
        pw.println("Last Activated Session ID   :\"
+
        SessionCounter.getActiveSessionID()+"<br>");
        pw.println("Last Inactivated Session ID :\"
+
        SessionCounter.getInactiveSessionID()+"<br>");
        session.setMaxInactiveInterval(2);
    }
}
```

Now register the listener in WEB-INF/web.xml file which makes aware the server about the SessionListener class that results in calling its event methods whenever a session is created and destroyed. Just add the following lines to the web.xml file.

```
web.xml:
.....
.....
<listener>
    <listener-class>SessionCounter</
listener-class>
</listener>
.....
.....
```

Now restart the server and call the servlet many times in different browser windows. The result will be shown like below:



4- Copy a file or directory to the specified location

This program explains copying a file or directory and all its sub-directories and files to any location.

```
import java.io.*;
public class CopyDirectory
{
    public static void main(String[] args) throws
    IOException
    {
        CopyDirectory cd = new CopyDirectory();
        BufferedReader in = new
        BufferedReader(new
        InputStreamReader(System.in));
        System.out.println("Enter the source
```

Tips & Tricks

```
directory or file name : ");
    String source = in.readLine();
    File src = new File(source);
    System.out.println("Enter the destination
directory or file name : ");
    String destination = in.readLine();
    File dst = new File(destination);
    cd.copyDirectory(src, dst);
    System.out.println("File or Directory
copied.");
}

public void copyDirectory(File srcPath, File
dstPath) throws IOException
{
    if (srcPath.isDirectory())
    {
        if (!dstPath.exists())
        {
            dstPath.mkdirs();
        }

        String files[] = srcPath.list();
        for (int i = 0; i < files.length; i++)
        {
            copyDirectory(new File(srcPath, files[i]),
new File(dstPath, files[i]));
        }
    }
    else
    {
        if (!srcPath.exists())
        {
            System.out.println("File or directory
does not exist.");
            System.exit(0);
        }
        else
        {
            InputStream in = new
FileInputStream(srcPath);
            OutputStream out = new
FileOutputStream(dstPath);

            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) > 0)
            {
                out.write(buf, 0, len);
            }
            in.close();
        }
    }
}
```

```
        out.close();
    }
}
}
```

Output:

If the user enter valid path of the file or directory then it shows a message that it has been copied to the proper destination.

```
C:\Program Files\Java\jdk1.6.0\bin>javac
CopyDirectory.java
C:\Program Files\Java\jdk1.6.0\bin>java
CopyDirectory
Enter the source directory or file name :
C:\JavaJazzUp
Enter the destination directory or file name :
D:\RoseIndia\JavaJazzUp
File or Directory copied.
C:\Program Files\Java\jdk1.6.0\bin>
```

When the user enters wrong source file or directory name to be copied that does not exist then it shows a message that the source file or directory does not exist.

```
C:\Program Files\Java\jdk1.6.0\bin>java
CopyDirectory
Enter the source directory or file name :
C:\JavaJazzUpNew
Enter the destination directory or file name :
D:\JavaJazzUp
File or directory does not exist.

C:\Program Files\Java\jdk1.6.0\bin>
```

Advertise with JavaJazzUp

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

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

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

JavaJazzUp Network comprises of :

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

Advertisement Options:

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

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

Deepak Kumar
deepak@roseindia.net

Coffee with JavaFX

Enjoy A New Flavour



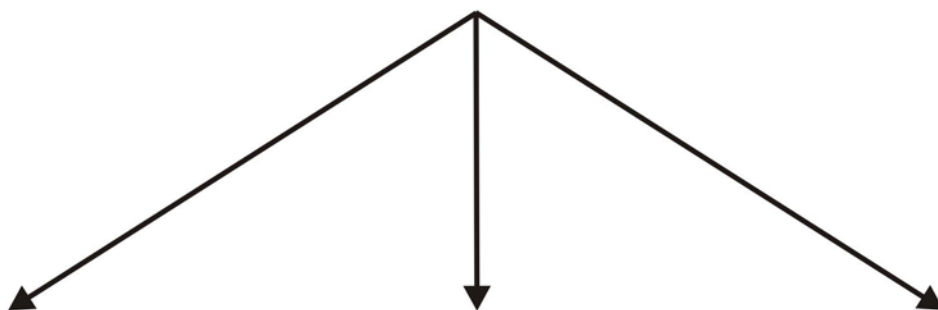
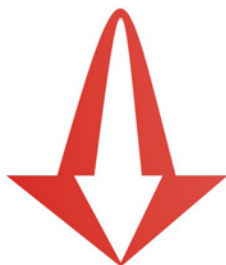
Visit us
<http://www.roseindia.net>

All Cool Jobs

Your Beginning Career



It's the best way to set **smiles** to your Growing career.
logon to <http://www.allcooljobs.com>



Post Resume

A Global Path

**Search
Latest Jobs**

Original Perception



CBI's failure to act against influential

For some time judiciary has pronounced such welcoming judgements which have been instrumental in enhancing the faith of people in the criminal judicial system of the country. But there is one agency which has often failed to deliver its honest services on this front. The central investigating agency CBI has been chided by the court in many cases for not dealing with the case seriously especially when the accused ones are from the influential background mainly from political and administrative.