



Java™ 6 Update 5 Consumer Platform Release

Simon Ritter
Technology Evangelist
Sun Microsystems, Inc.



How Much Java Technology Is Out There?

- >91% of all PCs run Java platform*
- ~77% of all Java technology-enabled PCs run Sun's Java Platform, Standard Edition (Java SE platform)**
- Distribution through PC OEMs
 - > Nine of the top ten PC OEMs ship the Sun JRE software
 - > Representing >60% of all shipped PCs
 - > 58 white box vendors have signed JRE software redistribution agreements
- Download/installs
 - > ~44m installations / month for the last six months on Windows
 - > >50M in Jan, Feb, April, 2007

* Omniture, April 2007

**Mapsolute/Map24.com, April 2007

Three Deployment Routes

There are three major ways a Java platform program can be distributed today

- Applets (Java Plug-In software)
- JNLP (Java Web Start software)
- Standalone Programs (Custom Installers)

All three paths share similar challenges

Java Network Launch Protocol (JNLP)

Deployment Challenges

- What versions of Java Technology are installed?
- How do I launch Java Technology?
- What's the best way to install a new version?

When you consider the number of platform and browser combinations Java technology supports, none of these questions have simple answers

Present-Day Solutions

- Possible to detect JRE version 1.4.2+ using JavaScript™ Technology
- Can't get exact versions
- Auto-install (Windows with Internet Explorer only) is somewhat complex
- Sun provides sample scripts only—no complete, working code
- Outside of the browser, you're on your own; The only reliable way to detect Java technology is to dig through the registry
- Installation could be smoother...

Next-Generation Installer

We will soon introduce a new look and feel for the Windows installer

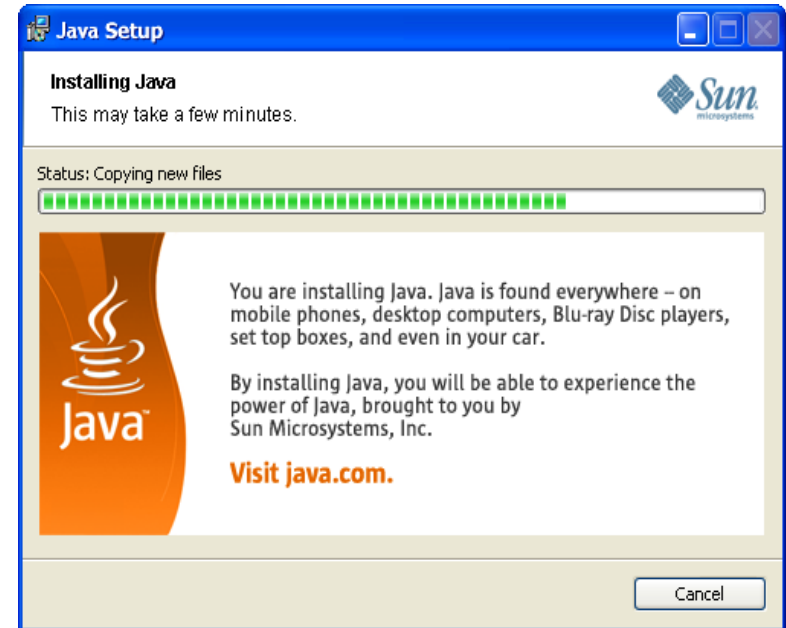
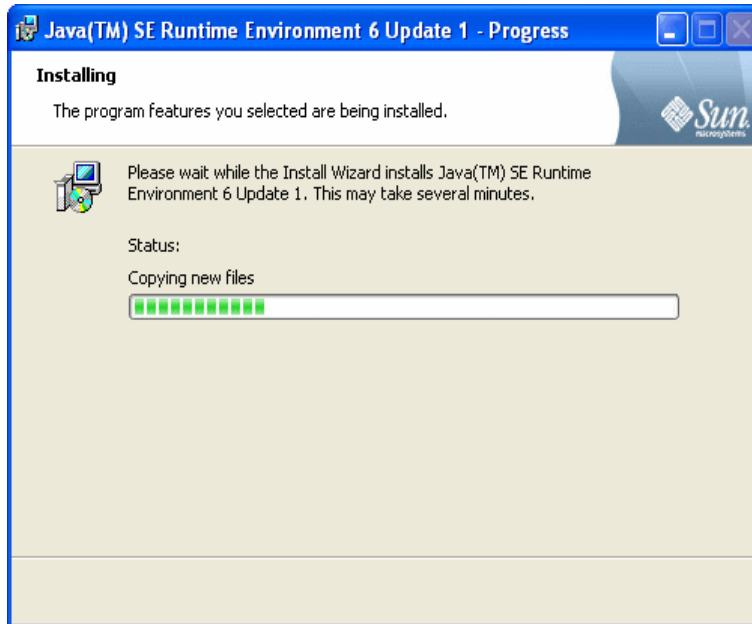
- More streamlined
- Less intimidating
- Improved messaging
- More visually appealing

New Installer Look and Feel

Progress

Before

After



Other Installer Improvements

- Patch-in-Place—no more duplicate JRE versions
- Reduced update sizes
- Faster installations and updates

Java SE, the Consumer Web Client

- Java's traditional sweet spots
 - > The first wave of Java 1.1 consumer applets
 - > Enterprise applications and applets
 - > Developer tools (Netbeans, Eclipse, IDEA, ...)
 - > Large consumer desktop applications (LimeWire, Azureus, ...)
- The world has changed
 - > Java SE runtime is now installed on most desktops
 - > Rich consumer clients and media are rapidly gaining importance
- The time is right for Java SE to become a primary consumer web client platform

Java SE, the Consumer Web Client

- JDK6 update for consumers
 - > Based on JDK6
 - > Planned 1st half 2008
 - > Eliminates the most important consumer issues
 - > Startup time, Download size, Install experience, Modern cross platform L&F
- JDK7
 - > Will have a new focus on consumer relevant features
 - > Media, Graphics, Animation, 3D support

Problem: JRE Software Detection

- No good way to detect JRE software existence and version from browser
- Developers use nifty “Get Java” button
 - > Which takes users away from the site
- Applets constrained to lowest-common-denominator APIs
 - > MS VM, circa Java Development Kit (JDK™) 1.1

Solution: Deployment Toolkit

- Deployment Toolkit JavaScript™ Technology hosted by Sun at <http://java.com/js/deployJava.js>
- Developer uses simple script on their site
 - > Link to Sun's script
 - > Simplified applet or JNLP Application launching
- Detects JRE software existence and version
- Depending on app requirements
 - > Redirects to download site
 - > Polls for successful install
 - > Redirects back to original site and launches app or applet

Solution: Deployment Toolkit

- ActiveX Control and NPAPI Plugin included in JRE
- Script will use the plugins if available
 - > Fine grained JRE detection
 - > Pure javascript can only detect at the family granularity.
 - > Allows application or applet to check that a specific version, a specific family, or a specific minimum version is installed.
 - > Install any available JRE version
 - > Pure javascript will only install latest JRE version
 - > Select installation type (kernel or online installer)
 - > Declare application or applet required packages
- Plugins remain installed after JRE uninstallation

JavaScript Technology Limitations

The script by itself...

- Can only detect highest installed JRE version
- Detects “family version” only (1.5 vs. 1.5.0_11)
- JRE software installation isn’t seamless
- User may be asked whether to open or save a JNLP file

The Solution: Browser Plug-Ins

- Upcoming JRE software will include browser plug-ins that perform these functions using native code
- The JavaScript Technology API will check for the presence of these plug-ins and, if found, delegate to them
- Either way, you call the same JavaScript technology functions; They just work better/more accurately if the user has the right JRE version installed

Plug-In Advantages

	JavaScript Technology Only	With Plug-In
<code>getJREs ()</code>	"1.6"	"1.6.0_00", "1.5.0_10", "1.5.0_06"
<code>installLatestJRE ()</code>	Redirect to java.com download page	Immediate installation with progress bar
Click on JNLP launch button	Sometimes "Would you like to open or save this file?"	Launches without prompting

GetJava.exe

Standalone detection and installation services

> `getjava -version 1.6+`

Detects whether a JRE version 1.6 or higher is installed, and if not installs the most recent version

> `getjava -detect`

Detects the highest installed JRE version, writing the results to stdout; A zero exit code indicates success

GetJava.exe

```
> getjava -detect -version  
1.5.0_11
```

Checks for the presence of a specific JRE version

Other arguments allow you to specify the path or URL to the desired JRE software installer, control HTTP proxies, and otherwise fine-tune the process

GetJava.exe Usage

- GetJava.exe can be used as part of a native installer; Simply embed in it in your installer and delegate to it for detection and installation

...no more manually digging through the registry!

Problem: Too Many Java Versions

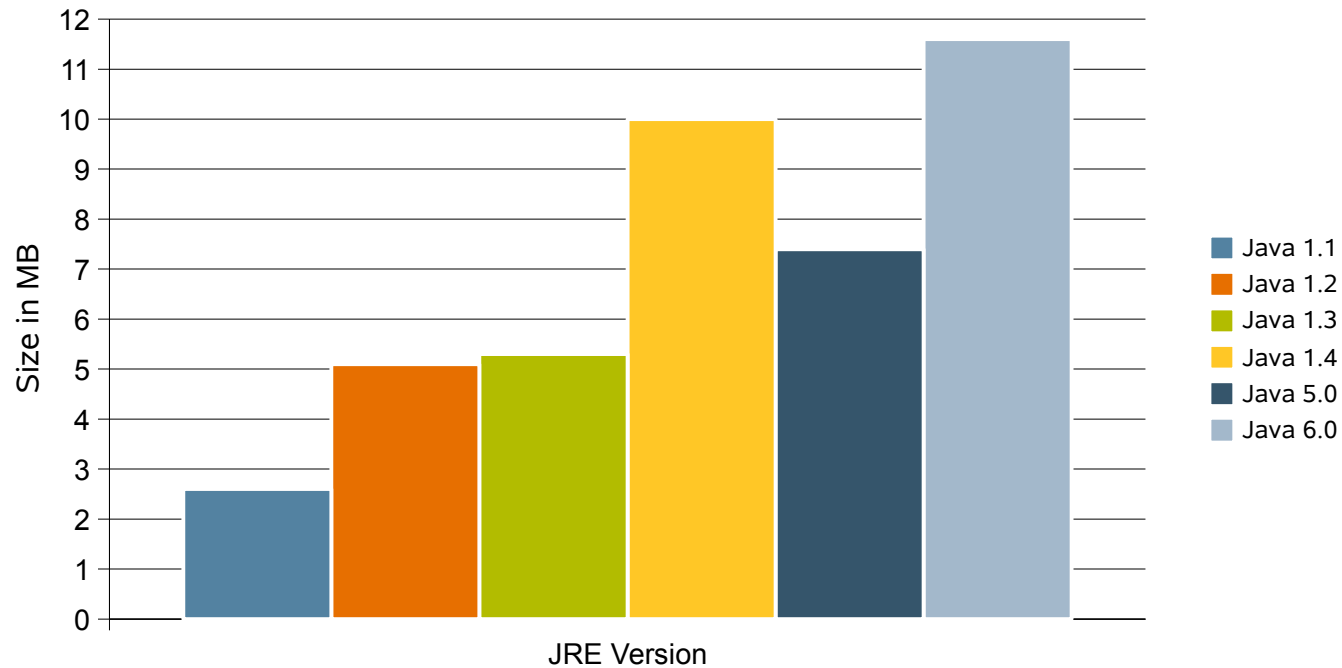
- Every Java Update version is installed as a separate program.
- Java Auto-Update cause many version to be installed.
- Add / Remove Programs is filled with many versions of the JRE.

Solution: Patch in Place

- Patch in Place allows updating an existing Java version to a later update of the same family.
- Static installation will be available for enterprises that need to rely on static versioning.
- Add / Remove Programs will only see one non-static Java Runtime installation in each family.
 - > (Existing installations are all Static.)
 - > New versions explicitly installed statically will also be shown.

JRE Release Size

No end in sight



Size Impact

- The JRE software contains an enormous number of APIs
- While they are all useful and important, few programs use even half
- Small programs in particular are heavily affected by this

Auto Update Download Size

- Initial install of Java for any major release (family) includes a “base image” .
 - > Base images make initial install take longer to install
- Java Auto Update downloads patches from this base image.
- Over time, the patches become very large, since they contain all the changes since the first release of the family.

Solution: Modularization

- Modularize the JRE software
- The core part of the JRE software (the kernel) contains just enough functionality to run “Hello World”
- The remaining JRE software components will be downloaded on demand or lazily downloaded

Solution: Incremental Update

- Download only the incremental changes from the version that the client machine actually has.
 - > Separate patches depending on existing install.
 - > For example, for 6 update 8, this would include : 6u5 -> 6u8, 6u6 -> 6u8, and 6u7 -> 6u8.
- No longer need to copy and install base images during initial install of the first version of a family.
 - > Faster installation of initial version
- Occupy less disk space by not retaining base images.

Problem: Download and Install Time

- The JRE implementation is, well, big
 - > 7-15 MB download
 - > Extracted to 40+ MB for rt.jar
 - > Plus other jarfiles, native libraries, resource files, ...
- All of this takes time
 - > Download
 - > Unzip
 - > Unpack200
 - > Copying bits around
- It's a wonder that it's as quick as it is...

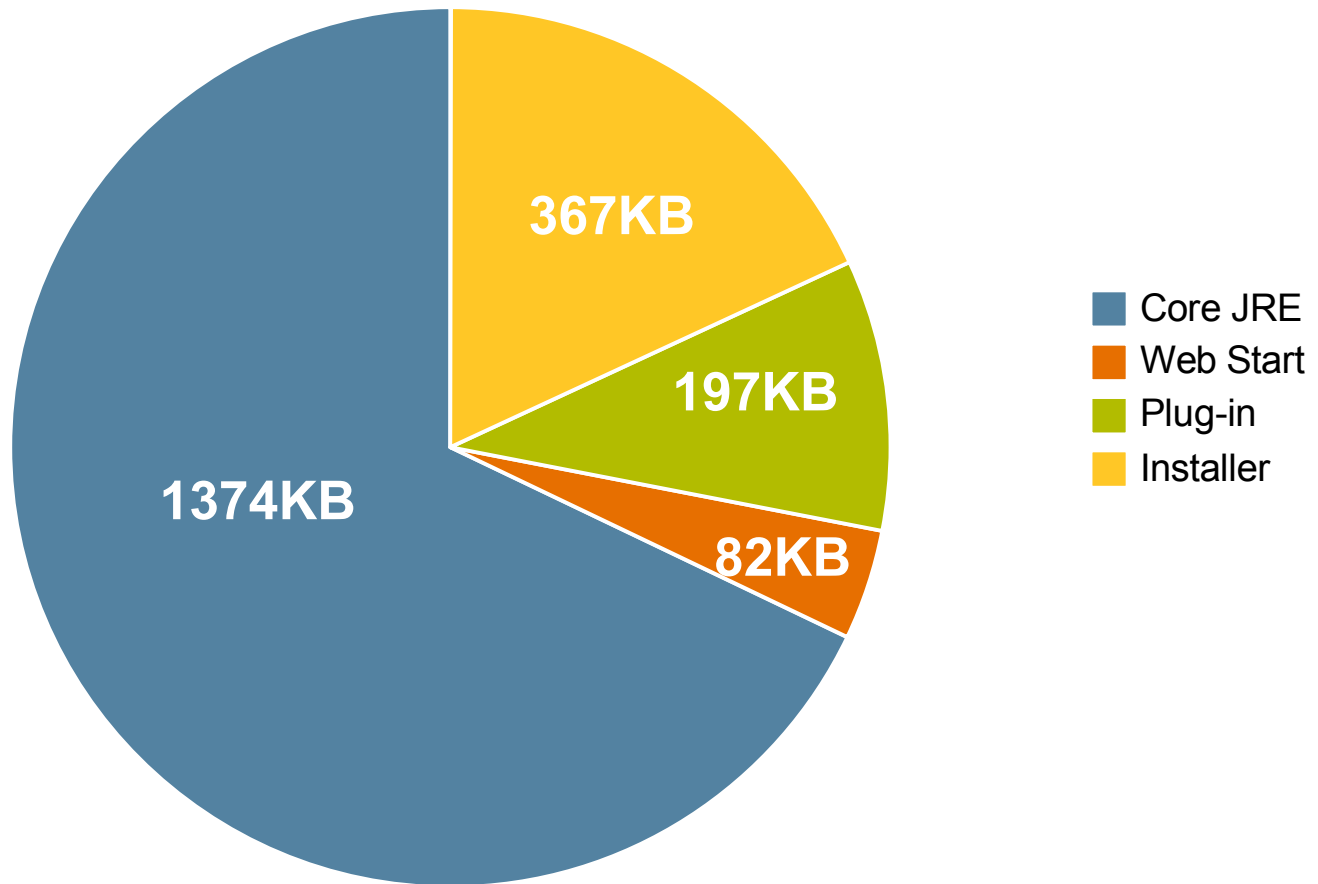
Solution: Java Kernel

- Every app needs some core functionality
 - > VM, networking, security, classloader
- ... plus other stuff on demand
 - > Swing, AWT, 2D, ~~CORBA~~
- Kernel downloads and installs:
 - > Bare essentials immediately
 - > Additional dependencies on demand
 - > Referencing a class
 - > Class.getResource() or equivalent
 - > System.loadLibrary() or equivalent
 - > Everything else in the background

Kernel Structure

- The Kernel JRE software is quite similar to today's JRE software
- The primary differences are that `rt.jar` is much smaller and many files are not initially present
- The missing classes and files are grouped into components, with names like "javax_swing" and "java_net"
- The component breakdown is largely along package boundaries

Kernel: Bare Essentials



On-Demand Downloading

Automatically downloading missing features

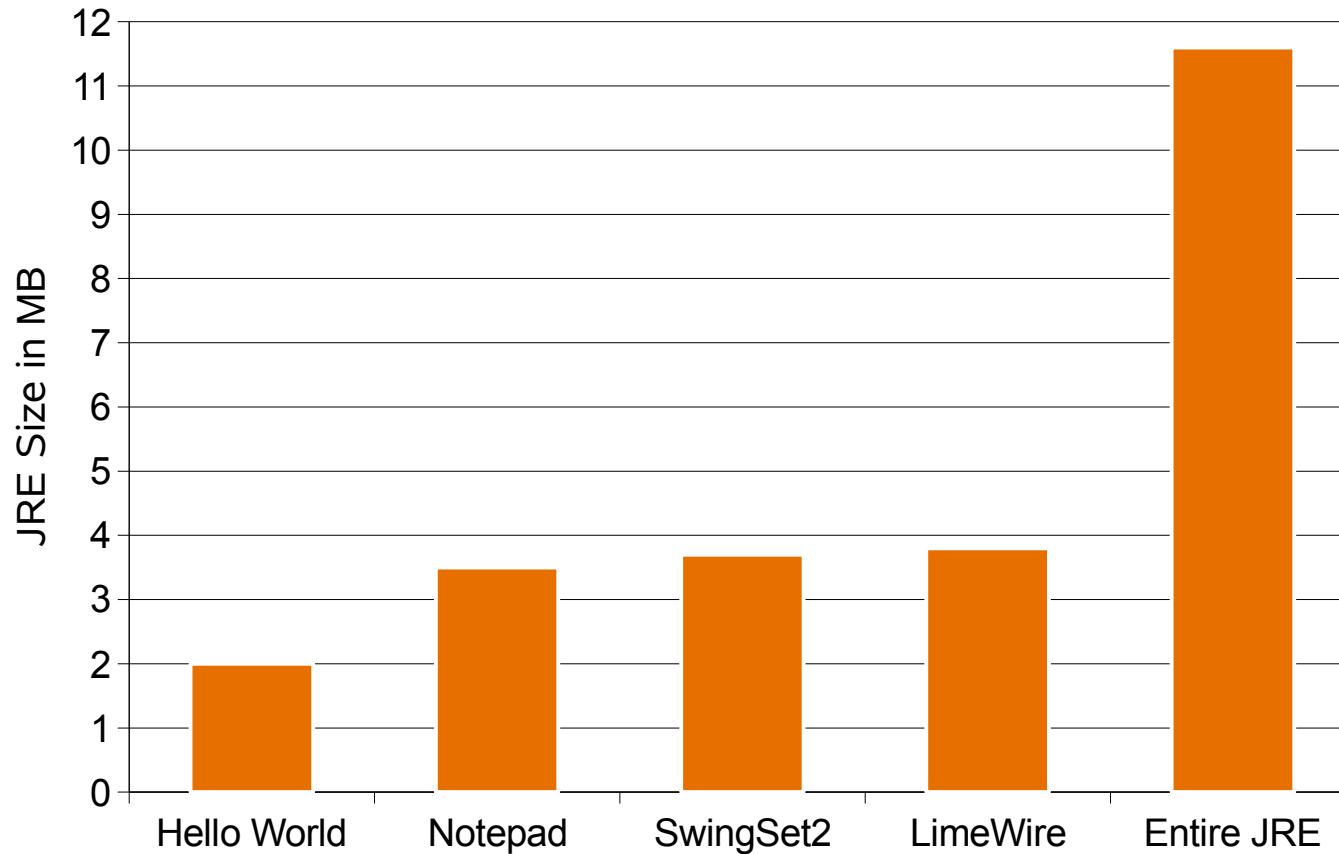
There are three ways to trigger a download:

- Referencing a class
- `Class.getResource()` or equivalent
- `System.loadLibrary()` or equivalent

Eliminating Pauses

- While the Java platform is downloading additional components, the user is forced to watch a progress dialog
- There **will** be a mechanism for developers to avoid these pauses, by ensuring that all required components are present before the program starts
- Details are still under wraps...

Estimated Download Sizes



Problem: Startup Time

- There are two different aspects to the Java platform startup time: “cold start” and “warm start”
- Cold start means that the JRE software is not currently in memory and must be read from disk
- Warm start means that the JRE software is already in memory and just needs to initialize itself
- Cold start is (basically) disk read time + warm start
 - > Typical times: 5-10+ seconds

Cold Start vs. Warm Start

- A lot of work has gone into both of these metrics over the last eleven years
- On a typical modern desktop machine, warm starts are now nearly instant; There isn't much left to improve
- Cold start continues to be an issue; No matter how much we optimize things, hard drives are still slow
- Warmstart is acceptable
 - > Lots of work over the years + faster machines
- Coldstart is **not**

Problem: Startup Time

- Coldstart is an OS issue
 - > All about the disk cache
 - > Java platform reads a **lot** from disk at startup
- Some improvements over the years
 - > Class Data Sharing
 - > Rearranging rt.jar
 - > OS-level prefetch (native libraries only)
- But it's still not nearly enough
- What to do?

Solution: Java Quickstarter

- Pre-load the disk cache, **before** launch
 - > At boot, at browser launch, whenever
- **Note: Not** the same as having a running VM
- Cooperates with the OS
 - > Memory still available for other apps
 - > OS will flush disk cache pages as necessary

Java QuickStart Service

- It's not a universal fix; On memory-constrained computers Java QuickStart Service will automatically disable itself to avoid thrashing the system
- But, in most cases, the Java platform startup time issue is effectively solved

Coming Soon: Consumer Platform Release JDK 6 Update 5 Features

- Features
 - > Deployment Toolkit
 - > Installer Enhancements
 - > Java Kernel
 - > Java Quick Starter
 - > Windows graphics acceleration
 - > Nimbus look & feel
- Caveat: This is the plan. Plans can change.
 - > Initial builds with all of these features exist now
 - > Early access builds available soon on java.sun.com

JDK 6 Update 5 Features

- JRE Software Detection
- Patch in Place, Java Installer Enhancements
- Java Kernel
- Java Quick Starter
- Windows Graphic Acceleration
- Nimbus Look and Feel

Summary

- A new installer will deliver a simpler, cleaner Java platform installation experience
- Detecting and installing the JRE software will be much simpler
- Standalone Java applications will finally receive equal treatment
- The “cold start” delay on initial Java platform startup has been mostly eliminated

Summary

- The JRE software is being modularized, so that users need only download the necessary components
- A complex Swing program can be delivered in under 4MB

Coming soon to a JRE version near you!



Java™ 6 Update 5 Consumer Platform Release

Simon Ritter
Technology Evangelist
Sun Microsystems, Inc.

