Java Web Start

Technology for Rich Client Graphical User Interfaces

Java Web Start & JNLP

- Enables highly interactive Swing-based graphical user interfaces.
  - Swing is Java's collection of graphical user interface widgets.

- Uses *Java Network Launching Protocol* (JNLP)
  - 'Java Web Start' is Sun Microsystems' implementation of JNLP.
  - The term 'Java Web Start' is often loosely used to include all JNLP implementations.

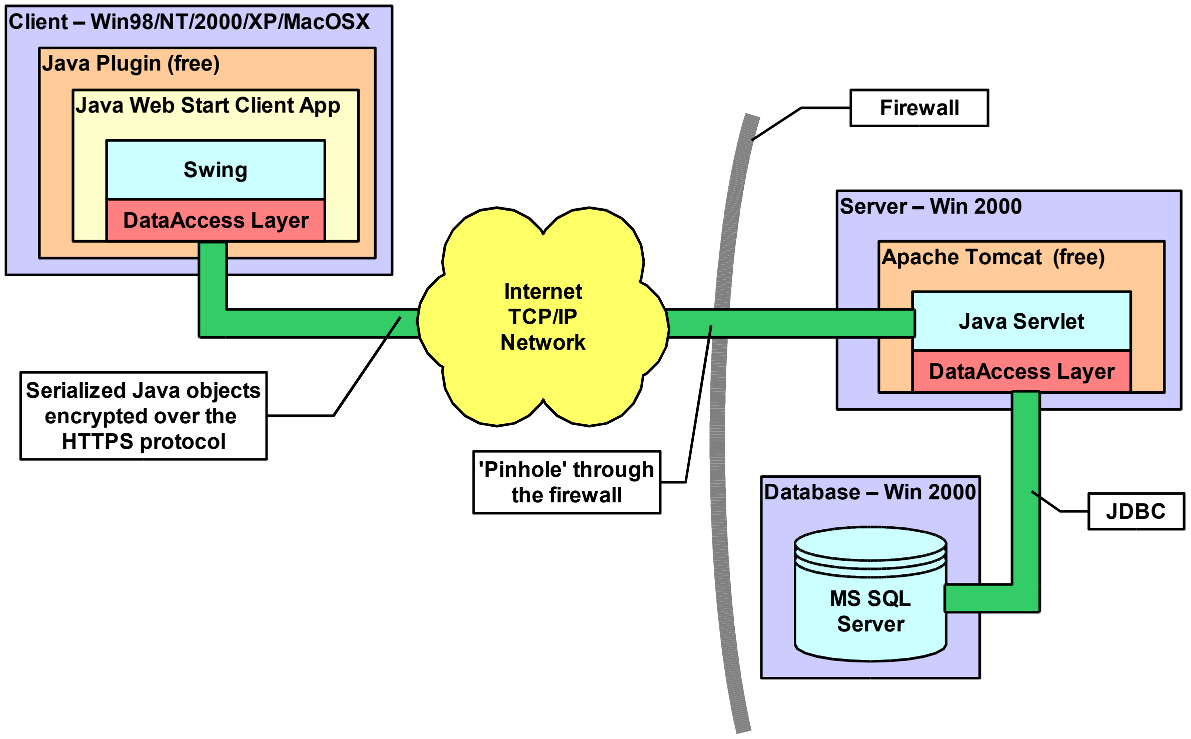
Keeping Client-side Code Up-To-Date

- The client portion of the application code is automatically downloaded to the users' machines for the first run of the application.
- This client-side code is cached on the users' machines.
- Every time the application is launched, Java Web Start *automatically* checks with the server to see if there are any code updates and *automatically* downloads them if necessary.

Getting Java Web Start

- Java Web Start is part of Sun Microsystems' free *Java Plugin.*
- If the plugin is not already installed, a *one-time* download and install of the plugin must be done.
- Many machines come with the Java Plugin pre-installed.
- Apple's Max OSX has built-in support for JNLP clients (Apple's version of Java Web Start).
- Sun provides a free Java Plugin for Windows, Linux, and Solaris operating systems.

Thin Java Client 3-tier Architecture



- Java Web Start is used on the client side to provide a thin, yet fully interactive rich graphical client.
- Because the client-side code is run with Java Web Start, it is automatically kept up-to-date.

- The Java Web Start client talks to a Java Servlet middle tier over SSL encrypted network channel
  - The client serializes up a request object.
  - The servlet returns a serialized object as a response.

- This use of the servlet middle tier enables the Java Web Start client to remain a thin client.
- The Java Servlet middle tier talks to the relational database using JDBC (Java Database Connectivity). This architecture uses the following J2EE technologies: JDBC, XML, and Servlets.

- This architecture is compatible with the addition of a browser-based client for future features for more casual users that may not want to download a plugin.
- A Java Web Start client can open up links in a user's browser.
- A link in a web page in a browser is used to launch the Java Web Start client.

Pros of this Java Web Start Architecture:

- Increased user interactivity over a browser due to Swing-based Java Web Start client:
  - Capable of using keyboard shortcuts (accelerators/mnemonics).
  - Modal dialogs (user must respond before proceeding).
  - Client-side sorting.
  - Client-side precision printing (can control page-breaks, headers, etc).
  - Client-side validation of user input.
  - Drag-and-drop user-interaction is possible.
  - Menus—both pulldown and "right-click" popup.
  - Background Threads—multiple background tasks can be simultaneously underway allowing the user to move on to the next task without waiting for everything to complete.

- More secure:
  - No disk caching of data. All data is in memory and simply closing the application window ensures that all data is gone.
  - After a specified time of inactivity, the client application can flip to a blank background (hiding sensitive data) and display a password prompt to block access. When the user enters the correct password, they are back in the application exactly where they left off—no need to redo work. Unlike a policy request that users employ a screen saver with password protection, this mechanism is independent of any settings on the client machines.

- Client-side data processing to whatever extent is deemed appropriate.
- Reduced load on the middle tier compared to a Servlet/JSP solution (only the data is sent down to the client, not the data *plus* the HTML and JavaScript).
- Unlike a traditional 2-tier solution, Java Web Start easily keeps client-side code automatically up-to-date.
- Thin client (although not quite as thin as a browser).
- Significantly simpler (therefore faster and less expensive) to develop and maintain than a Servlet/JSP solution.

- User's **operating system** does not matter (Windows, Mac OSX, Linux, Solaris).
- User's **browser brand** does not matter (IE, Netscape, Mozilla, Firefox, Opera, Safari).
- User's **browser version** does not matter.
- It does not matter if user has turned off **cookies** for security or privacy reasons.
- It does not matter if user has turned off **JavaScript** for security or privacy (or popup annoyance) reasons.
- The Java Plugin is **free**.

Cons:

- Requires one-time download of free Java plugin on the client machine.

User Perspective on a Java Web Start Client

- For the first run, user goes to a web page with a link to a JNLP file.
  - If necessary, user performs the one-time download of the FREE Java Plugin.
  - User clicks on the link or graphic to launch the application.
  - The web browser delegates the processing to the Java Web Start plugin.
  - Java Web Start downloads all the necessary code and starts the application running.
  - User is given the option of creating a shortcut to this application on their desktop and Programs folder.
    - In the future, this shortcut can be used to launch the application without having to use a web browser at all!

- The next time the user wants to run the application, the user can go to the same web page with a link to a JNLP file *or* the user can launch the application directly by using the desktop icon.
  - If necessary, Java Web Start automatically downloads any code updates before launching the application.
    - If there are no updates, the application immediately runs.
    - Java Web Start accomplishes this by caching the application code, images, etc. on the client machine.

Better User Experience

- Fewer data are transferred over the network for Java Web Start clients.
  - Browsers must be given the data and the HTML markup for every page—which is significantly more overhead.
  - Client-side sorting and printing saves a re-request over the network.
- Keyboard shortcuts boost user productivity.
- Extensive client-side validation of input.
- Documents can be printed locally.
- Step-by-step wizards can guide the user through a complex process.
- Users can capture the current screen and popup a window for text feedback to report a potential bug or to suggest a new feature.

- Help Desk personnel could “look over the shoulder” of a user.
    - Java Web Start technology would allow a help desk worker to have a view-only window that peers directly at the Java Web Start client running on the user's machine.
- Users can be contacted asynchronously with messages.
    - Perhaps a message about an emergency maintenance requirement to take the system down in 15 minutes.

Cost Savings

- Java Web Start clients can be developed less expensively than Servlet/JSP clients.
- Due to light client-side processing, there is less load on the middle-tier server, the database server, and the network bandwidth.

What About a Browser-Based Client Instead of Java Web Start?

- The development time frame would increase:
    - It takes longer to build the user interface portions for a browser-based client than it does for a Swing-based (Java Web Start) client.
    - User input validation must be written in JavaScript for the client side and then must be re-implemented in Java for the server-side validation (for data security—in case the browser has JavaScript turned off).
    - With a browser, a user can go anywhere as any time, can press the Refresh button, can press the Back button, can have JavaScript turned off, can have cookies turned off, and other issues that make application development more complex.
    - Each Edit, Compile, Run, Test cycle with a browser-based solution requires a full deployment. With the Java Web Start architecture shown at the beginning of this document, developers can run in a two-tier mode for most development—saving time on every Edit, Compile, Run, and Test cycle.
    - Most of the technologies used for a browser-based solution (JSP, HTML, CSS, JavaScript) require run-time testing to see if something is working. This results in far more Edit, Compile, Run, and Test cycles. With a Swing-based solution, trouble can be found earlier either by Eclipse *before* compilation or during compilation—resulting in less time being wasted chasing down bugs.
    - The complexity of a browser-based solution—so many technologies are intertwined for the client: HTML, JavaScript, CSS, JSP, XML, and Struts.

- The user would have a less-efficient user interface. Browser-based interfaces are not able to be as efficient as a rich-client interface built using Swing.
- The user would have a slower interface due to the fact that browser-based clients must receive not only the data for display, but also the HTML formatting and JavaScript validation code for every page. A Java Web Start client can receive just the data—compressed and encrypted. This is a significant difference for dial-up users.
- Many features would be impossible:
    - Help desk personnel "looking over your shoulder"
    - Automatic hiding of sensitive data after a timeout

- Right-click popup menus
- Integrated hover-over help
- No keyboard shortcuts—reducing worker efficiency
- No client-side page formatting and printing
- ...and other items that were mentioned earlier in this document.

Summary

- Java Web Start enables the development of a rich client user interface.
- Java Web Start automatically keeps client-side code up-to-date.
- Swing gives the application developer more options for user interaction.
- Users have a more efficient interaction with the software.
- Java Web Start solutions can be developed more quickly, more robustly, and less expensively than browser-based solutions.