Xito Platform White Paper

prepared by: Deane Richan

Table of Contents

Introduction

From the introduction of Java 1.0, the Java runtime environment has revolutionized the software industry. Before Java most applications were written to run on client computers running the Windows operating system. The applications were primarily written in C, or C++, or often the easy to use Visual Basic.

Java's original purpose in the 1.0 JDK was the creation of client Applets. These applets could be deployed in a web browser and accessed from a web page. The applets had special security restrictions enabling them to be executed in a safe protected environment where they could not harm or gain access to local computer resources. Many companies saw the vision that applets provided, and predicted that the future of computing would involve the execution of these server deployed applets.

Large software companies such as Lotus and Corel released preview versions of Office Suite applications written using Java Applet technology. Companies such as Oracle, Sun, and IBM, proposed Network computers where the applications would all be deployed from servers and run in a client environment primarily written in Java.

The combination of Java's security environment and Object Oriented Cross Platform design made it the platform of choice for this new client software revolution.

Problems with the Initial Java Client Vision

Although Java client applications gained large scale interest in these early years there were **three main problems** that caused the initial excitement to dim:

1. **Slow Performance and Lack of User Interface functionality.** Java's initial slow execution time and lack of superior client user interface libraries made it difficult to create applications that performed well. Also user interfaces were rudimentary.

2. **Software did not add value over traditional Office Suites.** Software companies had not made a paradigm shift in the types of applications they felt end users needed or wanted in this new deployment scheme. Companies like Corel, and Lotus simply wrote stripped down versions of word processors, and spreadsheets. There applications didn't add any value to existing office suite applications and did not take advantage of any unique features of the Internet.

3. **Lack of Standards and involvement of third party software and ISVs.** The application environments created by Corel, Lotus, SUN, and Oracle where not compatible and were written specifically for their limited applications. Independent software companies had little motivation or interest in developing

on top of the platforms.

Because of these problems and the increasing interest in HTML based web applications Java shifted to focus on Server Side development. With the language richness and cross platform benefits of Java, this approach has been extremely successful for Java. Today the vast majority of Java applications are deployed as web applications with the Java code running on a web or application server.

Hindrance to rebirth of Java on the Client

Applications written in Java to be run on the client have not gone away. There are many Java development projects that have continued to evolve. Most of these projects revolve around the software development tools category with the major efforts being Sun's NetBeans project and IBM's Eclipse project. Like these two projects, most well used Java applications that run on the client are written like standard native, shrink wrapped, monolithic applications. These applications do not take any particular advantage of Java for security and network deployment, and are only geared towards technical computer users not the mass market.

In 2000, SUN introduced a new Java client deployment technology called Java Network Launching Protocol (JNLP) with an implementation called Java WebStart. This technology allows Java application developers to describe the pieces of their application using XML based descriptor files and deploy the applications through a web browser. When users click on links to these descriptor files the applications are automatically downloaded and executed.

Although promising technology, JNLP has not produced large interest by developers in the technology. The vast majority of Java client applications continue to be deployed as ZIP files that must be extracted and run with a Batch or Shell script file. Also even with the introduction of JNLP, there is very little commercial interest in Java applications on the client. JNLP shows an interesting technology but not one that has captured the vision or excitement of the industry. Another issue that occurs with the use of JNLP and traditional applets is the continued use of Web Browsers as the User interaction point. Although Launching applications from a web browser is simple for the initial execution, always using a browser tends to make Java Applets and JNLP applications feel like an addition to the browser system rather then an improved mechanism in their own right.

In summary there are **three core existing Java client issues** that need to be addressed for Java to succeed on the client.

1. **Large Memory requirements and slow performance running several Java Application Virtual Machines.** Although Java application performance has improved drastically, the default behavior of Java applications is to run a single application in an instance of a Java Virtual Machine. This is fine for running one

or two applications but if users want to run 10 or 20 applications together the behavior of using a single VM per application will quickly bring a computer to its knees. Technology to share VM instances needs to be introduced on the client. JDK 5.0 introduces functionality in this direction but more improvements need to be made.

2. **Poor user experience and interface to access and use Java applications compared to native applications.** Java application deployment is often more complex then necessary. Users often have to deal with issues such as Java VM locations, VM Versions, Batch and Script files, Classpaths etc. Java Applet and JNLP help reduce this road block but their user interface experience is a hindrance to wide spread adoption. The technology is strong but there is a great lack of compelling user interactions. Also traditional client Java application developers must invest more time in the quality of the User Interface of their applications to make them appeal to the mass market.

3. **Complex and poorly understood security model for running applications.** Computer novices don't understand Java security implications nor do they want to. Java applications have the unique advantage of running in an environment called the sandbox. This sandbox isolates Java applications from restricted areas of the computer allowing them to be run safely even when the trust of their author can not be validated. Applications also have the ability to be given permission on a case by case basis allowing applications to have broad control over the computer if allowed by the user. This functionality has yet to be exposed in a simple and easy to understand interface that non-technical computer users can use.

Xito Platform for Java Client Deployment

The Xito platform has been created to address these issues and develop a rich Java client experience for end users. The platform is comprised of modular services that work together to produce a network centric architecture with an easy to use user interface integrated with the native host operating system.

The Xito Platform will address the core issues of existing Java client technology, namely: large memory and performance requirements, a poor user interface to manage Java applications, and complex security model that is not understood by end users.

The Platform will also address the initial problems with Java client software which hindered Java adoption for client applications in the beginning. The main approach to addressing these issues are: Create very user friendly user interfaces for the environment, create applications that take advantage of the network and Java's unique capabilities, and create a platform that third-party developers are excited to write for, creating a broad market for network deployed Java applications.

Xito Platform Architecture

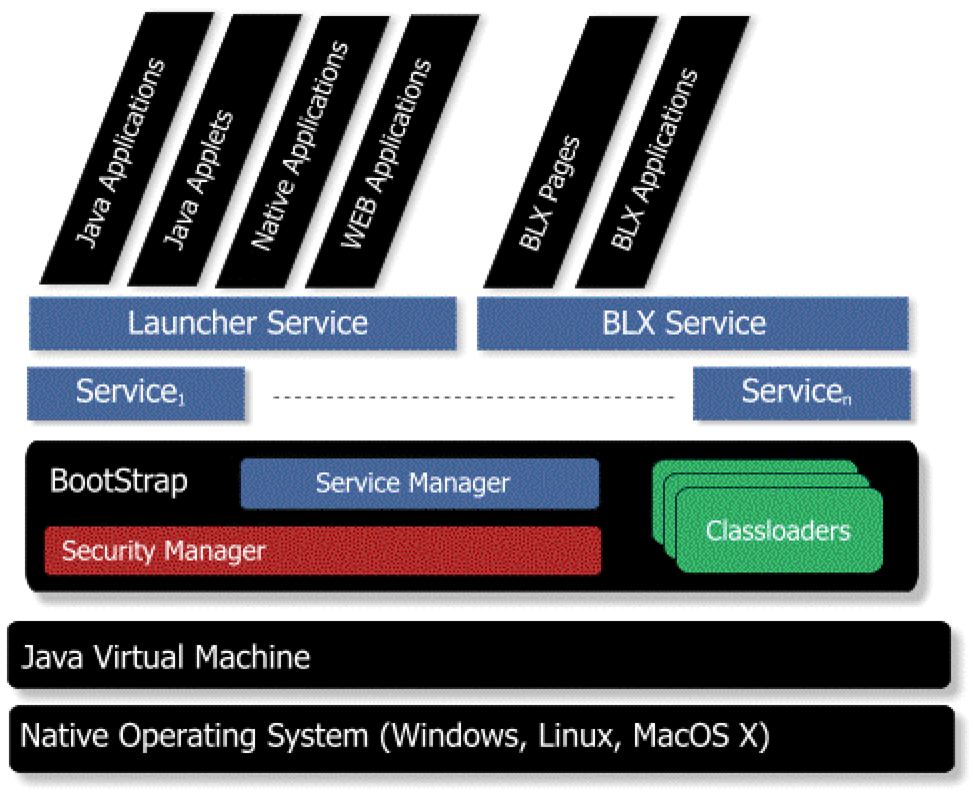
The Xito platform architecture is based on a lightweight bootstrap and service framework. The bootstrap initializes the environment, sets up the security manager, and then loads a set of services. Services are just Java Applications and don't require any interface implementation.

The BootStrap launches services and applications into a single Java virtual machine. This reduces memory requirements of all applications and services running in the environment and also enables very quick execution of downloaded applications.

Sharing of a single Virtual Machine increases performance and reduces memory requirements and allows untrusted code to run along side trusted code. However sharing a VM requires applications to cooperate with relation to the VM resources. Applications that want full control over the virtual machine can be launched into a new VM instance from the bootstrap.

Applications and services running in the bootstrap are designed to be executed directly from a web server. Therefore there is no installation required for services or applications. They are also always up to date because they are running from remote servers. A caching mechanism is used in the BootStrap to cache Java applications and service resources on the local hard drive so that they do not need to be downloaded each time they are executed. Also the local cache enables applications to be executed when the client is not connected to the network, a feature not available for traditional HTML based web applications. This network launching functionality allows applications deployed through Xito to have the benefits of a web application and also the benefits of a rich client application.

The following diagram describes the major components of the Xito Platform.



Bootstrap

The Xito BootStrap is the entry point to the platform. The Bootstrap is simply a Java Application that initializes the environment. The Bootstrap reads system properties, starts a Security Manager, and then executes a set of services. The Bootstrap is the only application that is allowed to shutdown the VM. This enables many applications to be run in the Virtual Machine together as long as they cooperate. Applications can also be configured to run in their own virtual machine. Applications that attempt to exit the VM will have their Windows Disposed and be shut down. This enables many existing Java applications to work in a single VM out of the box.

The BootStrap also handles all loading of application resources and execution of applications. It provides this functionality through four main components: Service Manger, Cache Manager, ClassLoaders and Executable Descriptors.

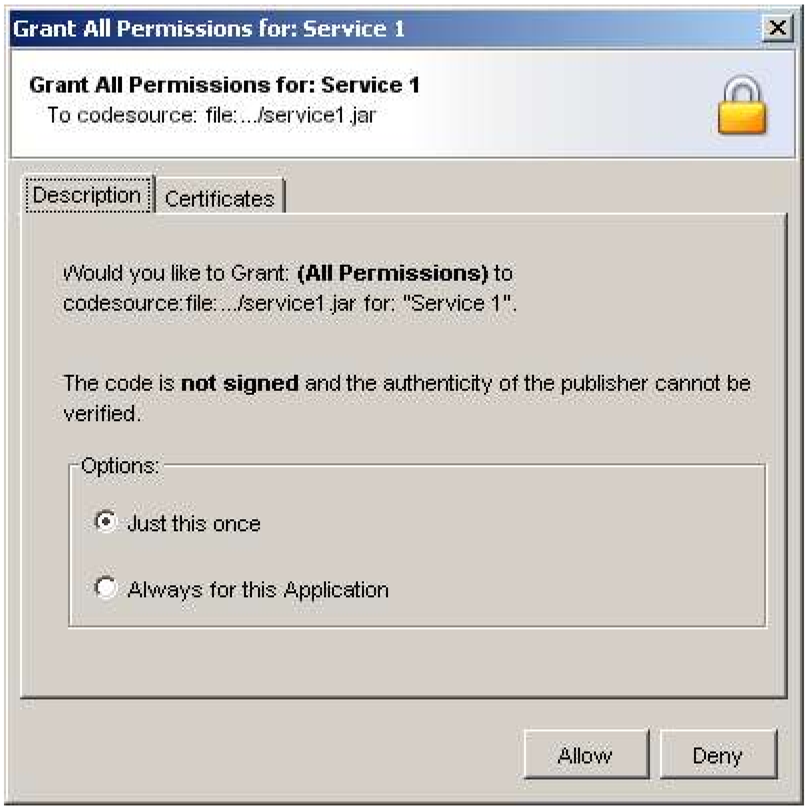
Bootstrap also contains a win32 native executable that can be used to launch the application on Windows platforms. The native exe can be renamed to what ever application name the developer wants for their application.

Security Manager

The Xito Bootstrap Security Manager enables applications downloaded from the Internet to be run in a secure environment. The Security Manager maintains permissions that have been granted to applications or services. When each Java application is executed it specifies the Permissions that it needs to execute. Based on the level of permissions requested the Security Manager will prompt the user to grant permissions to the application. The permissions can be granted for each codesource, each application, or each signer of an application. Applications that only request restricted permissions will not cause a security prompt to be displayed to the user. It is the intention that the vast amount of applications deployed through Xito run in the restricted environment.

Consider that nearly all browser based web applications do not require access to the local computer. Network deployed Java applications should be similar and only rarely do they need access to local machine resources. **Making applications default to restricted permissions will enable faster adoption by end users to applications since they do not have to worry about the security ramifications of the application**.

When applications do require additional permissions, the user interface for granting security permissions does not overwhelm the user with choices and should provide a simple selection.


Grant All Permissions for: Service 1

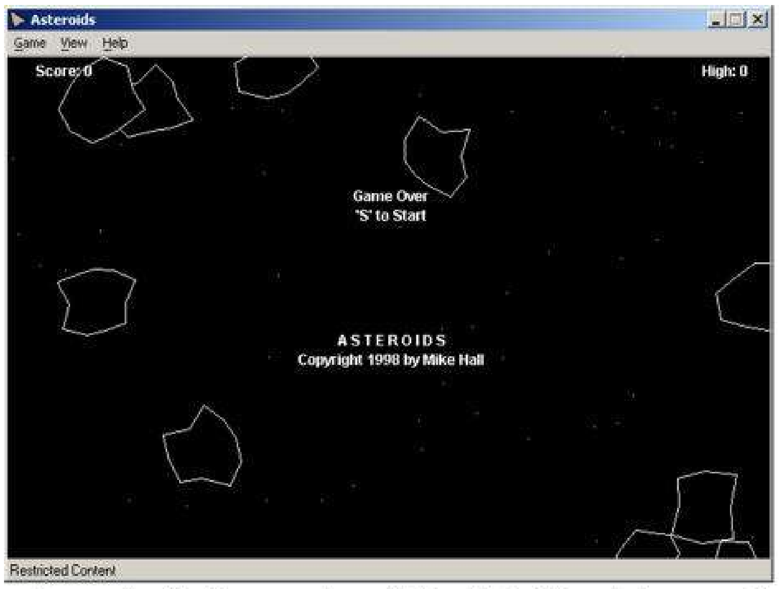
Grant All Permissions for: Service 1
To codesource: file:.../service1.jar

Would you like to Grant: **(All Permissions)** to codesource:file:.../service1.jar for: "Service 1".

The code is **not signed** and the authenticity of the publisher cannot be verified.

Options:
[✓ Just this once]
Always for this Application

Applications run in the restricted environment or applications that have been denied All Permissions would not have the ability to access arbitrary resources on the user's Hard Drive or make network conntections to servers other then the server it was deployed from.



Applications running with Restricted Permissions contain a notice on each window.

Service Manager

The Service Manager is started after BootStrap has been initialized and the Security Manager installed. The Service Manager will download and start 2 sets of services. These are Boot Services and Startup Services. Boot Services are required services that must be started without exception for the environment to boot. Startup services on the other hand are optional services that may start with errors but the environment will continue to run.

Services are Java Applications deployed in Jar files and deployed with a BootStrap installation or executed over the internet at startup. Each Service is described with an Service XML Descriptor. The Service Manager knows which services to load based on two XML Descriptors named boot_services.xml and startup_services.xml

There are no special interfaces that have to implemented by these services. In fact they may not even know they are running in the BootStrap environment. This enables many existing service oriented applications to be deployed as BootStrap Services.

Some examples of Services could include:

SplashScreen Service that monitors the boot process and display a splash screen.

Launcher Service which enables the execution of Java Applets and JNLP Web Applications.

Preferences Service which implements a Preference API for storing application settings

Chat Messaging Service that supports the connection to various Chat servers

Web App Service that provides an embedded web server in the environment

Database Service which could provide an embedded Database service in the environment

Desktop Service which would provide the main Desktop User Interface for the environment

Or a developer could use the Bootstrap to simple deploy their Application which would only require that they list their application as a service in the boot or startup services.

Application Cache

BootStrap includes a CacheManager that enables application code and resources to be downloaded and then cached on the local client. Most applications executed from within BootStrap should be downloaded from remote servers. This enables applications to be deployed as easy as web applications but allows them to have rich user interfaces not possible with web applications.

Applications can be configured to always download the latest code from the server or to use the Cached resources if they are up to date. These settings are configured in the Executable Descriptor for the application or Service.

Executable Descriptor

An Executable Descriptor maintains the information needed to execute an application. The BootStrap maintains two such implementations a ServiceDesc and an AppDesc. The ServiceDesc is an Executable Descriptor modelled on top of the Service XML Decriptor file. This enables developers to deploy their applications as services in the BootStrap environment. An AppDesc is also provided which enables basic execution of an embedded Java application. Developers can use this mechanism to launch applications inside the BootStrap environment.

Launcher Service

One of the main optional Services provided by the Xito Platform is the Launcher Service. This services provides mechanisms to launch several application types inside the environment. These include:

Java Applets: Traditional web based Java Applets can be executed inside the environment. These applets run in their own Window and run in the same restricted environment expected by the user.

Java Applications: These are standalone Java Applications that have not been configured to use JNLP technology. A user must supply a list of Jars and the name of the Main application Class. However Jar resources can be loaded over the Internet or other network.

JNLP Applications: These are standalone Java Applications that have been deployed using the JNLP descriptor files. The Launcher Service implements the JNLP Sepcification to enable it to launch these Java Applications.

Native Applications: Launcher service provides an easy straightforward way to launch local applications.

Web Applications: These browser applications can be launched through the Launcher service. The default web browser on the client will be launched and the web app loaded.

BLX (Blocks) Service

The BLX prounced Blocks Service is a Java Component framework for deploying application components called BLX (Blocks) remotely on web servers and allowing those BLX to be downloaded on demand. The BLX Framework enables entire rich clients to be configured as separated components and then deployed using simple XML pages. The BLX XML pages are well formed but do not conform to a single DTD this allows developers to enhance and extend it to their needs. More information will be coming later about this interesting technology and its ability to stream applications to client machines.

Summary

The Xito Platform is in the early development phases. The first release of the platform is the Xito BootStrap. This BootStrap can be used now by third-party developers to distribute their applications as services. Using Property and config files developers can configure the BootStrap so that it appears to be their application.

The first release of the platform that is available to end users is the Xito Application Manager. This simple application will be written to provide a easy to use professional consumer interface that allows any user to execute and manage Java Applications and Applets along with local applications and web sites.