IcedTeaNPPlugin

[LiveConnect design]

Deepak Bhole

Senior Software Engineer, Red Hat
2009-10-05

Plugin Architecture

- Divided into C++ and Java component
  - C++ side talks to the browser
  - Java side talks to JVM
  - Linked via FIFO link
  - Common string exchange format (UTF-8)
- Over 95% of changes in NPPlugin have been on C++ side
- Java side has been reused as much as possible to re-use the proven stable code

C++ Architecture

- Encompassing classes form the LiveConnect engine (which previously resided in Mozilla and was exposed via OJI)
- Each JavaScript var corresponding to a Java object has a corresponding IcedTeaScriptableJavaObject
- Engine controls the object life, and services all requests (get field, set field, invoke method, etc.)

C++ Architecture (Browser Interface)

- Browser interface consists primarily of IcedTeaScriptableJavaPackageObject, IcedTeaScriptableJavaObject and IcedTeaPluginRequestProcessor
- Above classes are unaware of Java interactions, and delegate to the Java interfaces for Java interaction
- They process all requests coming from the browser and going to the browser (getMember, call, eval, etc.)

C++ Architecture (Java Interface)

- Java interface consists primarily of IcedTeaJavaRequestProcessor
- This class has full knowledge of how the Java side works, and constructs appropriate requests to get all information from Java.
- The class process all requests to the JVM

Java Architecture

- Java side has 2 code classes aside from helpers, PluginAppletViewer and PluginAppletSecurityContext
- PluginAppletViewer is an interface to NetX, and processes JS related requests to and from NetX (the applet)
- PluginAppletSecurityContext is a direct reflection based interface to the VM. It processes all LiveConnect requests to and from the JVM
- Request processing is asynchronous, with scalable generic request processing workers

Java Architecture (PluginAppletViewer)

- Control of applet (initialize, resize, destroy, etc.) from browser
- Access to JavaScript from the applet (getMember, setMember, call, eval, etc.)

Java Architecture (PluginAppletSecurityContext)

- Direct access to the JVM from the browser (LiveConnect) via reflection.
- All reflection is built-in, so C++ side never needs to be aware of the complexities, unlike how OJI was.
- All VM calls are inside a sandbox, so JavaScript cannot do things that the default sandboxed VM can't.

Java Architecture (PluginAppletSecurityContext)

- Direct access to the JVM from the browser (LiveConnect) via reflection.
- All reflection is built-in, so C++ side never needs to be aware of the complexities, unlike how OJI was.
- All VM calls are inside a sandbox, so JavaScript cannot do things that the default sandboxed VM can't.

MessageBus architecture (C++)

- The link to Java is exposed to the rest of the code via a uniform “MessageBus” interface
- Since the code is unaware of the link specifics and has no synchronicity guarantee, the communication medium can be switched relatively easily.
- Whatever class is interested in the messages implements a “BusSubscriber” class and subscribes to the bus of interest.
- When messages come in, the bus notifies all subscribers

Example JS->Java workflow

- Example shows how NPP_HasProperty() works
- Browser has a link to an IcedTeaScriptableJavaObject representing a Java object instance
- It calls IcedTeaScriptableJavaObject::HasProperty()
- HasProperty() creates an IcedTeaJavaRequestProcessor (“java processor”)
- The java processor exposes all necessary APIs to the VM, including hasProperty (called hasField for Java naming consistency)

Example JS->Java workflow (contd.)

- Before making a hasField request, the processor subscribes itself to the “from Java” bus, so that it can read the response
- hasField request is made by the processor, posted to the “to java” bus
- Processor waits for response, or until timeout
- Once response is received, processor unsubscribes itself from the “from Java” bus and does postprocessing, and returns
- The IcedTeaScriptableJavaObject object reads the response, and sends it to the browser

Example Java->JS workflow

- All access to JS is via “JSObject”'s, as defined in the LiveConnect specification
- If applet wants to access a member of JSObject “window” for example, it will call getMember on the windows JSObject
- getMember calls a similarly named function in PluginAppletViewer
- PluginAppletViewer constructs a request for the C++ side, and posts it on the FIFO link

Example Java->JS workflow (contd.)

- On the C++ side, IcedTeaPluginRequestProcessor (plugin processor) is always subscribed to the “from Java” bus
- When the getMember request comes through, the plugin processor gets notified
- The embedded request