Jini for Desktop Integration

ServiceUI provides a lot of possibilities – What else is needed?

Gregg Wonderly
Cyte Technologies Inc.

The Desktop is a Context

- The desktop environment, for the user is a context where everything comes together.
- We all appreciate Cut-and-Paste between windows, drag and drop placement etc.
- Jini ServiceUIs have no immediately discernable relationship.
- The user needs to have control and the ServiceUI needs to know the users intent.

All of the ServiceUI Factories

- The initial development of ServiceUI includes a great, bare bones example of how to use the concepts in a GUI environment.
- The factories allow one to see how you might use ServiceUI to turn a Jini Service into a Web Service by providing a class that looks up a service, gets a particular UI role, and delegates to that service.
- For desktop integration, some extra functionality is essential.

JDesktopComponentFactory

- The JDesktopComponentFactory is an interface that extends JComponentFactory and adds a JDesktopContext parameter to allow the ServiceUI GUI to:
  - Find the top level component before it is realized.
  - Register a close handler
  - Notify the context that the GUI is closing
  - Request focus to be raised to the top.

JMenuBarFactory

- This interface is implemented by JDesktopComponentFactory. It provides the ServiceUI GUI a mechanism to advertise a JMenuBar.
- This allows a desktop integration component the chance to provide the container frame/window and the GUI to not loose functionality.

Security…

- When you have multiple ServiceUIs running in the same JVM, you need to be able to maintain classloader isolation as well as policy enforcement.
- The user needs to be able to trivially allow new things to become part of their desktop without great risk to corruption of their environment.
- This is not a trivial task!

Security by Example

- The com.sun.jini.start.ServerStarter class and the associated ServiceDescriptor implementations provide an example of the level of separation that should be used to provide sufficient isolation for the innocent user and harmless ServiceUI.
- It would be nice if this mechanism was extracted and made available for use by containers of any type that need to load multiple classpath/codebase based applications into the same JVM.

What are the Problems?

- A big issue from a GUI perspective is the use of JDialog by GUI ServiceUIs.
  - JOptionPane.showInternal*() methods still block the whole application instead of just one JInternalFrame.
  - Providing a dialog mechanism on JDesktopContext would be a potential solution for new applications to use. Existing ones would need code changes to use that API.

What are the Problems? (cont)

- ClassLoader isolation is important.
  - I've had issues with class versions between the desktop implementation and ServiceUIs which I've been able to solve with PerferredClassLoader.
  - I still don't have complete isolation by providing a PreferredClassLoader for the desktop that hangs below the application ClassLoader so that it is not visible to the ServiceUIs.

What are the Problems? (cont)

- Security and Policy
  - In my current implementation, I use blanket policies on a codesource host through DynamicPolicy and ProtectionDomains.
  - I'd like to extend reef.dev.java.net to provide access to the codebase so that I could show the user what they need to allow for code source level, and then get constraints for Principals needed.

What's on the Horizon?

- I'm working on extracting the JiniDeskTop class out of startnow.dev.java.net to create a new project for just it.
- I'm trying to include user controlled security based on codebase/codesource information in the base version.
  - That implies I am also trying to make reef.dev.java.net into a standalone lookup service implementation too.

Participation

- I have a lot of things in progress and work is keeping me from making all the progress I want.
- If you want to help, send me an email: gregg@wonderly.org
- If I get some interest, I'll create the new project now, and let those who are interested take the lead.

Discussion/Demo

Gregg Wonderly
Cyte Technologies Inc.