Porting a Swing Application to the NetBeans Platform

Anton Epple
www.eppleton.de/blog
Agenda

- Some examples of former “plain Swing” Apps
- Our experience porting a former Swing Application
- A generic process for porting an existing Swing Application to NetBeans Platform
Examples of ported Applications

1. Sepix CRM (Aljoscha Rittner)
2. Instant JChem (Tim Dudgeon, Petr Hamerik)
3. Blue Marine (Fabrizio Giudicci)
4. ChipInspector
Examples of ported Applications

CRM System
http://www.sepix.de/
Examples of ported Applications

http://bluemarine.tidalwave.it

Demo
ChipInspector: Starting Point

• Based on own Framework:
  > Weblauncher
  > XML based configuration
  > Update Manager
  > Download Manager
  > Proxy Handling
  > Options/Settings
  > JavaHelp
  > Docking
  > WebBrowser Component
Why (NetBeans) RCP

- Less framework maintenance
- Generic features for free (Update Manager, docking...)
- Less boilerplate code
- Application Lifecycle Management
- Standardized look & feel
- Modular structure (Dependency Management, Versioning)
Common Pitfalls

- Some features are hard to remove (Example “Web Browser” in Options Panel)
- Some settings are difficult to change (Example SecurityManager)
- ClassLoader hierarchy problematic with some libraries (Spring, JasperReports)
- Some things are harder to do (Obfuscation...)
Extras that made life easier

- You can exclude stuff from standard Java (Example JAX-WS) -> More control over environment
- You can use ServiceLoader like functionality in java < 1.6
- SystemFileSystem for storing your data
- You can easily create distributions (Zip Distro, Webstart, OpenInstaller, Os X Application)
Generic Process - Conversion

1. Third-party Libraries -> Library wrapper
2. Swing Panel -> TopComponent
3. Action -> CallableSystemAction, CallbackSystemAction...
4. Adjust Menus -> Layer (SystemFileSystem)
5. Dialogs, Wizards -> Dialogs API
6. Beans -> Node Wrappers
7. Synchronizing Views / Selection -> Lookup
Demo

• Porting a simple Swing application to the NetBeans Platform
9. Options, settings -> Options API
10. Libraries/features/packages -> modules
11. Extension Points -> META-INF/services, Lookup, SystemFileSystem
12. JavaHelp -> Help System Integration
13. Branding -> Resource Bundles
14. Mimetypes -> Datasystem API
15. Add IDE stuff: Project Tree, Web Browser, Database Explorer...
Summary

- First steps are simple,
- then there's a steep learning curve.
- NetBeans really helped reducing boilerplate code

- The real gain from porting comes from adapting NetBeans architecture
Hints

• Well written applications are easy to port

• To make sure your application remains portable:
  > Separate UI code from Business Logic code
  > Minimize dependencies
  > Know the APIs your application depends on (and their dependencies...)
  > Use wizards, don't depend on them
  > Make sure you still understand your application
Resources

Tutorial for getting started:
http://platform.netbeans.org/tutorials/60/nbm-porting-basic.html

Blue Marine developers experience:

XING Group
http://www.xing.com/group-20148.82db20