NetBeans Platform Compared with Eclipse Rich Client Platform

Kai Tödter, Siemens AG
Geertjan Wielenga, Sun Microsystems

ID# ????
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
MP3 Manager: Eclipse RCP based
MP3 Manager: NetBeans Platform based
Demo: MP3 Manager on both Platforms
Outline

- MP3 Manager: A demo application
- Software Architecture
- Component Model & Module Concept
- UI Toolkits & Customization
- Starting Application Development
- Project Structure
- Actions
- Extension Points & Lookups
- Update Functionality & Help System
- Misc
- Conclusion
Eclipse RCP Architecture

Rich Client Application

App Plug-in... ... ... ...

OS
Java VM
NetBeans Platform Architecture

Rich Client Application

App Plugin...

Java VM
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Eclipse RCP: Based on OSGi (1)

- Dynamic modules for Java
- Highly adopted standard
- OSGi Bundle is the unit of modularization
  - Eclipse Plug-in == OSGi Bundle
  - Roughly equivalent to a JAR
  - Self-described using MANIFEST.MF metadata
Eclipse RCP: Based on OSGi (2)

- **The OSGi Runtime**
  - Manages dependencies and lifecycle of bundles
  - Explicitly supports dynamic scenarios

- **Bundles interact through**
  - Java package sharing
  - OSGi Service registry
  - Eclipse Extension Registry

- It is possible to run two or more versions of the same bundle in one application
NetBeans Module System (1)

- Dynamic modules for NetBeans Platform
- Proprietary, but
  - Basic idea taken from the Java Extension Mechanism
- NetBeans Module is the unit of modularization
  - Roughly equivalent to a JAR
  - Module attributes in MANIFEST.MF metadata
  - Extra XML descriptor needed by the platform runtime
NetBeans Module System (2)

● The NetBeans Runtime
  • Manages dependencies and lifecycle of NetBeans modules
  • Some support of dynamic scenarios (ModuleInstall)

● NetBeans modules interact through
  • Java package sharing
  • Service registry
  • XML Layer

● It is possible to run two or more versions of the same NetBeans module in one application
Outline

- MP3 Manager: A demo application
- Software Architecture
- Component Model & Module Concept
- UI Toolkits & Customization
- Starting Application Development
- Project Structure
- Actions
- Extension Points & Lookups
- Update Functionality & Help System
- Misc
- Conclusion
NetBeans UI: Swing

- Very mature UI toolkit
  - But more a widget set than an UI application framework
  - But NetBeans adds framework functionality
- Good performance since Java 1.3
- Java Standard included in the JRE
- Very good free GUI builder Matisse out of the box with the NetBeans IDE
- Native Look & Feels are emulated
  - Since Java 6, native rendering is used if possible
- Very good customizable through the pluggable Look & Feel mechanism
NetBeans Platform based Example
NetBeans MP3M with Napkin Look & Feel
Eclipse UI: SWT/JFace

- Very mature UI toolkit
  - SWT provides rich widget set
  - JFace adds UI application framework, with
    - Viewers, Forms, Data binding, Wizards, etc.
- Excellent performance through native widgets
- Some GUI builders available for the Eclipse IDE
  - But the most of them are commercial
- Native Look & Feels
  - Highest OS Look & Feel fidelity
- Partly customizable through Presentation API and custom widgets
SWT Examples
Eclipse MP3 Manager with customized UI
More UI Customization: IBM Lotus Notes
Docking Systems

- A docking system is a windowing system, where the windows can be layouted in several regions.
- These regions use tab containers for the containing windows.
- The windows can be dragged and dropped into other regions.
- The windows can be minimized and maximized.
- The windows can be undocked (Stand alone on the OS desktop).
- Both, Eclipse RCP and NetBeans Platform provide excellent docking systems!
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Starting Application Development

• NetBeans Platform
  • Start with a suite
  • Remove all IDE specific modules
    • Creates application shown on the next slide
  • Remove all the UI elements you don’t want to reuse

Eclipse RCP
  • Start with a plug-in
  • Choose a RCP template, e.g. Hello World
    • Creates application shown on the next slide
  • Add new UI contributions
Eclipse RCP Hello World
NetBeans Platform Hello World
Outline

- MP3 Manager: A demo application
- Software Architecture
- Component Model & Module Concept
- UI Toolkits & Customization
- Starting Application Development
- Project Structure
- Actions
- Extension Points & Lookups
- Update Functionality & Help System
- Misc
- Conclusion
Eclipse RCP Project Structure

• Plug-ins
  • Provide the functionality
  • Often separation between core and UI plug-ins

• Features
  • Collection of plug-ins that implement the feature’s functionality
  • Needed for Update functionality
  • Provide Feature Branding and licensing info

• Product Configuration
  • Can be put in a feature or in a plug-in
  • Contains launching, configuration and product branding info
NetBeans Project Structure

- Modules
  - Provide the functionality
- Suite
  - Collection of modules that implement the application’s functionality
  - Provides application branding and licensing info
  - Usually one suite per application
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Creating an Action in NetBeans

- Use the “New Action...” wizard
- Fill out the form
- The Action’s Java class template is generated
- The XML layer (layer.xml) for contributing UI to the menu bar and tool bar is created automatically
Action Class in NetBeans (1)

```java
public final class AddMusicFolderAction extends CallableSystemAction {

    final JFileChooser fc = new JFileChooser();

    public void performAction() {
        // Action’s business logic
    }

    public String getName() {
        return NbBundle.getMessage(AddMusicFolderAction.class,
                                   "CTL_AddMusicFolderAction");
    }
}
```
Action Class in NetBeans (2)

```java
protected String iconResource() {
    return "com/siemens/ct/nb/mp3m/actions/add_folder.gif";
}

public HelpCtx getHelpCtx() {
    return HelpCtx.DEFAULT_HELP;
}

protected boolean asynchronous() {
    return false;
}
```
The XML Layer

<filesystem>
  <folder name="Actions">
    <folder name="File">
      <file name="com-siemens-ct-nb-mp3m-actions-AddMusicFolderAction.instance"/>
    </folder>
  </folder>
  <folder name="Menu">
    <folder name="File">
      <attr name="AddMusicFolderAction.shadow/RemoveMusicFolderAction.shadow" boolvalue="true"/>
    </folder>
  </folder>
  <folder name="Toolbars">
    <folder name="File">
      <file name="AddMusicFolderAction.shadow">
        <attr name="originalFile" stringvalue="Actions/File/com-siemens-ct-nb-mp3m-actions-AddMusicFolderAction.instance"/>
      </file>
    </folder>
  </folder>
</filesystem>
Creating an Action in Eclipse

- Extend the Extension Point “ActionsSets”
- Fill out the form
- The Action’s Java class template is generated by clicking the class attribute
- The XML layer (plugin.xml) for contributing UI to the menu bar and tool bar is created automatically
Creating the ActionSets Extension

Extension Element Details
Set the properties of "action"

- id*: com.siemens.ct.mp3m.actions.AddMusicFolderAction
- label*: %AddMusicFolderAction.label
- accelerator:
- definitionId: com.siemens.ct.mp3m.commands.AddMusicFolder
- menubarPath: file/additions
- toolbarPath: additions
- icon: icons/add_folder.gif
- disabledIcon:
- hoverIcon:
- tooltip:
- helpContextId: com.siemens.ct.mp3m.actions.AddMusicFolderAction
- style:
- state:
- pulldown:
- class: com.siemens.ct.mp3m.actions.AddMusicFolderAction
- retarget:
- allowLabelUpdate:
- enablesFor:
Action Class in Eclipse (1)

```java
public class AddMusicFolderAction implements IWorkbenchWindowActionDelegate {

    private IWorkbenchWindow window;

    public void run(IAction action) {
        // Action’s business logic
    }

    public void init(IWorkbenchWindowWindow window) {
        this.window = window;
    }
}
```
public void dispose() {
    window = null;
}

public void selectionChanged(IAction action, ISelection selection) {
    // Selection changes can be handled here
}

Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Eclipse Extensions and Extension Points

- Extension
- Extension Point

Rich Client Platform

Platform Runtime
Lazy loading with Eclipse Extension Points

- Create an Interface and expose it to the public
- Create an Extension Point that provides an attribute for the implementing class
- A using plug-in could extent the Extension Point and provide an implementation of the given interface
- The provider of the Extension Point can check at runtime, what extensions are available and what to do with them
Interface and Extension Point Editor

```java
public interface IMP3InfoProvider {
    public IMP3Info getMP3Info();
}
```
The Extensions Check at Runtime

```java
IConfigurationElement[] providers = Platform.getExtensionRegistry()
    .getConfigurationElementsFor("com.siemens.ct.mp3m.mode",
    "mp3info");

for (IConfigurationElement provider : providers) {
    try {
        IMP3InfoProvider provider = (IMP3InfoProvider)
            provider.createExecutableExtension("class");
        // do something useful with the dynamically created class...
    }

```
Benefits of the Extension Point Mechanism

● Very good scalability
● Loose coupling of components
  • Using String IDs rather than Java objects
● Very good startup time
  • UI contributions specified in the XML layer are processed at startup
  • Lazy loading of Java classes due to extension check at runtime
The Tip of the Iceberg

- Startup time: $O(\text{plug-ins used at startup})$ rather than $O(\text{plug-ins that add UI contributions})$. 

![Diagram of an iceberg with Declarative Definition (manifest) and Procedural Implementation (Java JAR) sections.](image-url)
Lazy Loading in NetBeans

- Provide an Interface and expose it to the public
- Create a directory META-INF/services
- Create a file with the fully qualified name of the interface in that directory, e.g. `com.siemens.ct.mp3m.model.IMP3InfoProvider`
- Put the fully qualified name of the implementing class as content in the file
- The user that checks the global lookup at runtime for implementations of the interface
Using the Global Lookup

IMP3InfoProvider provider = (IMP3InfoProvider) Lookup.getDefault().lookup(
    IMP3InfoProvider.class);
NetBeans Service Approach Benefits

- Very good scalability
- Loose coupling of components
- Easy to use
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Integrating Update Functionality

- Both Platforms provide sophisticated Update Mechanisms
  - Eclipse Update using update sites
  - NetBeans Update Center
- Both out of the box update functionalities are often not applicable to domain specific applications
  - Eclipse/NetBeans address experienced software developers
How to create customized update?

- Integrate the out of the box update functionality of the given platform into your application. That lets you test the basic mechanisms and can be done easily with both platforms.
- Then decide, what kind of granularity and complexity you would like to provide in your application.
- Try to reuse some fine granular APIs of the corresponding platform that does the job.
Integrating Help

● NetBeans
  • Using JavaHelp standard
  • Static content only
  • Good & mature help system
  • Help infrastructure is moderate (800 KB)

Eclipse
  • Using web server and indexing/search engine
    • Currently Jetty & Lucene
  • Static and dynamic content
  • Very good & mature help system
  • Help infrastructure is big (7 MB)
NetBeans MP3 Manager Help

User Documentation

This application is meant to be just a demo, but has at least some functionality:

- You could add directories of your file system, using the "Add music folder" menu or toolbar button. These directories are scanned for mp3 files.
- The application provides 2 views on all found mp3 files:
  - A physical file system view
  - A logical view, containing artists, albums, tracks and song titles
- By double clicking a title (or mp3 file in the file system view) an editor is opened
  - You can edit and save some of the id3 tags like artist, album, title, track, year, genre, etc.
Eclipse MP3 Manager Help

User Documentation

This application is meant to be just a demo, but has at least some functionality:

- You could add directories of your file system, using the "Add music folder" menu or toolbar button. These directories are scanned for mp3 files.
- The application provides 2 views on all found mp3 files:
  - A physical file system view
  - A logical view, containing artists, albums, tracks and song titles
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Long Running Operations & User Interaction

• Eclipse:
  • Jobs API
  • Many utilities for asynchronous long running operations in the platform
  • Interactive & Cancelable

NetBeans:
  • Progress API
  • Interactive & Cancelable
Deployment

• Both IDEs provide the deployment of the whole application to the local file system
  • NetBeans as ZIP file & Web-Start
  • Eclipse as directory structure, can be zipped

• Java Web Start
  • NetBeans IDE supports direct deployment for Java Web Start
  • Eclipse RCP apps can be made ready for Java Web Start with some manual configuration

• Native installers (e.g. NSIS) can be used in conjunction with both
Licensing

- **Eclipse RCP**
  - Eclipse Public License (EPL)
  - ICU4J license

**NetBeans Platform**

- Dual licensed
  - Common Development and Distribution License (CDDL)
  - GPL v2 with Classpath Exception
    - The Classpath exception allows you to link an application available under any license to a library that is part of software licensed under...
Eclipse RCP Documentation

- Lots of information at www.eclipse.org
- Good RCP Wiki
- Highly recommended book:
  Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications (The Eclipse Series) by Jeff McAffer and Jean-Michel Lemieux
NetBeans Platform Documentation

- Lots of information (Articles, Tutorials, etc.) at platform.netbeans.org
- Highly recommended book:

  Rich Client Programming: Plugging into the NetBeans Platform
  by Tim Boudreau, Jaroslav Tulach, and Geertjan Wielenga
Outline

● MP3 Manager: A demo application
● Software Architecture
● Component Model & Module Concept
● UI Toolkits & Customization
● Starting Application Development
● Project Structure
● Actions
● Extension Points & Lookups
● Update Functionality & Help System
● Misc
● Conclusion
Conclusion

• Both platforms provide
  • Module system with
    • Dependency management
    • Dynamic modules
    • Module-private classpaths
  • Service infrastructure and lazy loading support
  • Mature UI toolkits with huge widget sets
  • Very good docking system
  • Update support
  • Support for interactive, long running operations
  • Integration of help system
  • And MUCH MORE!
But the Question is…

- Which platform is the better one?
- Which platform should you use?

Recommendation: Get the requirements for YOUR rich client application first! There might be non-functional requirements like scalability, extensibility, reliability, usability and so on as well as functional requirements. After prioritizing the requirements, make your platform choice.

Both platforms Eclipse RCP and NetBeans Platform offer you a lot and help you to build better Java
THANK YOU

Kai Tödter, Siemens AG
Geertjan Wielenga, Sun Microsystems

ID#, Misc., 16 pt.