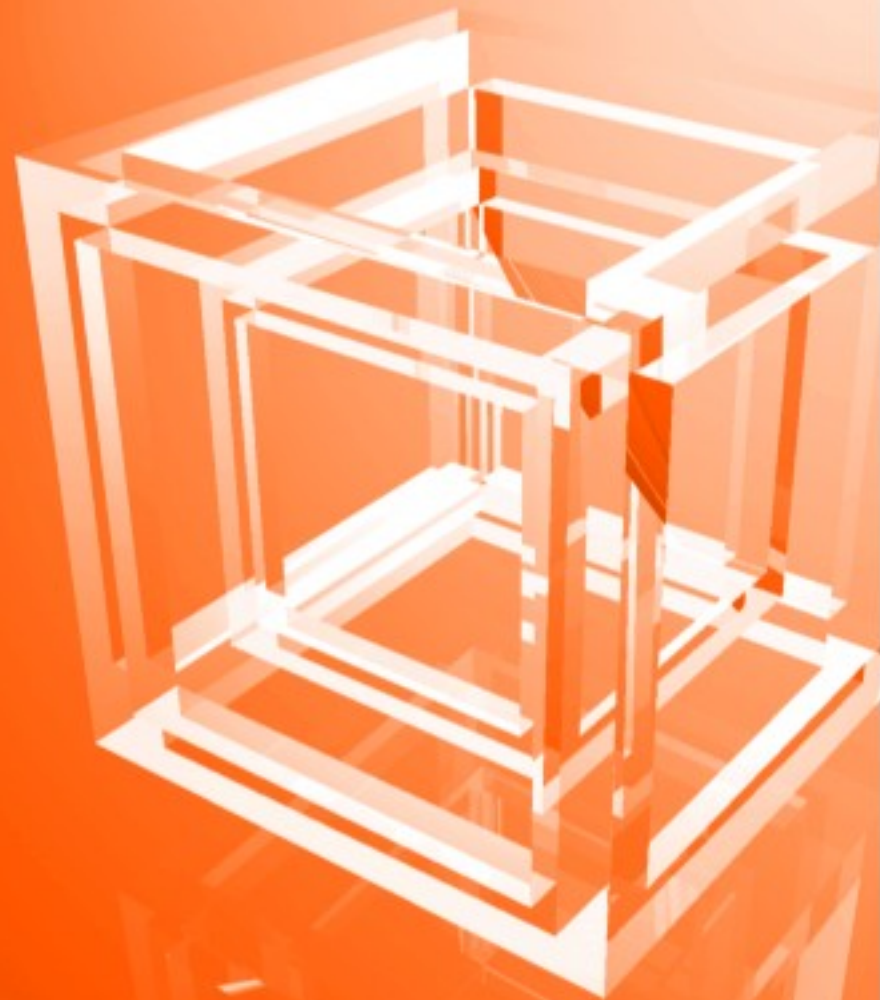


NetBeans Platform

Jaroslav Tulach

NetBeans Platform Architect

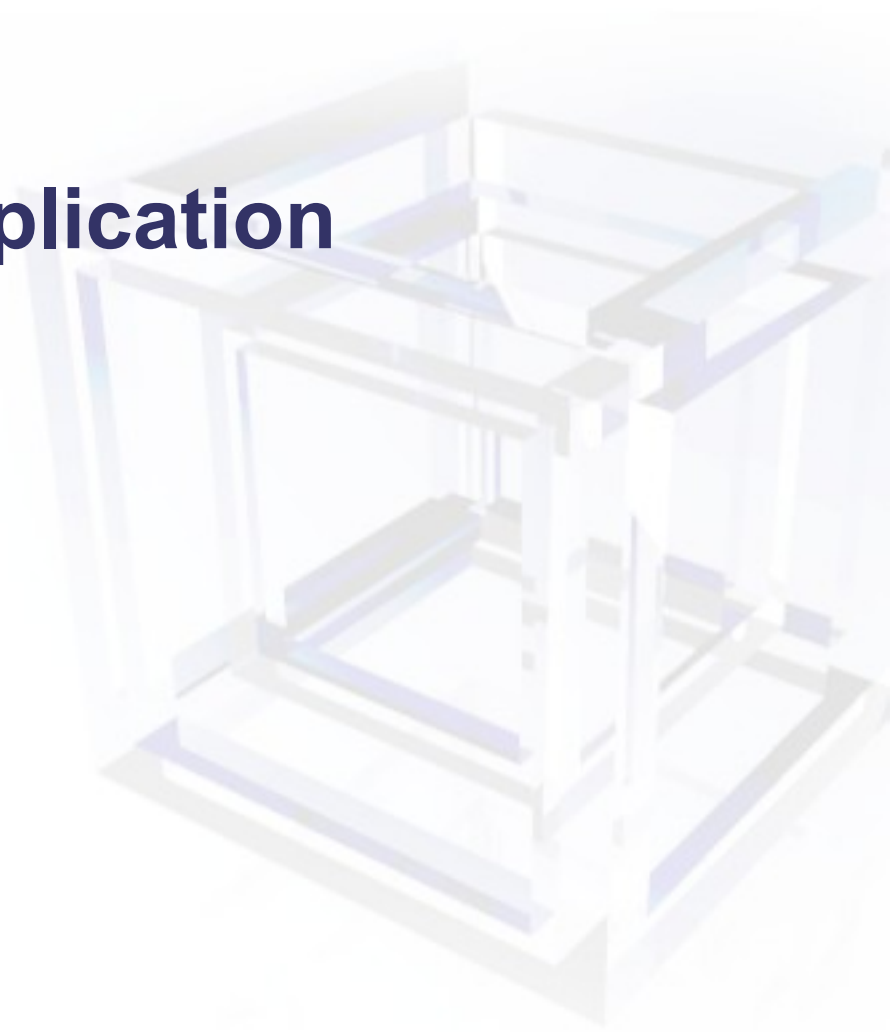


It is Easy and it is Fun!

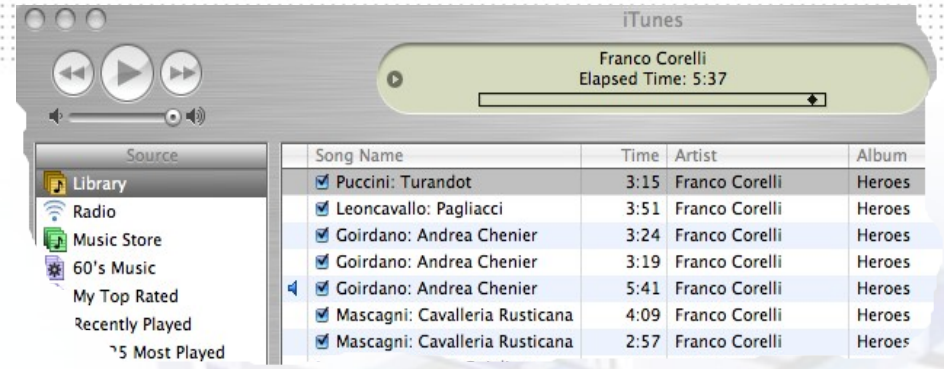
Writing applications on top of NetBeans Platform is easy and productive. It is going to prevent reinvention of the wheel.

Agenda

- **Why NetBeans Platform?**
- **Converting an Existing Application**
- **NetBeans is a Lego**
- **NetBeans Patterns**
- **Questions and Answers**



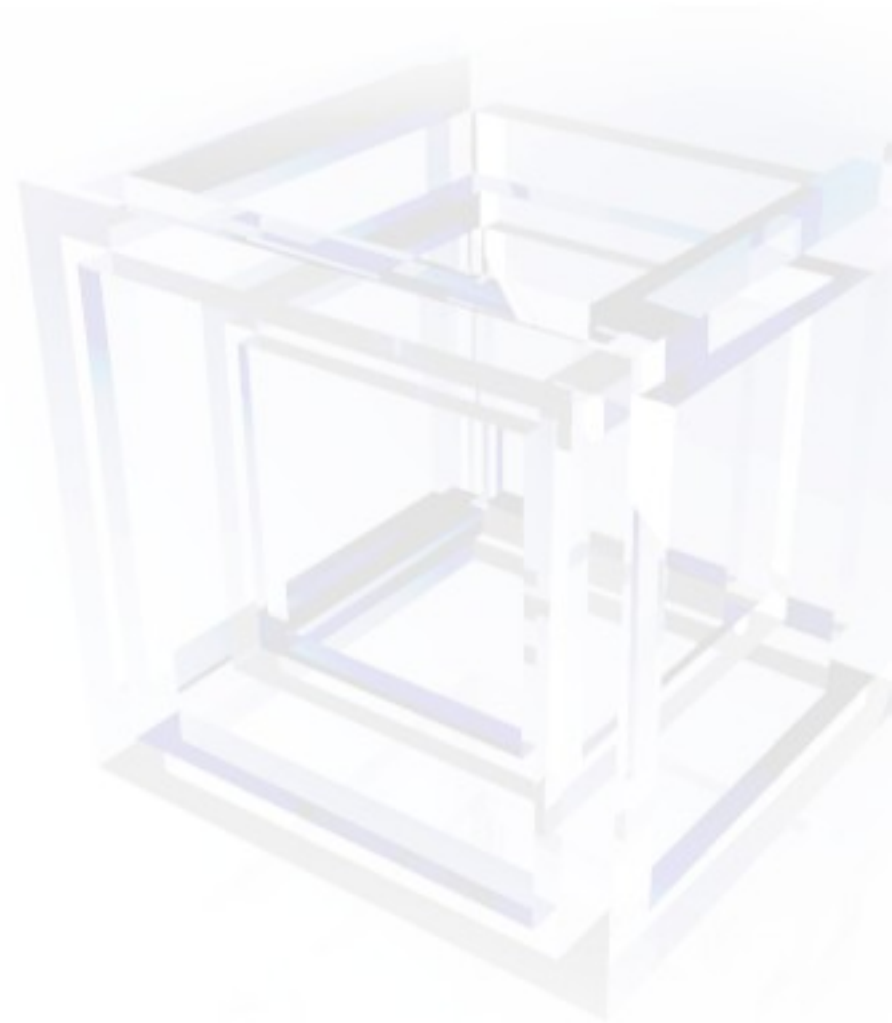
The Need For Rich Desktop Clients



- Web will not do it all
 - Real time interaction (dealing, monitoring)
 - Integration with OS (sound, etc.)
- 100% Java counts
 - Ease of administration and distribution
- NetBeans Platform
 - The engine behind NetBeans IDE

Why NetBeans Platform?

- **People want to write applications**
- **Not:**
 - Windowing System
 - A help set indexer
 - Menu, toolbar code, etc.
- **Choices:**
 - Plain Old Swing
 - Eclipse RCP
 - NetBeans Platform



Everything ends up being a Framework!



Rest In Piece Home-made Frameworks 1995-2005

Costs Savings

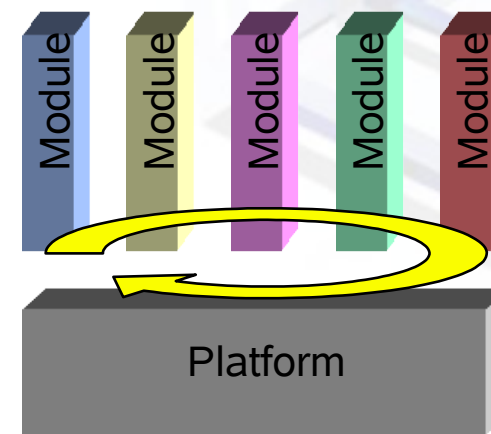
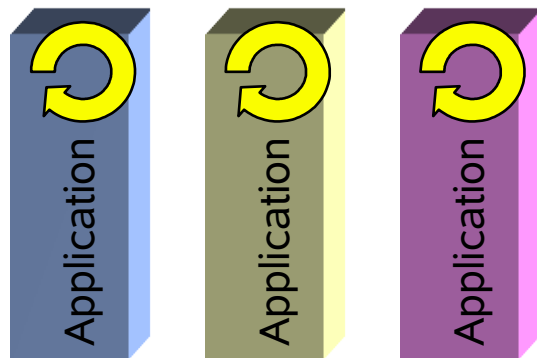
The advantages of reusing NetBeans platform

- Avoid reinventing the wheel
 - Windowing, Menu, Toolbars, etc.
 - Modularization, Branding, Localization
- Concentrate on guts only
 - Write business logic, reuse the framework
- Assemble Applications from Components
 - A lot of existing components on netbeans.org
 - Designed in modular way

Modularity of Components

Important for good UI application

- Discipline: Developers write cleaner code
- Final assembly of application is independent
- Solutions tailored to customer needs
- Much tighter UI task flow



Modularity in NetBeans

Modify your application to be NetBeans module

- Module is any JAR file with enhanced manifest

```
Manifest-Version: 1.0
```

```
OpenIDE-Module: org.netbeans.modules.text/1
```

```
OpenIDE-Module-Specification-Version: 1.13
```

- What is this good for?
 - Identification of each library
 - Version specification
 - Dependencies

Modularity in NetBeans

Dependencies between modules

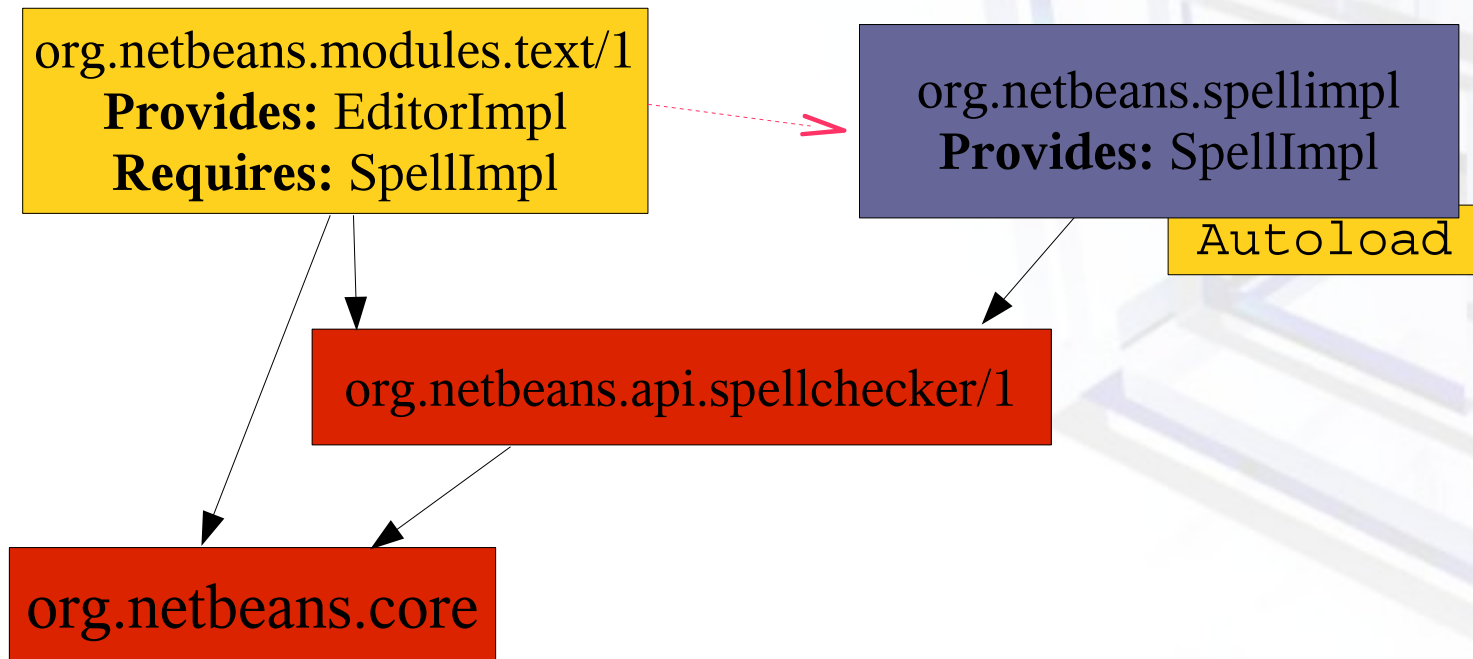
```
OpenIDE-Module-Specification-Version: 1.13
OpenIDE-Module-Provides: EditorImpl
OpenIDE-Module-Module-Dependencies:
    org.netbeans.api.spellchecker/1 > 1.3,
    org.netbeans.core > 4.32
OpenIDE-Module-Requires: SpellImpl
```

- Public packages
- Types of modules
 - Regular
 - Autoload
 - Eager

Modularity in NetBeans

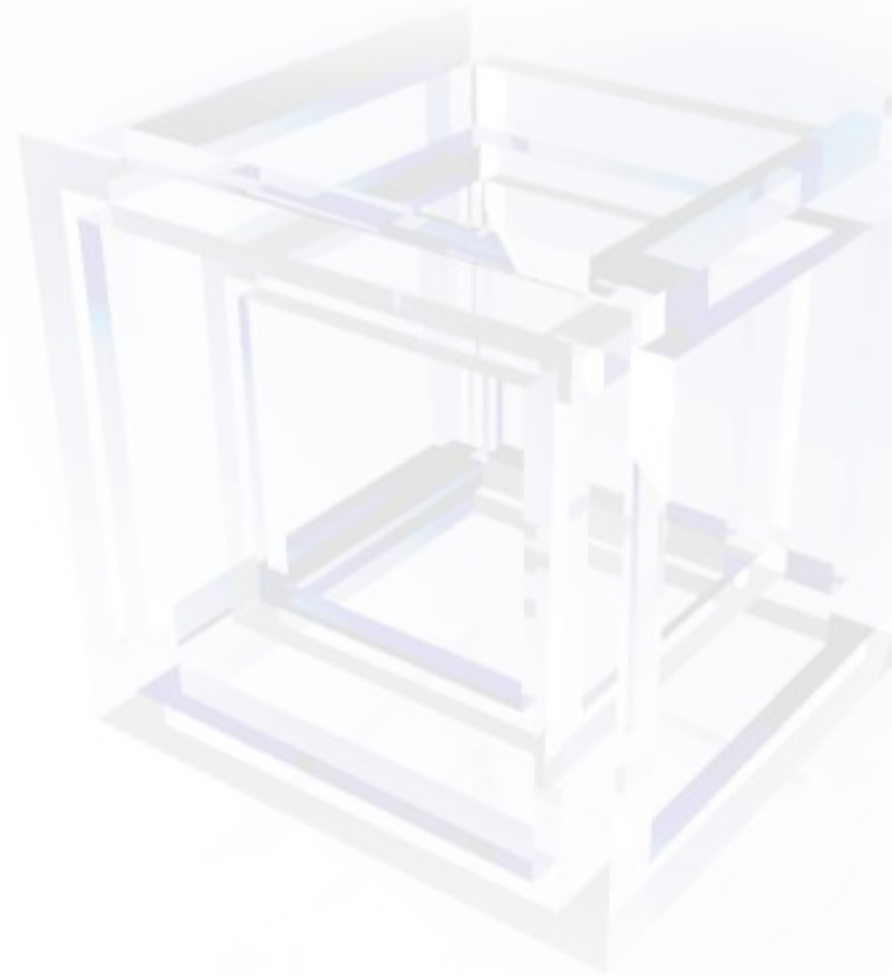
Module Enablement and ClassLoader Hierarchy

- Dependencies influence runtime classpath
- A module can turn other modules on



Convert Hello World

Demo



Levels of Compliance

0: Launchable

- Enhance your manifest
- Use dependencies between your JARs
- Register a menu item



Levels of Compliance

1: Integrated

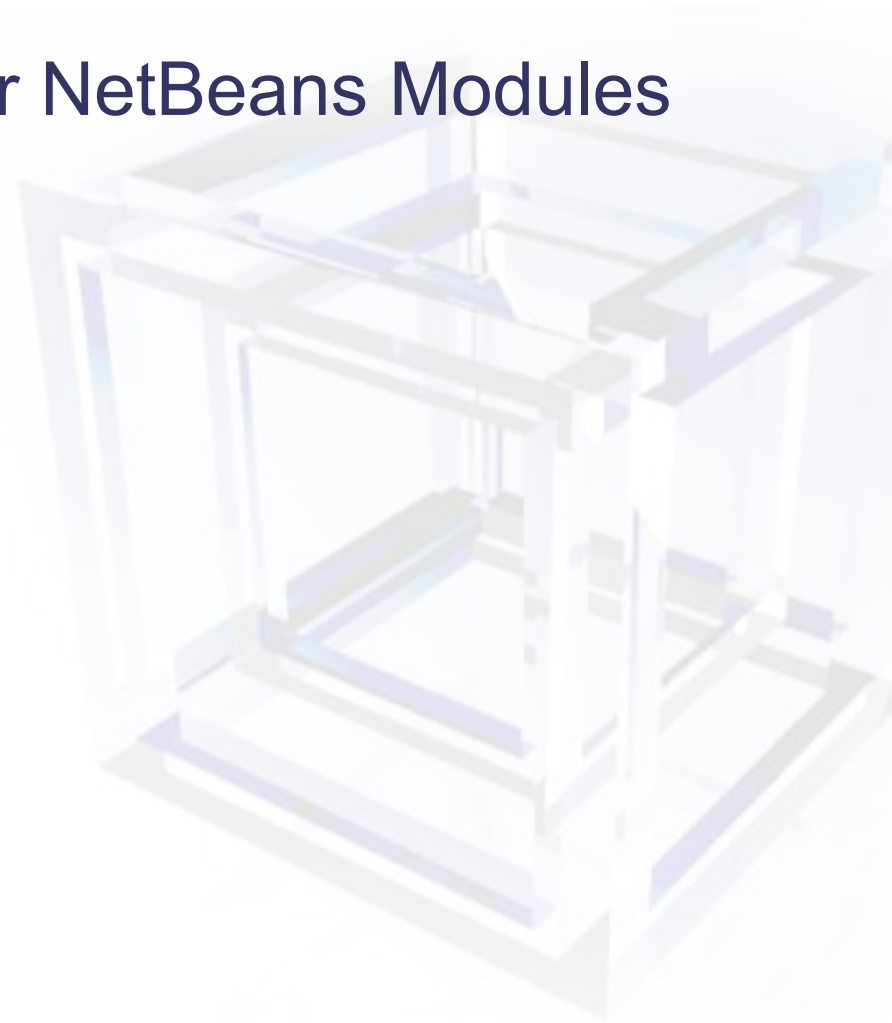
- Integrate the application visually
- Use NetBeans APIs
 - TopComponent
 - DialogDisplayer
- Revise the initialization code of the application
- About 1% of the existing code.



Levels of Compliance

2: Use Case Support

- Bind Your Application to Other NetBeans Modules
- Easier Work flow
- Listening to Global Selection
- Modest Effort
- Brings New Functionality



Intermezzo: Cooperation of Modules

Exposing module state

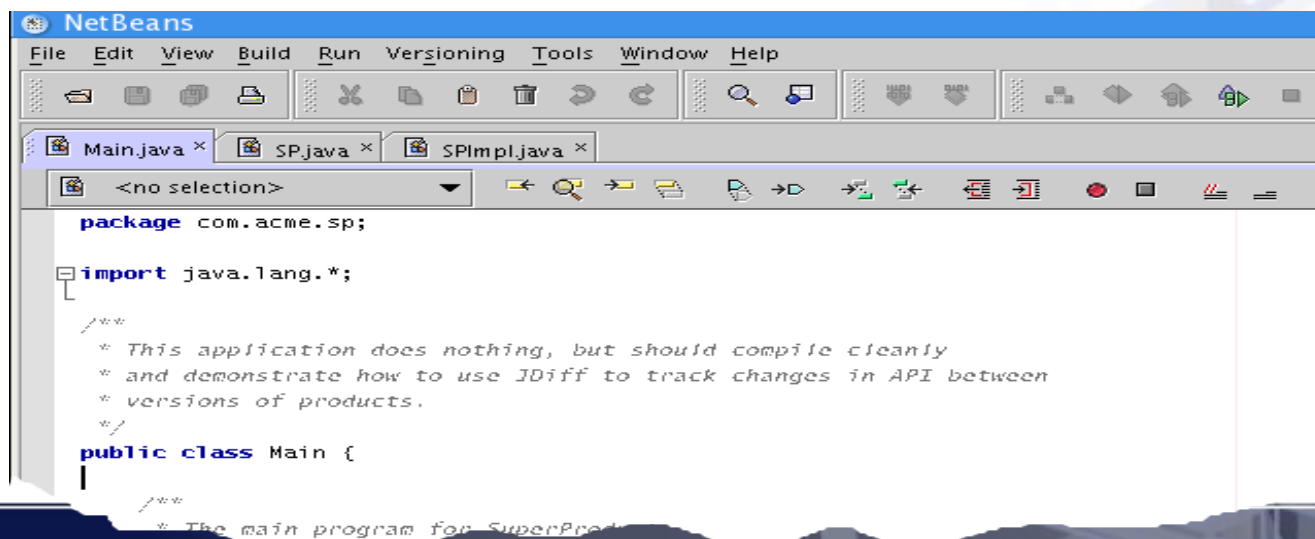
- A module owns a Window, another an action
- Windows can have associated context
- Selected window defines the global context
- UI Elements update their state according to it
- What is inside context?
 - Anything important (javax.swing.Document, etc.)
- What operations does a context need?
 - Changes of its content
 - Observability

Intermezzo: Cooperation of Modules

Exposing Window State

```
class MyWindow
extends org.openide.windows.TopComponent {
    private JEditorPane pane;

public org.openide.util.Lookup getLookup () {
    return Lookups.singleton(pane.getDocument());
}
}
```



Intermezzo: Cooperation of Modules

Querying and Listening to State

```
import org.openide.util.Utilities;

class MyInsertAction extends AbstractAction {
    public void actionPerformed (ActionEvent ev) {
        Lookup lkp = Utilities.actionGlobalContext();
        Document doc = lkp.lookup (Document.class);
        if (doc != null) {
            doc.insertString ("Hello world!", null, 0);
        }
    }
}
```

In similar way one can listen to changes in Lookup using `LookupListener`.

Final Level of Compliance

3: Aligned

- Expose Your State to Other NetBeans Modules
- Eliminate Duplicated Functionality
- Cooperate
- Adapt to NetBeans
- Brings Full Compliance



NetBeans is a Lego

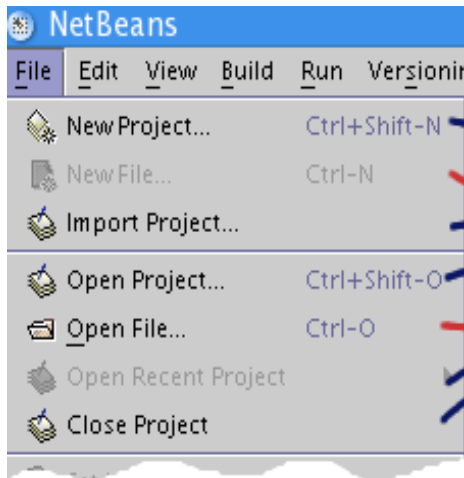
Composition of UI Elements

- Menu, toolbar elements
 - Get merged together by the NetBeans framework
- Windows Layout
- Registration solved by Layers

```
<folder name="Menu" >
  <folder name="File" >
    <file name="Open.instance" >
      <attr name="instanceCreate"
        newvalue="org.openide.actions.OpenAction" />
    </file>
  </folder>
</folder>
```

NetBeans is a Lego

Composition of Menu



```
<folder name="Menu/File">
  <file name="NewProject.instance" />
  <file name="ImportProject.instance"
/>
  <file name="Separator.instance" />
  <file name="OpenProject.instance" />
  <file name="OpenRecent.instance" />
  <file name="CloseProject.instance" />
</folder>
```

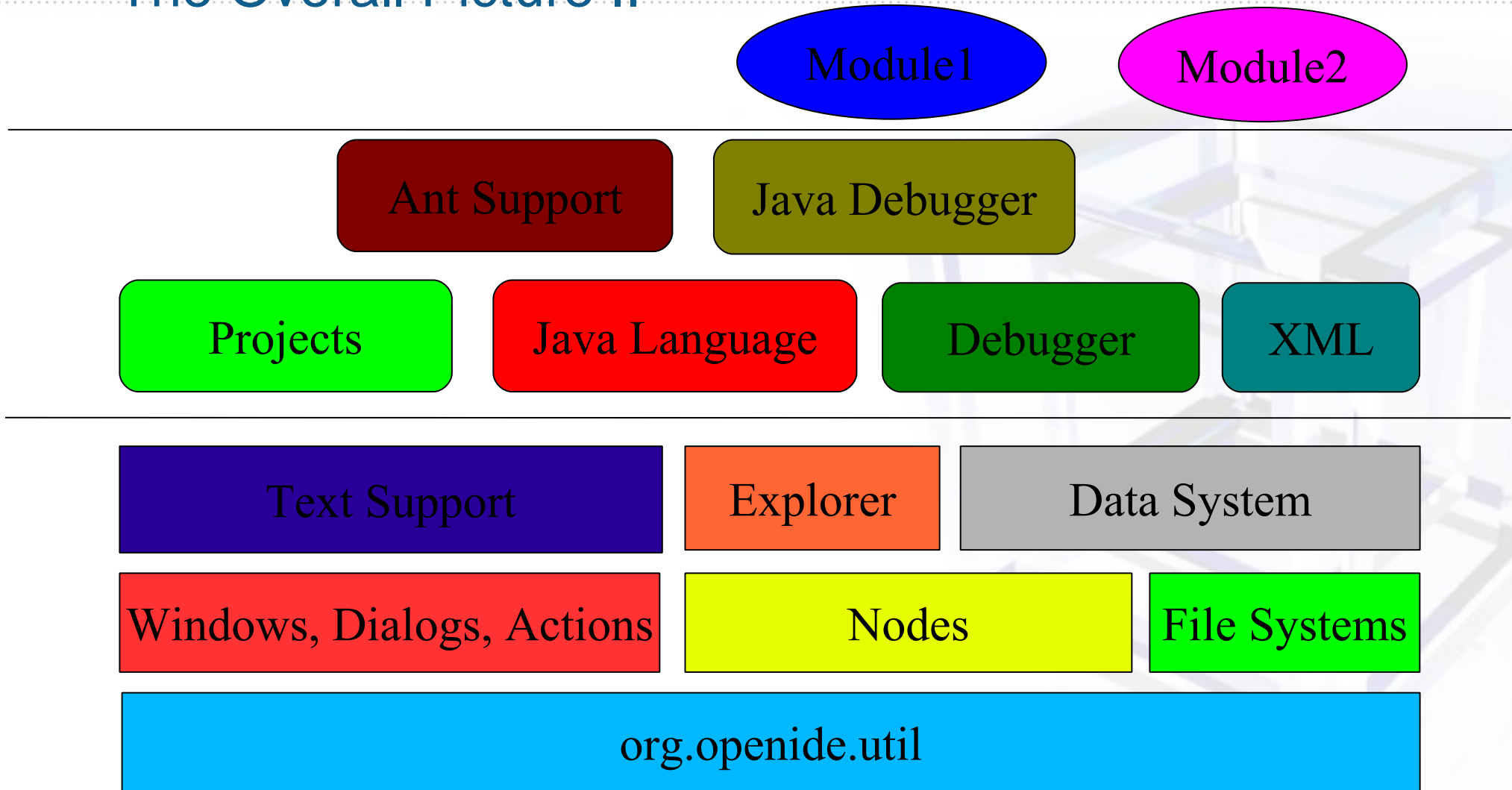
```
<folder name="Menu/File">
  <file name="NewFile.instance" />
  <file name="Open.instance" />
</folder>
```

```
<folder name="Menu/File">
  <attr name="NewProject.instance/NewFile.instance" boolvalue="true" />
  <attr name="NewFile.instance/ImportProject.instance" boolvalue="true"
/>
  <attr name="OpenProject.instance/Open.instance" />
  <attr name="Open.instance/OpenRecent.instance" />
</folder>
```

Windows, Toolbars, etc. work in the same way

NetBeans is a Lego

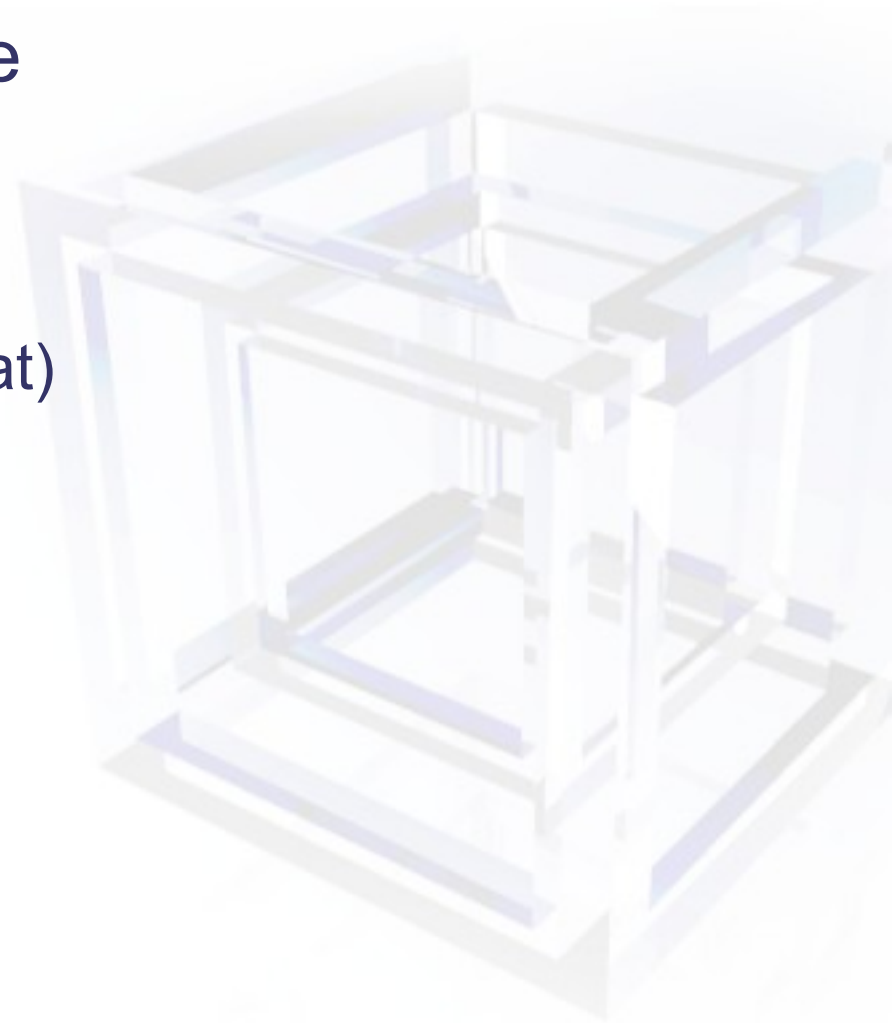
The Overall Picture II



Branding and Customization

Build on NetBeans

- Provide an additional module
 - suitable for IDE extensions
- Build own application
 - subsets of a functionality (jswat)
 - completely different than IDE
 - suite

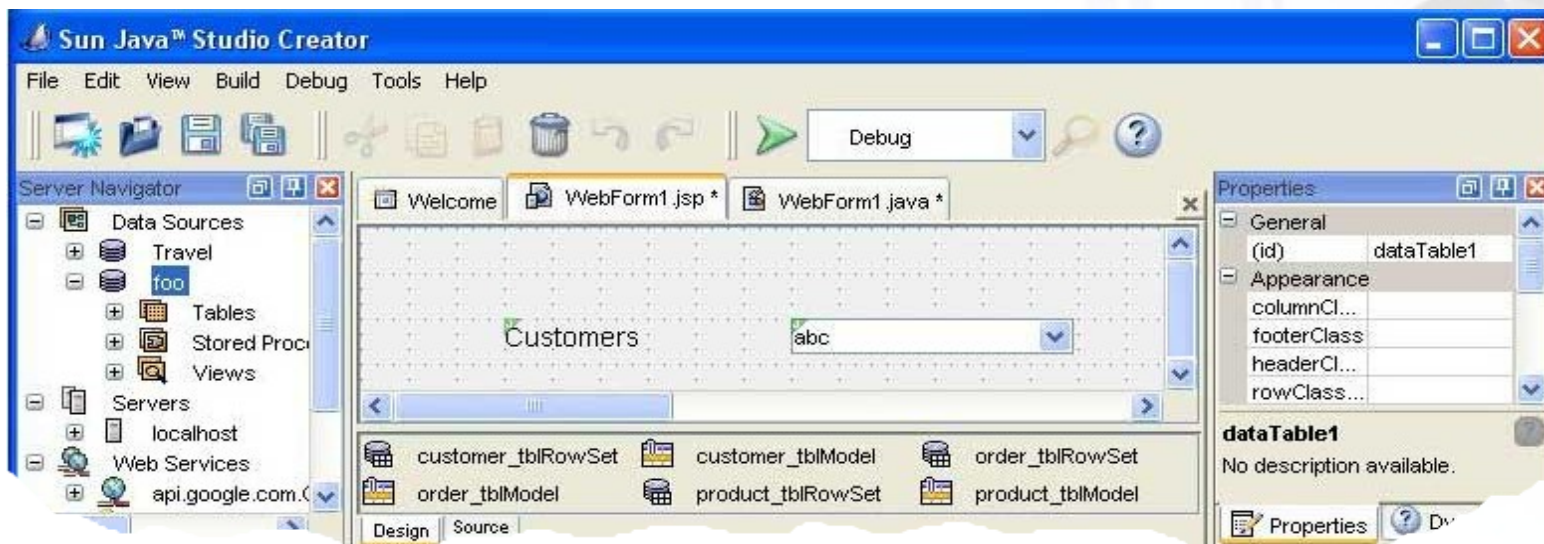


Branding and Customization

- Sun products are branded, L10N and A11Y
- Customization by selection of modules
- Overriding or masking resources

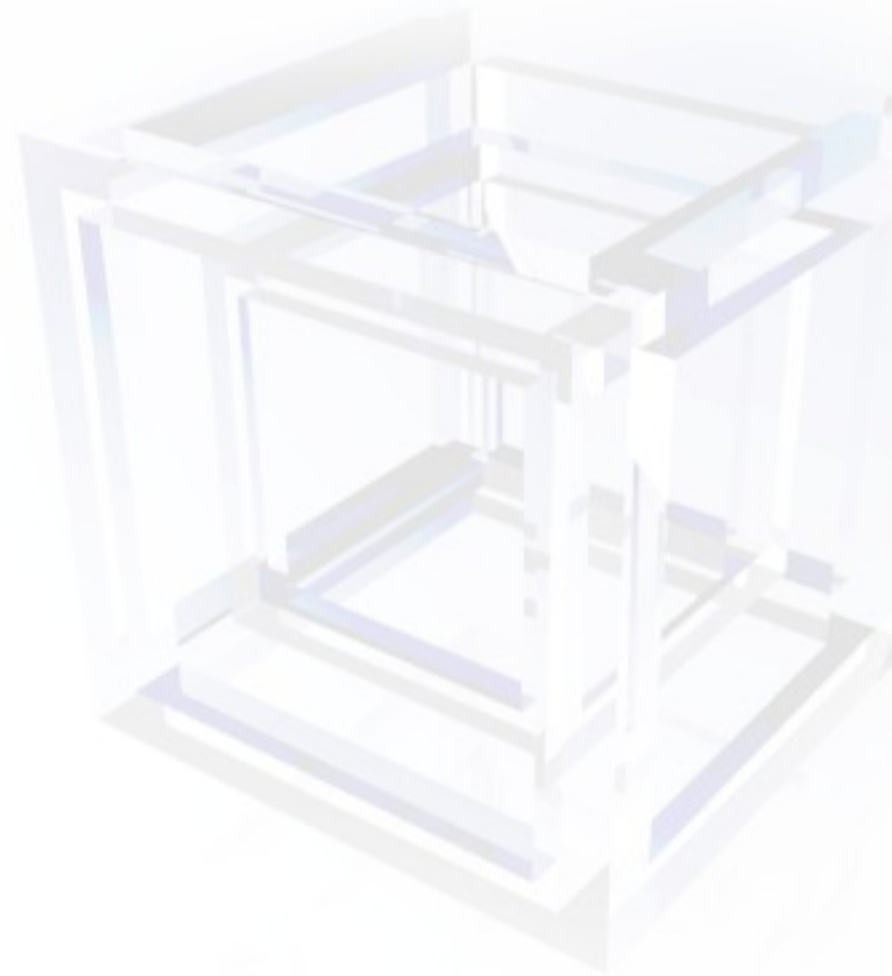
```
<file name="Open.instance" />
```

```
<file name="Open.instance_hidden" />
```



Demo Branding and Distribution

Demo



NetBeans Design Pattern

Write your own Extension Point

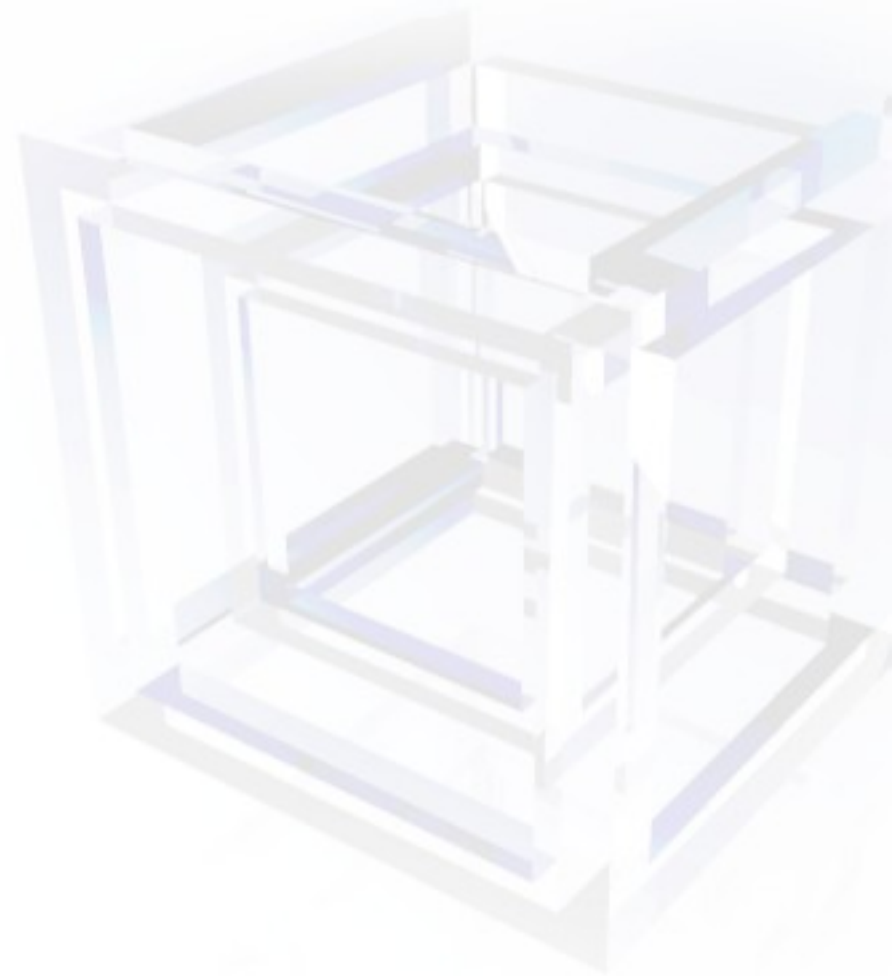
- Module exports an interface
- Uses Lookup to get all implementations
- Other modules registers in META-INF/services
- JDK standard - java.util.Services

```
Lookup lkp = Lookup.getDefault();
Lookup.Template t = new Lookup.Template(YourAPI.class);
Lookup.Result res = lkp.lookup(t);
for (yourAPI : res.allInstances()) {
    yourAPI.doWhatYouWant();
}
res.addLookupListener(new LookupListener() { ... });
```

http://www.netbeans.org/download/5_0/javadoc/usecases.html

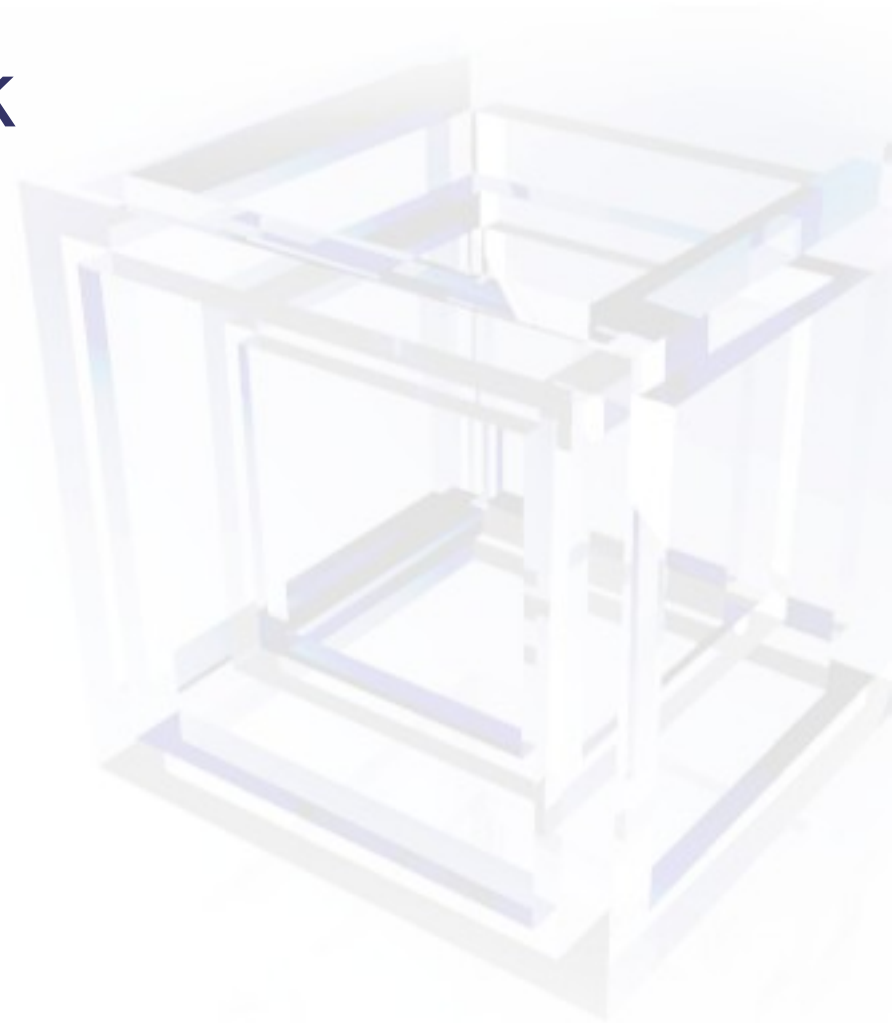
Use Extension Point in Hello World

Demo



NetBeans Platform

- Swing based Framework
- Modular Design
- Lego like Composition
- Years of experience
- Joy to use



References

- Platform Page
 - <http://platform.netbeans.org>
- Geertjan's blog
 - <http://blogs.sun.com/roller/page/geertjan>
- API Documentation
 - <http://www.netbeans.org/download/dev/javadoc/>
- Mailing lists
 - dev@openide.netbeans.org
 - nbdev@netbeans.org



Q & A

