

Consumer IDE



**Jiri Rehtacek
Geertjan Wielenga
Sun Microsystems**

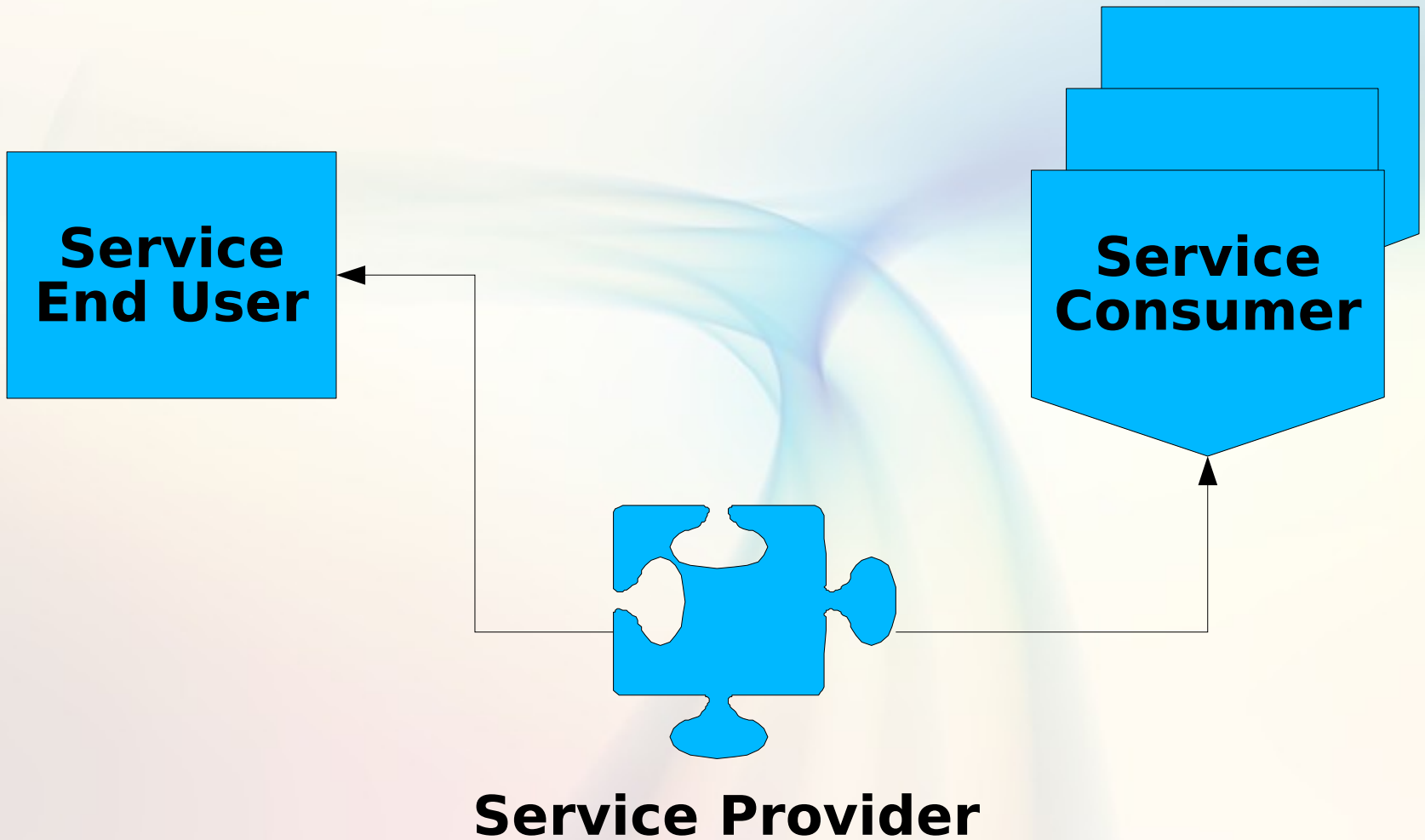
Agenda

- Introduction
- Problem Statement
- Demo 1: End user
- Demo 2: Service Consumer
- Demo 3: Service Provider
- Demo 4: Special Feature on Demand Tools
- Conclusion
- Q/A

Problem Statement

- Choice
- Download size
- Startup time
- Performance

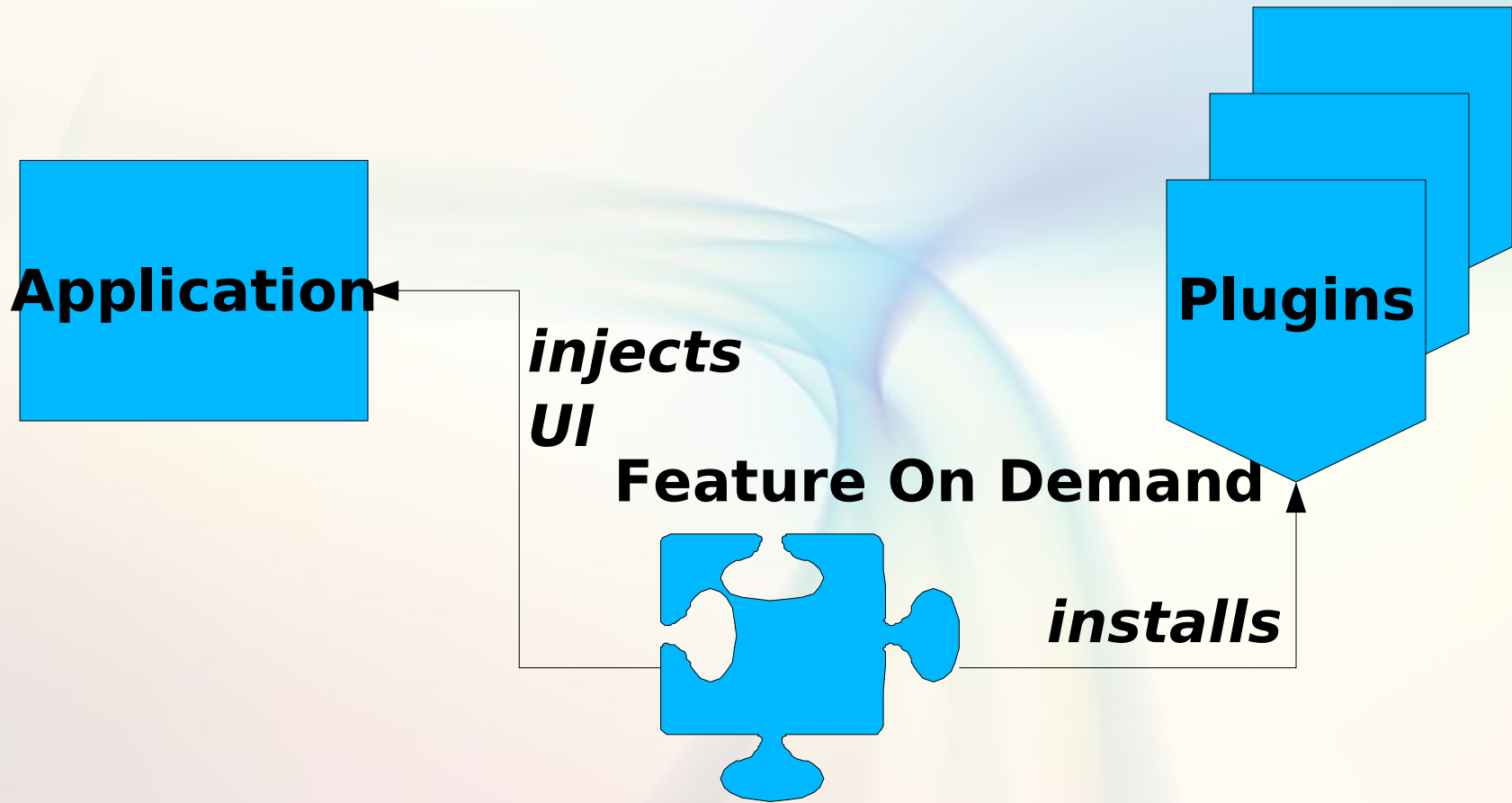
Actors



Demo 1: Service end user

- Show Feature on Demand in Action
 - > Concept can be used in multiple applications on the NetBeans Platform
 - > Example: Maven in NetBeans IDE
 - > Open Maven project without Maven
 - > Create new Maven project without Maven
 - > Example: VisualVM
 - > GlassFish plugin

Demo 1: Service end user (cont.)

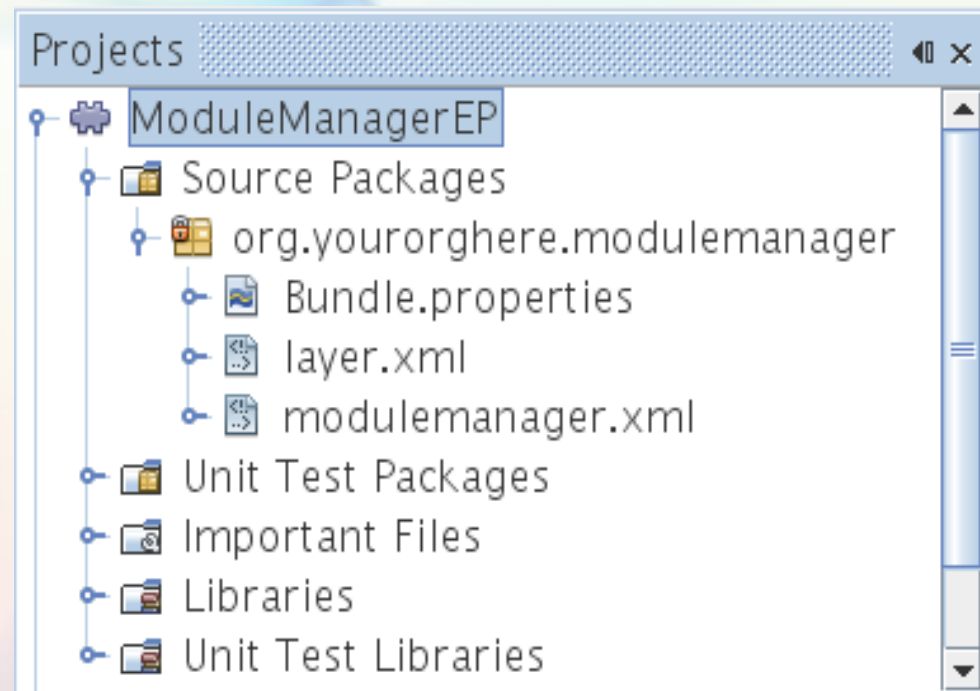


Demo 2: Service consumer

- Service consumer wants to:
 - > provide their own module, which will be installed on demand
 - > promote a different module, which will be installed on demand
- Service consumer doesn't want to:
 - > learn about the application's infrastructure

Demo 2: Service consumer

- > Entrypoint exists
- > Developer uses XML file and layer to let service end user install modules on demand



Demo 2: Service consumer

- How to consume an entry point
 - > Know the concept
 - > Know how to declare ui in XML
 - > Know public methods in API
 - > Know how to declare in layer.xml

Entry Point Registration

```
1. <filesystem>
2.   <folder name="VisualVM">
3.     <folder name="ExplorerPopupSelection">
4.       <file name="org-netbeans-modules-consumerentrypoints-GlassFishApplicationTypeAction.instance">
5.         //Call to the API:
6.         <attr name="instanceCreate"
7.           methodvalue="org.netbeans.modules.consumervisualvm.api.ApplicationTypeAction.newAction"/>
8.         //Application type for which the menu item will be shown:
9.         <attr name="mainClassName" stringvalue="com.sun.enterprise.server.PELaunch"/>
10.        //Code name base of plugin in update center:
11.        <attr name="pluginCodeName" stringvalue="net.java.visualvm.modules.glassfish"/>
12.        //Which bundle:
13.        <attr name="SystemFileSystem.localizingBundle"
14.          stringvalue="org.netbeans.modules.consumerentrypoints.resources.Bundle"/>
15.        //Which key:
16.        <attr name="ActionName" stringvalue="org-netbeans-modules-consumerentrypoints
17.          GlassFishApplicationTypeAction" />
18.      </file>
19.    </folder>
20.  </folder>
21. </filesystem>
```

Layer Registration

```
1. <filesystem>
2.   <folder name="ConsumerVisualVM">
3.     <file name="glassfish.instance">
4.       //API class, holds the delegate layer and name:
5.       <attr name="instanceCreate"
6.         methodvalue=
7.         "org.netbeans.modules.consumervisualvm.api.PluginInfo.create"/>
8.       //Presence of this plugin determines whether delegate is injected:
9.       <attr name="codeName"
10.        stringvalue="net.java.visualvm.modules.glassfish"/>
11.      //Registers the delegate layer:
12.      <attr name="delegateLayer"
13.        urlvalue="nbresloc:/org/netbeans/modules/consumerentrypoints
14.        resources/glassfish.xml"/>
15.    </file>
16.  </folder>
17.</filesystem>
```

Demo 3: Service provider

- How to provide an entry point
 - > Developer creates own entry point
 - > Example: MIME-specific Action

Demo 3: Service provider (cont.)

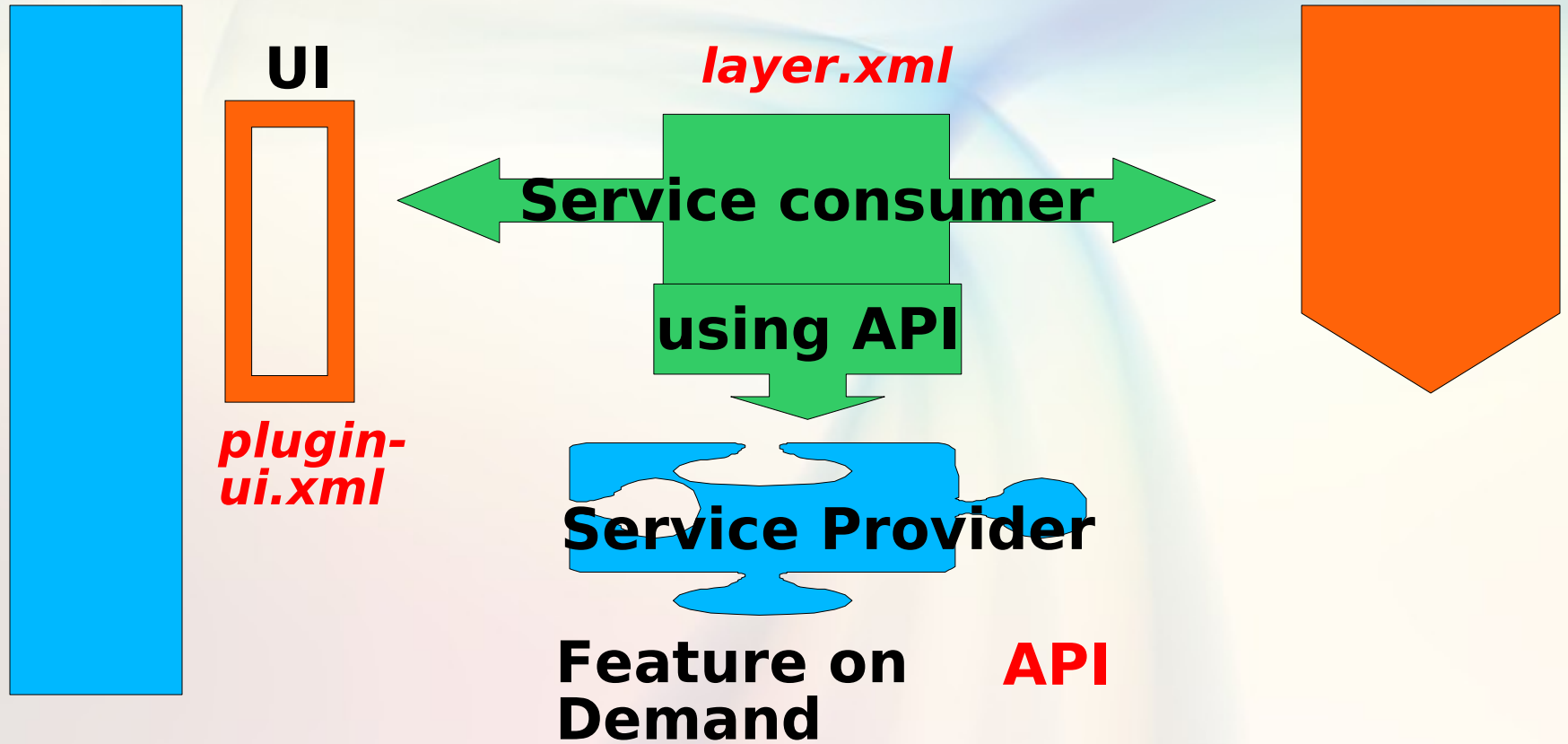
```
public void open () {
    WizardDescriptor.Iterator<WizardDescriptor> iterator = new InstallMissingModulesIterator ()
    WizardDescriptor wizardDescriptor = new WizardDescriptor (iterator);
    wizardDescriptor.setModal (false);
    wizardDescriptor.setTitleFormat (new MessageFormat (NbBundle.getMessage (InstallMissingDisplayer.class, "InstallMissingDisplayer_Title")));
    wizardDescriptor.setTitle
    wizard = DialogDisplayer.getDefault ().createDialog (wizardDescriptor);
    wizard.setVisible (true);
    wizard.addWindowListener(new WindowListener () {
        public void windowOpened (WindowEvent e) {}
        public void windowClosing (WindowEvent e) {
            wizard = null;
        }
        public void windowClosed (WindowEvent e) {
            wizard = null;
        }
        public void windowIconified (WindowEvent e) {}
        public void windowDeiconified (WindowEvent e) {}
        public void windowActivated (WindowEvent e) {}
    });
}
```

Demo 3: Service provider (cont.)

- How to achieve own MIME-Type-specific Action?

Application

Plugin



Demo 3: Service provider (cont.)

- Declare FeatureInfo in layer.xml

```
<file name="enable-i18n-hyperlinking.instance">
  <attr name="instanceCreate" methodvalue="org.
    [...]FeatureInfo.create"/>
  <attr name="codeName"
    stringvalue="org.netbeans.modules.editor.hyperlinking.i18n"/>
  <attr name="delegateLayer" urlvalue="nbresloc:/org/
    [...]resources/enable-i18n-hypelinking-layer.xml"/>
</file>
```

Demo 3: Service provider (cont.)

- Declare Own Action

```
<folder name="Actions">
  <folder name="Tools">
    <file name="I18N-Action.instance">
      <attr name="instanceCreate"
methodvalue="org.netbeans.spi.actions.support.Factory.delegate"
/>
      <attr name="delegate"
methodvalue="org.netbeans.modules.autoupdate.featureondemand
.api.Factory.newAction"/>
      <attr name="SystemFileSystem.localizingBundle"
stringvalue="org.[...].resources.Bundle"/>
    </file>
  </folder>
</folder>
```


Demo 3: Service provider (cont.)

- Declare **Java Source** specific action in NetBeans editor

```
<folder name="Editors">
  <folder name="text">
    <folder name="x-java">
      <folder name="Popup">
        <file name="I18N-Action.shadow">
          <attr name="originalFile"
stringvalue="Actions/Tools/I18N-Action.instance"/>
        </file>
      </folder>
    </folder>
  </folder>
</folder>
```

Demo 3: Service provider (cont.)

- Feature On Demand API **Factory** class

```
public final class Factory {  
    private Factory() {}  
    // declared in Service Consumer declaration  
    public static ActionListener newAction(FileObject fo) {  
        return new FeatureAction (fo);  
    }  
    ...  
}
```

Demo 3: Service provider (cont.)

- Implementation of FeatureAction

```
public class FeatureAction implements ActionListener, Runnable {
    public FeatureAction(FileObject fo) {this.fo = fo;}
    public void actionPerformed(ActionEvent e) {
        RequestProcessor.getDefault().post(this);
    }
    public void run() {
        URL url = // XML layer
        FoDFileSystem.getInstance().getDelegateFileSystem(fo);
        String codeName = // plugin code name
        Feature2LayerMapping.getInstance().getCodeName(url);
        ModulesInstaller.installModules (codeName);
    }
}
```

Demo 4: Feature on Demand Tools

- Entry Point Consumer Wizard
 - > Template that generates all the files.
 - > Everything generated in two clicks.
 - > Installs into NetBeans Platform.
 - > Each entry point might need a wizard.

Conclusion

- When: Today
- Where: NetBeans sources/contrib
<http://hg.netbeans.org/main/contrib>
 - > autoupdate.featureondemand
 - > featureentries
 - > spi.actions.support
- More info:
 - > <http://wiki.netbeans.org/FeatureOnDemar>
 - > <http://blogs.sun.com/geertjan/>
 - > <http://blogs.sun.com/rechtacek/>

- Questions?