NetBeans and NetBeans Platform
Overview

• History
  > originally MFF student project (Xelfi)
• IDE
  > Java, C/C++, PHP, Python,...
• Platform
  > rich clients development
  > Swing
Sources

• NetBeans source code
  > http://www.netbeans.org/downloads/zip.html

• API Javadoc
  > http://bits.netbeans.org/dev/javadoc/index.html

• Planet NetBeans
  > http://planetnetbeans.org/

• Numerous NetBeans bloggers
  > e.g. https://blogs.oracle.com/geertjan/
Extending NetBeans

• Possibilities
  > single module
  > suite of modules
  > standalone application
    > like a suite of modules
  > wrapper module of an existing JAR
Extending NetBeans

• Possibilities
  > single module
  > suite of modules
  > standalone application like a suite
  > wrapper module of an existing JAR
Single module creation

Steps
1. Choose Project
2. **Name and Location**
3. Basic Module Configuration

**Name and Location**
- **Project Name:** Example Module
- **Project Location:** /home/hnetynka/devel/NB2014
- **Project Folder:** /home/hnetynka/devel/NB2014/ExampleModule

**Standalone Module**
- **NetBeans Platform:** Development IDE

**Add to Module Suite**
- **Module Suite:**

Controls:
- **Back**
- **Next**
- **Finish**
- **Cancel**
- **Help**
Suite & standalone application

- **Suite**
  > set of modules that have to be loaded together
- **Standalone application**
  > same as the suite
  > configured to be run as a standalone application
## Dependencies

![Project Properties - StandaloneApp1](image)

**Categories:**
- Sources
- Libraries
- Application
- Installer
- Hints

**Java Platform:** JDK 1.8 (Default)
- Manage Java Platforms...

**NetBeans Platform:** Development IDE
- Manage Platforms...

### Platform Modules

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Included</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>apisupport</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>enterprise</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>extide</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>harness</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>ide</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>java</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>javafx</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>mobility</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>nb</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>php</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>platform</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>profiler</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>webcommon</td>
<td></td>
<td>Platform</td>
</tr>
<tr>
<td>websvccommon</td>
<td></td>
<td>Platform</td>
</tr>
</tbody>
</table>

---

⚠️ Module JavaFX wrapper in platform requests the token org.openide.modules.jre.JavaFX but there are no known providers.

![Resolve](image) ![Add Project...](image) ![Add Cluster...](image) ![Remove](image) ![Edit...](image)

[OK] [Cancel] [Help]
Branding application

[Screen shot of a branding application setup window in NetBeans IDE]
Executing application/module

- Run
  - executes new instance of IDE with installed modules
- Install /Reload in Development IDE
  - runs module in the development instance of IDE
    - no new instance is executed
  - available for standalone modules only
Distribution

- Modules ~ NBM files
  - common JAR file
  - with extra info in its manifest
- Standalone apps
  - ZIP files or
  - JNLP application
Converting an existing applications
Generic process

• “Library” without UI => library wrapper

• Application with UI
  > converting the application by parts
    > Swing panel => TopComponent
    > Actions => CallableSystemAction, CallbackSystemAction
    > Menu => NB menu via layer
    > ...
Converting application

• Levels of conformance
  > Level 0: Launchable
    > enhancing the manifest file with NetBeans entries
    > adding dependencies to other modules
    > adding menu item to “launch” the application
  > Level 1: Integration
    > using NetBeans Window system and Dialog API
    > initialization via ModuleInstal or META-INF/services
  > Level 2: Use case support
    > follow NetBeans paradigms
  > Level 3: Aligned
    > reusing as much as possible, cooperating with other modules
Example

• Converting the Anagram Game
  > available as a std example
    > New Project → Samples → Java → Anagram game

• Step 1 – create new module
Step 1

**Name and Location**

- **Project Name:** NBAanagram
- **Project Location:** /home/hnetynka/devel/NB2014
- **Project Folder:** /home/hnetynka/devel/NB2014/NBAanagram

**Standalone Module**

- **NetBeans Platform:** Development IDE

**Add to Module Suite**

- **Module Suite:**
Step 1

Basic Module Configuration

- Code Name Base: cz.cuni.mff.nb.anagram
- Module Display Name: NBAngram
- Localizing Bundle: cz/cuni/mff/nb/anagram/Bundle.properties

Generate OSGi Bundle
Getting Level 0

• Copy classes of the anagram game to our module
• Add new action
• Implement the action to show the anagram game window
Copying game classes

NetBeans
Adding an action
public final class AnagramGameShowAction implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        new Anagrams().setVisible(true);
    }
}
Finished

- Execute new IDE with out module
  > “Run” in the right-click menu

- Pack module as NBM file

- Distribute the module ;-)
Converting an existing applications
Obtaining Level 1
Converting to Level 1

• Using TopComponent and Dialog API
  > JFrame → TopComponent
Process

- Create new TopComponent
  > “Window component”
- Copy Anagram panel to the created window
- Copy local variables
Creating new TopComponent

• Choosing position
  > in which are the component has to appear
Creating new TopComponent

- Name prefix for created classes etc.

![NetBeans New Window dialog](image-url)

- Steps
  1. Choose File Type
  2. Basic Settings
  3. Name, Icon and Location

- Name, Icon and Location
  - Class Name Prefix: AnagramGame
  - Icon: [Browse...]
  - Project: NBAagram
  - Package: cz.cuni.mff.nb.anagram
  - Created Files:
    - src/cz/cuni/mff/nb/anagram/AnagramGameTop
    - src/cz/cuni/mff/nb/anagram/AnagramGameTop
  - Modified Files:
    - manifest.mf
    - nbproject/project.xml
Creating new TopComponent
Copying panel to the window

- Select JPanel in the Anagram class
- Copy it
- Paste it to the TopComponent class

Ctrl + C

Ctrl + V
Copying variable

- Copy variables from the Anagrams class
- Paste them to the TopComponent

public class Anagrams extends JFrame {
    public static void main(String[] args) {
        new Anagrams().setVisible(true);
    }
    private int wordIdx = 0;
}

Ctrl+C

final class AnagramTopComponent extends TopComponent {
    private int wordIdx = 0;
    private static AnagramTopComponent instance;
}

Ctrl+V
Window System
Overview

- **Window system**
  - management of windows (panels) in the NetBeans

- **Basic Elements**
  - **TopComponent**
    - JPanel with additional methods
  - **Mode**
    - in which are the component has to be placed
      - i.e. docking mode
  - **WindowManager**
    - managing state of UI
  - **TopComponentGroup**
    - set of windows that should be activated together
  - **Roles (Perspectives)**
    - switching between window layouts (new in 7.1)

- **UI = Swing**
TopComponent

- open()
- close()
- requestVisible()
- requestActive()
- componentHidden()
- componentShowing()
- componentDeactivated()
- componentActivated()
- componentClosed()
- componentOpened()
TopComponent

• Persisting session across sessions
  > TopComponent implements Externalizable

• Persistence modes
  > PERSISTENCE_ALWAYS
  > PERSISTENCE_NEVER
  > PERSISTENCE_OPENED
TopComponent

- Changing persistence – old style (till 6.5)
  - change ResolvableHelper
    - and writeReplace()
  - default persistence code

```java
public int getPersistenceType() {
    return TopComponent.PERSISTENCE_ALWAYS;
}
/** replaces this in object stream */
public Object writeReplace() {
    return new ResolvableHelper();
}
protected String preferredID() {
    return PREFERRED_ID;
}
final static class ResolvableHelper implements Serializable {
    public Object readResolve() {
        return XTopComponent.getDefault();
    }
}
```
TopComponent

- Persistence – current style
  - annotation @ConvertAsProperties
    - defines public ID of a DTD for the storing file
      - identification of the file
  - methods
    readProperties(Properties p)
    writeProperties(Properties p)
    - reading/saving via them
Mode

- Position in application
- Many predefined
  - editor, navigator, output,....
- Own one can be defined
  - defined by XML
  - new editor in NB 7.1
  - in NB 7.0 and older – no editor available
  - “little hack” for creation
    - launch IDE with module
    - move the component to the desired area
    - exit IDE
    - copy automatically created mode description
Mode

- Position in application
- Many predefined editors, navigator, output, etc.
- Own one can be defined:
  > defined by XML
  > new editor in NB 7.1
  > in NB 7.0 and older – no editor available

- "little hack" for creation:
  - launch IDE with module
  - move the component to the desired area
  - exit IDE
  - copy automatically created mode description
<mode version="2.3">
  <name unique="anonymousMode_1" />
  <kind type="view" />
  <state type="joined" />
  <constraints>
    <path orientation="vertical" number="20" weight="0.7"/>
    <path orientation="horizontal" number="20" weight="0.32"/>
    <path orientation="vertical" number="21" weight="0.2909090909090909"/>
  </constraints>
  <bounds x="0" y="0" width="0" height="0" />
  <frame state="0"/>
  <active-tc id="AnagramTopComponent" />
  <empty-behavior permanent="false"/>
</mode>
Mode

- Opening a component in a particular mode programmatically

```java
public void open() {
    Mode mode = WindowManager.getDefault().findMode("mode");
    if (mode != null) {
        mode.dockInto(this);
    }
    super.open();
}
```
TopComponent groups

- Set of windows that should be activated together
- Defined by file descriptors
  > wsgrp
  > wstcgrp
TopComponent groups

<group version="2.0">
   <module name="org.netbeans.modules.windowgroupsample" spec="1.0" />
   <name unique="MyGroup" />
   <state opened="false" />
</group>

<tc-group version="2.0">
   <module name="org.netbeans.modules.windowgroupsample" spec="1.0" />
   <tc-id id="OneTopComponent" />
   <open-close-behavior open="true" close="true" />
</tc-group>
Roles (Perspectives)

- New in 7.1
- Easy switching between window layouts

- `@TopComponent.Registration(mode = "editor", openAtStartup = true, role="admin")`
- `WindowManager.getDefault().setRole("admin");`