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/

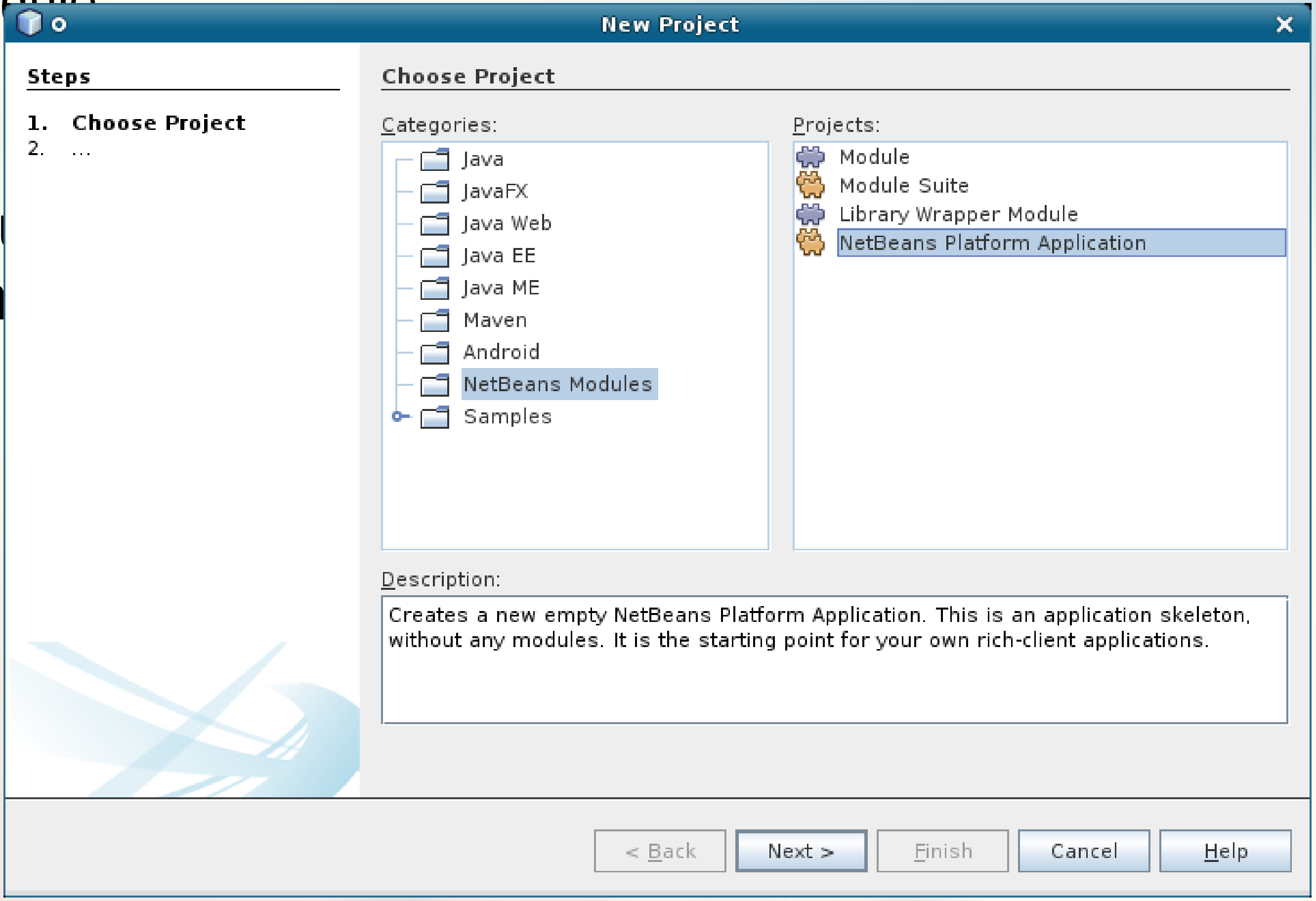
Getting Started with the NetBeans Platform

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 m
  - standalon
    - like a s
  - wrapper m


New Project

Steps

1. Choose Project
2. ...

Choose Project

Categories:

- Java
- JavaFX
- Java Web
- Java EE
- Java ME
- Maven
- Android
- NetBeans Modules
- Samples

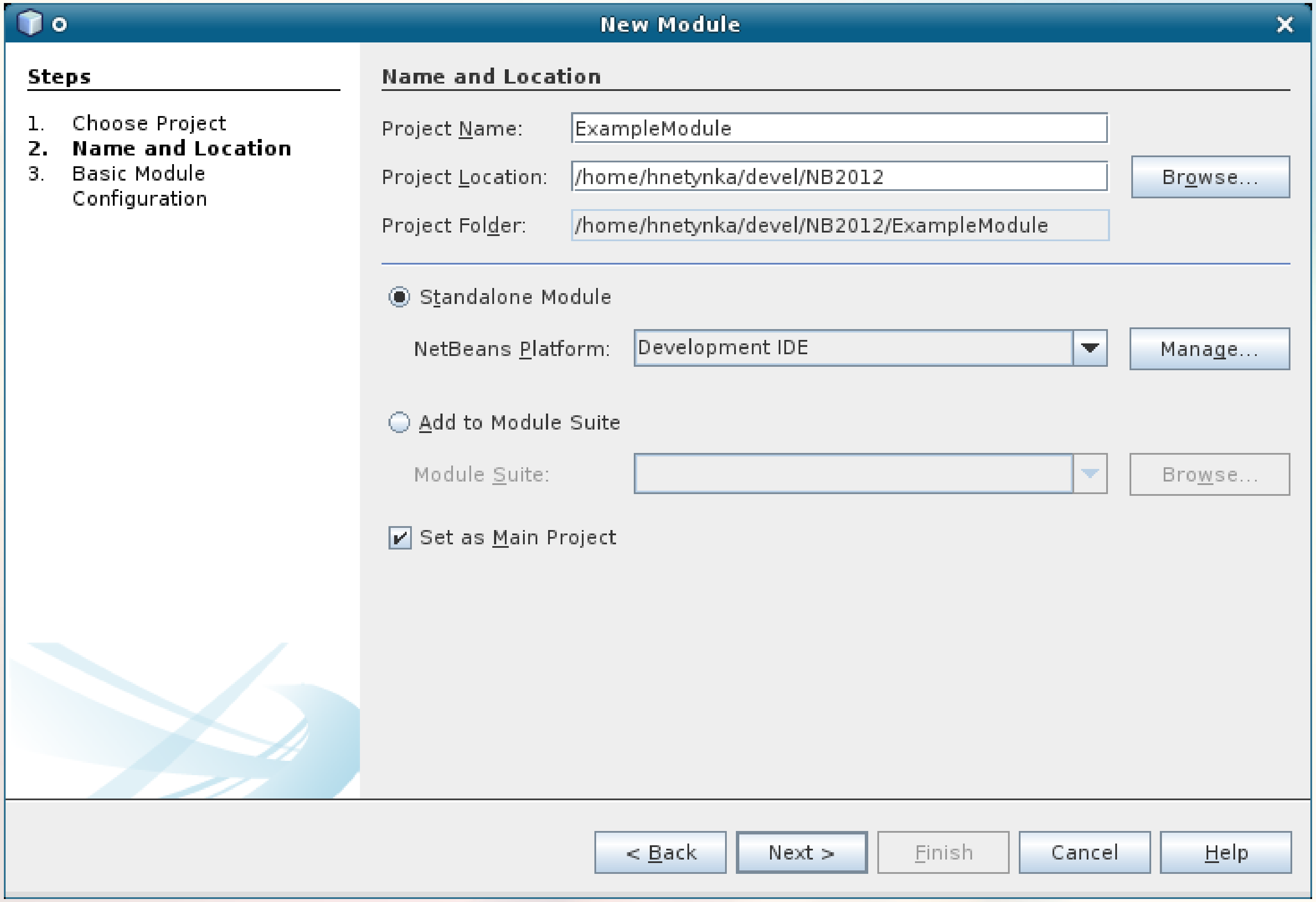
Projects:

- Module
- Module Suite
- Library Wrapper Module
- NetBeans Platform Application

Description:

Creates a new empty NetBeans Platform Application. This is an application skeleton, without any modules. It is the starting point for your own rich-client applications.

Single module creation


New Module

Steps

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

Name and Location

Project Name: ExampleModule

Project Location: /home/hnetynka/devel/NB2012

Project Folder: /home/hnetynka/devel/NB2012/ExampleModule

● Standalone Module

NetBeans Platform: Development IDE

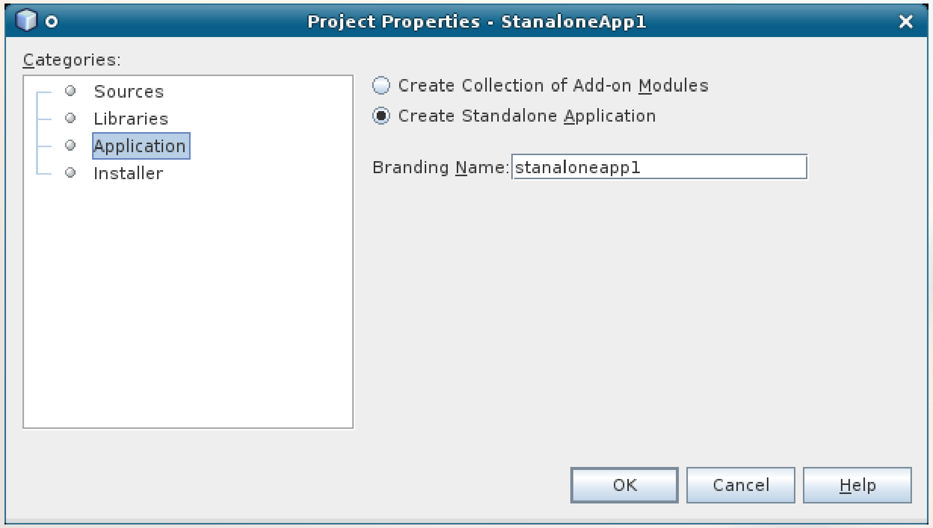
○ Add to Module Suite

Module Suite:

☑ Set as Main Project

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

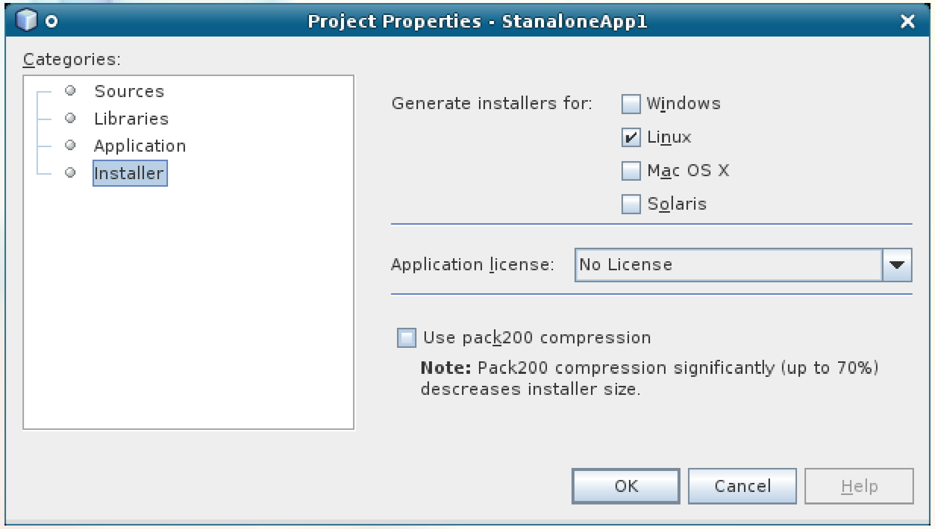


Project Properties - StanaloneApp1

Categories:
- Sources
- Libraries
- Application
- Installer

( ) Create Collection of Add-on Modules
(●) Create Standalone Application

Branding Name: stanaloneapp1



Project Properties - StanaloneApp1

Categories:
- Sources
- Libraries
- Application
- Installer

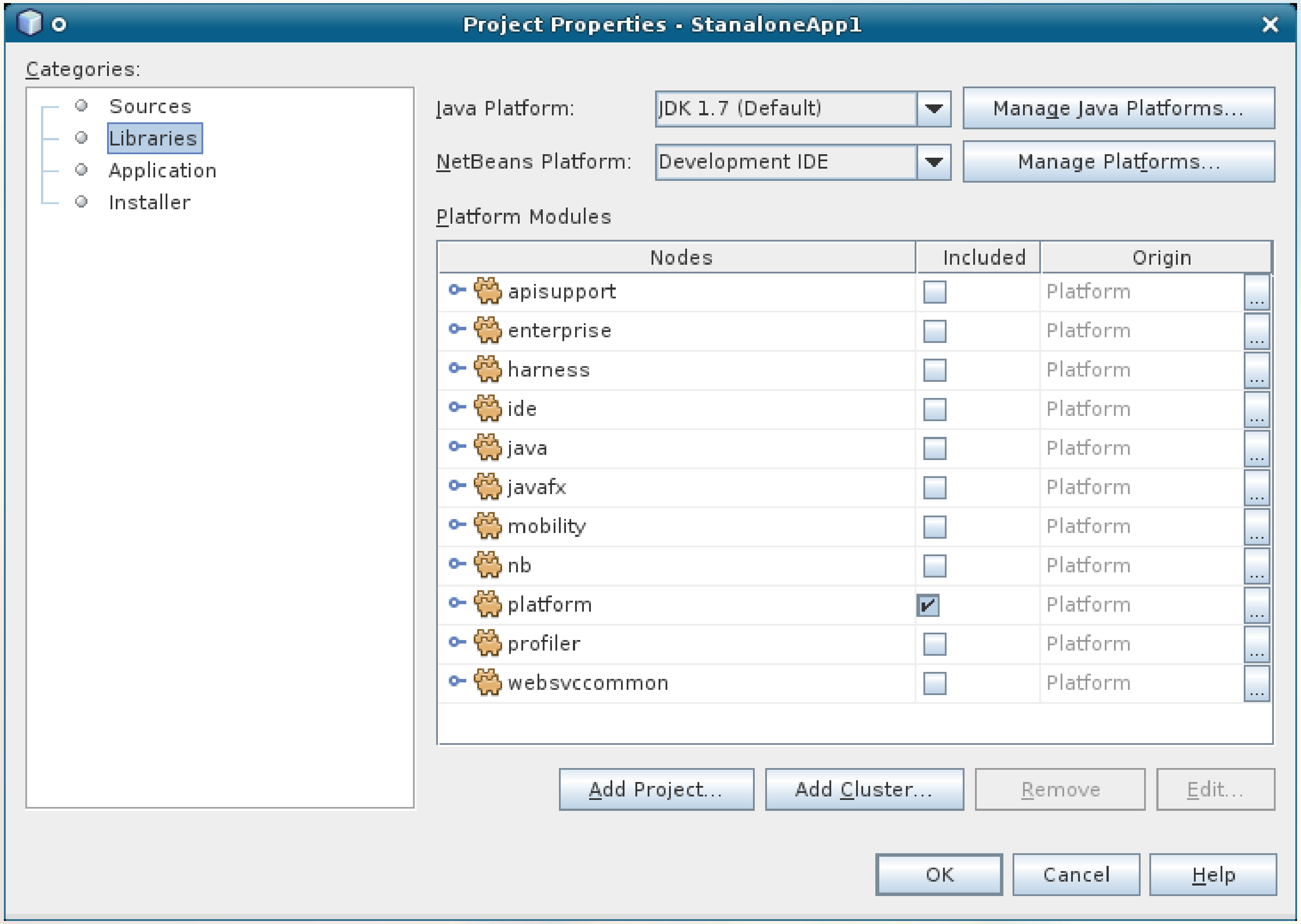
Generate installers for: ☐ Windows ☑ Linux ☐ Mac OS X ☐ Solaris

Application license: No License

☐ Use pack200 compression

Note: Pack200 compression significantly (up to 70%) descreases installer size.

Dependencies



Project Properties - StanaloneApp1

Categories:
- Sources
- Libraries
- Application
- Installer

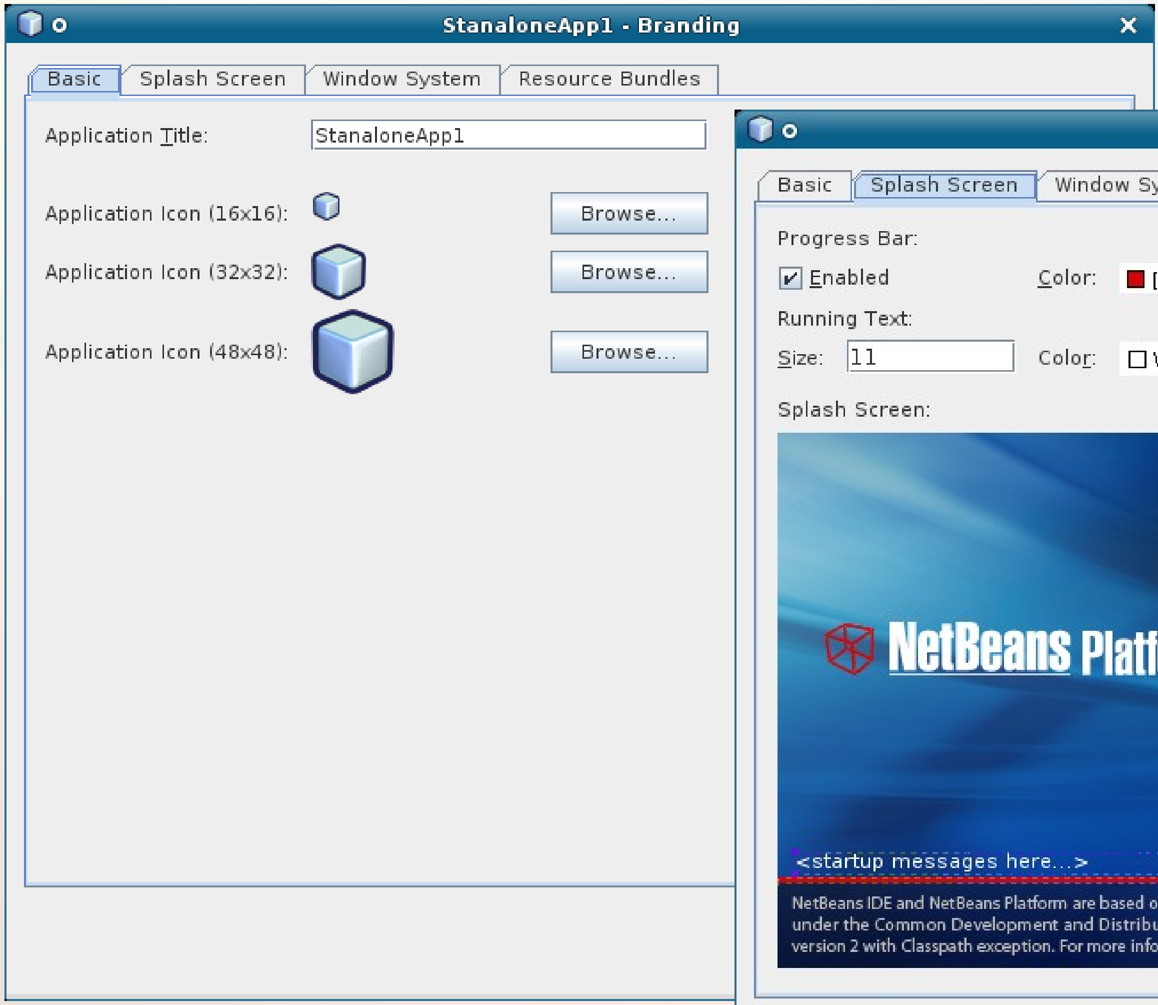
Java Platform: JDK 1.7 (Default) — Manage Java Platforms...

NetBeans Platform: Development IDE — Manage Platforms...

Platform Modules

Nodes	Included	Origin
apisupport	☐	Platform
enterprise	☐	Platform
harness	☐	Platform
ide	☐	Platform
java	☐	Platform
javafx	☐	Platform
mobility	☐	Platform
nb	☐	Platform
platform	✓	Platform
profiler	☐	Platform
websvccommon	☐	Platform

Branding application



StanaloneApp1 - Branding

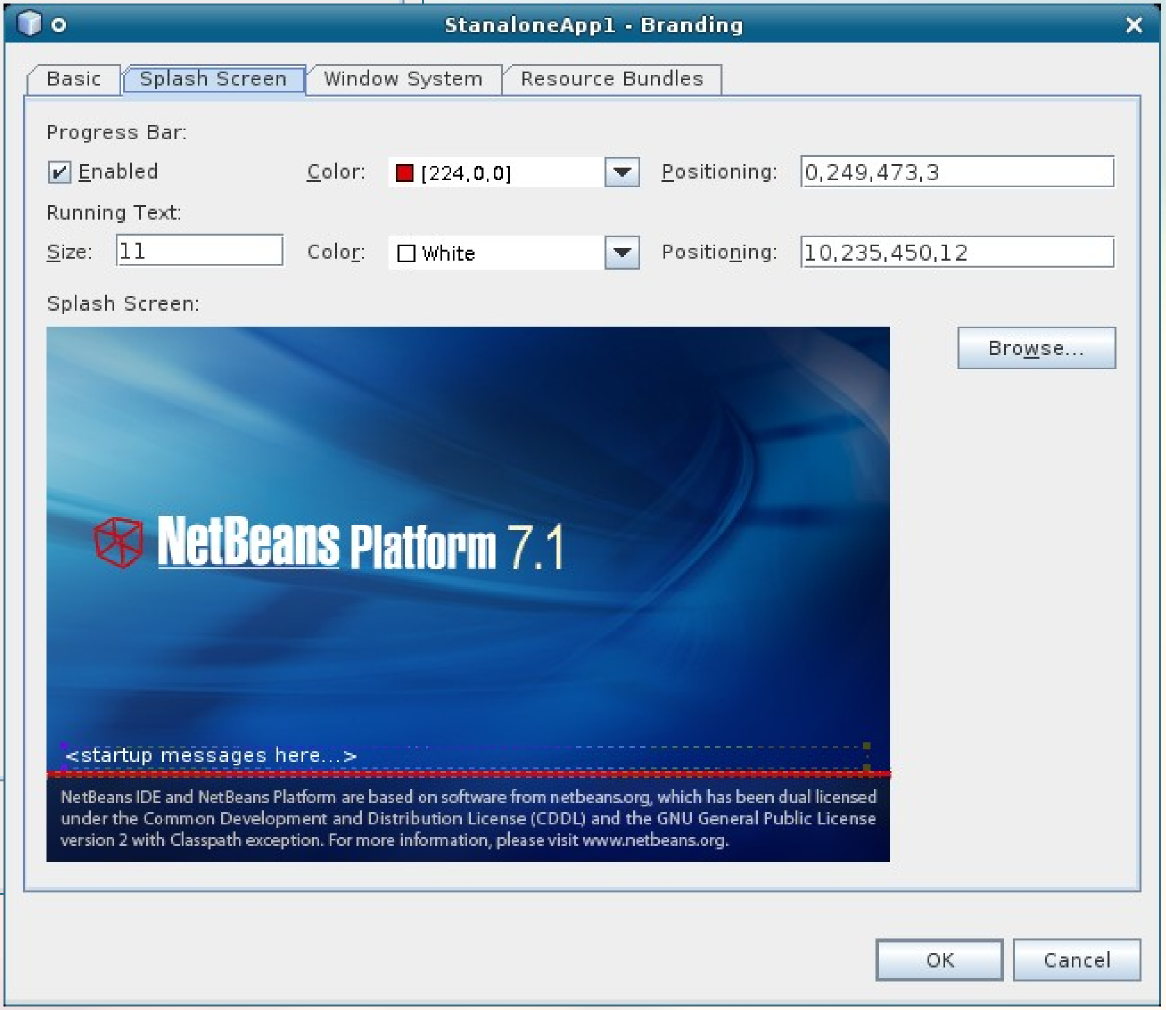
Basic | Splash Screen | Window System | Resource Bundles

Application Title: StanaloneApp1

Application Icon (16x16): Browse...

Application Icon (32x32): Browse...

Application Icon (48x48): Browse...



StanaloneApp1 - Branding

Basic | Splash Screen | Window System | Resource Bundles

Progress Bar:

✓ Enabled   Color: 🟥 [224,0,0]   Positioning: 0,249,473,3

Running Text:

Size: 11   Color: ☐ White   Positioning: 10,235,450,12

Splash Screen: Browse...

NetBeans Platform 7.1

<startup messages here...>

NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.

OK | Cancel

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

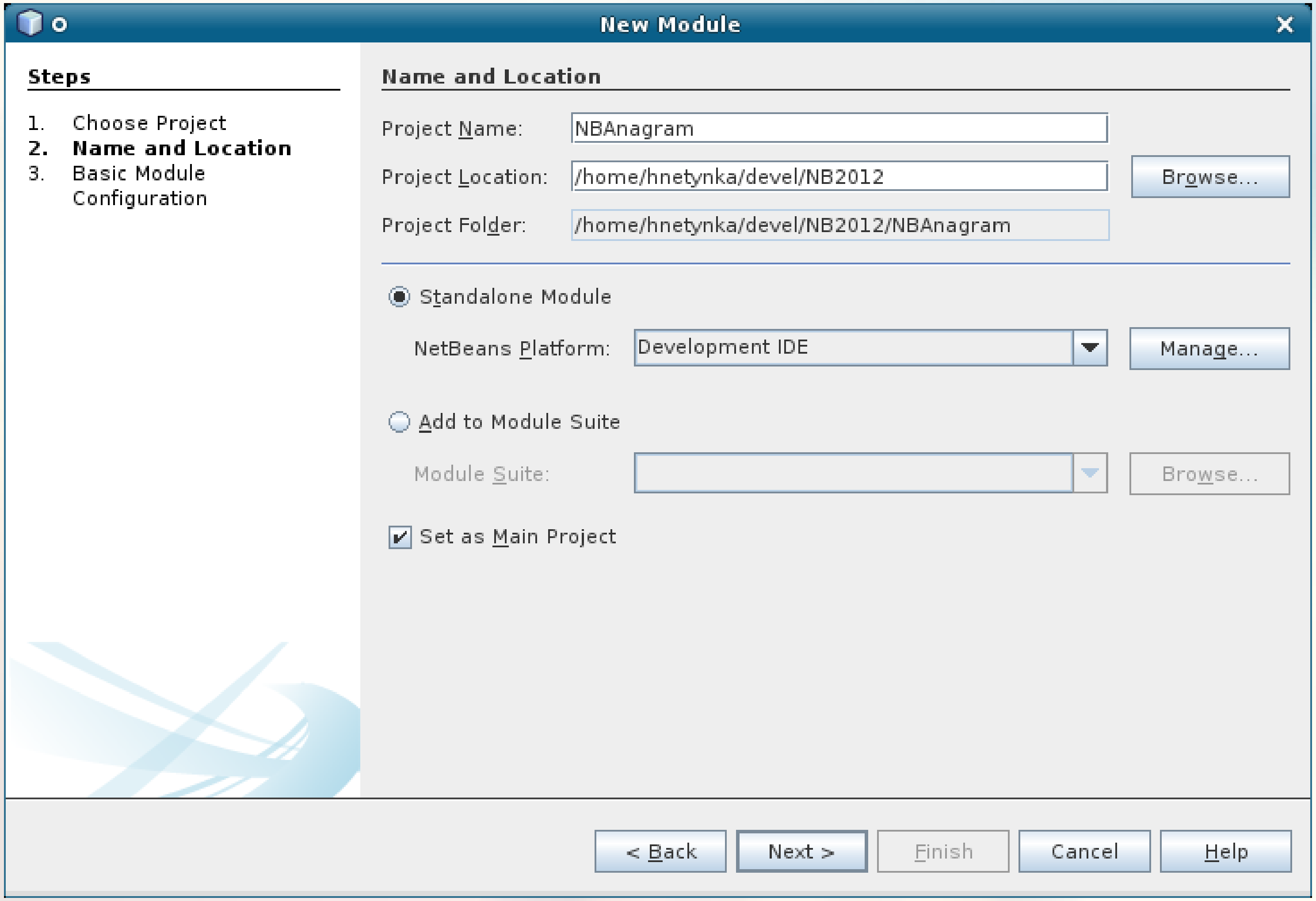
- Levels of conformance
  - Level 0: Launchable
    - enhancing the manifest file with NetBeans entries
    - adding dependencies to other modules
    - adding menu item to “launch” the application
  - Level 1: Integration
    - using NetBeans Window system and Dialog API
    - initialization via `ModuleInstal` 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



New Module

Steps

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

Name and Location

Project Name: NBAnagram

Project Location: /home/hnetynka/devel/NB2012

Project Folder: /home/hnetynka/devel/NB2012/NBAnagram

● Standalone Module

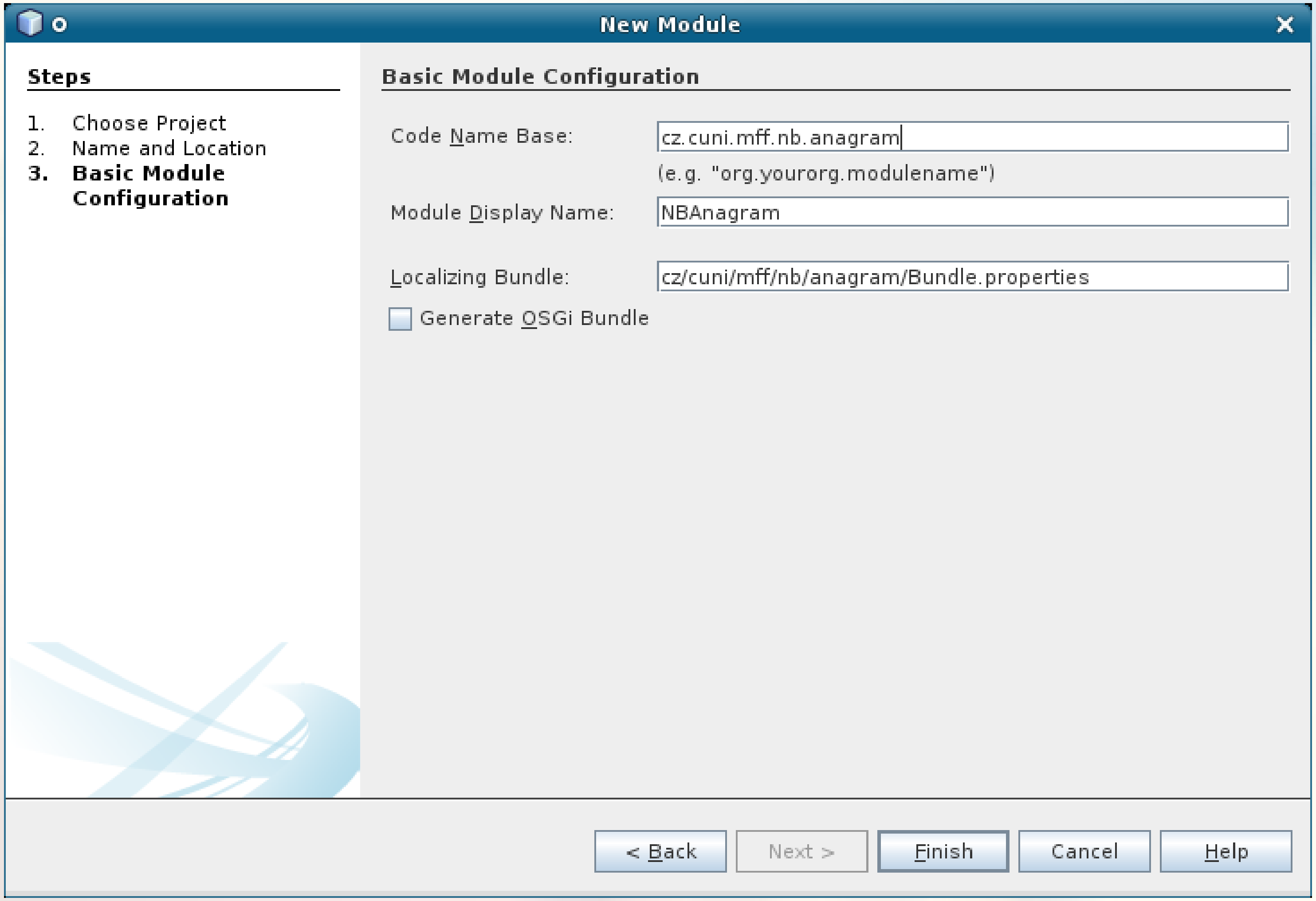
NetBeans Platform: Development IDE

○ Add to Module Suite

Module Suite:

☑ Set as Main Project

Step 1



New Module

Steps

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

Basic Module Configuration

Code Name Base: cz.cuni.mff.nb.anagram

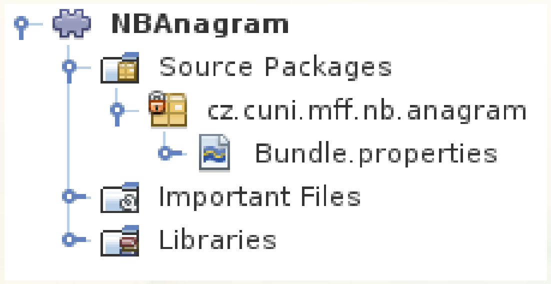
(e.g. "org.yourorg.modulename")

Module Display Name: NBAnagram

Localizing Bundle: cz/cuni/mff/nb/anagram/Bundle.properties

[ ] Generate OSGi Bundle

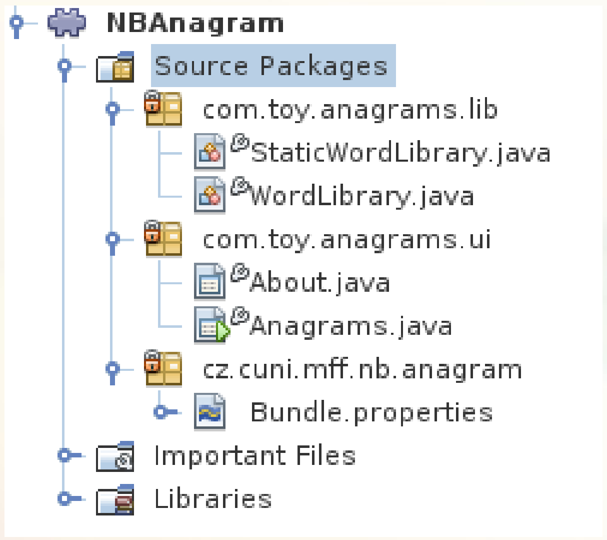
Step 1


NBAnagram
Source Packages
cz.cuni.mff.nb.anagram
Bundle.properties
Important Files
Libraries

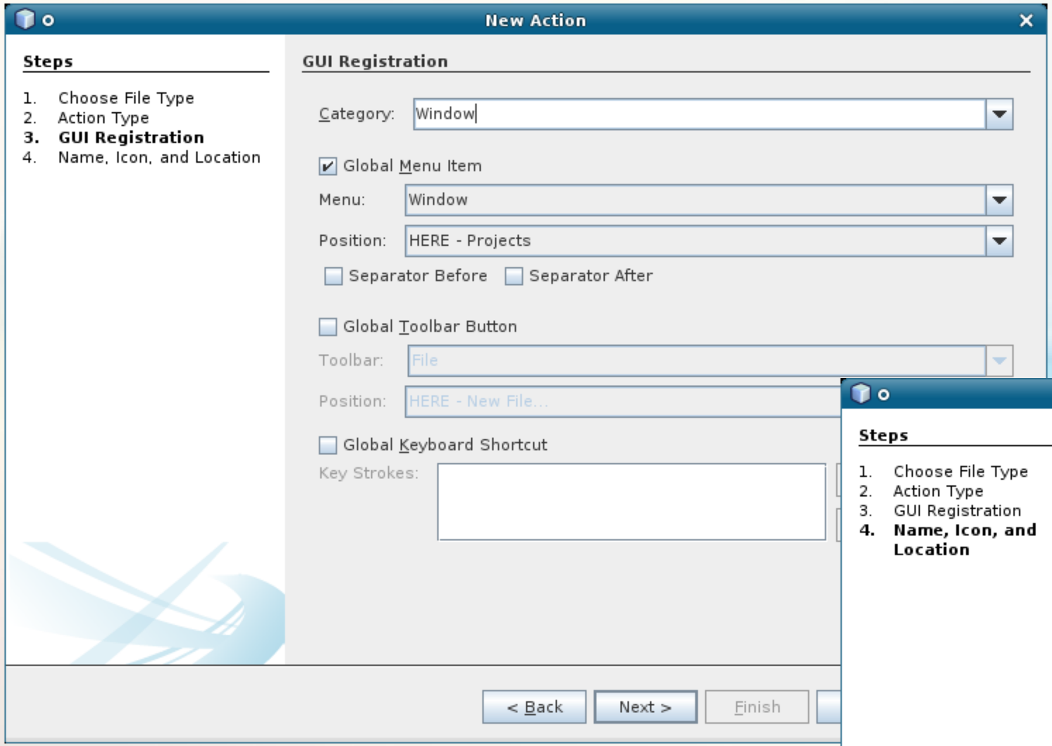
Getting Level 0

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

Copying game classes



Adding an action



New Action

Steps
1. Choose File Type
2. Action Type
3. **GUI Registration**
4. Name, Icon, and Location

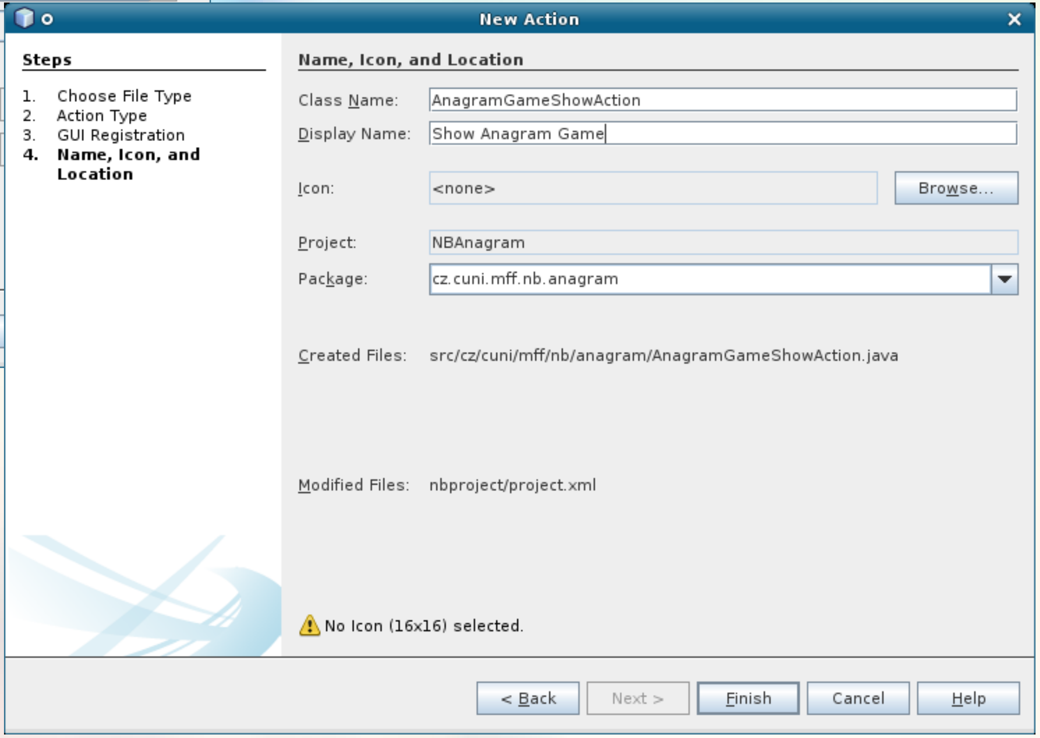
GUI Registration

Category: Window

[✓] Global Menu Item
Menu: Window
Position: HERE - Projects
[ ] Separator Before [ ] Separator After

[ ] Global Toolbar Button
Toolbar: File
Position: HERE - New File...

[ ] Global Keyboard Shortcut
Key Strokes:



New Action

Steps
1. Choose File Type
2. Action Type
3. GUI Registration
4. **Name, Icon, and Location**

Name, Icon, and Location

Class Name: AnagramGameShowAction
Display Name: Show Anagram Game
Icon: <none>
Project: NBAnagram
Package: cz.cuni.mff.nb.anagram

Created Files: src/cz/cuni/mff/nb/anagram/AnagramGameShowAction.java

Modified Files: nbproject/project.xml

No Icon (16x16) selected.

Implementing action

```
package cz.cuni.mff.nb.anagram;

import com.toy.anagrams.ui.Anagrams;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

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

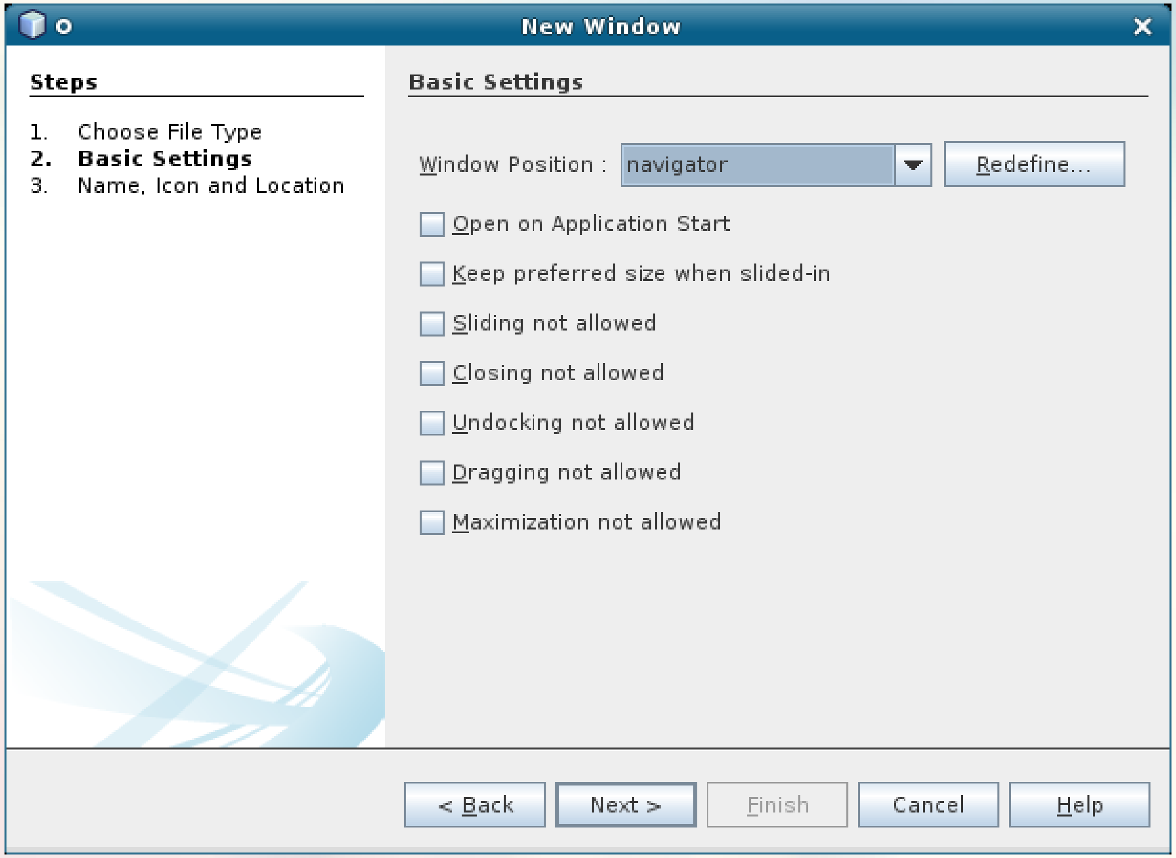
- Using TopComponent and Dialog API
  - JFrame → TopComponent

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



Steps

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

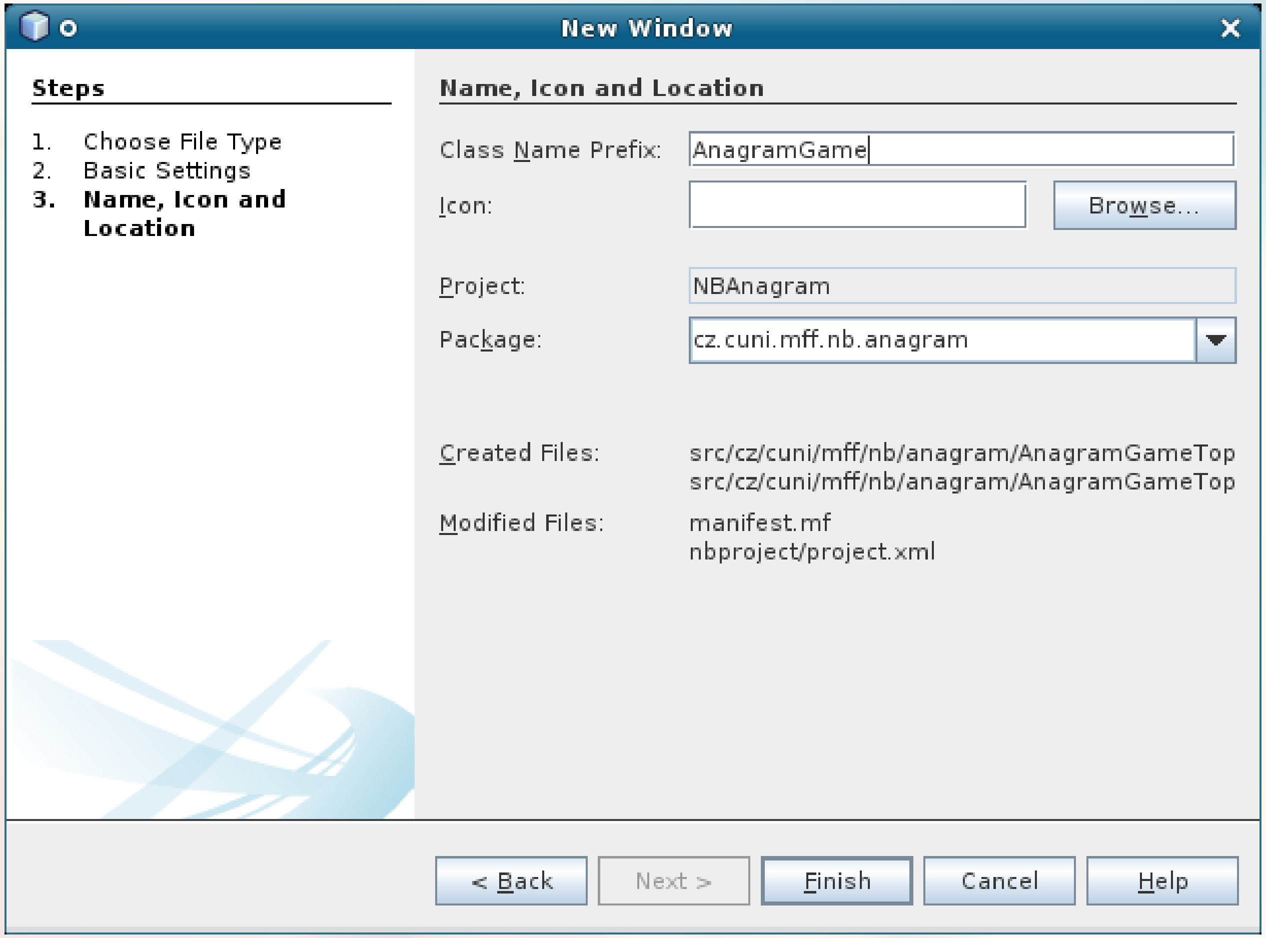
Basic Settings

Window Position : navigator

- [ ] Open on Application Start
- [ ] Keep preferred size when slided-in
- [ ] Sliding not allowed
- [ ] Closing not allowed
- [ ] Undocking not allowed
- [ ] Dragging not allowed
- [ ] Maximization not allowed

Creating new TopComponent

- Name prefix for created classes etc.



Steps

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

Name, Icon and Location

Class Name Prefix: AnagramGame

Icon:

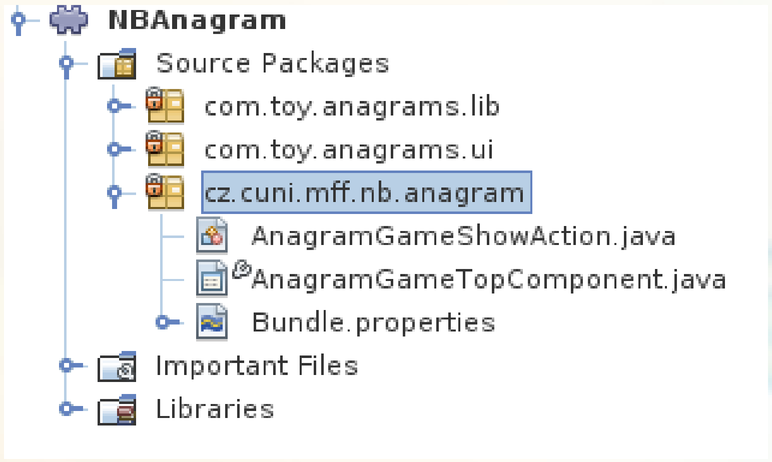
Project: NBAnagram

Package: cz.cuni.mff.nb.anagram

Created Files: src/cz/cuni/mff/nb/anagram/AnagramGameTop
src/cz/cuni/mff/nb/anagram/AnagramGameTop

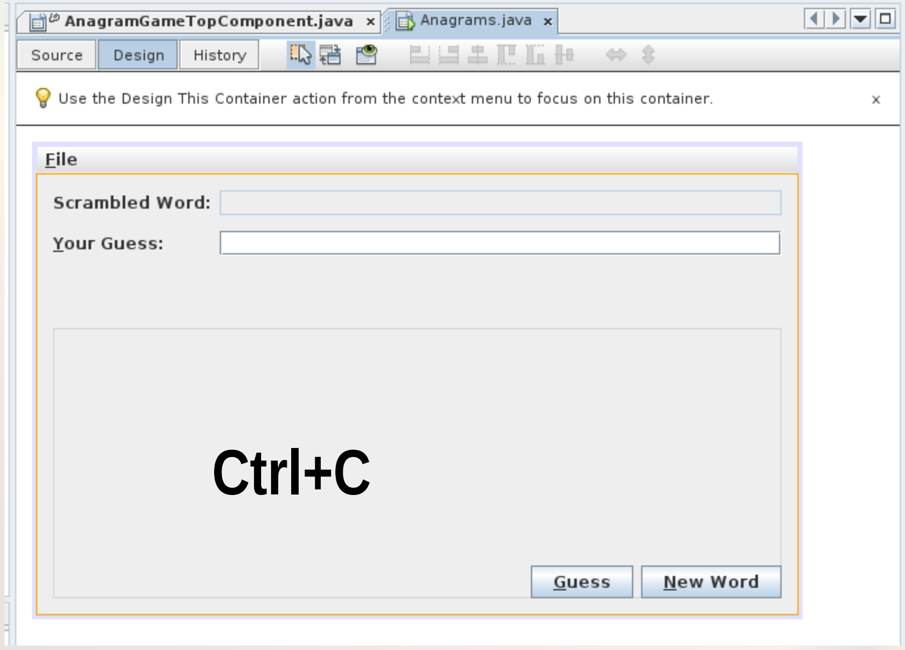
Modified Files: manifest.mf
nbproject/project.xml

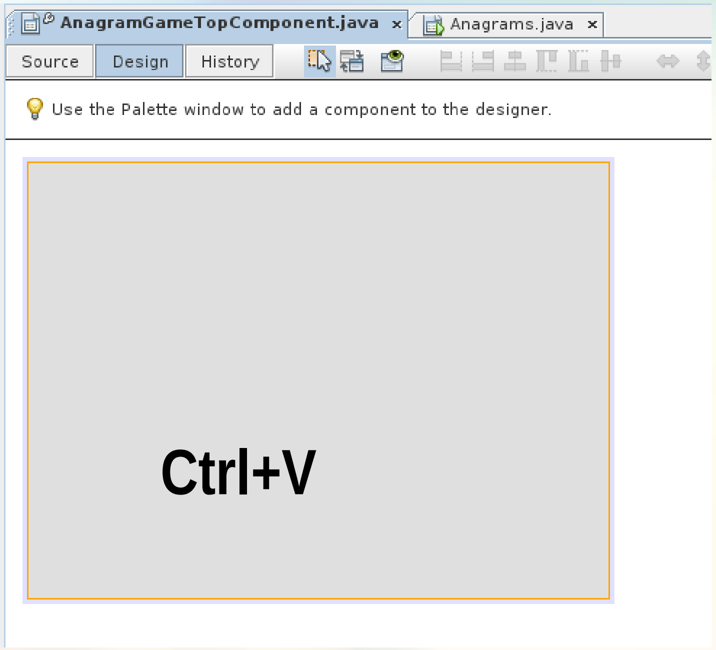
Creating new TopComponent


NBAnagram
Source Packages
com.toy.anagrams.lib
com.toy.anagrams.ui
cz.cuni.mff.nb.anagram
AnagramGameShowAction.java
AnagramGameTopComponent.java
Bundle.properties
Important Files
Libraries

Copying panel to the window

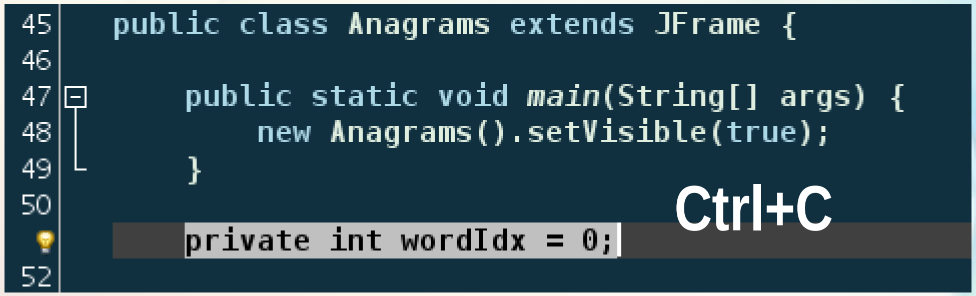
- Select JPanel in the Anagram class
- Copy it
- Paste it to the TopComponent class


AnagramGameTopComponent.java
Anagrams.java
Source
Design
History
Use the Design This Container action from the context menu to focus on this container.
File
Scrambled Word:
Your Guess:
Ctrl+C
Guess
New Word


AnagramGameTopComponent.java
Anagrams.java
Source
Design
History
Use the Palette window to add a component to the designer.
Ctrl+V

Copying variable

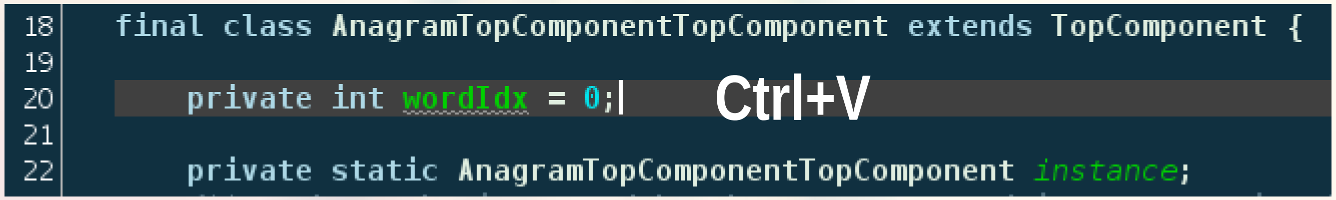
- Copy variables from the Anagrams class
- Paste them to the TopComponent



```
public class Anagrams extends JFrame {

    public static void main(String[] args) {
        new Anagrams().setVisible(true);
    }

    private int wordIdx = 0;
```



```
final class AnagramTopComponentTopComponent extends TopComponent {

    private int wordIdx = 0;

    private static AnagramTopComponentTopComponent instance;
```

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

```
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

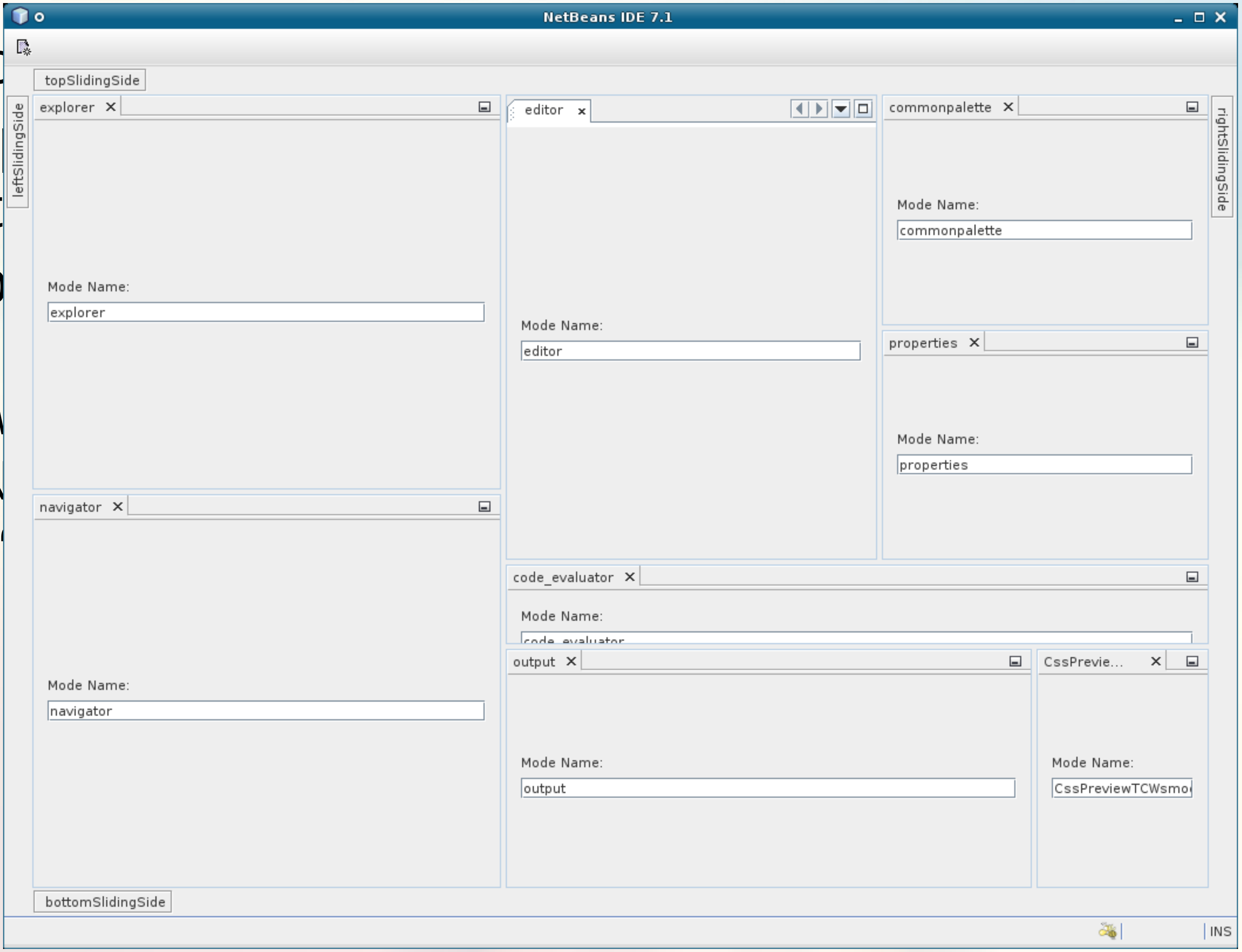
- Persistence – current style
  - annotation **`@ConvertAsProperties`**
    - defines public ID of a DTD for the storing file
      - identification of the file
  - methods
    `readProperties(Properties p)`
    `writePropertes(Properties p)`
    - reading/saving via them

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

- Positic
- Many
  - > edit
- Own o
  - > defi
  - > new
  - > in N
    - > “



NetBeans IDE 7.1

topSlidingSide

leftSlidingSide

rightSlidingSide

bottomSlidingSide

explorer
Mode Name: explorer

editor
Mode Name: editor

commonpalette
Mode Name: commonpalette

properties
Mode Name: properties

navigator
Mode Name: navigator

code_evaluator
Mode Name: code_evaluator

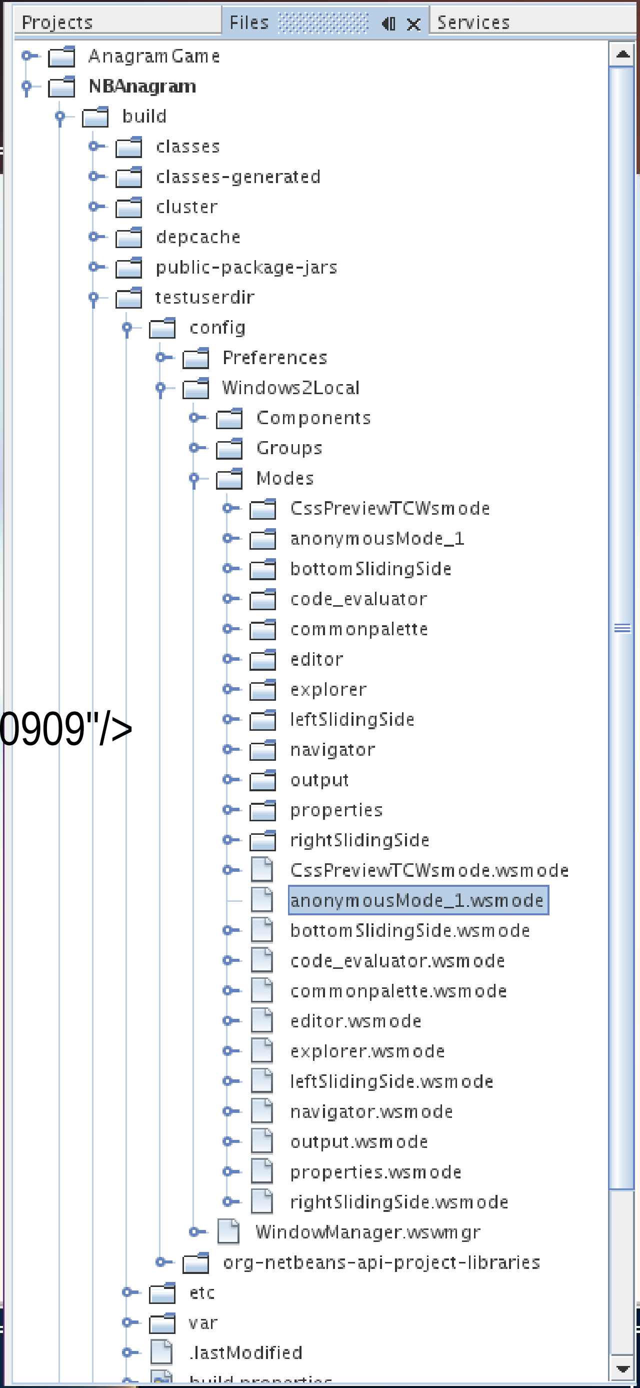
output
Mode Name: output

CssPrevie...
Mode Name: CssPreviewTCWsmo

INS

Mode

```
<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>
```



```
AnagramGame
NBAnagram
  build
    classes
    classes-generated
    cluster
    depcache
    public-package-jars
    testuserdir
      config
        Preferences
        Windows2Local
          Components
          Groups
          Modes
            CssPreviewTCWsmode
            anonymousMode_1
            bottomSlidingSide
            code_evaluator
            commonpalette
            editor
            explorer
            leftSlidingSide
            navigator
            output
            properties
            rightSlidingSide
            CssPreviewTCWsmode.wsmode
            anonymousMode_1.wsmode
            bottomSlidingSide.wsmode
            code_evaluator.wsmode
            commonpalette.wsmode
            editor.wsmode
            explorer.wsmode
            leftSlidingSide.wsmode
            navigator.wsmode
            output.wsmode
            properties.wsmode
            rightSlidingSide.wsmode
          WindowManager.wswmgr
        org-netbeans-api-project-libraries
      etc
      var
      .lastModified
```

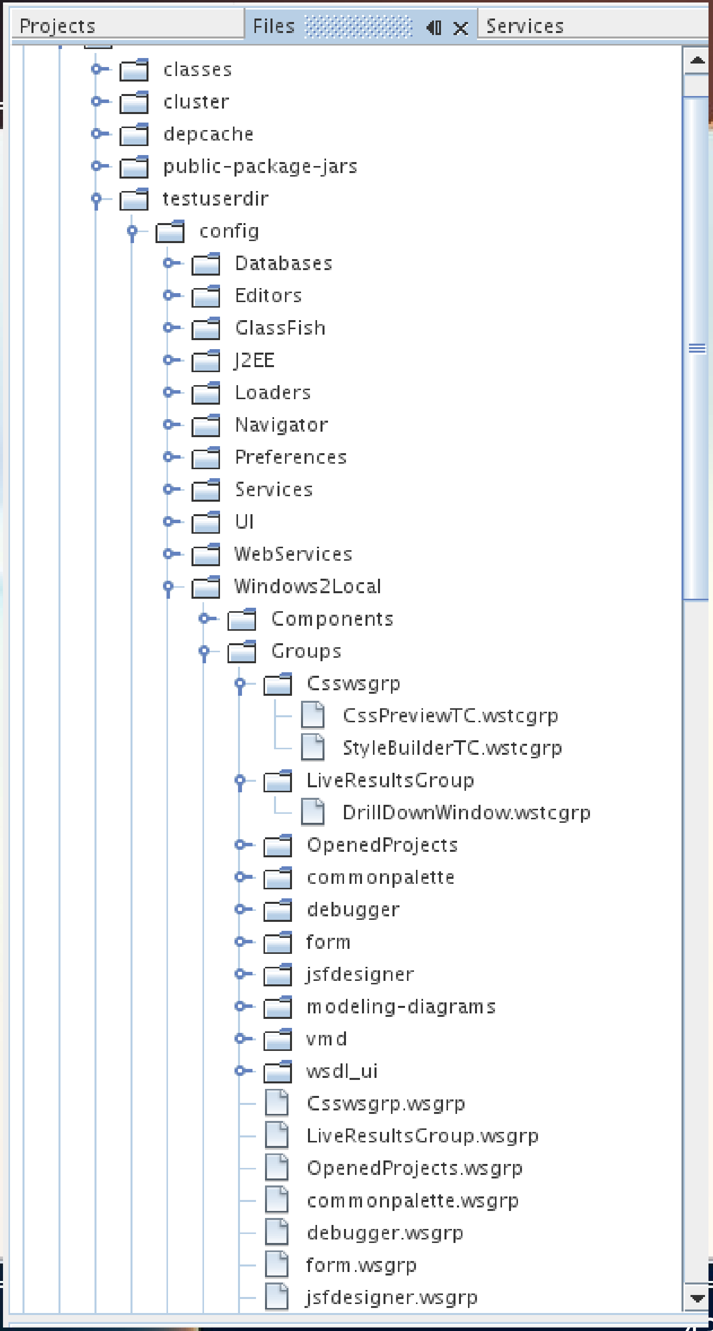
Mode

- Opening a component in a particular mode programmatically

```
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



```
classes
cluster
depcache
public-package-jars
testuserdir
  config
    Databases
    Editors
    GlassFish
    J2EE
    Loaders
    Navigator
    Preferences
    Services
    UI
    WebServices
    Windows2Local
      Components
      Groups
        Csswsgrp
          CssPreviewTC.wstcgrp
          StyleBuilderTC.wstcgrp
        LiveResultsGroup
          DrillDownWindow.wstcgrp
        OpenedProjects
        commonpalette
        debugger
        form
        jsfdesigner
        modeling-diagrams
        vmd
        wsdl_ui
        Csswsgrp.wsgrp
        LiveResultsGroup.wsgrp
        OpenedProjects.wsgrp
        commonpalette.wsgrp
        debugger.wsgrp
        form.wsgrp
        jsfdesigner.wsgrp
```

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"**);