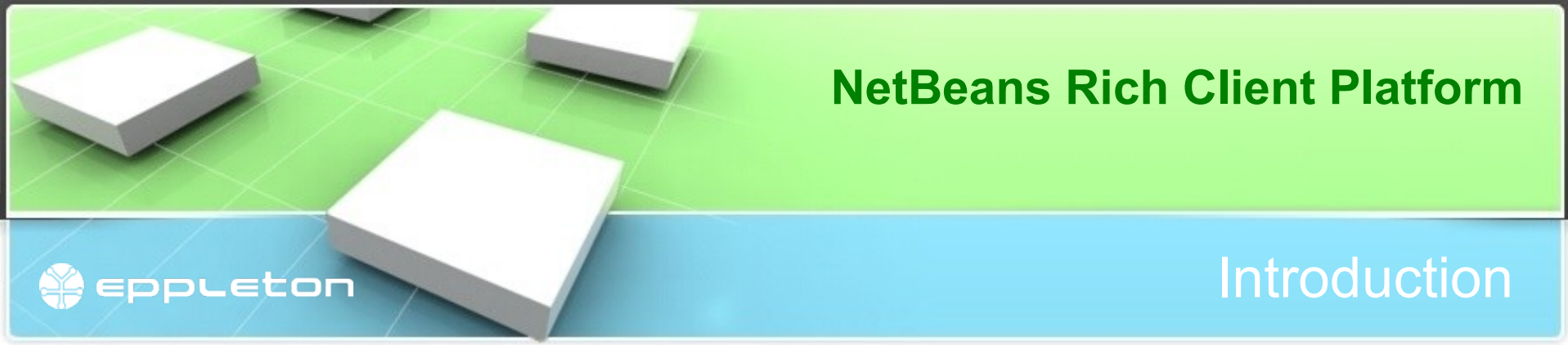
Dialogs API

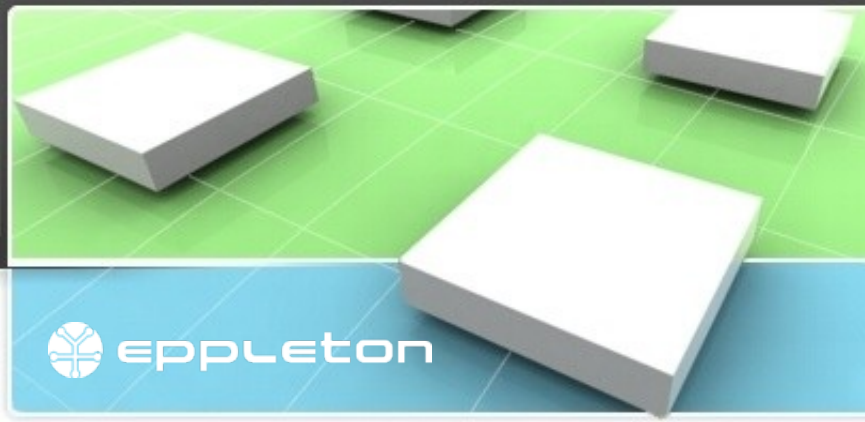
Anton Epple
Eppleton IT Consulting

Anton Epple
http://www.eppleton.de



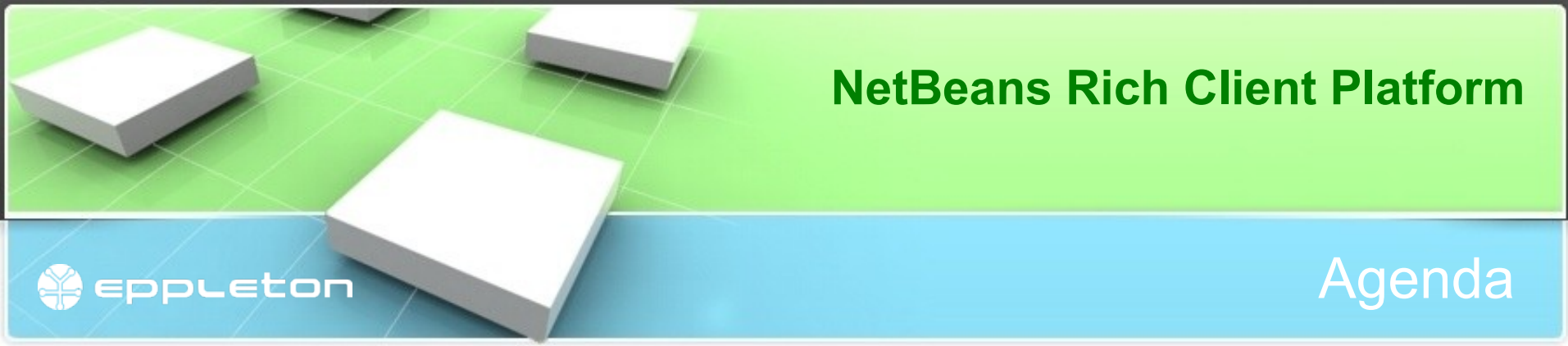
Introduction

- The **Dialogs API** helps in creating dialogs & wizards
- Is based on **java.awt.Dialog**
- Helps creating standard dialogs as well as custom ones
- Resembles **JOptionPane** in many aspects
- Integrated with NetBeans **Window System** & **Help System**
- Simplifies maintaining a standardized Look & Feel



Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards



Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards

Notifications:

- **NotifyDescriptor** for setting the Properties of a Dialog
  - Message as a String, Icon or Component
  - Arrays for more than one message
  - Specify Type of Message (set's the Icon)
- **DialogDisplayer** to show the message

Types of Messages:

- Defined as constants in **NotifyDescriptor**

Constant	Message type / Symbol
PLAIN_MESSAGE	Neutral message without symbol
INFORMATION_MESSAGE	Standard info symbol
QUESTION_MESSAGE	Questionmark
WARNING_MESSAGE	Warning sign
ERROR_MESSAGE	Error sign

Buttons:

- Defined as constants in **NotifyDescriptor**

Constant	Controls
DEFAULT_OPTIONS	Default controls for the chosen dialog type: e.g. OK Button for PLAIN_MESSAGE
OK_OPTION	OK Button
OK_CANCEL_OPTION	OK- and Cancel- Button
YES_NO_OPTION	Yes- and No- Button
YES_NO_CANCEL_OPTION	Yes-, No- and Cancel-Button

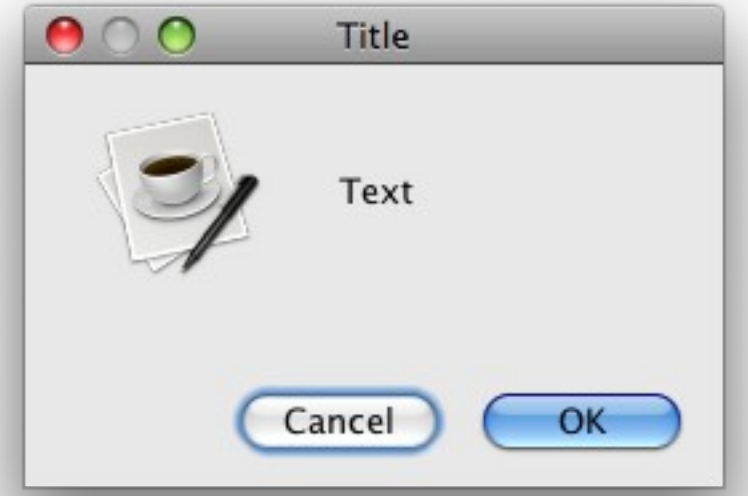
Example:

1. Create new Action “**ShowDialog**” in Menu “**Window**”:
2. Add to **actionPerformed**:

```
public void actionPerformed(ActionEvent e) {
    NotifyDescriptor d = new NotifyDescriptor(
            "Text", // Message or Component to show
            "Title", // Dialog title
            NotifyDescriptor.OK_CANCEL_OPTION, // Controls
            NotifyDescriptor.INFORMATION_MESSAGE,// Symbol
            null, // Custom Controls (Object [])
            null // initial value
            );
    Object response = DialogDisplayer.getDefault().notify(d);
}
```

Example:

3. Run and invoke:



Possible return values:

Constant	Action
OK_OPTION	OK-button was pressed
YES_OPTION	Yes-button was pressed
NO_OPTION	No-button was pressed
CANCEL_OPTION	Cancel-button was pressed
CLOSED_OPTION	Dialog closed without pressing button

Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards
5. Recap

Standard Dialogs

- For the most common dialog types NotifyDescriptor has four subclasses:
  - **NotifyDescriptor.Message**
  - **NotifyDescriptor.Confirmation**
  - **NotifyDescriptor.InputLine**
  - **NotifyDescriptor.Exception**

NotifyDescriptor.Message

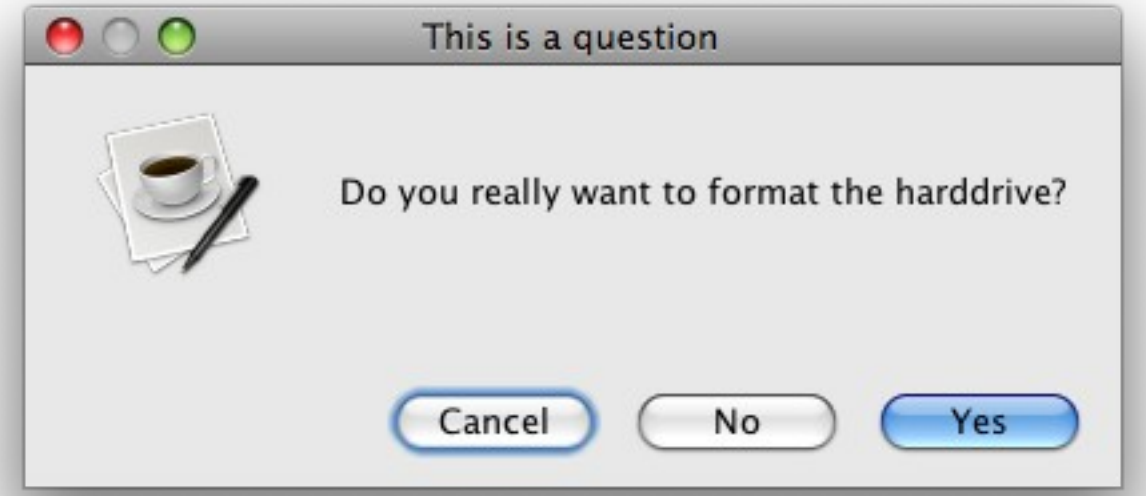
- Change the NotifyDescriptor in actionPerformed to:

NotifyDescriptor d = new NotifyDescriptor.Message("Place any String here");

NotifyDescriptor.Confirmation

- Change the NotifyDescriptor in actionPerformed to:

```
NotifyDescriptor d = new NotifyDescriptor.Confirmation("Do you really want to format the harddrive?", "This is a question");
```



NotifyDescriptor.InputLine

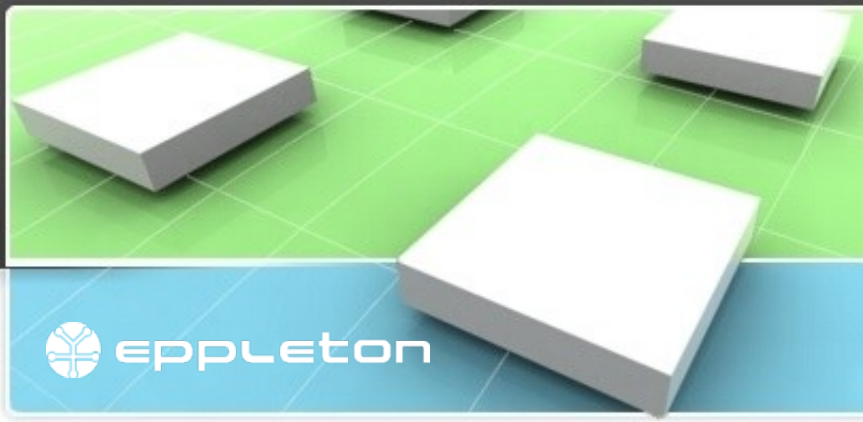
- Change the NotifyDescriptor in actionPerformed to:

```
NotifyDescriptor.InputLine d = new
    NotifyDescriptor.InputLine("Name:","Please enter your name");
Object response = DialogDisplayer.getDefault().notify(d);
System.out.println("name "+ d.getInputText());
```

NotifyDescriptor.Exception

- Change the NotifyDesccriptor in actionPerformed to:

```
NotifyDescriptor d = new NotifyDescriptor.Exception(new
Exception("Something went wrong!"));
```



Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards

DialogDescriptor

- For more customized dialogs you can use the DialogDescriptor class
- DialogDescriptor is an Extension of NotifyDescriptor
- It accepts a Component to display, an ActionListener to react on controls.
- When supplying a HelpCtx it will automatically display a Help button

DialogDescriptor Example: Asynchronous Login Dialog

- As an example we'll create a Login Dialog to display on startup, so we'll create a Module Installer to display it
- We'll use the Form Builder to design the JPanel
- To display it asynchronously (so it doesn't halt the startup process) we'll give it an ActionListener to react on input
- If the login is incorrect we'll use the LifecycleManager to shutdown the application

DialogDescriptor Example: Login Panel

1. Create a new JPanel “LoginPanel” with the Form Builder:

2. Add methods to get the UserName and Password, and set a failure message:

```
public String getUserName() {
    return jTextField1.getText();
}
public char[] getPassword(){
    return jPasswordField1.getPassword();
}
public void setInfo(String info){
    jLabel3.setText(info);
}
```

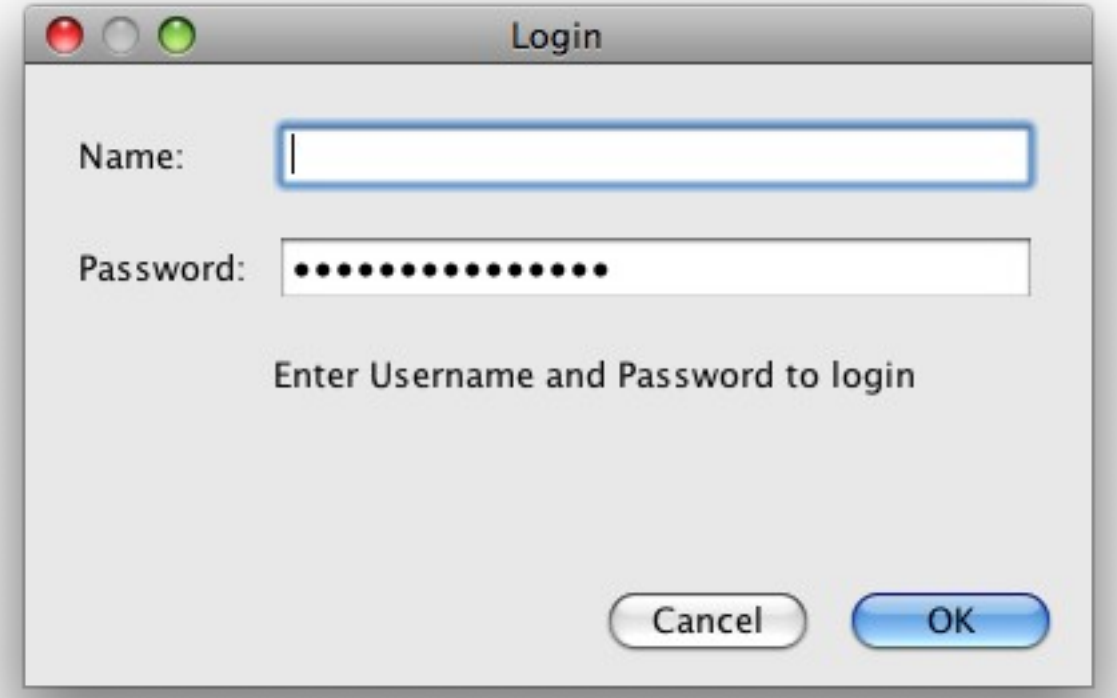
3. Create a new Module Installer (new File → Module Development → Module Installer) and override restored:

```
private LoginPanel loginPanel = new LoginPanel();
private DialogDescriptor d = null;

@Override
public void restored() {
    d = new DialogDescriptor(loginPanel, "Login", true, this);
    d.setClosingOptions(new Object[]{}); // not closeable
    DialogDisplayer.getDefault().notifyLater(d);
}
```

4. Dialogdescriptor expects an ActionListener as the constructor's fourth parameter, we fix this by implementing ActionListener:

```
public class Installer extends ModuleInstall implements ActionListener {
…
public void actionPerformed(ActionEvent arg0) {
        if (arg0.getSource() == DialogDescriptor.CANCEL_OPTION){
            LifecycleManager.getDefault().exit();
        } else if (!loginPanel.getUserName().equals("Toni")){// do real check here
            loginPanel.setInfo("Wrong Username or Password");
        } else {
            d.setClosingOptions(null);// can close
        }
    }
```

5. Run the application:



6. Finally fix the bug in restored() method :-) :

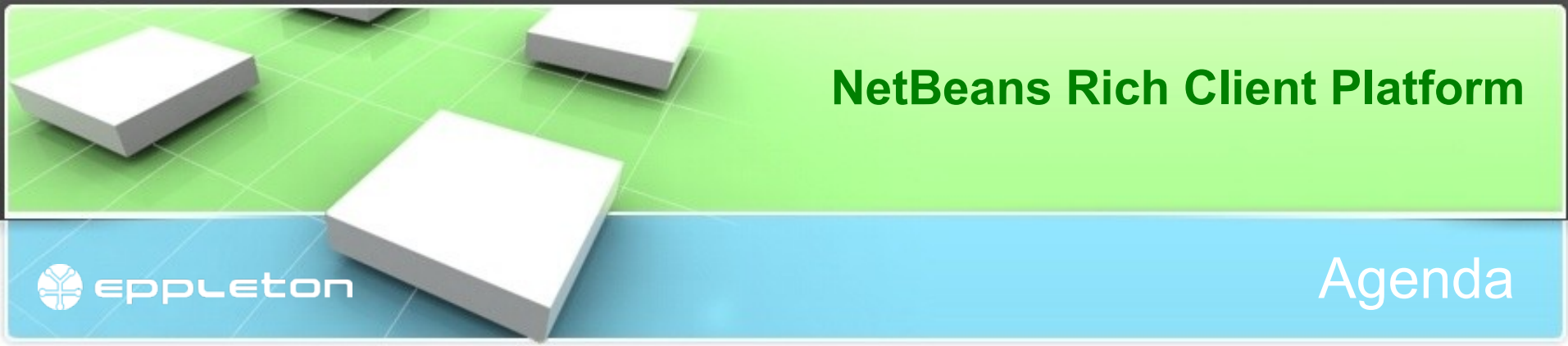
```
...
// if someone simply closes the dialog
d.addPropertyChangeListener(new PropertyChangeListener() {

        public void propertyChange(PropertyChangeEvent e) {
            if (e.getPropertyName().equals(DialogDescriptor.PROP_VALUE)
                    && e.getNewValue()== DialogDescriptor.CLOSED_OPTION){
                    LifecycleManager.getDefault().exit();
            }
        }
    });
DialogDisplayer.getDefault().notifyLater(d);
```

Creating Dialogs without the help of Dialogs API:

- File → New File → Java GUI Forms → JDialog
- Getting the main Frame as parent:

Creating Dialogs without the help of Dialogs API:

- File → New File → Java GUI Forms → JDialog
- Getting the main Frame as parent:
  - Frame f = WindowManager.getDefault().getMainWindow();



Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards

NetBeans Wizard Architecture

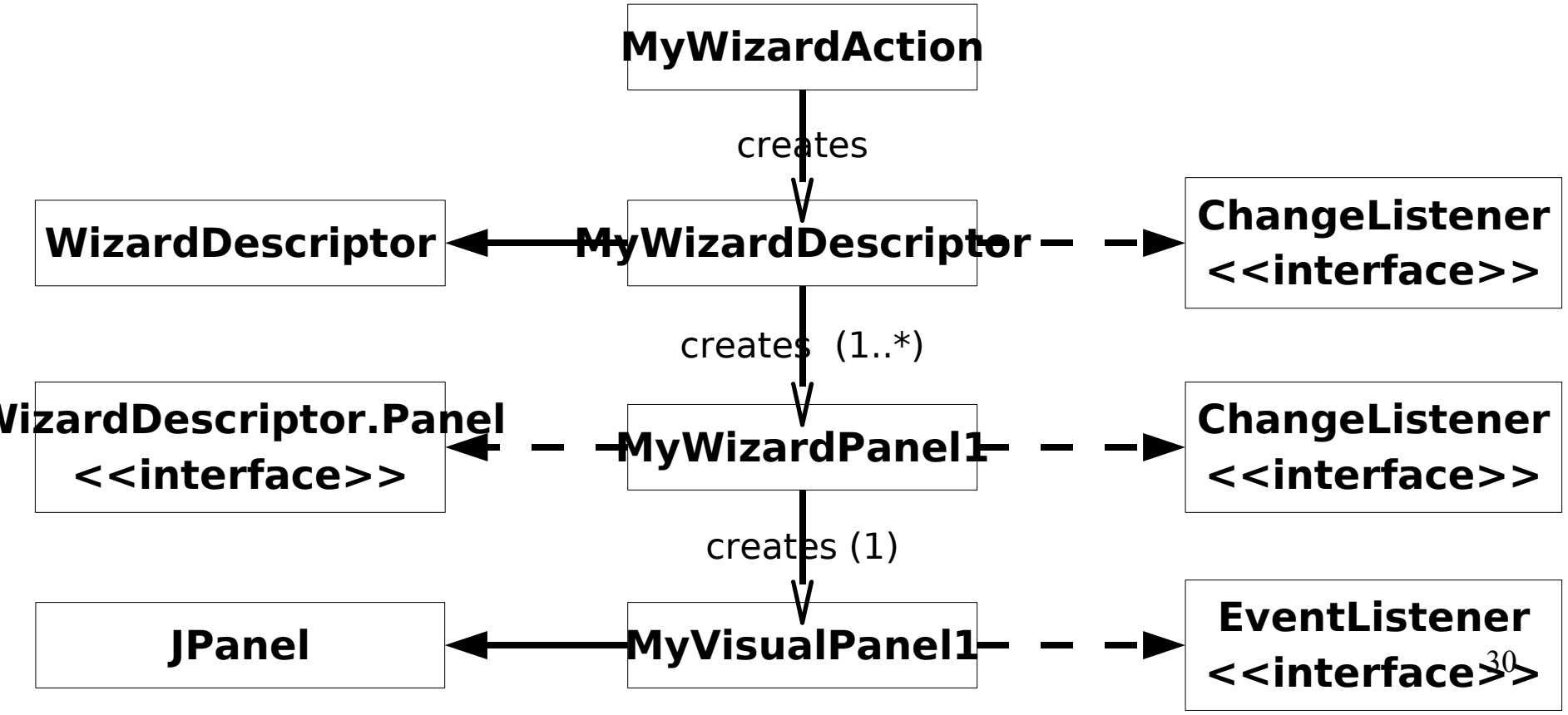
WizardDescriptor:

- **Controller** for wizard, manages the Panels
- Subclass of **DialogDescriptor**
- Is typically used as **DataModel** to store the data collected between individual Wizard steps as **Properties**

Each wizard step typically consists of two classes

- A **Visual Panel**, normally a JPanel without dependencies on Wizard specific classes
- A **WizardDescriptor.Panel <Data>** acting as a Controller

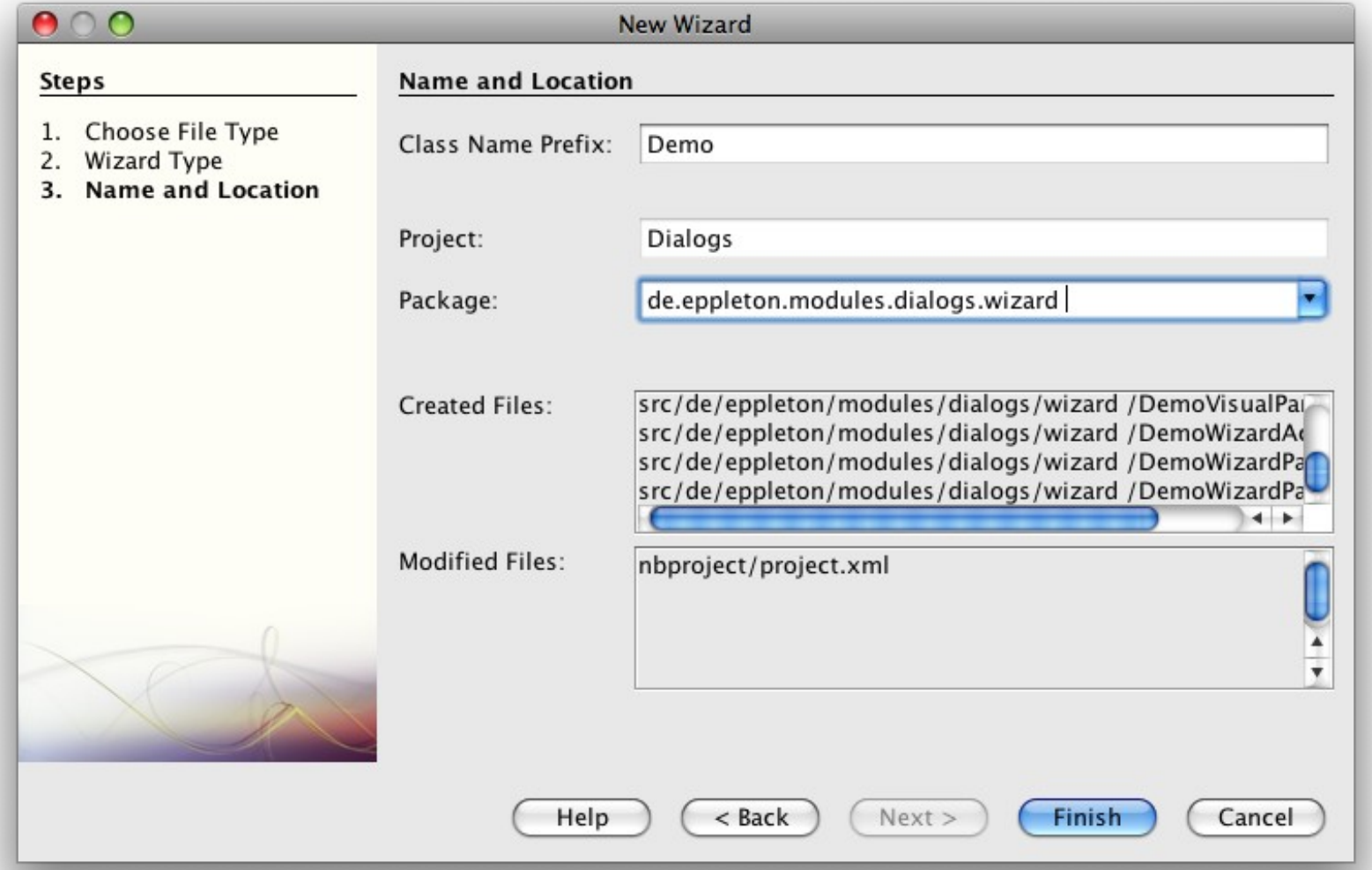
NetBeans Wizard Architecture



The **Wizard Wizard** creates a skeleton implementation

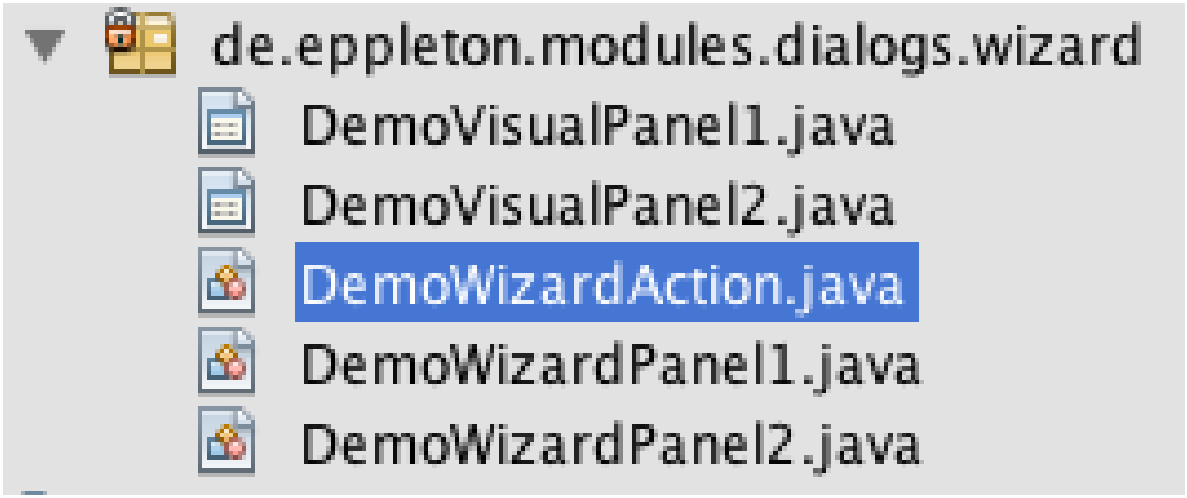
1. New File → Module Development → Wizard

2.


New Wizard
Steps
1. Choose File Type
2. Wizard Type
3. Name and Location
Name and Location
Class Name Prefix:
Demo
Project:
Dialogs
Package:
de.eppleton.modules.dialogs.wizard
Created Files:
src/de/eppleton/modules/dialogs/wizard /DemoVisualPa
src/de/eppleton/modules/dialogs/wizard /DemoWizardA
src/de/eppleton/modules/dialogs/wizard /DemoWizardPa
src/de/eppleton/modules/dialogs/wizard /DemoWizardPa
Modified Files:
nbproject/project.xml
Help
< Back
Next >
Finish
Cancel

3. The Wizard generates 5 Files:

- The generated WizardAction uses the standard WizardDescriptor to invoke the Wizard



Generated code **DemoWizardAction**:

```
WizardDescriptor wizardDescriptor = new WizardDescriptor(getPanels());
// {0} will be replaced by WizardDesriptor.Panel.getComponent().getName()
wizardDescriptor.setTitleFormat(new MessageFormat("{0}"));
wizardDescriptor.setTitle("Your wizard dialog title here");
Dialog dialog = DialogDisplayer.getDefault().createDialog(wizardDescriptor);
dialog.setVisible(true);
dialog.toFront();
boolean cancelled = wizardDescriptor.getValue() !=
                                        WizardDescriptor.FINISH_OPTION;
if (!cancelled) {
   // Use WizardDescriptor.getProperty to read the values entered by User
}
```

Generated code **DemoWizardAction**:

```
private WizardDescriptor.Panel[] getPanels() {
        if (panels == null) {
            // creates the Panels
            panels = new WizardDescriptor.Panel[]{
                        new DemoWizardPanel1(),
                        new DemoWizardPanel2()
                    };
            String[] steps = new String[panels.length];
            for (int i = 0; i < panels.length; i++) {
            … // sets Properties like display name, etc.
```

Client Properties for **WizardDescriptor.Panel**:

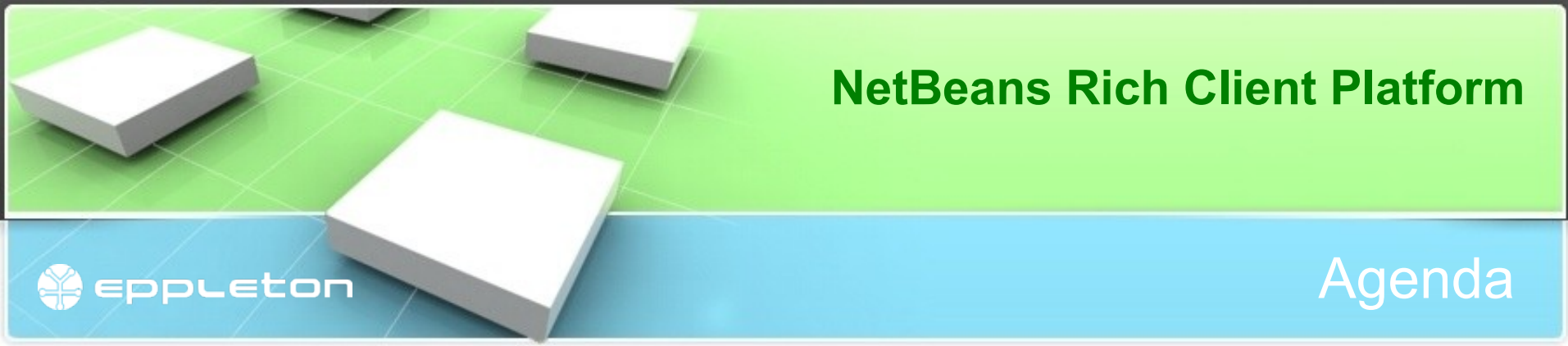
Property (WizardPanel_...)	Type	Meaning
...autoWizardStyle	Boolean	Turn on subtitle creation
...contentDisplayed	Boolean	Show step overview Panel
...helpDisplayed	Boolean	Show help in extra tab
...contentNumbered	Boolean	Turn on numbering of steps
...contentSelectedIndex	Integer	Index number of step (starts with 0)
...contentData	String []	Names of steps
...image	Image	Background for overview Panel
...errorMessage	String	Shown when invalid state
...helpURL	URL	URL for this Panel's help

Generated code **DemoWizardPanel1**:

```
public Component getComponent() { // returns the Visual Panel
    return component;
}

public boolean isValid() { // can Proceed?
    return true;
}

public final void addChangeListener(ChangeListener l) {
   // the WizardDescriptor registers here
}

public final void removeChangeListener(ChangeListener l) {
}
```

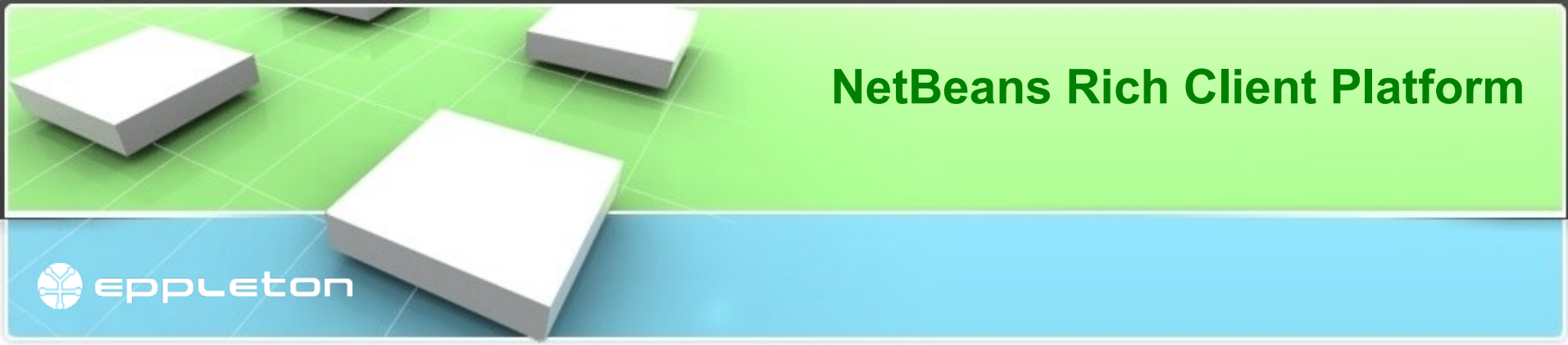
Generated code **DemoWizardPanel1**:

```
// You can use a settings object to keep track of state. Normally the
// settings object will be the WizardDescriptor, so you can use
// WizardDescriptor.getProperty & putProperty to store information entered
// by the user.
public void readSettings(Object settings) {
    this.model = (WizardDescriptor) settings;
}

public void storeSettings(Object settings) {
    model.putProperty(MyVisualPanel1.PROPERTY_SAMPLE,
                      getComponent().getSample() );
}
```



Agenda

1. Notifications
2. Standard Dialogs
3. Custom Dialogs
4. Wizards



Q&A