

Introduction à l'API Palette de NetBeans

par Mhamed Ben Jmaa ([home page](#))

Date de publication : 16/05/2008

Dernière mise à jour : 16/05/2008

Dans cet article nous allons découvrir l'API Palette de NetBeans Platform.

0 - Introduction.....	3
I - Mise en place du projet.....	4
II - Préparation de l'éditeur :.....	5
III - Utilisation de la Palette :.....	7
IV - Remerciements.....	13


0 - Introduction

La plateforme Netbeans est un outil très puissant pour la réalisation d'application swing, elle offre un gain de temps en matière de productivité, reste à se familiariser avec ses outils rapidement, je vais donc vous faire part de quelques trucs et astuces pour vous éviter beaucoup d'heures de recherche dans les forums et autres articles, pour mener à bien ce tutoriel, on utilisera les source de **Christophe Dujardin** issue de son article " **Introduction au dessin en Java** ", on aura pour résultat final, un petit outil graphique pour dessiner des formes géométriques.

I - Mise en place du projet

1. Nous allons commencer par créer un nouveau projet, File -> new Project -> NetBeans plug-in modules -> Module suite Project.

Le module suite est un conteneur de modules vous pourrez y déployer vos modules afin de les tester.

 *RQ : il est tout à fait possible de passer outre cette étape et aller directement à l'étape 4 si vous voulez que vos modules soient directement greffés dans l'IDE Netbeans lui-même.*

2. Cliquez sur next, donnez un nom à votre module suite, on le nommera " masuite ", cliquez sur " finish ".

3. Lancez l'application. (Appuyez sur F6)

Première remarque :

C'est la copie conforme de Netbeans, tout les modules sont là, ils sont chargés par défaut, on ne va pas avoir besoin de tous ces modules. On va donc les enlever et travailler avec une plateforme basique, on ajoutera les modules dont on aura besoin au fur et à mesure.

Fermez l'instance de " masuite ", click droit sur le projet -> properties -> applications -> Create a standalone application.

Vous aurez un avertissement qui vous demandera d'exclure tous les modules relatifs à Netbeans, cliquez sur " exclude ", vous pourrez désormais changer le titre, l'icône, le splash screen " écran de démarrage " de votre application.


Une fois que vous avez fini, cliquez sur OK, vous pouvez relancer l'application, on voit bien que le temps de démarrage se réduit considérablement, cela est dû au fait qu'il ya beaucoup moins de modules à charger lors du lancement.

4. Une fois que le modulesuite est en place, on va créer un module dans lequel on placera notre travail :

File -> new project -> netbeans plug-in modules -> Module Project, cliquez sur Next,

Vous appelez votre module comme vous voulez, dans mon cas j'ai choisie " moduledessin ".

On va placer notre module dans le module suite " masuite " que l'on vient juste de mettre au point.

 *RQ : si vous avez choisi de ne pas créer de module suite, il suffit de placer votre module dans " Standalone Module ".*

5. Cliquez sur next, donnez un " code base name " à votre module, prenez l'habitude de bien nommer vos modules, afin d'avoir une bonne structure et pour éviter des confusions.

6. Cliquez finish.

II - Préparation de l'éditeur :

Pour mettre en place la palette il faut qu'il y ait un TopComponent qui va de paire avec elle, on va donc commencer par créer un éditeur

Cliquez droit sur moduledessin -> new ->file or folder ->netbeans module developpement -> window component -> click next.

Ce sera un "editor" dans "Window Position" -> Cliquez sur next.

Pour les noms je vous laisse le choix, pour ma part j'ai choisi " editeurdedessin " pour le nom de la class de l'éditeur et je l'ai placé dans le package " org.yourorghere.moduledessin.editeur ", vous pouvez ajouter une icône au menu, pour un résultat plus esthétique, sachez que le format PNG est supporté, vous pouvez donc créer des icônes avec des effets de transparence.

Prenez l'habitude de bien nommer vos modules, afin d'avoir une bonne structure et pour éviter des confusions.

Cliquez sur finish.

Voilà qu'on obtient un nouvel éditeur vierge, on va y placer le JCanvas de M. **Christophe Dujardin**, voici les liens :

Il faut télécharger les sources via **FTP** ou **HTTP**.

Le fichier zip est constitué ainsi : intro-dessin.zip\drawing\src\.

Dans le répertoire " src " il ya les fichiers dont on aura besoin, le package " listeners " et le package " demo ", en ce qui concerne le package " demo ", vous pouvez l'enlever, il ne va pas nous servir, pour le reste, il faut les copier dans le sous répertoire " src " de votre module.

Puis, on fera un

- 1 " build clean " du projet (maj+F11)
- 2 Dans Netbeans on clique sur tool -> palette manager -> swing/ awt componenet-> new categorie que j'ai nommé " maCategorie "
- 3 Cliquez sur ok
- 4 Cliquez sur add from project
- 5 On ira chercher dans notre projet moduledessin
- 6 Cliquez sur next
- 7 On choisi l'objet JCanvas
- 8 Cliquez sur Next. On le placera dans maCategorie
- 9 Cliquez sur close.

On ajoute enfin ce JCanvas dans notre fenêtre de l'éditeur et on l'étale sur toute la surface.

On va juste rajouter quelques lignes dans l'initialisation de ce JCanvas :

```
private editeurdedessinTopComponent() {  
    initComponents();  
    jCanvas1.setBackground(Color.WHITE);  
}
```

```
new NonOverlapMoveAdapter(jCanvas1);

setName(NbBundle.getMessage(editeurdedessinTopComponent.class, "CTL_editeurdedessinTopComponent"));

setToolTipText(NbBundle.getMessage(editeurdedessinTopComponent.class, "HINT_editeurdedessinTopComponent"));
//      setIcon(Utilities.loadImage(ICON_PATH, true));
}
```



Je vous conseille de jeter un oeil sur l'article "[Introduction au dessin en Java](#)" si vous voulez en savoir plus sur l'utilisation de ce JCanvas.

Maintenant que notre éditeur est prêt passons à la palette.

III - Utilisation de la Palette :

Les éléments d'une palette sont définis grâce des fichiers XML, on va donc commencer par définir les éléments :

Click droit sur module dessin -> new file/folder -> XML -> XML Document -> Cliquez sur next

Cliquez sur next.

On choisira " DTD -constrained document ", Cliquez sur next

Cliquez sur finish.

On obtiendra un fichier XML pour la description des éléments il faut juste rajouter cette ligne au fichier XML obtenu :

```
Carre.XML

<?xml version="1.0" encoding="UTF-8"?>

<!--
  Document   : carre.xml
  Created on : 9 juin 2007, 23:40
  Author    : Mhamed
  Description:
    Purpose of the document follows.
-->
<!-- Ligne aajouter -->
<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.0//EN" "http://www.netbeans.org/dtds/editor-palette-item-1_0.dtd">
<editor_palette_item>

</editor_palette_item>
```

On refait la même opération pour obtenir un fichier XML, son nom sera cercle.XML

```
Cercle.XML

<?xml version="1.0" encoding="UTF-8"?>

<!--
  Document   : cercle.xml
  Created on : 9 juin 2007, 23:41
  Author    : Mhamed
  Description:
    Purpose of the document follows.
-->
<b><!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.0//EN"
"http://www.netbeans.org/dtds/editor-palette-item-1_0.dtd"></b>
<editor_palette_item>

</editor_palette_item>
```

On va ajouter les icones, à vous de créer 4 icones une icône : de 16x16 et une de 32x32 pour chaque élément, on aura donc ceci :

On ira ajouter les informations concernant les icones, le nom, le tooltip, la classe correspondante de chaque élément dans le fichier XML :

On obtiendra ce résultat :

Carre.XML

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  Document   : carre.xml
  Created on : 9 juin 2007, 23:49
  Author    : Mhamed
  Description:
    Purpose of the document follows.
-->

<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.0//EN"
"http://www.netbeans.org/dtds/editor-palette-item-1_0.dtd">
<editor_palette_item version="1.0">
  <class name="drawing.RectangleDrawable"/>
  <icon16 urlvalue="org/yourorghere/moduledessin/editeur/carre16.JPG" />
  <icon32 urlvalue="org/yourorghere/moduledessin/editeur/carre32.JPG" />
  <description localizing-bundle="org/yourorghere/moduledessin/editeur/Bundle"
display-name-key="carre_nom"
tooltip-key="carre_Tooltip" />
</editor_palette_item>
```

Et

Cercle.XML

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  Document   : cercle.xml
  Created on : 10 juin 2007, 00:16
  Author    : Mhamed
  Description:
    Purpose of the document follows.
-->

<!DOCTYPE editor_palette_item PUBLIC "-//NetBeans//Editor Palette Item 1.0//EN"
"http://www.netbeans.org/dtds/editor-palette-item-1_0.dtd">
<editor_palette_item version="1.0">
  <class name="drawing.CircleDrawable"/>
  <icon16 urlvalue="org/yourorghere/moduledessin/editeur/cercle16.JPG" />
  <icon32 urlvalue="org/yourorghere/moduledessin/editeur/cercle32.JPG" />
  <description localizing-bundle="org/yourorghere/moduledessin/editeur/Bundle"
display-name-key="cercle_nom"
tooltip-key="cercle_Tooltip" />
</editor_palette_item>
```

Il faudra alors ajouter les propriétés qu'on vient de définir dans le fichier `Bundle.properties`. Vous y trouverez par la même occasion, le nom de votre éditeur, ainsi que le nom de l'action.

Bundle.properties

```
CTL_editeuredessinAction=Open editeuredessin Window
CTL_editeuredessinTopComponent=editeuredessin Window
HINT_editeuredessinTopComponent=This is a editeuredessin window
cercle_nom=Un cercle
cercle_Tooltip=Un cercle
carre_nom=Un carré
carre_Tooltip=Un carré
```

En ce qui concerne les catégories, il faut les ajouter dans le fichier layer.xml, de cette manière : On ajoute une nouvelle balise <folder> juste après la balise racine <filesystem> cette balise contiendra ceci :

Layer.XML

```
<folder name="MaPalette">
  <folder name="Angulaire">
    <file name="objet_1.xml" url="carre.xml"/>
  </folder>
  <folder name="Circulaire">
    <file name="objet_2.xml" url="cercle.xml"/>
  </folder>
</folder>
```

Chaque "Folder" dans le fichier XML représente une catégorie.

Maintenant qu'on a défini les catégories et les éléments de la palette, on va la lier au TopComponent, grâce à un **Lookup**, on va donc rajouter tout d'abord cette méthode dans le fichier editeurdedessinTopComponent.java,

editeurdedessinTopComponent.java

```
private PaletteController initializePalette() {
try {
return PaletteFactory.createPalette( "MaPalette", new PaletteActions() {
public Action[] getCustomCategoryActions(Lookup lookup) {
return new Action[0];
}
public Action[] getCustomItemActions(Lookup lookup) {
return new Action[0];
}
public Action[] getCustomPaletteActions() {
return new Action[0];
}
public Action[] getImportActions() {
return new Action[0];
}
public Action getPreferredAction(Lookup lookup) {
return null; //TODO
}
});
} catch (IOException ex) {
ex.printStackTrace();
}
return null;
}
```

 **A voir aussi : [PaletteActions](#)**

Vous remarquez que pour le nom de la palette on a mis le nom du folder qu'on a précédemment défini dans le fichier layer.xml, " MaPalette "

On aura des problèmes d'imports, il faut donc rajouter les dépendances :

Cliquez droit sur masuite -> propriétés -> librairies -> ide7 -> Core Component Palette -> Cliquez sur ok -> Cliquez sur ok.

Cliquez droit sur modulede dessin->propriétés -> librairies -> add dependency -> Core-component palette -> Cliquez sur ok -> add dependency -> Nodes API -> Cliquez sur ok -> add dependency -> Text API -> Cliquez sur OK -> Cliquez sur OK.

On aura besoin de Text API plus tard pour l'utilisation du drag'ndrop.

On pourra enfin ajouter les imports nécessaires a notre fichier, pour les objets Action, choisissez java.swing.Action ;

On va ensuite modifier le constructeur de editeurdedessinTopComponent.java pour qu'il puisse initialiser un objet de type PaletteController appelé via la méthode lookup, on aura donc ceci :

```
private editeurdedessinTopComponent(){
    this(new InstanceContent());
}
private editeurdedessinTopComponent(InstanceContent content) {
    super( new AbstractLookup( content ) );
    content.add( initializePalette() );
    initComponents();
    jCanvas1.setBackground(Color.WHITE);
    new NonOverlapMoveAdapter( jCanvas1);

    setName(NbBundle.getMessage(editeurdedessinTopComponent.class, "CTL_editeurdedessinTopComponent"));

    setToolTipText(NbBundle.getMessage(editeurdedessinTopComponent.class, "HINT_editeurdedessinTopComponent"));
    // setIcon(Utilities.loadImage(ICON_PATH, true));
}
```

On obtient alors le résultat suivant :

On va ajouter quelque bout de code en plus pour que notre palette et notre éditeur puissent supporter le drag'n drop :

On commence par modifier le fichier FormDrawable.java :

FormDrawable


```
public abstract class FormDrawable implements IMovableDrawable, ActiveEditorDrop {

    protected Rectangle rect ;
    protected Color color;
    public FormDrawable(){

    }
    public FormDrawable(Color color, Point pos, Dimension dim){
        this.color=color;
        this.rect = new Rectangle(dim);
        setPosition(pos);
    }
    public abstract void draw(Graphics g) ;

    public Rectangle getRectangle(){
        return (Rectangle) rect.clone();
    }
    public Point getPosition() {
        Point p= rect.getLocation();
        p.x = (p.x+rect.width/2);
        p.y = (p.y+rect.height/2);
        return p;
    }
    public void setPosition(Point p) {
        rect.x = (p.x-rect.width/2);
        rect.y = (p.y-rect.height/2);
    }

    public boolean handleTransfer(JTextComponent jTextComponent) {
        return true;
    }
}
```

 *Il faut rajouter un constructeur vide dans RectangleDrawable et CircleDrawable, ce constructeur servira pour l'instanciation des objets lors du drag'n drop*

On va maintenant ajouter un PropertyChangeListener pour détecter si l'utilisateur à changer l'élément qu'il a sélectionné dans la palette et la méthode qui permet de savoir sur quel élément il a cliqué :

Dans le fichier editeurdedessinTopComponent.java, on va ajouter l'implements PropertyChangeListener, on ajoutera deux variables globales :

```
private PaletteController controller;
private ActiveEditorDrop selNode;
```

Et la méthode qui suit :

```
public void propertyChange(PropertyChangeEvent evt) {
    if( PaletteController.PROP_SELECTED_ITEM.equals( evt.getPropertyName() ) ) {
        Lookup selItem = controller.getSelectedItem();
        if( null != selItem ) {
            selNode = (ActiveEditorDrop)selItem.lookup( ActiveEditorDrop.class );
        }
    }
}
```

Puis on ajoute ce code dans le constructeur :

```
super( new AbstractLookup( content ) );
//*****
// code a ajouter
//*****
controller = initializePalette();
content.add( controller);
controller.addPropertyChangeListener( this);
setDropTarget(new DropTarget(this,new DropTargetListener() {
    public void dragEnter(DropTargetDragEvent dropTargetDragEvent) {
    }
    public void dragExit(DropTargetEvent dropTargetEvent) {
    }
    public void dragOver(DropTargetDragEvent dropTargetDragEvent) {
    }
    public void drop(DropTargetDropEvent dropTargetDropEvent) {

        if (selNode != null){
            if(selNode instanceof CircleDrawable){
                CircleDrawable CD = new CircleDrawable(Color.BLUE,new
Point(dropTargetDropEvent.getLocation().x-10,
dropTargetDropEvent.getLocation().y-10),new Dimension(20,20));
                if (jCanvas1.isFree(CD.getRectangle()))
                    jCanvas1.addDrawable(CD);
            }
            if(selNode instanceof RectangleDrawable){
                RectangleDrawable RD = new RectangleDrawable(Color.RED,new
Point(dropTargetDropEvent.getLocation().x-10,
dropTargetDropEvent.getLocation().y-10),new Dimension(20,20));
                if (jCanvas1.isFree(RD.getRectangle()))
                    jCanvas1.addDrawable(RD);
            }
        }

        validate();
        repaint();
    }
});
```

```
//accepter le drop
dropTargetDropEvent.acceptDrop(DnDConstants.ACTION_COPY_OR_MOVE);
}
public void dropActionChanged(DropTargetDragEvent dropTargetDragEvent) {
}
});
//*****
// fin du code a ajouter
//*****

initComponents();
```

build clean, puis on lance l'application :

On a en plus tous les avantages du tutoriel du M. **Christophe Dujardin** qui sont la possibilité de déplacer nos objets et éviter la superposition.

IV - Remerciements

Tout d'abord je remercie **Christophe Dujardin** pour son excellent tutoriel de dessin qui m'a appris pleins de choses, et bien entendu tout ceux qui ont contribué à l'amélioration de ce tutoriel.

Merci

Cordialement

Mhamed Ben Jmaa