

Utilisation de PopUpManager

par Olivier Bugalotto ([Mes articles](#))

Date de publication : 10/03/2008

Ce tutoriel vous expliquera comment réaliser des fenêtres modales à partir de composants existants mais aussi avec des composants personnalisés.

- I - Introduction
- II - Création du composant
- III - Gestion du composant
- IV - Sources

I - Introduction

Nous allons voir dans ce tutoriel comment réaliser des fenêtres modales à partir de composants existants mais aussi avec des composants personnalisés.

Mais avant toute chose qu'est ce qu'une fenêtre modale ? C'est une fenêtre qui se superpose au reste de l'application en figeant cette dernière ; elle est accompagnée le plus souvent d'une question ou d'un formulaire de saisie.

L'utilisation de la classe *PopUpManager* permet la construction de ce type de fenêtre où nous allons pouvoir choisir si cette fenêtre est ou non modale.

Cette classe propose plusieurs méthodes statiques qui vont nous permettre de construire, détruire et manipuler les fenêtres :

PopUpManager
<code>+addPopUp()</code> <code>+bringToFront()</code> <code>+centerPopUp()</code> <code>+createPopUp()</code> <code>+removePopUp()</code>

II - Création du composant

Dans un premier temps, nous allons créer un composant que nous appelons *DialogBox* de la manière suivante :

1. Créons un nouveau composant MXML
2. Nommons-le *DialogBox* et indiquons qu'il est construit sur le composant *TitleWindow*

New MXML component

Create a new MXML component.



Enter or select the parent folder:

CreatePopUpApp

Home ← →

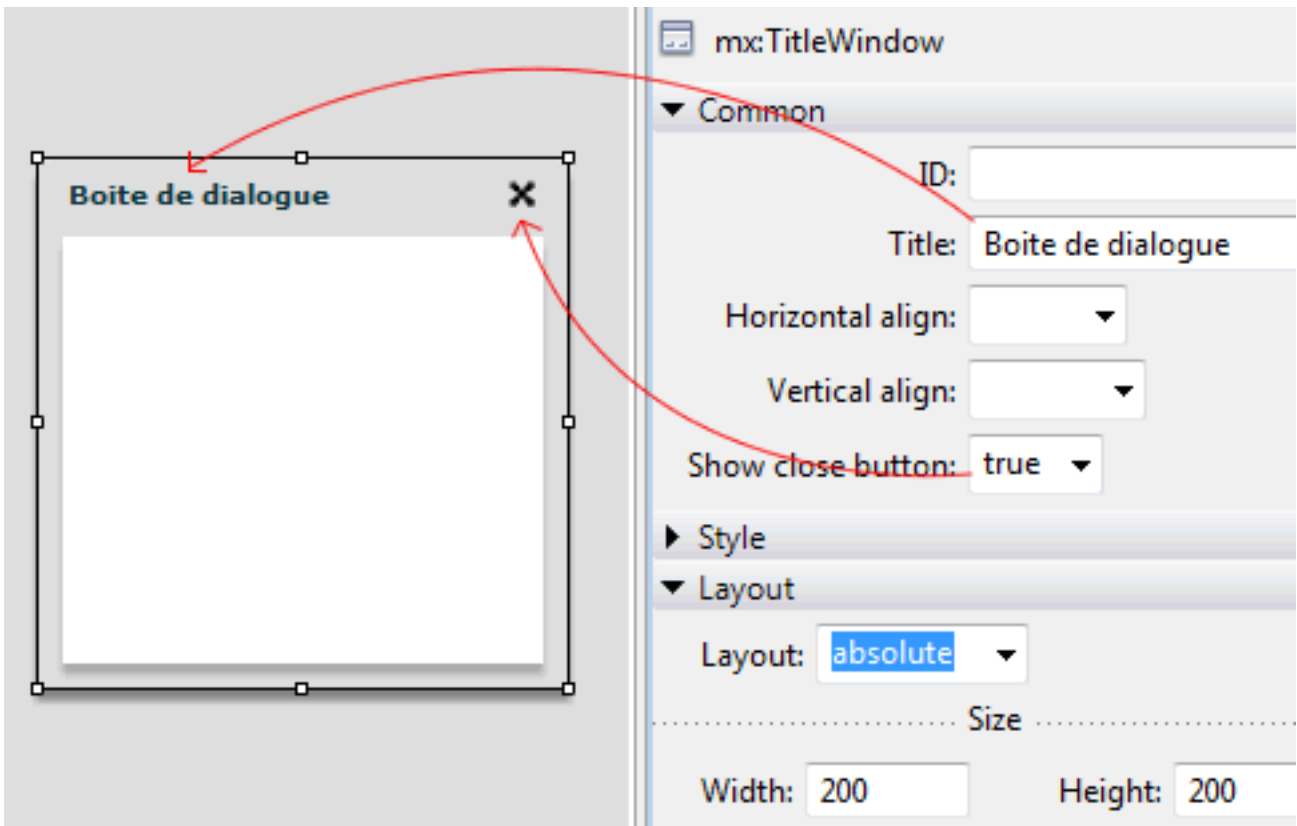
+ CreatePopUpApp

Filename: DialogBox

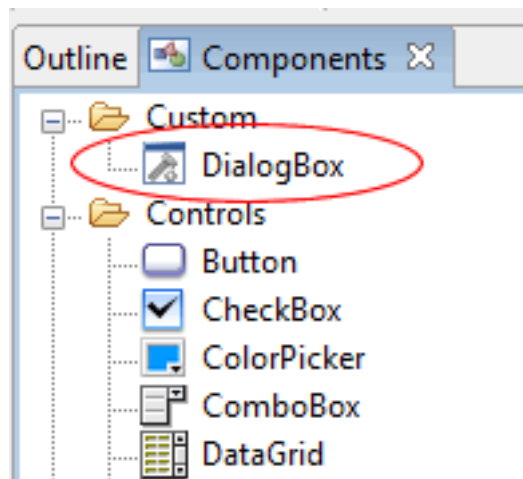
Based on: TitleWindow Layout: absolute

Width: 200 Height: 200

3. Ajoutons un titre à notre composant et un bouton de fermeture



Voilà, notre composant est prêt et disponible dans le panneau custom des composants.



C'est aussi cela la puissance de flex builder de construire des composants rapidement avec une integration facile dans l'environnement. Vous ne le feriez pas aussi facilement en .Net et encore moins en Java.

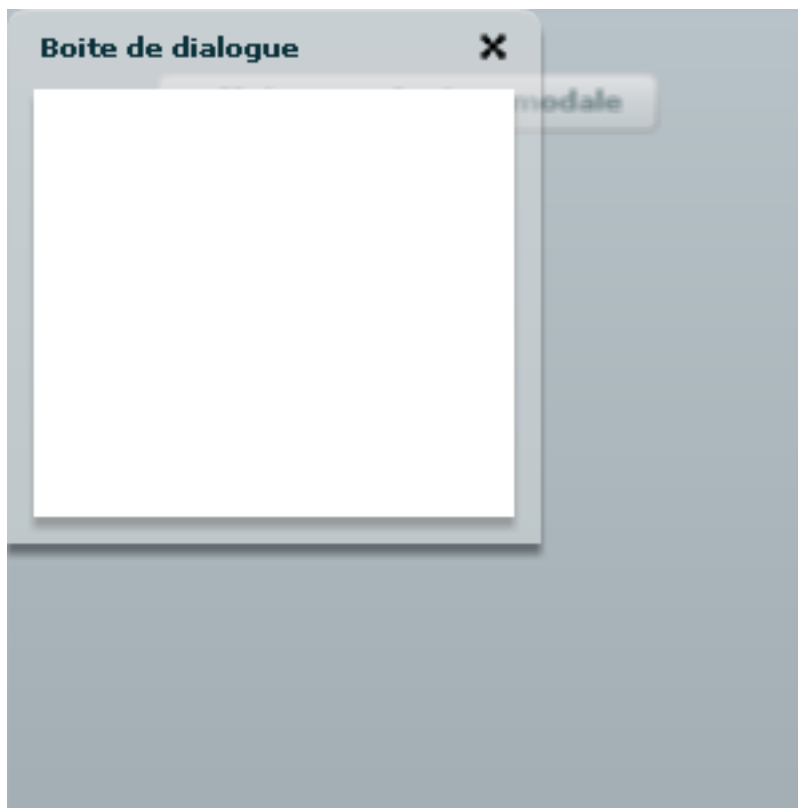
III - Gestion du composant

Lorsque nous voulons créer une fenêtre modale à partir d'un composant *DialogBox*, nous allons utiliser la méthode *createPopUp()*, exemple :

```

/*
 * Creation d'une fenetre modale
 * @param parent DisplayObject Le parent utiliser pour construire la fenetre
 * @param className Class La reference de la classe du composant qui sera utilise comme fenetre
 * @param modal Boolean La fenetre est oui ou non modale
 */
PopUpManager.createPopUp(this,DialogBox,true);

```



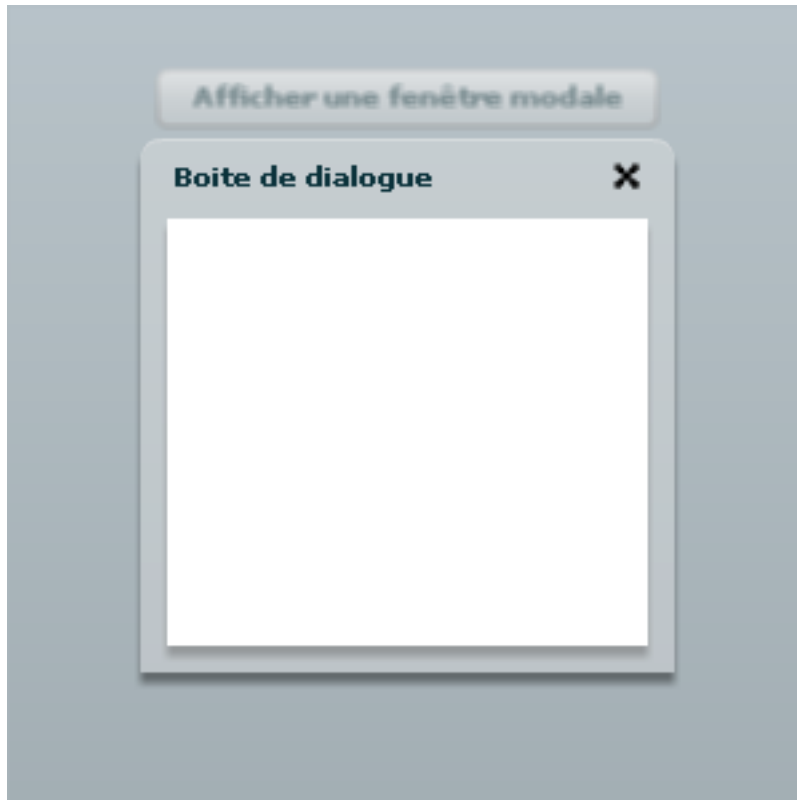
Vous pouvez constater que la fenêtre modale est créée en haut à gauche, ce qui je l'avoue n'est pas très beau. Nous allons pouvoir centrer cette dernière à partir du parent où nous l'avons construite. Nous utilisons pour cela la méthode *centerPopUp()* qui nous demande une instance de la fenêtre nouvellement créée.

Pour ce faire, nous récupérons la référence à partir de la méthode *createPopUp()* :

```

var dBox:IFlexDisplayObject = PopUpManager.createPopUp(this,TitleWindow,true);
PopUpManager.centerPopUp(dBox);

```



i Vous avez pu constater l'utilisation de l'interface *IFlexDisplayObject* qui est autorisée ici puisque tous les composants implémentent cette interface ce qui est donc le cas de notre composant. Nous appliquons une des bases de la POO : le polymorphisme (renvoi vers une référence).

Notre composant ne fait pas grand chose, il nous faut donc pouvoir fermer ce dernier pour revenir à notre application, nous allons pour cela déclencher sa fermeture à partir de l'événement *close* diffusé par le bouton de fermeture (la croix en haut à droite du composant).

```

...
var dBox:IFlexDisplayObject = PopUpManager.createPopUp(this,TitleWindow,true);
PopUpManager.centerPopUp(dBox);
dBox.addEventListener(CloseEvent.CLOSE,onClose);

```

Pour fermer cette fenêtre, nous utilisons la méthode *removePopUp()*. Elle nous demande l'instance de la fenêtre à retirer que nous récupérerons grâce à l'événement :

```

private function onClose(e:CloseEvent):void {
// L'evenement contient la reference du diffuseur (celui a l'origine de l'evenement)
// dans ce cas notre fenetre.
var dBox:IFlexDisplayObject = e.target as IFlexDisplayObject;
PopUpManager.removePopUp(dBox);
}

```

Dans les exemples précédents, vous avez pu constater que c'était la classe *PopUpManager* qui s'occupait d'instancier notre fenêtre mais dans le cas où nous avons déjà une instance sous la main comment aurions-nous pu en faire une fenêtre modale ? Ils ont pensé à tout et notre classe *PopUpManager* nous propose pour cela la méthode *addPopUp()* :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical" width="300"
height="300" xmlns:ns1="*" verticalGap="20">

<mx:Script>
<![CDATA[
import mx.events.CloseEvent;
import mx.core.IFlexDisplayObject;
import mx.managers.PopUpManager;

private var dBox:DialogBox;

private function onShowDialogBox():void {
if(!dBox) {
dBox = PopUpManager.createPopUp(this,DialogBox,true) as DialogBox;
dBox.addEventListener(CloseEvent.CLOSE,onClose);
} else {
PopUpManager.addPopUp(dBox,this,true);
}

PopUpManager.centerPopUp(dBox);
}

private function onClose(e:CloseEvent):void {
PopUpManager.removePopUp(dBox);
}
}]>
</mx:Script>

<mx:Button label="Ouvrir une fenetre" click="onShowDialogBox()"/>

</mx:Application>
```

Comme vous pouvez le voir, la création de fenêtres modales est aisée puisque la charge en revient à *PopUpManager*. Nous pouvons aussi utiliser cette dernière pour créer des fenêtres volantes en mettant le paramètre modal à false (valeur par défaut) :

```
PopUpManager.createPopUp(this,TitleWindow,false);
```

IV - Sources

Vous trouverez les codes sources des exemples ici : **PopUpManager.zip** ([miroir](#))

