

Utiliser les ToolTips sous Flex 2

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

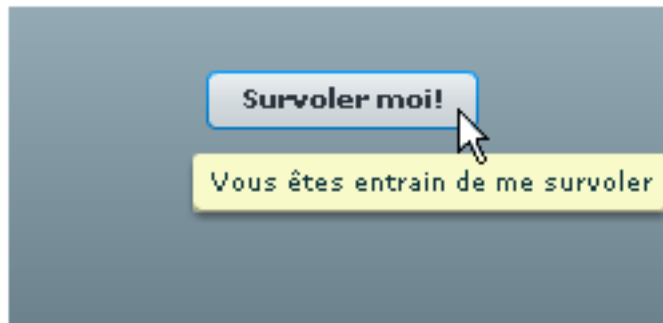
Date de publication : 06/07/2007

Ce tutoriel vous expliquera comment utiliser les infobulles des composants Flex.

Nous allons voir comment utiliser la bulle d'informations sur les composants ou plus communément appelée "ToolTip".

Tous les composants permettent d'afficher des informations dans une bulle (ToolTip) lors de leur survol, à l'aide de la propriété tooltip définie dans la classe UIComponent :

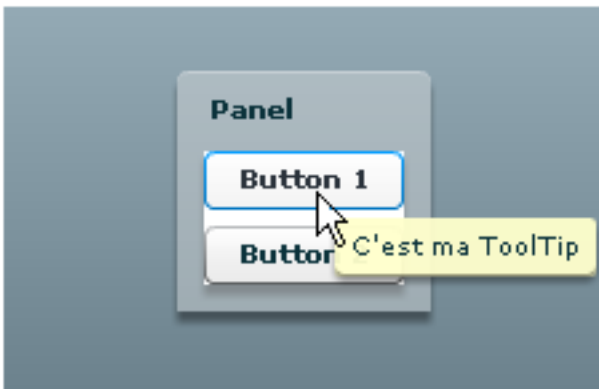
```
<mx:Button label="Survolez moi!" tooltip="Vous etes en train de me survoler" />
```



ToolTip sur un bouton

Si nous appliquons une ToolTip sur un conteneur, se sont les enfants qui l'afficherons. Bien entendu dans le cas ou l'enfant lui même n'as pas de ToolTip, exemple :

```
<mx:Panel title="Panel" tooltip="Je suis la tooltip du conteneur">
  <mx:Button label="Button 1" tooltip="C'est ma Tooltip"/>
  <mx:Button label="Button 2"/>
</mx:Panel>
```



ToolTip sur un container

Nous pouvons modifier l'apparence de la ToolTip, en modifiant la classe de style ToolTip, exemple :

```
<mx:Style>
  ToolTip {
    fontFamily: "Arial";
    fontSize: 16;
    color: white; /* #FFFFFF */
    backgroundColor: red; /* #FF0000 */
    corner-radius: 0;
  }
</mx:Style>
<mx:Button label="Survole moi!" tooltip="Je suis trop belle ..." />
```

Observations : vous remarquerez que nous pouvons indiquer une color par son nom comme en CSS.



Style de Tooltip personnalisée

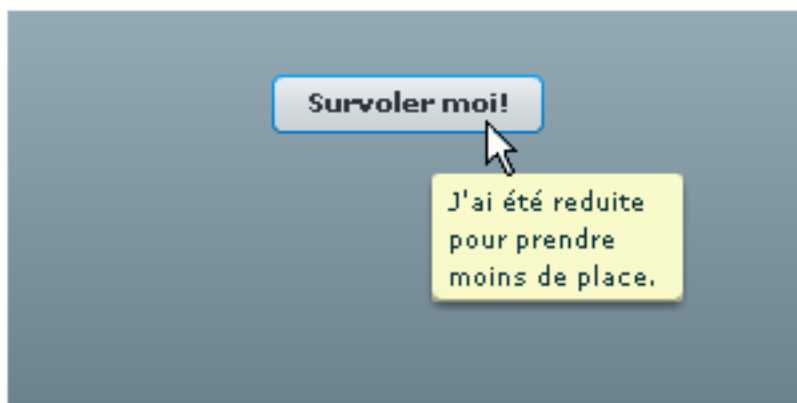
Nous pouvons modifier la largeur d'une Tooltip en passant par la classe Tooltip utilisée pour créer un objet Tooltip (valeur par défaut : 300px), exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
initialize="onInit()">

<mx:Script>
<![CDATA[
import mx.controls.ToolTip;
private function onInit():void {
    Tooltip.maxWidth = 100;
}
]]>
</mx:Script>

<mx:Button label="Survoler moi!"
tooltip="J'ai été reduite pour prendre moins de place."/>

</mx:Application>
```



Largeur de la Tooltip modifiée

Il est aussi possible s'utiliser le caractère d'échappement '\n' en Actionsript et '' dans un attribut d'une balise MXML pour effectuer un retour à la ligne, exemple :

```
<mx:Button label="Survoler moi!"
  tooltip="J'utilise le caractere d'echappement&#13;pour mettre en
  page mon contenu."/>
```

La classe TooltipManager responsable de la création des ToolTips diffusent à des moments précis les évènements suivants :

- **TOOL_TIP_CREATE** : Emit avant la création de la Tooltip
- **TOOL_TIP_END** : Emit lorsque la Tooltip disparaît toute seule
- **TOOL_TIP_HIDE** : Emit lorsque la propriété visible passe à false
- **TOOL_TIP_SHOW** : Emit lorsque la propriété visible passe à true
- **TOOL_TIP_SHOWN** : Emit après que la Tooltip est apparue
- **TOOL_TIP_START** : Emit dès le survole de la souris sur un composant.



Visual application AllEventTooltip

Nous permettant par exemple de déclencher des actions comme la lecture d'un son à l'apparition de la Tooltip :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical"
  initialize="onInit()">

<mx:Script>
  <![CDATA[
    import mx.events.TooltipEvent;

    [Embed(source="medias/Exclamation.mp3")]
    private var ExclamationSound:Class;

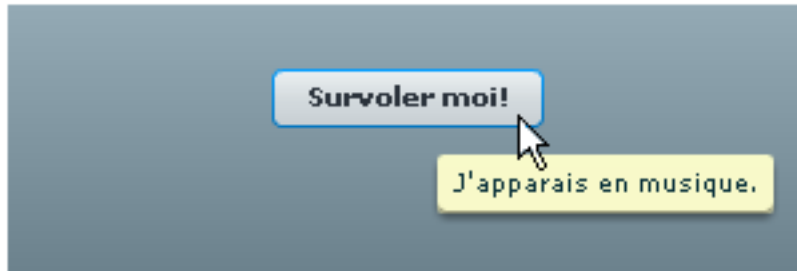
    private var exclamationSnd:Sound;

    private function onInit():void {
      exclamationSnd = new ExclamationSound();
    }
  ]]>
</mx:Script>
```

```

private function onToolTipShow(e:ToolTipEvent):void {
    exclamationSnd.play();
}
]]>
</mx:Script>
<mx:Button label="Survoler moi!"
  tooltip="J'apparais en musique."
  tooltipShow="onToolTipShow(event)" />
</mx:Application>

```



Déclencher un son

Il est possible de désactiver les infos bulle sur les composants à partir de la propriété enabled de ToolTipManager, exemple:

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="horizontal">

  <mx:Style>
    .chkComment {
      up-icon: Embed(source="assets/comment.png");
      over-icon: Embed(source="assets/comment.png");
      down-icon: Embed(source="assets/comment.png");
      disabled-icon: Embed(source="assets/comment.png");

      selected-up-icon: Embed(source="assets/comment_delete.png");
      selected-over-icon: Embed(source="assets/comment_delete.png");
      selected-down-icon: Embed(source="assets/comment_delete.png");
      selected-disabled-icon: Embed(source="assets/comment_delete.png");
    }
  </mx:Style>

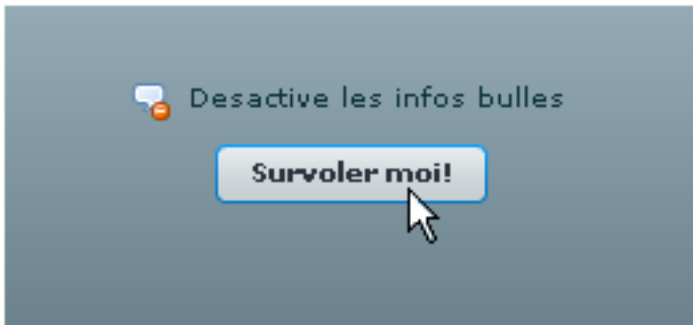
  <mx:Script>
    
      import mx.managers.ToolTipManager;
      private function onEnabledToolTip():void {
        ToolTipManager.enabled = !chkEnable.selected;
      }
    
  </mx:Script>

  <mx:CheckBox id="chkEnable" styleName="chkComment"
    click="onEnabledToolTip()" />
  <mx:Button label="Survoler moi!"
    tooltip="Vous etes en train de me survoler." />

</mx:Application>

```

Pour désactiver la Tooltip, il vous suffit de cliqué sur l'icone commentaire :



Activer/desactiver une Tooltip

Nous pouvons configurer le delai d'affichage d'une Tooltip ainsi que sa durée d'affichage, exemple :

```

<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical" width="275"
height="175">

<mx:Script>
<![CDATA[
import mx.managers.ToolTipManager;
import mx.utils.StringUtil;

private function onSetting():void {
var showDelay:Number = nsShowDelay.value;
var hideDelay:Number = nsHideDelay.value;

ToolTipManager.showDelay = showDelay;
ToolTipManager.hideDelay = hideDelay;
}
]]>
</mx:Script>

<mx:Button id="btn" label="Survoler moi!"
tooltip="Cette Tooltip est configuré@e par vos soins."/>
<mx:Form>
<mx:FormItem label="ShowDelay">
<mx:NumericStepper minimum="0" maximum="10000" stepSize="100"
id="nsShowDelay" change="onSetting()" value="500"/>
</mx:FormItem>
<mx:FormItem label="HideDelay">
<mx:NumericStepper minimum="0" maximum="10000" stepSize="100"
id="nsHideDelay" change="onSetting()" value="10000"/>
</mx:FormItem>
</mx:Form>

</mx:Application>

```



Configuration de la Tooltip

Jusqu'à maintenant, nous avons vu comment configurer et personnaliser une Tooltip, nous allons voir comment en créer.

La création d'une Tooltip peut se faire à l'aide de la méthode `ToolTipManager.createToolTip()`, exemple :

```

<?xml version="1.0"?>
<!-- tooltips/CreatingToolTips.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" width="297" height="127"
horizontalAlign="left">
  <mx:Script>
    <![CDATA[
import mx.managers.ToolTipManager;
import mx.controls.ToolTip;

public var myTip:ToolTip;

private function createToolTip(e:MouseEvent):void {
  var gap:uint = getStyle("verticalGap") as uint;
  var paddingTop:uint = getStyle("paddingTop") as uint;
  trace(gap);
  var s:String = "Cette Tooltip du " + e.target.label +
" est creer a l'aide de la methode createToolTip."
  myTip = ToolTipManager.createToolTip(s,btn1.x + btn1.width + gap,paddingTop) as ToolTip;
  myTip.setStyle("backgroundColor", "white");
  myTip.width = 150;
  myTip.height = 80;
}

private function destroyToolTip(e:MouseEvent):void {
  ToolTipManager.destroyToolTip(myTip);
}
]]>
</mx:Script>
<mx:Button id="btn1" label="Button 1"
rollOver="createToolTip(event)"
rollOut="destroyToolTip(event)"/>
<mx:Button label="Button 2"
rollOver="createToolTip(event)"
rollOut="destroyToolTip(event)"/>
<mx:Button label="Button 3"
rollOver="createToolTip(event)"
rollOut="destroyToolTip(event)"/>
</mx:Application>

```



Créer un Tooltip

Mais il est encore plus intéressant de créer une mise en page différente pour votre Tooltip en utilisant un composant MXML. Il suffit pour cela de créer un nouveau composant MXML et d'implémenter l'interface IToolTip, exemple :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:HBox xmlns:mx="http://www.adobe.com/2006/mxml" implements="mx.core.IToolTip"
alpha=".8"
borderThickness="2"
backgroundColor="0xCCCCCC"
dropShadowEnabled="true"
borderColor="black"
borderStyle="solid">

<mx:Script>
<![CDATA[
[Bindable]
public var contact:Object;

public function get text():String {
return null;
}

public function set text(value:String):void {

}

]]>
</mx:Script>

<mx:Image source="{contact.avatar}"/>
<mx:Form paddingBottom="5" paddingLeft="5" paddingRight="5" paddingTop="5">
<mx:FormItem label="Nom :">
<mx:Label text="{contact.nom}"/>
</mx:FormItem>
<mx:FormItem label="Prenom :">
<mx:Label text="{contact.prenom}"/>
</mx:FormItem>
<mx:FormItem label="Email :">
<mx:Label text="{contact.email}"/>
</mx:FormItem>
</mx:Form>

</mx:HBox>
```

Dans mon cas, je ne me sers pas de la propriété text qui est obligatoire pour implémenter l'interface IToolTip. Venons en maintenant à l'application qui va utiliser ce Tooltip :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical">
```

```
<mx:Script>
  <![CDATA[
    import mx.events.ToolTipEvent;
    private function onCreateToolTip(e:ToolTipEvent):void {
      var tooltip:ContactToolTip = new ContactToolTip();
      tooltip.contact = e.target.data;
      e.tooltip = tooltip;
    }
  ]]>
</mx:Script>

<mx:ArrayCollection id="contacts">
  <mx:Object nom="Dubreuil" prenom="Henry" email="dubreuil@cretin.fr" avatar="assets/avatar1.jpg" />
  <mx:Object nom="Pecchi" prenom="Marcello" email="pecchi@free.fr" avatar="assets/avatar2.jpg" />
  <mx:Object nom="Perez" prenom="Joseph" email="perez@orange.fr" avatar="assets/avatar3.jpg" />
</mx:ArrayCollection>

<mx:Repeater id="rp" dataProvider="{contacts}">
  <mx:Label text="{rp.currentItem.nom + ' ' + rp.currentItem.prenom}"
    tooltip=" " data="{rp.currentItem}"
    tooltipCreate="onCreateToolTip(event)" />
</mx:Repeater>

</mx:Application>
```

Tous les composants diffuse l'évènement createToolTip ce qui permet de récupérer dans l'évènement ToolTipEvent, la référence de la ToolTip que va utiliser ToolTipManager et ainsi de la modifier avec notre propre composant :

```
private function onCreateToolTip(e:ToolTipEvent):void {
  var tooltip:ContactToolTip = new ContactToolTip();
  tooltip.contact = e.target.data;
  // Je passe l'instance de ma ToolTip
  e.tooltip = tooltip;
}
```

Voici donc le rendu :



L'utilisation des ToolTips peut s'avérer très intéressant lorsque de l'information à afficher ne se limite plus à du texte. Vous trouverez dans le zip suivant le code source de ce tutorial (attention j'utilise **Flex builder 3**) : [source.zip](#) ([miroir](#))

