


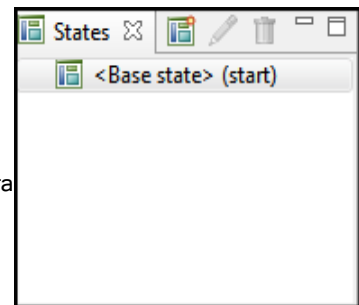
### 3.1. States, Entendendo a disposição dos states

Já vimos como adicionar um painel ao nosso "stage" principal por via de action script chamado através de um botão, vamos agora iniciar uma pequena demonstração de "states". Os states, são os "estados" da nossa aplicação, funcionam como sub-páginas em html as quais chamamos assim que formos precisando delas.

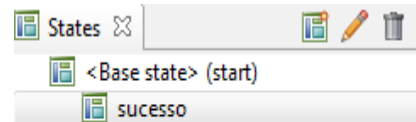
Vamos focar a janela "States" que foi apresentada em cima e mostrada na imagem em baixo.

Nesta janela apenas se encontra no nosso "Base state" ou seja, a base da nossa aplicação e como podem verificar encontra-se com a indicação (start) que indica que este deve ser o painel a apresentar quando iniciar a nossa aplicação.

Vamos criar um novo state, para isso clicamos com o rato no icon  que se encontra nesse painel, é-nos pedido o nome desse novo state, uma opção **based on** que permite escolher a base desse painel, neste momento vamos deixar estar por defeito



(base state) e também tem a opção **"Set as start state"** que ao clicarmos vamos definir esse painel com o principal, mas não vamos seleccionar essa opção por agora. Vamos dar o nome de "sucesso", e manter as opções sem alterar nada e clicar apenas no botão ok. Devemos então ter na janela "States" algo como:



Se carregarem no base state e de novo no sucesso não acontece nada, porque os estados são iguais (o sucesso foi criado como base do base state).

**Selecionamos o state sucesso** e clicamos por exemplo no nosso painel com o

título **Painel adicionado via MXML** e clicamos na tecla DEL, para apagar esse painel, agora se clicarem no "Base state" verão que o painel ainda se mantém lá, mas ao clicar no sucesso esse painel desapareceu. Ou seja, temos criados 2 estados da nossa aplicação. Se executarmos a nossa aplicação agora, veremos que se mantém tudo na mesma, porque o nosso painel que arranca em primeiro é o Base state e não o sucesso. Para arrancarmos o sucesso em primeiro teríamos que o definir como "set as start state", opção que podemos definir clicando no pequeno lápis no painel de states.

Se verificarmos o nosso MXML (código) teremos algo como:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
width="609" height="393">
  <mx:states>
    <mx:State name="sucesso">
      <mx:RemoveChild target="{panell1}"/>
    </mx:State>
  </mx:states>
  <mx:Panel x="107" y="89" width="250" height="200" layout="absolute"
title="Teste olaMundo">
    <mx:Button x="39.5" y="108" label="Olá" click="aoClicar();" />
    <mx:Button x="92.5" y="108" label="Adiciona Painel"
click="addPanel();" />
  </mx:Panel>
  <mx:Script source="olaMundo.as" />
  <mx:Panel x="365" y="89" width="180" height="115" layout="absolute"
title="Painel adicionado via MXML" id="panell1"/>
</mx:Application>
```

se repararem foi adicionado automaticamente o código:

```
<mx:states>
  <mx:State name="sucesso">
    <mx:RemoveChild target="{panel1}"/>
  </mx:State>
</mx:states>
```

que indica que ao entrar nesse state (sucesso), o panel1 (Painel adicionado via MXML) será retirado (RemoveChild).

Vamos então agora adicionar um botão para alterar entre states, reaproveitando o mesmo código para alterar entre os 2 states. O reaproveitamento de código é muito importante em todas as linguagens de programação, porque além de reduzir o tempo de execução reduz também a “confusão” e extensão do nosso código.

Vamos colocar um botão no nosso painel (vou fazê-lo via MXML), para isso no dentro do nosso primeiro `<mx:Panel>` que aparece no código, acrescentamos o seguinte:

```
<mx:Button x="92" y="135" label="Muda estado" width="113.5"
click="mudaEstado()" />
```

De seguida no nosso `olaMudo.as` vamos criar a nossa função, como a função vai ser usada para alternar entre os 2 painéis, teremos que verificar qual deles está “activo” para alternar para o outro, vamos então criar a nossa função:

```
private function mudaEstado():void
{
  if(currentState!="sucesso") currentState="sucesso";
  else currentState="";
  //usamos o currentState="" para mudar para o state definido como (start)
}
```

poderíamos usar código action script directamente no evento click do nosso botão, como por exemplo:

```
click="currentState=&quot;sucesso&quot;;" (os &quot;; significam "") mas com este código não poderíamos utilizar o mesmo botão para mudar de estado, apenas seria utilizado para alterar para o estado “sucesso”.
```

Se correrem agora a vossa aplicação verão que ao carregar no botão Muda Estado, o estado foi alterado para o sucesso (Que não apresenta o painel adicional via MXML), e ao clicar de novo o estado volta ao estado inicial (start).

Uma nota bastante importante, se desejarem alterar alguma coisa em determinado State devem seleccioná-lo primeiro, e quando procederem a alterações devem verificar sempre se as alterações foram feitas no state correcto.

### 3.2. Transições e seus efeitos

Vamos agora falar um pouco de transições, o flex contém alguns efeitos possíveis de fazer nas transições/alternâncias entre os states. Esses efeitos são executados quando se muda de state. Neste caso vamos apenas demonstrar 2 usos de efeitos um em cadeia (sequence) e outro em paralelo (parallel).

A primeira coisa temos que confirmar é a existência de uma identificação (id) nos nossos elementos que vão ser algo destes efeitos de transição. Neste caso vamos usar o primeiro painel **Teste Ola Mundo** e vamos acrescentar o id="panel2" assim:

```
<mx:Panel x="107" y="89" width="250" height="200" layout="absolute" title="Teste
olaMundo" id="panel2">
```

e no **botão olá** dar a identificação "botaoOla" assim:

```
<mx:Button x="39.5" y="108" label="Olá" click="aoClickar();" id="botaoOla"/>
```

E de seguida no final logo antes do </mx:Application> vamos escrever:

```
<mx:transitions>
  <mx:Transition fromState="*" toState="sucesso" id="TransicaoMudar" >
    <mx:Sequence duration="1000" >
      <mx:Glow target="{panel2}" alphaFrom="0" alphaTo="1"
color="#99cc00" />
      <mx:Move yBy="25" target="{botaoOla}" />
      <mx:Glow target="{panel2}" alphaFrom="1" alphaTo="0"
color="#99cc00" />
      <mx:Move yBy="-25" target="{botaoOla}" />
    </mx:Sequence>
  </mx:Transition>
</mx:transitions>
```

Vamos começar por explicar, como este código que coloquei em cima serve apenas para demonstração, usei alguns efeitos e depois repus esses efeitos como de origem.

Iniciamos com **<mx:transitions>** depois podemos colocar mais que uma transição de 2 ou mais estados da nossa aplicação, no nosso caso apenas teremos uma transição propriamente dita, que nos indica com o parâmetro **fromState** e **toState** quando devem ser executados os nossos efeitos, neste caso **fromState="\*" significa** de qualquer state, qualquer alteração de states, com destino ao estado sucesso (**toState="sucesso"**), e de seguida colocamos as acções a efectuar nessa alteração. Aqui podemos utilizar principalmente 2 tipos de efeitos, efeitos sequenciais, um é executado a seguir ao outro, ou efeitos paralelos que são executados todos ao mesmo tempo.

Como indico no código:

```
<mx:Sequence duration="1000" >
  <mx:Glow target="{panel2}" alphaFrom="0" alphaTo="1" color="#99cc00" />
  <mx:Move yBy="25" target="{botaoOla}" />
  <mx:Glow target="{panel2}" alphaFrom="1" alphaTo="0" color="#99cc00" />
  <mx:Move yBy="-25" target="{botaoOla}" />
</mx:Sequence>
```

consegue-se perceber que temos uma sequencia de efeitos, com a duração de 1 segundo (1000 ms), poderíamos definir aqui os objectos destinatários dos efeitos, mas como quero 2 efeitos diferentes, coloquei o destinatário (**target**) em cada um. Será efectuado primeiro um **mx:Glow** no **panel2**, de seguida um **mx:Move** para mover o **botaoOla** verticalmente 25px, logo depois um **mx:Glow** de novo para restaurar a cor original do painel e finalmente para restaurar a posição original do botão um **mx:Move**.

Todos estes efeitos são efectuados por ordem e nenhum é iniciado antes do anterior ter terminado. Vamos agora ver como efectuar a mesma animação, mas ao mesmo tempo e com os mesmos efeitos, para isso misturando efeitos paralelos.

Vamos escrever o nosso código assim:

```
<mx:transitions>
    <mx:Transition fromState="*" toState="sucesso" id="TransicaoMudar" >
        <mx:Sequence duration="1000" >
            <mx:Parallel duration="1000" >
                <mx:Glow target="{panel2}" alphaFrom="0"
alphaTo="1" color="#99cc00" />
                <mx:Move yBy="25" target="{botaoOla}" />
            </mx:Parallel>
            <mx:Parallel duration="1000" >
                <mx:Glow target="{panel2}" alphaFrom="1"
alphaTo="0" color="#99cc00" />
                <mx:Move yBy="-25" target="{botaoOla}" />
            </mx:Parallel>
        </mx:Sequence>
    </mx:Transition>
</mx:transitions>
```

A ordem será:

Na transição de qualquer estado (**fromState="\*"**) para o estado sucesso (**toState="sucesso"**) serão executada uma sequencia (**mx:Sequence**) de efeitos que engloba em primeiro a executado ao mesmo tempo (**mx:Parallel**) de 2 efeitos (**mx:Glow** e **mx:Move**) e de seguida será executada outra serie de efeitos em paralelo para restaurar os efeitos como origem. Os efeitos estão agora agrupados em 2 **mx:Parallel** que serão executado por ordem (**mx:Sequence**), e de notar que os efeitos dentro do **mx:Parallel** são executados ao mesmo tempo e não um a seguir ao outro, como acontecia com o **mx:Sequence**.

## 4. Programação do exemplo olá mundo em Action Script

Iremos agora aprofundar um pouco o conhecimento de action script, bem como a disposição de childs no mais stage e iniciação aos eventListeners.

Vamos criar o painel criado anteriormente (**teste OlaMundo**) via MXML, mas desta vez via Action Script, vamos apenas criar esse painel e não o **state Sucesso** e o **Painel adicionado via MXML**. Vamos então começar um novo action Script File (File-> New -> Action Script File) e vamos dar o nome de **mainScript** e dentro desse script vamos começar por criar uma função para começar colocar os nossos elementos no Stage.

No `mainScript.as` escrevam: `// ActionScript file`

```
import flash.events.MouseEvent;
import mx.containers.Panel;
import mx.controls.Alert;
import mx.controls.Button;
import mx.effects.WipeDown;
import mx.effects.Effect;

private function criaTudo():void {
    //#####
    //declaramos o painel numa nova variavel
    var painel2:Panel = new Panel();
    //definimos os parametros "graficos" do painel
    painel2.x=107;
    painel2.y=89;
    painel2.width=250;
    painel2.height=200;
    painel2.layout="absolute";
    painel2.title="Teste olaMundo";
    painel2.id="panel2";
    //adicionamos como filho do "mainStage"
    this.addChild(painel2);

    //#####
    //vamos criar os mesmos botões: Ola e adiciona painel
    var btnOla:Button = new Button();
    //definimos as propriedades
    btnOla.x=39.5;
    btnOla.y=108;
    btnOla.label="Olá";
    btnOla.id="botaoOla";
    /*adicionamos um Event Listner, algo novo que nunca falei antes, mas como
    o botão não aceita a definição da propriedade .click em action script
    temos que lhe adicionar um "espia" para disparar a função aoClickarEvent
    assim que for clickado pelo rato (MouseEvent.CLICK). no final iremos ver
    como construir esta função aoClickarEvent(). Se repararem ao esceverem
    btnOla.addEventListener apareceram variados tipos de eventListeners para
    adicionar ao botão, esses event listeners são as maneiras possiveis de
    controlar as acções do utilizador nesse botão e chamr funções consoante a
    acção.
    */
    btnOla.addEventListener(MouseEvent.CLICK, aoClickarEvent);
    //agora vamos adicionar este botão como filho (child) do nosso painel2
    //ueremos o botão dentro desse painel.
    painel2.addChild(btnOla);

    //#####
    //vamos criar o botão adiciona Paniel
    var btnAdd:Button = new Button();
    //definimos as propriedades
    btnAdd.x=92.5;
    btnAdd.y=108 ;
    btnAdd.label="Adiciona Painel";
    //adicionamos um event listner no caso do utilizador clicar
    btnAdd.addEventListener(MouseEvent.CLICK, addPanelEvent);
    //adicionamos ao painel como child.
    painel2.addChild(btnAdd);
    //#####
}
```

E temos a função `criaTudo()` que irá criar o nosso painel e os 2 botões, falta-nos agora criar as funções que serão chamadas pelos `eventlisteners` que temos em cima, estas funções serão praticamente iguais às que foram criadas no início deste tutorial no ficheiro `olaMundo.as` mas com uma pequena alteração, a função tem que receber dados de um event, e por isso as funções neste `mainScript.as` ficarão assim:

```
private function aoClickarEvent(event:MouseEvent):void {
    Alert.show("Mundo");
}

private function addPanelEvent(event:MouseEvent):void {
    var novo:Panel = new Panel;
    var efeito:Effect = new WipeDown;
    novo.width=180;
    novo.height=115;
    novo.x=0;
    novo.y=0;
    novo.id="panel3";
    efeito.target=novo;
    this.addChild(novo);
    efeito.play();
}
```

Nas funções em cima estão declarados um `event:MouseEvent` porque foi um listener com um evento/Ação do rato que chamou essa função. Estes `AddListnerEvent` são muito uteis já que nos permitem saber determinadas ações e chamar funções em certos elementos, como `ROLL_OVER`, `ROLL_OUT`, `MOVE`, `RESIZE`, `DRAG`, `DROP` e muitas mais funções.

Agora voltando ao nosso código MXML do `olaMundo.mxml`, vamos apagar tudo e deixar apenas o seguinte:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
width="609 height="393" creationComplete="criaTudo();" >

<mx:Script source="mainScript.as" />

</mx:Application>
```

Estamos a colocar a nossa aplicação vazia, sem qualquer elementos, e chamamos apenas o script para importar as nossas funções para criar o painel e botões. Mas agora para que o painel e os botões sejam criados temos que chamar a nossa função `criaTudo()`, mas como não queremos juntar mais nenhum botão para chamar essa função, e queremos que ela seja logo executada assim que a aplicação arranca, colocamos na tag `<mx:Application>` o parâmetro `creationComplete="criaTudo()"` este parâmetro indica à aplicação que quando estiver "iniciada" e a sua "criação completa" vai chamar a função `criaTudo()`; função esta que vai criar o nosso Painel juntamente com os botões...

Salvem a vossa aplicação e corram-a, se clicarem nos botões, estes devem estar a funcionar como se tivessem sido criados por via de MXML.

## 5-Efeitos e EventListners

Existem dezenas de efeitos possíveis no flex, embora o flex não tenha sido criado com grandes potencialidades de animação, o mesmo às vezes chega a surpreender pela simplicidade de criação de efeitos e animações personalizadas, em baixo vamos falar de alguns efeitos, eventListners e de como aplicar EventListners ao componente dataGrid.

Vamos pegar no nosso exemplo criado em cima, deixando estar tudo como está, e vamos acrescentar ao código, a seguir ao `<mx:script>`, um painel (**painel3**):

```
<mx:Panel x="372" y="48" width="227" height="272" layout="absolute" id="painel3"
title="DestinoEfeitos">
    <mx:DataGrid x="10" y="10" width="187" id="meusDados" >
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1" dataField="col1"/>
            <mx:DataGridColumn headerText="Column 2" dataField="col2"/>
            <mx:DataGridColumn headerText="Column 3" dataField="col3"/>
        </mx:columns>
    </mx:DataGrid>
    <mx:Button x="10" y="183" label="Efeito 1" width="89" id="efeito1"/>
    <mx:Button x="107" y="183" label="Efeito 2" width="90" id="efeito2"/>
</mx:Panel>
```

Painel este com uma dataGrid (**meusDados**), 2 botões de efeitos (**efeito1** e **efeito2**), e vamos aproveitar já para encher com dados a nossa dataGrid com um arrayCollection, que funciona com se fosse uma matriz de dados.

Voltando ao **mainScript.as** vamos scerver a seguir ao "imports" e antes/fora de qualquer função, o seguinte:

```
import mx.collections.ArrayCollection;

[Bindable]
public var dadosDataGrid:ArrayCollection=new ArrayCollection([
    {campo1: "10", campo2: "12.5", campo3: "0.025"},
    {campo1: "1", campo2: "2.5", campo3: "0.115"},
    {campo1: "5", campo2: "1.5", campo3: "0.005"}]);
```

E assim temos o nosso fornecedor de dados para a nossa dataGrid, faltando agora na dataGrid indicar que estes são os dados a apresentar, Colocamos a instrução **[Bindable]** antes da declaração do arrayCollection porque queremos/indicamos que este array pode alterar, e se isso acontecer os objectos dependentes dele deverão ser actualizados. **dataProvider="{dadosDataGrid}"**

Voltamos ao nosso mxml, e na tag `<mx:DataGrid>` colocamos a tag **dataProvider="{dadosDataGrid}"** onde os `{ }` devem ser usados para que se houver alguma alteração nos dados a tabela refrescar os seus dados também, e alterando também nas linhas da tabela `<mx:DataGridColumn>` o dataField para campo1, campo2, campo3 respectivamente, já que são os campos definidos no array de dados.

Ficará algo como:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
width="609" creationComplete="criaTudo()" height="393">

    <mx:Script source="mainScript.as" />
<mx:Panel x="372" y="48" width="227" height="272" layout="absolute" id="painel3"
title="DestinoEfeitos">
    <mx:DataGrid x="10" y="10" width="187" id="meusDados"
dataProvider="{dadosDataGrid}">
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1" dataField="campo1"/>
            <mx:DataGridColumn headerText="Column 2" dataField="campo2"/>
            <mx:DataGridColumn headerText="Column 3" dataField="campo3"/>
        </mx:columns>
    </mx:DataGrid>
    <mx:Button x="10" y="183" label="Efeito 1" width="89" id="efeito1"/>
    <mx:Button x="107" y="183" label="Efeito 2" width="90" id="efeito2"/>
</mx:Panel>
</mx:Application>
```

E se correrem agora a aplicação já tem a dataGrid preenchida com dados.

Vamos agora criar um efeito via MXML para animar o aparecimento da data grid, colocamos a seguir ao `</mx:Panel>` o seguinte:

```
<mx:WipeLeft duration="1000" id="deLado" target="meusDados" />
<mx:Resize target="{[efeito1, efeito2]}" duration="1000" heightBy="20"
id="resizeo" />
```

Temos 2 efeitos, um, **WipeLeft**, com o "target" (objecto a afectar) para a dataGrid **meusDados** e o outro, **Resize**, para os 2 botões de efeitos. Usamos o **creationCompleteEffect** nos componentes para que quando acabarem de ser criados executem os efeitos, efeitos que são identificados com o **id**, neste caso tempos o **id deLado** para o **WipeLeft** e o **resizeo** para o **Resize**. Acrescentamos então o **creationCompleteEffect** na **dataGrid** com o efeito "deLado" nos 2 botões de efeitos com o efeito "resizeo" e ficaremos com algo como isto:

```
<mx:Panel x="372" y="48" width="227" height="272" layout="absolute" id="painel3"
title="DestinoEfeitos">
    <mx:DataGrid x="10" y="10" width="187" id="meusDados"
creationCompleteEffect="{deLado}" dataProvider="{dadosDataGrid}">
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1" dataField="campo1"/>
            <mx:DataGridColumn headerText="Column 2" dataField="campo2"/>
            <mx:DataGridColumn headerText="Column 3" dataField="campo3"/>
        </mx:columns>
    </mx:DataGrid>
    <mx:Button x="10" y="183" label="Efeito 1" width="89" id="efeito1"
creationCompleteEffect="{resizeo}" />
    <mx:Button x="107" y="183" label="Efeito 2" width="90" id="efeito2"
creationCompleteEffect="{resizeo}" />
</mx:Panel>
```

Se correrem a aplicação versão os efeitos na dataGrid e nos botões.

Vamos regressar ao **mainScript.as** e vamos adicionar dentro da função **criaTudo**, antes do **}** final, o seguinte:

```
meusDados.addEventListener(DataGridEvent.HEADER_RELEASE, playEffectEvent);
```

e no mesmo ficheiro, fora de qualquer função, vamos criar uma função como aqui apresento:

```
private function playEffectEvent(event:DataGridEvent):void {
    deLado.play();
}
```

e no topo, a seguir aos imports, adicionar (caso tenha usado copy & past deste documento, caso tenha escrito o **eventLisnter**, esse import será adicionado automaticamente):

```
import mx.events.DataGridEvent;
```

Adicionamos um **eventLisnter** para chamar a função **playEffectEvent** se o cabeçalho da **dataGrid** fosse "released" (quando clicamos e largamos o botão do rato), e na função chamamos o efeito que criámos antes e iniciamo-lo: **deLado.play()**;

Vamos adicionar mais um **eventLisnter** na função **criaTudo()** a seguir ao que criámos, algo como:

```
meusDados.addEventListener(ListEvent.ITEM_CLICK, itemListEvent);
```

e no topo a seguir ao ultimo import:

```
import mx.events.ListEvent;
```

e por final, criamos outra função:

```
private function itemListEvent(event>ListEvent):void {
    Alert.show("Item clickado!");
}
```

Por final, se correrem a vossa aplicação, além dos efeitos iniciais, são executados também os efeitos **deLado** ao clicar no título das colunas da **dataGrid**, e também é apresentado um **Alert** chamado pelo evento click na lista.

Recapitulando os ficheiros criados:

**olaMundo.mxml:**

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" width="609"
creationComplete="criaTudo()" height="393">
<mx:Script source="mainScript.as" />
<mx:Panel x="372" y="48" width="227" height="272" layout="absolute" id="painel3"
title="DestinoEfeitos">
    <mx:DataGrid x="10" y="10" width="187" id="meusDados" creationCompleteEffect="{deLado}"
dataProvider="{dadosDataGrid}">
        <mx:columns>
            <mx:DataGridColumn headerText="Column 1" dataField="campo1"/>
            <mx:DataGridColumn headerText="Column 2" dataField="campo2"/>
            <mx:DataGridColumn headerText="Column 3" dataField="campo3"/>
        </mx:columns>
    </mx:DataGrid>
    <mx:Button x="10" y="183" label="Efeito 1" width="89" id="efeito1"
creationCompleteEffect="{resize}" />
    <mx:Button x="107" y="183" label="Efeito 2" width="90" id="efeito2"
creationCompleteEffect="{resize}" />
</mx:Panel>

    <mx:WipeLeft duration="1000" id="deLado" target="meusDados" />
    <mx:Resize target="{[efeito1, efeito2]}" duration="1000" heightBy="20" id="resize" />
</mx:Application>
```

## mainScript.as:

```
import flash.events.MouseEvent;

import mx.collections.ArrayCollection;
import mx.containers.Panel;
import mx.controls.Alert;
import mx.controls.Button;
import mx.effects.Effect;
import mx.effects.WipeDown;
import mx.events.DataGridEvent;
import mx.events.ListEvent;

[Bindable]
public var dadosDataGrid:ArrayCollection=new ArrayCollection([{campo1: "10", campo2: "12.5", campo3:
"0.025"}, {campo1: "1", campo2: "2.5", campo3: "0.115"},{campo1: "5", campo2: "1.5", campo3:
"0.005"}]);

private function criaTudo():void {
    #####
    //declaramos o painel numa nova variavel
    var painel2:Panel = new Panel();
    //definimos os parametros "graficos" do painel
    painel2.x=107;
    painel2.y=89;
    painel2.width=250;
    painel2.height=200;
    painel2.layout="absolute";
    painel2.title="Teste olaMundo";
    painel2.id="panel2";
    //adicionamos como filho do "mainStage"
    this.addChild(painel2);
    #####
    //vamos criar os mesmos botões: Ola, muda estado e adiciona painel
    var btnOla:Button = new Button();
    //definimos as propriedades
    btnOla.x=39.5;
    btnOla.y=108;
    btnOla.label="Olá";
    btnOla.id="botaoOla";
    /*adicionamos um Event Listener, algo novo que nunca falei antes, mas como o botão não
    aceita a definição da propriedade .click em action script temos que lhe adicionar um "espia"
    para disparar a função aoClickarEvent assim que for clicado pelo rato (MouseEvent.CLICK).
    no final iremos ver como construir esta função aoClickarEvent(). Se repararem ao esceverem
    esses event listeners são as maneiras possíveis de controlar as acções do utilizador nesse botão e
    chamr funções consoante a acção.*/
    btnOla.addEventListener(MouseEvent.CLICK, aoClickarEvent);
    //agora vamos adicionar este botão como filho (child) do nosso painel2
    //ueremos o botão dentro desse painel.
    painel2.addChild(btnOla);
    #####
    //vamos criar o botão adiciona Painel
    var btnAdd:Button = new Button();
    //definimos as propriedades
    btnAdd.x=92.5;
    btnAdd.y=108 ;
    btnAdd.label="Adiciona Painel";
    //adicionamos um event listener no caso do utilizador clicar

    btnAdd.addEventListener(MouseEvent.CLICK, addPanelEvent);
    //adicionamos ao painel como child.
    painel2.addChild(btnAdd);

    #####

    meusDados.addEventListener(DataGridEvent.HEADER_RELEASE, playEffectEvent);
    meusDados.addEventListener(ListEvent.ITEM_CLICK, itemListEvent);
}

private function itemListEvent(event>ListEvent):void {
    Alert.show("Item clickado!");
}

private function playEffectEvent(event>DataGridEvent):void {
    deLado.play();
}
```

```
private function aoClickarEvent (event:MouseEvent):void {
    Alert.show("Mundo");
}

private function addPanelEvent (event:MouseEvent):void {
    var novo:Panel = new Panel;
    var efeito:Effect = new WipeDown;
    novo.width=180;
    novo.height=115;
    novo.x=0;
    novo.y=0;
    novo.id="panel3";
    efeito.target=novo;
    this.addChild(novo);
    efeito.play();
}
```

Esta terceira parte do tutorial está terminada, na próxima parte falarei de como criar componentes e módulos, e importá-los/chamá-los para a nossa aplicação, e se não se tornar muito extenso falarei também da iniciação e instalação do AMFPHP para futuro exemplo de comunicação com um banco de dados.