

2. Entendendo a ordenação e estruturação do código.

Bom, quero antes de mais lembrar que vamos usar por enquanto o exemplo criado anteriormente.

Tínhamos criado a nossa primeira aplicação... bom, neste exemplo vou aprofundar mais um pouco da estrutura e organização de código... já antes falei um pouco, mas agora vou exemplificar mais um pouco...

O que aconselho, é estruturar e organizar bem o código por uma questão de fácil identificação de erros, e também por uma questão de desempenho

Exemplo de mau código de um script:

```
<mx:Script>
<![CDATA[
    private function clickado():void {
        var x:String;
        x="Olá";
import mx.controls.Alert;
Alert.show(x+"Mundo");
    }
    ]]>
</mx:Script>
```

Bom, se alguns de vocês estão habituados a boas práticas de programação, já repararam que podia ter simplificado bastante esta função começando pela tabulação adequada do código (estilo tree).

Não vou estar a aprofundar muito este ponto, porque tornar-se-ia muito extenso e até porque o próprio flex builder informa muitos destes erros.

Ora reparem:

```
<mx:Script>
<![CDATA[
    import mx.controls.Alert;
    private function clickado():void
    {
        var x:String = "Olá Mundo";
        Alert.show(x);
    }
    ]]>
</mx:Script>
```

Como podem ver, as coisas simplificaram-se muito... principalmente a nível de erros (não se deve usar o import dentro de uma função) porque seria importado cada vez que a função for chamada sobrecarregando o aplicativo e invalidando o código.

Depois, ao declarar a string, podemos logo atribuir-lhe um valor, e neste caso como queríamos apresentar um alerta com a frase "Olá Mundo" declaramos logo a variável com esse valor, e dentro do alert chamamos a variável... podia ainda simplificar mais, usar somente o `Alert.show("Olá Mundo");` mas não o fiz para vocês aprenderem já a declarar variáveis :)

Como devem ter reparado (se não fizeram copy/paste) ao escrever "var x:" apareceu uma lista com todos os tipos de variáveis que podem ser criadas/declaradas. O código foi organizado também em forma de "tree", devidamente tabulado e estruturado.

Uma das grandes diferenças da maior parte das linguagens é que podemos atribuir objectos a uma variável, por exemplo: `var x:Panel;`

só isto não funcionaria devidamente porque esse componente não existe no script, nem foi devidamente atribuído. Este tipo de declarações de variáveis como objectos é muito útil, já que podemos inserir objectos no nosso layout por via de Action Script (já devem estar a imaginar centenas de formas úteis e essenciais que tornam isto muito importante), mas falarei disto mais à frente.

Já falámos da má organização de scripts, agora vamos falar um pouco da má organização de MXML.

Imaginem no nosso antigo código, que criamos no tutorial anterior:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
    <mx:Panel x="126" y="114" width="250" height="200" layout="absolute" title="Teste olaMundo">
        <mx:Button x="92.5" y="128" label="Olá" click="aoClickar()"/>
    </mx:Panel>
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;

            private function aoClickar():void {
                Alert.show("Mundo");
            }
        ]]>
    </mx:Script>
</mx:Application>
```

Imaginem que queremos outra função? não devemos criar de novo outro `<mx:Script>`, mas sim usar o existente para a criação de outra função, mesmo assim existem alguns pequenos pormenores que muitas vezes falham. Se não usarmos a organização como em cima, torna-se mais complicado, por exemplo, identificar devido painel, ou botão.

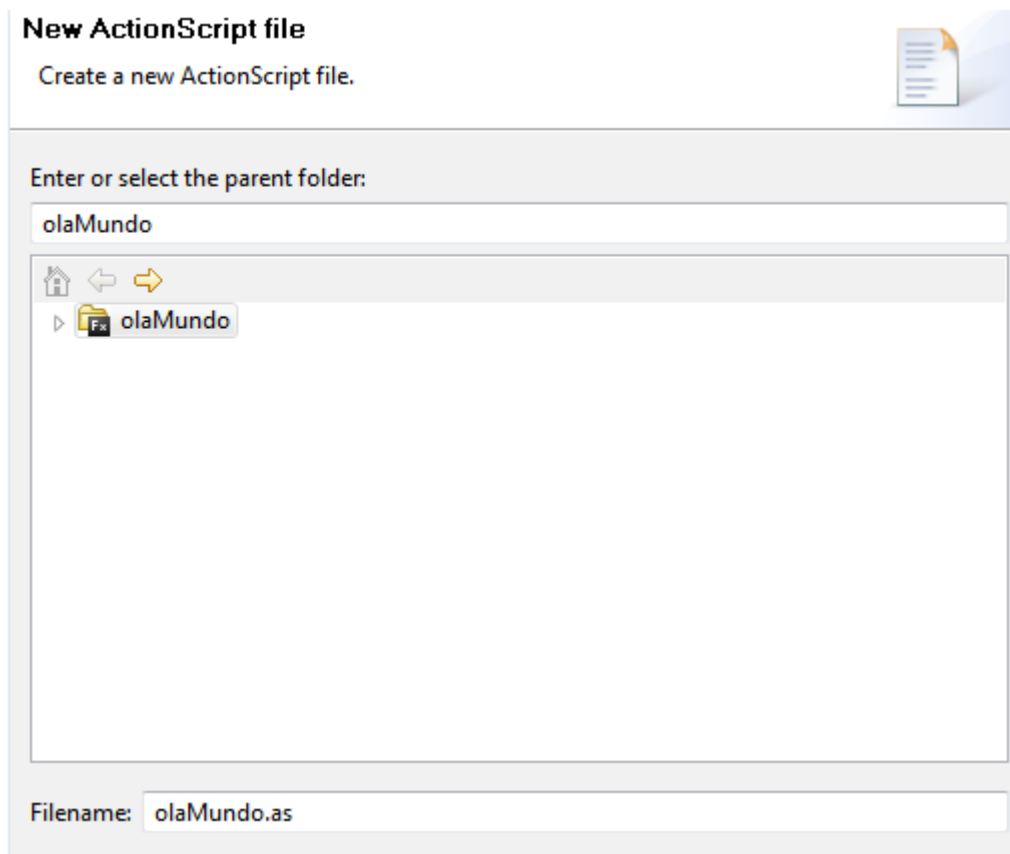
Bom, a nível de organização de MXML pouco mais há a dizer a não ser usar as boas práticas e "tabular" devidamente o código, para fácil uso, e não usar por exemplo a importação de componentes que não sejam necessários na aplicação, por exemplo, o uso do `import mx.controls.Alert;` e não depois não o usarmos no nosso código. (isto é uma falha muito comum, e que torna a compilação muito mais demorada e mais sujeita a erros).

Podemos simplificar mais ainda a criação dos nosso aplicativo a nível de código, criando para isso um documento à parte para todo o conteúdo Action Script que vou passar a explicar. (neste caso não é necessário o uso de um ficheiro externo contendo o nosso action script, visto que se trata de de um script muito simples, mas para scripts extensos é a melhor opção).

2. Criação de um Ficheiro Action Script Externo para uso no Flex.

Vão ao menu "File" -> "New" -> "ActionScript File"

Na janela que se irá abrir, coloquem em FileName (ao fundo) "olaMundo" como na figura em baixo:

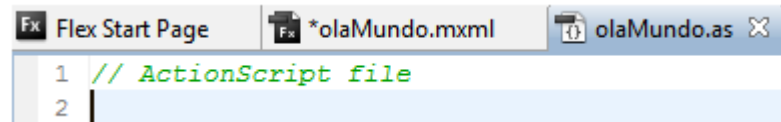


Clicam em "Finish".

A nossa área de desenho abriu agora o nosso recente criado “olaMundo”, contendo apenas `//ActionScript File` e do nosso lado esquerdo, no “Flex Navigator” foi criado o nosso olaMundo.as (a extensão indica que é um ficheiro ActionScript).

Como este exemplo serve para mostrar como criar o nosso script num ficheiro à parte e usa-lo no nosso aplicativo, vamos limitar-nos a copiar o nosso script para este recente criado olaMundo.as

Se repararem, no topo da área de desenho estão abertos os nosso ficheiros do projecto que estamos a trabalhar como na figura ao lado.



Vamos clicar no olaMundo.mxml, e cortar o nosso script do “Source” e cola-lo no olaMundo.as, ficaríamos com o seguinte código no olaMundo.mxml:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Panel x="126" y="114" width="250" height="200" layout="absolute" title="Teste
olaMundo">
    <mx:Button x="92.5" y="128" label="Olá" click="aoClickar();"/>
  </mx:Panel>
  <mx:Script>
    <![CDATA[
      ]]>
  </mx:Script>
</mx:Application>
```

e no nosso olaMundo.as :


```
import mx.controls.Alert;

private function aoClickar():void
{
    Alert.show("Mundo");
}
```

Agora o que temos a fazer, é chamar o nosso script externo olaMundo.as no nosso olaMundo.mxml, para isso voltamos ao “Source” do olaMundo.mxml e na “tag” `<mx:Script>` vamos remover todo o seu conteúdo e remover também a “tag” `</mx:Script>` e colocar o seguinte:

```
<mx:Script source="olaMundo.as" />
```

notem que indicamos na “tag” o parâmetro “source” a indicar onde se encontra o código fonte desse script, e no final colocamos o caracter “/” porque não utilizaremos qualquer script na sua área.

Salvem os documentos. (olaMundo.mxml e olaMundo.as) no menu "File" -> "Save All" ou com as teclas de atalho CTRL+SHIFT+S. Após salvarem os vossos ficheiros clicam no botão  para compilar a vossa aplicação, se tudo correu bem devem ter o mesmo exemplo a correr, mas agora usando o script importado.

Isto é muito importante, já que no desenvolver de uma aplicação de tamanho médio/grande podemos separar o nosso código action script para mais fácil identificação e para futuramente podermos usar também esses scripts em outros projectos.

3. Entendendo os componentes internos, states e transições/efeitos.

Uma das partes mais difíceis da arquitectura do flex é a sua organização interna de "Packages"/"Componetes", como já disse e exemplifiquei anteriormente, para usarmos o Alert, temos antes de o importar para a nossa aplicação para ele estar disponível para uso, caso contrário ao compilar seriam apresentados erros.

Bom, todos os componentes disponíveis no modo gráfico, na janela componentes, estão também disponíveis em MXML e em Action Script, mas aqui já com algumas alterações para que sejam apresentados no nosso layout.

Vamos ver um exemplo para adicionar um novo painel na área de desenho pelo modo de MXML e pelo modo de Action Script. Usando o nosso código:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
  <mx:Panel x="126" y="114" width="250" height="200" layout="absolute"
title="Teste olaMundo">
    <mx:Button x="92.5" y="128" label="Olá" click="aoClickar();" />
  </mx:Panel>
  <mx:Script source="olaMundo.as" />
</mx:Application>
```

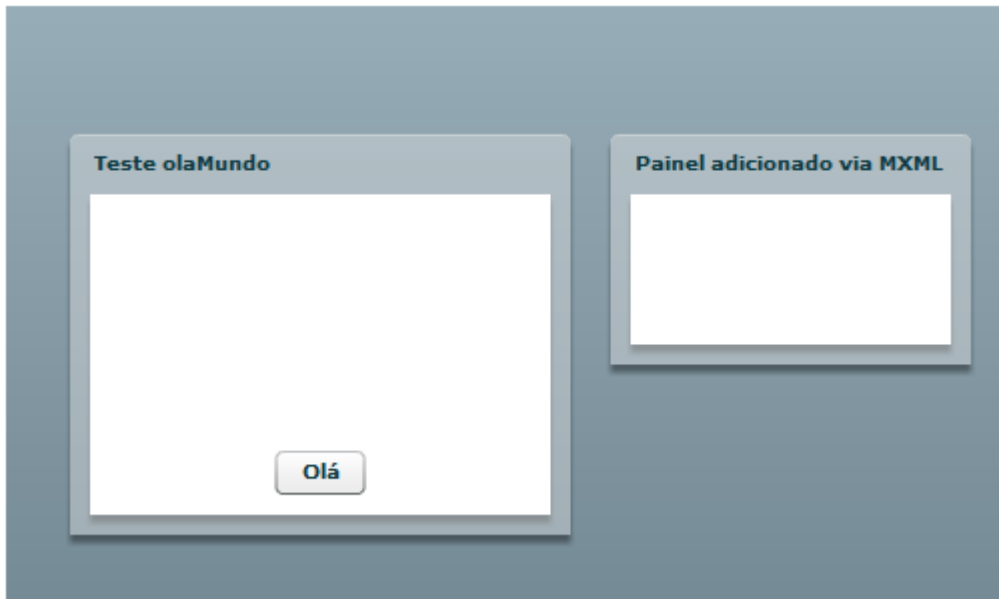
Vamos alterar o nosso tamanho da nossa aplicação para se adaptar ao tamanho do browser para que possamos ver melhor a nossa área, na "tag" <mx:Application ..> vamos escrever/alterar os seguintes parâmetros width e height para 100%, depois de alterado ficaria algo como:

```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
width="100%" height="100%">
```

de seguida vamos escrever o seguinte a seguir à nossa "tag" <mx:Script> a antes da <mx:Application> o seguinte:

```
<mx:Panel x="400" y="114" width="180" height="115" layout="absolute" title="Painel adicionado via MXML" />
```

Reparem que coloquei a "/" no final, o que quer dizer que o nosso painel para já não terá qualquer conteúdo, de seguida vejam o modo "Design" e verifiquem que o nosso painel já foi adicionado, ficando algo como a figura em baixo:



Agora vamos adicionar um terceiro painel, mas desta vez via action script, e só irá aparecer no nosso layout quando clicamos no botão olá.

No nosso olaMundo.as vamos aproveitar o nosso já criado olaMundo.as e alem de mostrar um novo painel vamos já adicionar um pequeno efeito (WipeDown) na altura da criação desse painel, um efeito disponível como package no flex. Escrevam o seguinte no olaMundo.as a seguir ao nosso import do Alert:

```
import mx.containers.Panel; //vamos importar o componente painel
import mx.effects.Effect; //importar a definição effect
import mx.effects.WipeDown; //importar o efeito
```

à seguir à função aoClickar(); criamos uma nova função addPanel(); :

```
private function addPanel():void
{
    var novo:Panel = new Panel; //declaramos a variavel novo como painel
    var efeito:Effect = new WipeDown; //decl. Efeito com novo wipeDown effect
    novo.width=180; //comprimento
    novo.height=115; //altura
    novo.x=0; //posição x onde o painel vai aparecer
    novo.y=0; //posição y onde o painel vai aparecer
    novo.id="panel3"; //identificamos o painel como "panel3"
    efeito.target=novo; //definimos o destino/target do efeito wipedown
    this.addChild(novo); //adicionamos como "filho" do "stage"/Mostra painel
    efeito.play(); //e finalmente iniciamos o efeito wipeDown
}
```

Se lerem atentamente, e ao escreverem, verão muitas das funções disponíveis, bem como componentes e efeitos, mas restringam-se apenas à função em cima.

Lembrem-se que temos sempre que importar os componentes/efeitos/instancias sempre que as pretendemos usar, mas apenas uma vez. O nosso `olaMundo.as` deve estar algo como:

```
// ActionScript file
import mx.controls.Alert;
import mx.containers.Panel;
import mx.effects.Effect;
import mx.effects.WipeDown;

private function aoClickar():void
{
    Alert.show("Mundo");
}

private function addPanel():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();
}
}
```


Agora no nosso `olaMundo.mxml` vamos criar um novo botão via MXML para chamar a nossa função para criar o novo panel, escrevam dentro do nosso primeiro panel, em baixo da "tag" do nosso botão "Olá" o seguinte:

```
<mx:Button x="80" y="148" label="Adiciona Painel" click="addPanel();"/>
```

O vosso código deve estar como 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">
    <mx:Panel x="107" y="89" width="250" height="200" layout="absolute"
title="Teste olaMundo">
        <mx:Button x="92.5" y="128" label="Olá" click="aoClickar();"/>
        <mx:Button x="80" y="148" 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" />
</mx:Application>
```

Devemos ter agora um novo botão "Adiciona Painel" no nosso layout que ao ser clickado chama a função `addPanel()`; que criamos anteriormente.

Salvem os vossos ficheiros (CTRL+SHIF+S) e corram a aplicação ()

Devem estar disponíveis apenas 2 painéis, o nosso painel “Olá Mundo” e o “Painel criado via MXML”.

Se clicarem no botão “Adiciona Painel” verão a criação de um novo painel, via Action Script com o efeito WipeDown do flex. Cada vez que clicarem será adicionado um novo painel, na mesma posição, com o mesmo efeito.

Claro que num possível uso desta função, e querendo só um novo painel, teríamos que controlar os cliques no botão e verificar se o painel já existe para evitar a duplicação do painel com o mesmo ID.

Falarei mais à frente sobre como evitar duplicação de painéis e eliminação dos mesmos.