SIMULADOS & TUTORIAIS

TUTORIAIS FLASH

O Componente Button

O Componente Button

Introdução

Esse componente é parecido com um botão convencional criado pelo usuário. Embora seja possível alterar suas dimensões, como largura e altura, não é possível editá-lo e modificar seus estágios como fazemos com um botão convencional.
Mas, você poderá adicionar um ícone personalizado a esse componente, utilizando suas propriedades existentes no ActionScript. Ele também permite selecionar as duas formas de comportamento existentes, tais como, **clicar** ou utilizá-lo como um interruptor **liga-desliga**.
Um componente **Button** pode estar habilitado ou desabilitado. No estado desabilitado, ele não poderá receber nenhuma entrada do mouse ou teclado. Quando habilitado, ele receberá o foco quando clicado com o mouse ou selecionado com a tecla **TAB**. Quando o componente **Button** estiver com o foco você poderá utilizar as seguintes teclas para controlá-lo:

Tabela 3.1 – Teclas de controle para o componente Button.

Tecla	Descrição
Shift + Tab	Move o foco para o objeto anterior.
Barra de Espaço	Pressiona ou libera o componente.
Tab	Move o foco para o próximo objeto.

Os parâmetros do componente Button

Quando você adiciona um componente **Button** na sua aplicação, são apresentados alguns parâmetros no **Inspetor de Propriedades**, que você deverá configurá-los para que você possa utilizá-los corretamente, conforme mostramos abaixo:

Tabela 3.2 – Parâmetros do componente Button.

Parâmetro	Descrição
emphasized	Um valor booleano que indica se uma borda deverá ser mostrada em volta do componente quando seu estado for "**para cima**". O valor padrão é: **false**.
enabled	Esse parâmetro é um valor booleano que indica se o componente deverá estar habilitado ou desabilitado quando a aplicação for executada. O valor padrão é **true**.
label	Esse parâmetro indica o rótulo do botão, ou seja, um identificador. O valor padrão é: **label**.
labelPlacement	Utilize esse parâmetro para posicionar o rótulo do botão (parâmetro **label**) em relação ao ícone inserido no botão. Ele permite uma das seguintes posições: **left** (esquerda), **right** (direita), **top** (em cima) ou **bottom** (em baixo). O valor padrão é: **right**.
selected	Se o parâmetro **toggle** for configurado para **true**, esse parâmetro indica que o botão está pressionado (**true**), caso contrário, estará liberado (**false**). O valor padrão é **false**.
toggle	Esse parâmetro é utilizado para tornar o componente **Button** uma espécie de chave **liga-desliga**. Se for configurado para **true**, o botão permanece no estágio "**para baixo**" quando clicado com o mouse. Volta ao estado normal quando pressionado novamente. Se for selecionado **false** para esse parâmetro, o botão funciona como um botão convencional. O valor padrão é **false**.
visible	Esse parâmetro é um valor booleano que indica se o componente deverá aparecer visível ou oculto quando a

	aplicação for executada. O valor padrão é: **true**.

Além desses parâmetros, você poderá também encontrar métodos e eventos para esse componente na linguagem ActionScript que você poderá utilizá-los em tempo de execução e criar aplicações mais robustas, conforme descrevemos a seguir:

Os métodos do componente Button

São os seguintes os métodos disponíveis para o componente **Button**:

Método	Descrição
Button	Cria uma nova instância do componente **Button**.
getStyleDefinition	Recupera o estilo de mapa padrão para o componente atual.

Os eventos do componente Button

Eventos são ações utilizadas pelos componentes para executar tarefas específicas, de acordo com a finalidade de cada situação. Veja a seguir os eventos disponíveis para esse componente e para que servem.

Evento	Descrição
buttonDown	É executado quando o usuário pressiona o componente **Button**. Se a propriedade **autoRepeat** estiver configurada para **true**, este evento é disparado em intervalos específicos até o botão ser liberado.
change	É executado quando o valor da propriedade selecionada de um componente **Button** do tipo **liga-desliga** for alterado. Um componente **Button** do tipo **liga-desliga** é aquele cuja propriedade **toggle** está configurada para **true**.
click	É executado após o botão do tipo **liga-desliga** receber entrada com o uso do mouse ou da barra de espaço.

Veja como utilizar esses recursos nos exemplos mais adiante neste capítulo e no DVD, que acompanha a presente obra.

Criando aplicações com o componente Button

Um botão é parte fundamental de qualquer formulário ou aplicação, tanto na Web quanto em uma aplicação desktop. Você poderá utilizá-lo para diversas finalidades, tais como, enviar um formulário para um servidor qualquer, exibir as imagens em um slide show, responder a algum questionário, e assim por diante.

Para entender melhor como funciona esse componente, mostraremos a seguir alguns exemplos práticos, inclusive em conjunto com outros componentes.

Exemplo 1

Nesse exemplo utilizaremos um componente **Button** como um botão convencional e um componente **Label**, de forma que quando o botão for clicado, seja mostrado no **Label** um número aleatório entre 1 e 50. Para isso, utilizaremos a função **Math.random()**. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **Button** para o palco. Veja sua aparência na **Figura 3.1** a seguir:

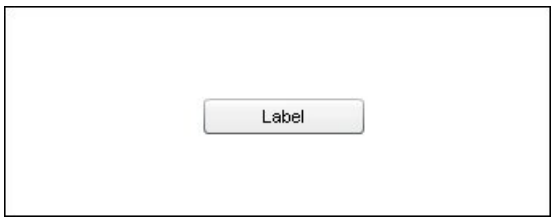

Figura 3.1 – Componente Button original.

3. Abra o painel **Component Inspector** (**Shift + F7**), e altere o parâmetro **label** para: **Sortear Número**;
4. Para que ele funcione como um botão normal, mantenha o parâmetro **toggle** como **false**;
5. No painel **Properties**, digite um nome para sua instância. Por exemplo: **sortear**;
6. Crie um campo dinâmico logo abaixo do botão;
7. No painel **Properties**, na caixa de texto **Instance Name**, digite: **valor**, para esse campo;
8. Nomeie essa camada de: **componentes**;
9. Crie uma nova camada e chame-a de: **ações**;
10. No primeiro frame dessa camada insira o seguinte código:

```
stop();
import flash.events.MouseEvent;
sortear.addEventListener(MouseEvent.CLICK, randomico);
//
function randomico(event:MouseEvent):void {
    var numero = Math.floor((Math.random()*50+1));
    valor.text = numero;
}
```

Vejamos os comentários sobre o código:

> **OBS**.: Em todos os exemplos deste tutorial incluiremos o comando **stop()** na primeira linha do código. Esse comando tem como objetivo interromper a aplicação quando a mesma for executada, mesmo que a aplicação não possua mais de um frame na sua linha do tempo. Consideramos um bom hábito essa prática. Portanto, não comentaremos mais essa linha nos próximos exemplos.

Na linha:

```
import flash.events.MouseEvent;
```

Importamos a classe de eventos do mouse necessária para a nossa aplicação.

O bloco de código logo abaixo:

```
sortear.addEventListener(MouseEvent.CLICK, randomico);
//
function randomico(event:MouseEvent):void {
    var numero = Math.floor((Math.random()*51));
    valor.text = numero;
}
```

Criamos um evento **CLICK** vinculado ao botão **sortear**, de forma que quando o mesmo for clicado, a função **randomico** logo adiante deverá ser executada. Em seguida, criamos essa função cuja finalidade é mostrar no texto dinâmico **valor** um número randômico toda vez que o botão for clicado. A variável **numero** recebe um valor randômico entre 1 e 50 a partir da função matemática **Math.random()**. Para que o resultado não apresente casas decimais, utilizamos também a função **Math.floor** para arredondar o valor gerado sempre para menor.

11. Execute a aplicação (**Ctrl + Enter**) e confira o resultado com o mostrado na **Figura 3.2** abaixo:

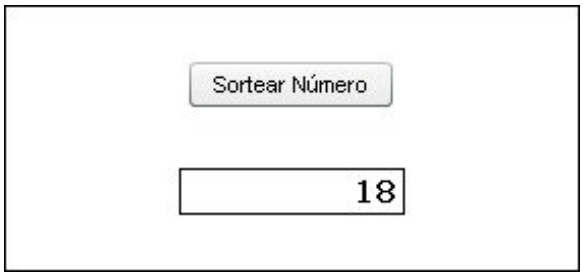

Figura 3.2 – Componente Button utilizado para gerar um número randômico entre 0 e 50.

*****

Exemplo 2

Nesse exemplo utilizaremos um componente **Button** como uma espécie de chave **liga-desliga**, de forma que quando o mesmo for clicado ele deverá executar uma ação, e quando for novamente clicado, executará uma nova ação. Vejamos como criá-lo:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **Button** para o palco;
3. Abra o painel **Component Inspector** (**Shift + F7**), e altere o parâmetro **label** para: **Liberado**;
4. Para que ele funcione como um botão **liga-desliga**, altere o parâmetro **toggle** para **true**;
5. Abra o painel **Properties** e digite no campo de texto **Instance Name**, um nome para a sua instância. Por exemplo: **estado**.
6. Nomeie essa camada de: **componentes**;
7. Crie uma nova camada e chame-a de: **ações**;
8. No primeiro frame dessa camada insira o seguinte código:

```
stop();
import flash.events.MouseEvent;
estado.addEventListener(MouseEvent.CLICK, chave);
//
function chave(event:MouseEvent):void {
    if (estado.selected) {
            estado.label = "Pressionado";
    } else {
            estado.label = "Liberado";
    }
}
```

<u>Vejamos os comentários sobre o código</u>:

<u>Na linha</u>:

```
import flash.events.MouseEvent;
```

Importamos a classe de eventos do mouse necessária para a nossa aplicação.

<u>No bloco de código abaixo</u>:

```
estado.addEventListener(MouseEvent.CLICK, chave);
//
function chave(event:MouseEvent):void {
    if (estado.selected) {
            estado.label = "Pressionado";
    } else {
            estado.label = "Liberado";
    }
}
```

Criamos um evento **CLICK** do mouse vinculado ao botão (instância: **estado**), de forma que quando o mesmo for clicado, a função **chave** logo adiante seja executada. Em seguida, criamos essa função cuja finalidade é alterar o estado do botão quando o mesmo for clicado, de **Liberado** para **Pressionado** e vice-versa.

9. Execute a aplicação e confira o resultado com o mostrado na **Figura 3.3** abaixo:

Figura 3.3 – Componente Button utilizado como uma chave liga-desliga.

*****

Exemplo 3

Nesse exemplo utilizaremos um componente **Button** e um campo de texto da classe **TextField** em tempo de execução, de forma que quando o botão for clicado, o mesmo ficará desabilitado, suas dimensões e sua posição no palco também serão alteradas em tempo de execução, e essas informações serão mostradas no mesmo campo de texto. Vejamos como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **Button** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
import fl.controls.Button;
import flash.events.MouseEvent;
//
var meuTexto:TextField = new TextField();
meuTexto.autoSize = TextFieldAutoSize.LEFT;
meuTexto.multiline = true;
meuTexto.x = 160;
meuTexto.y = 10;
addChild(meuTexto);
//
var meuBotao:Button = new Button();
meuBotao.label = "Clique me";
meuBotao.setSize(120, 40);
meuBotao.move(10, 10);
```

```
addChild(meuBotao);
//
meuBotao.addEventListener(MouseEvent.CLICK, alteraBotao);
//
function alteraBotao (event:MouseEvent):void {
    meuBotao.setSize(80, 20);
    meuBotao.move(30, 10);
    meuBotao.enabled=false;
    meuTexto.text="O tamanho do botão foi alterado para : "+meuBotao.width
+ " pixels de largura por " + meuBotao.height + " pixels de altura.\n";
    meuTexto.appendText("O botão foi movido para as coordenadas:
"+meuBotao.x + " e " + meuBotao.y + ".");
    meuTexto.x=120;
}
```

Vejamos como funciona o código:

Nas linhas:

```
import fl.controls.Button;
import flash.events.MouseEvent;
```

Importamos as classes do componente **Button** e os eventos do mouse necessárias para a nossa aplicação.

No bloco de código abaixo:

```
var meuTexto:TextField = new TextField();
meuTexto.autoSize = TextFieldAutoSize.LEFT;
meuTexto.multiline = true;
meuTexto.x = 160;
meuTexto.y = 10;
```

Criamos uma nova instância (**meuTexto**) da classe **TextField** para mostrarmos as informações referentes ao estado do botão, e em seguida alteramos alguns dos seus parâmetros conforme abaixo:

- O alinhamento do texto utilizando a propriedade **autoSize**.
- A propriedade **multiline** foi configurada para **true** de forma que o campo de texto aceite mais de uma linha.
- As coordenadas X e Y foram definidas para posicionar o respectivo campo de texto no palco.

Na linha:

```
addChild(meuTexto);
```

Utilizamos a propriedade **addChild()** para colocar no palco a instância do objeto **meuTexto** criada anteriormente.

Na linha seguinte:

```
var meuBotao:Button = new Button();
meuBotao.label = "Clique me";
meuBotao.setSize(120, 40);
meuBotao.move(10, 10);
```

Criamos uma nova instância (**meuBotao**) do componente **Button**, e em seguida alteramos os seguintes parâmetros:

- O seu rótulo, utilizando a propriedade **label**.
- O seu tamanho, utilizando a propriedade **setSize**.
- A sua posição no palco, utilizando a propriedade **move**.

Na linha seguinte:

addChild(meuBotao);

Utilizamos a propriedade **addChild()** para colocar o respectivo componente no palco (instância **meuBotao**).

Na linha:

meuBotao.addEventListener(MouseEvent.CLICK, alteraBotao);

Criamos um evento **CLICK** do mouse vinculado ao botão **meuBotao**, de forma que quando o mesmo for clicado, a função **alteraBotao** seja executada.

No bloco de código:

```
function alteraBotao (event:MouseEvent):void {
    meuBotao.setSize(80, 20);
    meuBotao.move(30, 10);
    meuBotao.enabled=false;
    meuTexto.text="O tamanho do botão foi alterado para : "+meuBotao.width
+ " pixels de largura por " + meuBotao.height + " pixels de altura.\n";
    meuTexto.appendText("O botão foi movido para as coordenadas:
"+meuBotao.x + " e " + meuBotao.y + ".");
    meuTexto.x=120;}
```

Criamos a função **alteraBotao**, de forma que quando a mesma for executada, ocorrerão os seguintes eventos:

- O tamanho do botão será alterado.
- O botão será movido para outra posição no palco.
- O botão ficará desabilitado.
- Serão mostradas no mesmo campo de texto duas mensagens informando as alterações ocorridas no botão. O caractere de escape "**\n**" no final da primeira linha significa que o texto a ser anexado deverá ser iniciado na linha seguinte. O método **appendText** anexa o segundo texto logo atrás do primeiro.
- O texto será posicionado na coordenada **x** a 120 pixels da margem do palco.

5. Execute a aplicação e confira o resultado com as figuras abaixo:

Figura 3.4 – Aplicação após a sua execução.

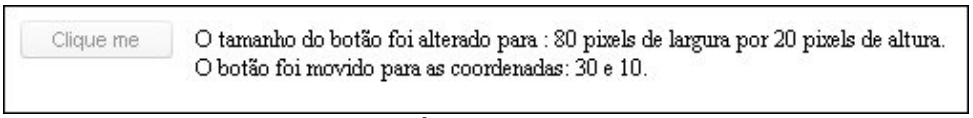

Figura 3.5 – Resultado após o componente Button ser clicado.

*****

Exemplo 4

Nesse exemplo utilizaremos um componente **Button** de forma que, ao ser clicado, carregará uma imagem para um outro componente chamado **UILoader** (que será explicado em detalhes no **Capítulo 19 – O Componente UILoader**). Essa imagem deverá estar no diretório local. Quando o botão for clicado a imagem será carregada nesse componente. Veja como fazer isso:

1. Crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste para o palco um componente **Button** e um componente **UILoader**. Veja layout na **Figura 3.6** a seguir:

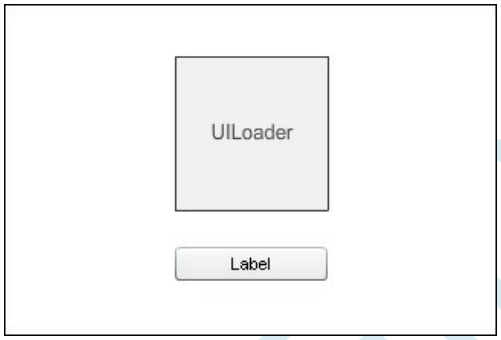

Figura 3.6 – Componente Button a ser utilizado para carregar uma imagem no componente UILoader.

3. Clique no componente **UILoader**, e no painel **Properties**, crie uma instância para ele. Por exemplo: **imagem**;
4. Como o componente **UILoader** não será visível quando a aplicação for executada, e para que você tenha uma idéia de sua posição no palco, crie um retângulo (somente as bordas) com as mesmas dimensões do **UILoader**, ou seja, 100 x 100;
5. Clique no componente **Button**, e no painel **Properties**, crie uma instância para ele. Por exemplo: **botao**;
6. Nomeie a camada atual para: **componentes**;
7. Crie uma nova camada e chame-a de: **ações**;
8. No primeiro frame dessa camada insira o seguinte código:

```
stop();
//
import flash.events.MouseEvent;
botao.label = "Carregar";
imagem.autoLoad = false;
imagem.source = "diego1.jpg";
//
botao.addEventListener(MouseEvent.CLICK, carregaImagem);
//
function carregaImagem(event:MouseEvent):void {
    imagem.load();
    botao.enabled=false;
}
```

Vejamos os comentários sobre o código:

Na linha:

```
import flash.events.MouseEvent;
```

Importamos a classe de eventos para o mouse, necessária para nossa aplicação.

Nas linhas seguintes:

```
botao.label = "Carregar";
imagem.autoLoad = false;
```

Criamos um rótulo em tempo de execução para o botão, e definimos o parâmetro **autoLoad** do componente **UILoader** (instância: **imagem**) como **false** para que a imagem não seja carregada quando a aplicação for executada.

Na linha:

```
imagem.source = "diego1.jpg";
```

Informamos através do parâmetro **source**, da instância **imagem**, o nome da imagem (ou o caminho onde ela se encontra, se for o caso) a ser carregada no componente **UILoader**, somente quando o botão for clicado.

No bloco de código a seguir:

```
botao.addEventListener(MouseEvent.CLICK, carregaImagem);
//
function carregaImagem(event:MouseEvent):void {
    imagem.load();
    botao.enabled=false;
}
```

Criamos um evento **CLICK** para o mouse, vinculado à instância **botao**, de forma que quando o mesmo for clicado, a função **carregaImagem** deverá ser executada. Posteriormente, criamos essa função cuja finalidade é carregar a imagem para o componente **UILoader** (instância: **imagem**), e em seguida, desabilitar o botão (instância **botao**) logo após a imagem ter sido carregada.

9. Execute a aplicação e confira o resultado com o mostrado na **Figura 3.7** a seguir:

Figura 3.7 – Componente Button utilizado juntamente com um componente UILoader para carregar uma imagem.

*****

Exemplo 5

Nesse exemplo criaremos um componente **Button** e alguns estilos de formatação utilizando a classe **TextFormat** em tempo de execução, de forma que cada vez que o botão for

clicado seu rótulo mudará de cor. Para isso, utilizamos uma matriz para armazenar algumas cores. Vejamos como fazer isso:

1. Inicialmente crie um novo documento (ActionScript 3.0);
2. Abra o painel **Components** (**Ctrl + F7**) e arraste um componente **Button** para a biblioteca;
3. Nomeie a camada atual para: **ações**;
4. No primeiro frame dessa camada insira o seguinte código:

```
stop();
import fl.controls.Button;
import flash.events.MouseEvent;
//
var cor=0;
var cores:Array=new Array
("0xFF0000","0x00FF00","0x0000FF","0xFFFF00","0x00FFFF","0xFF00FF");
//
var meuFormato:TextFormat = new TextFormat();
meuFormato.bold = true;
//
var meuBotao:Button = new Button();
meuBotao.label = "Clique me";
meuBotao.move(80, 20);
meuBotao.setStyle("textFormat", meuFormato);
addChild(meuBotao);
//
meuBotao.addEventListener(MouseEvent.CLICK, alteraCor);
//
function alteraCor(event:MouseEvent):void {
    meuFormato.color = cores[cor];
    if (cor>5) {
          cor=-1;
    }
    cor++;
}
```

Vejamos os comentários sobre o código:

Nas linhas:

```
import fl.controls.Button;
import flash.events.MouseEvent;
```

Importamos as classes referentes ao componente **Button** e aos eventos do mouse necessárias para podermos utilizá-los adequadamente em nossa aplicação.

Nas linhas seguintes:

```
var cor=0;
var cores:Array=new Array
("0xFF0000","0x00FF00","0x0000FF","0xFFFF00","0x00FFFF","0xFF00FF");
```

Criamos uma variável de nome **cor** e inicializamos a mesma com **0**. Ela será utilizada para selecionar as cores na matriz quando o botão for clicado. Em seguida, criamos uma nova instância (**cores**) da classe **Array** através de seu construtor para armazenar as cores que iremos utilizar em nosso exemplo. As cores deverão estar no formato hexadecimal.

Nas linhas abaixo:

```
var meuFormato:TextFormat = new TextFormat();
meuFormato.bold = true;
```

Criamos uma nova instância (**meuFormato**) da classe **TextFormat** para definirmos alguns formatos para o botão. Aqui definimos o atributo **bold** e na função mais adiante definimos a cor.

No bloco de código:

```
var meuBotao:Button = new Button();
meuBotao.label = "Clique me";
meuBotao.move(80, 20);
meuBotao.setStyle("textFormat", meuFormato);
```

Criamos uma nova instância (**meuBotao**) do componente **Button** com seu respectivo construtor, e em seguida definimos alguns parâmetros para ele, tais como o seu rótulo (**label**), a sua posição no palco (**move**) e o estilo de formatação definido anteriormente, ou seja, **negrito**.

Na linha:

```
addChild(meuBotao);
```

Utilizamos o método **addChild()** para colocarmos a instância do botão no palco.

Na linha seguinte:

```
meuBotao.addEventListener(MouseEvent.CLICK, alteraCor);
```

Utilizamos o evento **CLICK** do mouse vinculado à instância do componente **Button** (**meuBotao**), de forma que, quando o botão for clicado, a função **alteraCor** seja executada.

No código a seguir:

```
function alteraCor(event:MouseEvent):void {
    meuFormato.color = cores[cor];
    if (cor>5) {
          cor=-1;
    }
    cor++;
}
```

Criamos a função **alteraCor** para alterar a cor do botão com as cores armazenadas na matriz **cores** toda vez que o botão for clicado. Quando a aplicação for executada, a cor do rótulo do botão por padrão será preto. Quando o mesmo for clicado pela primeira vez, a cor do rótulo assumirá a primeira cor da matriz, ou seja, vermelho (**0xFF0000**), tendo em vista que a variável **cor** inicia com 0. Após o botão ser clicado a variável **cor** será incrementada de uma unidade, e o rótulo do botão mudará novamente para a segunda cor da matriz, e assim por diante. Como definimos apenas seis cores, cujo índice varia de 0 a 5, construímos uma condição para quando a variável **cor** atingir um valor maior que 5. Quando isso acontecer, a variável **cor** voltará a ser zero (0), ou seja, -1 + 1 = 0.

5. Execute a aplicação e confira o resultado com o mostrado na **Figura 3.8** a seguir:

Figura 3.8 – Componente Button utilizado para alteração de cor de seu rótulo.

*****

Exercícios de Fixação

1. Qual o parâmetro utilizado para tornar um componente **Button** uma espécie de chave **liga-desliga**?

   a) selected
   b) toggle
   c) trigger
   d) alternate
   e) label

2. Qual o parâmetro utilizado para se criar um rótulo em um componente **Button**?

   a) name
   b) title
   c) _name
   d) label
   e) toggle

3. Que classe precisamos importar para que possamos criar uma instância de um componente **Button** em tempo de execução?

   a) flash.control.Button
   b) fl.controlsButton
   c) flashControls.Button
   d) fl.controlButton
   e) fl.controls.Button

4. Qual das alternativas abaixo cria uma instância de um componente **Button** chamada **meuBotao** em tempo de execução?

   a) var meuBotao.Button = new Button();
   b) var meuBotao:Button = new.Button();
   c) var meuBotao:Button = new Button();
   d) var meuBotao:Button = newButton();
   e) var meuBotao.Button = newButton();

5. Que propriedade é utilizada para se criar uma borda em volta de um componente **Button**?

   a) setBorder
   b) setSize
   c) emphasized
   d) borderSize
   e) sizeBorder

6. Para se criar um componente **Button** em tempo de execução é necessário que o componente esteja presente na biblioteca do usuário.

a) Verdadeiro
b) Falso

7. Qual das alternativas abaixo altera corretamente a largura e a altura de um componente **Button** cuja instância se chama **meuBotao**?

   a) meuBotao.setSize("100,40")
   b) meuBotao.setStyle ("100","40")
   c) meuBotao.setSize(100;40)
   d) meuBotao.setStyle (100,40)
   e) meuBotao.setSize(100,40)

8. Quando um componente **Button** é criado em tempo de execução ele é posicionado, por padrão:

   a) No centro do palco
   b) No canto superior esquerdo do palco
   c) No canto superior direito do palco
   d) No canto inferior esquerdo do palco
   e) No canto inferior direito do palco

9. Qual das alternativas abaixo é utilizada para posicionar um componente **Button** em uma determinada posição do palco?

   a) _x e _y
   b) move()
   c) direction()
   d) moveTo()
   e) localTo()

10. Que evento de um componente **Button** devemos utilizar para criarmos uma ação de forma que ela seja executada somente quando o botão for pressionado, mas não liberado?

    a) buttonDown
    b) buttonClick
    c) buttonPressDown
    d) pressButton
    e) pressClick

*****

Exercícios Propostos

1. Crie uma aplicação utilizando três componentes **Button**, um componente **UILoader** e um campo de texto dinâmico, de forma que quando cada um dos botões for clicado carregue uma imagem diferente para o componente **UILoader** e mostre o nome dessa imagem no campo de texto. Veja uma sugestão de layout para esse exercício na figura abaixo:

2. Crie uma aplicação em tempo de execução, utilizando um componente **Button** e um componente **Slider**, de forma que, quando o controle deslizante do **Slider** for alterado, a borda em volta do botão seja expandida ou reduzida de acordo com o valor selecionado no **Slider**. Utilize a propriedade **emphasized**. Para desenvolver essa aplicação consulte o **Capítulo 14**, referente ao componente **Slider**.

3. Crie uma aplicação utilizando um componente **Button** e um componente **TextArea**, de forma que quando o componente **Button** for clicado apareça no **TextArea** a mensagem: "**Botão clicado.**". Por outro lado, se o botão for mantido pressionado, mostre a mensagem: "**Botão para baixo.**". Configure a propriedade **autoRepeat** para **true**.

*****