1 动画编程简介

本章将对 ActionScript 3.0 作简单介绍，并对 Adobe Flash Professional 中的 IDE 编程环境作简要说明，这些初步的认识和了解，为以后进一步学习 ActionScript 3.0 语言和训练动画编程技能打下了基础。

- Flash ActionScript 3.0 概况
- Adobe Flash Professional IDE
- Flash 中设计与 AS 代码的集成

1.1 动画编程语言 AS 3.0

Flash 是目前最为热门的二维交互式矢量动画制作软件。ActionScript（AS）是内置于 Flash 的编程语言，它在 Flash 内容和应用程序中实现交互性、数据处理以及其他许多功能。

ActionScript 在 ActionScript 虚拟机（AVM）中执行，运行时 AVM 包含在 Flash Player 中。ActionScript 代码通常由编译器转换为字节代码格式。字节代码是一种由计算机编写和识别的编程语言。在 Adobe Flash Professional 中内置有编译器。字节代码嵌入在 Flash Player 执行的 SWF 文件中。

ActionScript 3.0 已成了真正的面向对象的编程语言，提供了可靠的编程模型，方便创建拥有大型数据集和面向对象的可重用代码库的高度复杂应用程序。ActionScript 3.0 相对于早期版本改进的一些重要功能包括：

- 一个新增的 ActionScript 虚拟机，称为 AVM2，它使用全新的字节代码指令集，可使性能显著提高；

- 一个更为先进的编译器代码库，可执行比早期编译器版本更深入的优化；
- 一个扩展并改进的应用程序编程接口（API），拥有对对象的低级控制和真正意义上的面向对象的模型；
- 一个基于 ECMAScript for XML（E4X）规范的 XML API。E4X 是 ECMAScript 的一种语言扩展，它将 XML 添加为语言的本机数据类型；
- 一个基于文档对象模型（DOM）第 3 级事件规范的事件模型。

1.1.1 ActionScript 3.0 的优点

ActionScript 3.0 的脚本编写功能优于早期版本，它旨在方便创建拥有大型数据集和面向对象的可重用代码库的高度复杂应用程序。在 Flash Player 中运行的内容不要求使用 ActionScript 3.0。但是，拥有它可以得到只能通过 AVM2（ActionScript 3.0 虚拟机）实现的性能改善。与旧的 ActionScript 代码相比，ActionScript 3.0 代码的执行速度快 10 倍。

早期版本的 ActionScript 虚拟机 AVM1 执行 ActionScript 1.0 和 ActionScript 2.0 代码。Flash Player 9、10、11 等版本也支持 AVM1 以实现向后兼容性。

1.1.2 ActionScript 3.0 中的新功能

ActionScript 3.0 的新功能包括新增的核心语言功能，以及能够更好地控制低级对象的改进 API。

1. 核心语言功能

核心语言定义编程语言的基本构造块，例如语句、表达式、条件、循环和类型。ActionScript 3.0 包含许多加快开发过程的功能。

①运行时异常：运行时异常用于常见的错误情形；②运行时类型：类型信息在运行时保留，这样提高了性能，减少了内存使用量；③密封类：密封类只能拥有在编译时定义的一组固定的属性和方法；不能添加其他属性和方法；④闭包方法：ActionScript 3.0 使闭包方法可以自动记起它的原始对象实例；⑤ECMAScript for XML（E4X）：E4X 通过大大减少所需代码的数量来简化操作 XML 的应用程序的开发；⑥正则表达式：ActionScript 3.0 包括对正则表达式的固有支持，因此可以快速搜索并操作字符串；⑦命名空间：命名空间使用统一资源标识符（URI）以避免冲突，而且在使用 E4X 时还可以表示 XML 命名空间；⑧新基元类型：ActionScript 3.0 包含三种数值类型：Number、int 和 uint。而 ActionScript 2.0 只包含 Number 一种数值类型。

2. API 功能

ActionScript 3.0 中的 API 包含许多可用于在低级别控制对象的类。语言体系结构的设计比早期版本更为直观。虽然有太多的类需要详细介绍，但是一些重要的区别更值得注意。

①DOM3 事件模型：这种事件模型的设计允许应用程序中的对象进行交互和通信、维持其状态以及响应更改，ActionScript 3.0 事件模型的模式遵守 W3C 的 DOM 级别 3 事件规范；②显示列表 API：用于访问显示列表（包含应用程序中所有可视元素的树）的 API 由使用可视基元的类组成；③处理动态数据和内容：ActionScript 3.0 包含用于加载和处理应用程序中的资源和数据的机制，这些机制在 API 中是直观的并且是一致的；④低级数据访问：多种 API 都提供对数据的低级访问；⑤使用文本：ActionScript 3.0 包含一个用于所有与文本相关的 API 的 flash.text 包。

为了对文本进行更低级别的控制，flash.text.engine 包中的类组成了 Flash 文本引擎。这组类提供对文本的低级控制，是针对创建文本框架和组件而设计的。

1.2 Flash IDE 简介

Adobe Flash Professional IDE，从 CS3 开始直到现在的 CS6 版本，一直没发生过改变，是一种集 Flash 动画设计与编程于一体的面向对象集成开发环境。

1. “动作”面板

“动作”面板是用来编辑、调试时间轴代码、类代码等所有 AS3.0 程序代码的场所。“动作”面板分为 4 个区域，分别是脚本窗口、面板菜单、动作工具箱、脚本导航器，如图 1-1 所示。

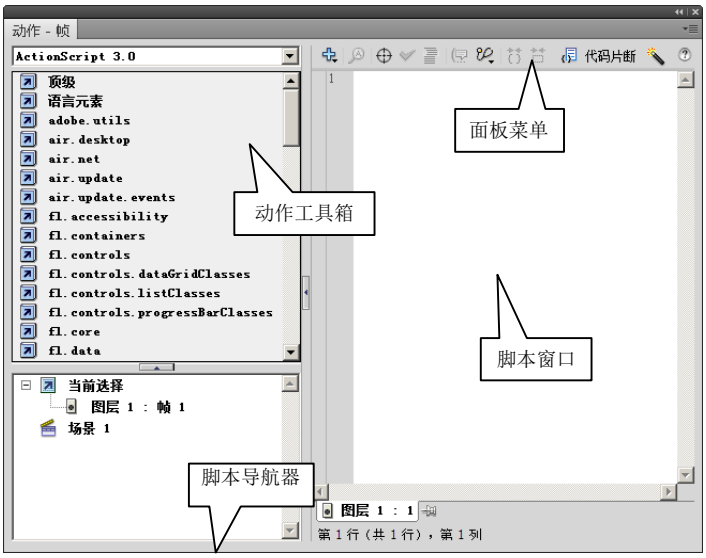

图 1-1 “动作”面板

2. “输出”面板

“输出”面板是测试程序的有效工具。程序中调用 trace()函数输出相关信息到“输出”面板，它在显示结果、测试代码的过程中经常用到。通过鼠标点击菜单【窗口】→【输出】可以将“输出”面板打开。

3. “编译器错误”面板

“编译器错误”面板是调试程序时经常使用的有效工具。通过查看面板中的错误，可以清晰地定位程序出错位置及错误的描述，有助于程序调试中将错误定位和改正，如图 1-2 所示。与“输出”面板类似，可以通过鼠标点击菜单【窗口】→【编译器错误】将“编译器错误”面板打开。

图 1-2 “编译器错误”面板

4. “脚本”窗口

“脚本”窗口是代码编辑的区域。“动作”面板是编辑时间轴代码的主要工具，更多的代码在外部的脚本文件（*.as）中保存。

1.3 案例——图形自动绘制程序

1.3.1 案例任务描述

本案例是一个不需要动画设计的脚本范例，程序代码以外部独立的 AS 文件存在，它实现的功能是在程序运行时自动地完成一个箭头图形的绘制，如图 1-3 所示。

图 1-3 箭头图形

1.3.2 操作流程

（1）创建并编辑脚本文件：打开 Adobe Flash Professional CS5/CS5.5/CS6，选择【文件】→【新建(N)…】，打开“新建文档”窗口，选择“ActionScript 文件”类型，点击【确定】按钮，创建一个空的脚本文件。在脚本窗口输入下述代码，文件命名为 Arrow.as 保存到指定的文件夹。注意，文件名与代码中的类名要一致，区分大小写。

```
package  {  //包定义:首先本程序需用到的内建类
    import flash.display.Sprite;
    import flash.display.Graphics;

    public class Arrow extends Sprite{ //自定义 Arrow 类
        public function Arrow() {
            init();
        }
        public function init():void {  //画线并填充
            graphics.lineStyle(1,0,1);
            graphics.beginFill(0xffff00);
            graphics.moveTo(0+100,25+100);
            graphics.moveTo(0+100,25+100);
            graphics.lineTo(50+100,25+100);
            graphics.lineTo(50+100,0+100);
            graphics.lineTo(100+100,50+100);
```

```
                graphics.lineTo(50+100,100+100);
                graphics.lineTo(50+100,75+100);
                graphics.lineTo(0+100,75+100);
                graphics.lineTo(0+100,25+100);
                graphics.endFill();
            }
        }
    }
```

（2）创建空的 Fla 文档：打开 Adobe Flash Professional CS5/CS5.5/CS6，创建新文档，选择“ActionScript”类型，创新空文档，进入设计环境。点击【文件】→【ActionScript 设置…】，打开“高级 ActionScript 3.0 设置”对话框，如图 1-4 所示。在“文档类”栏中输入名称：Arrow，并在“源路径”下填加保存 Arrow.as 的文件夹。

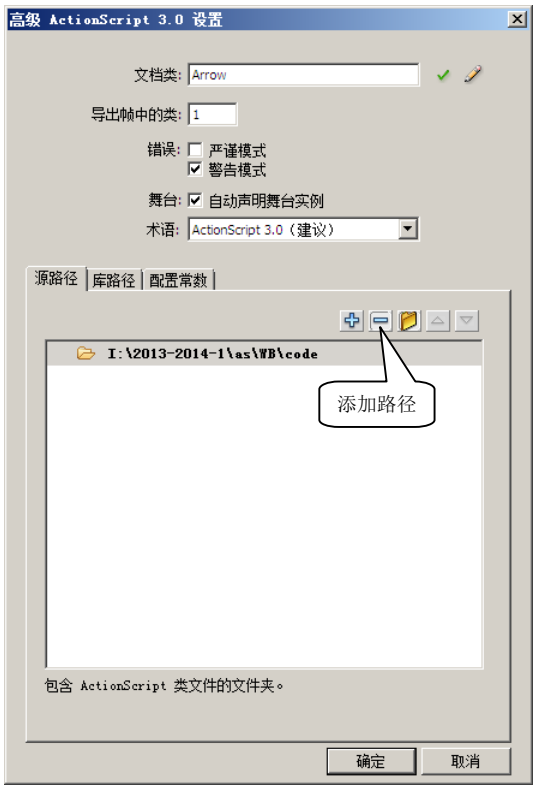

图 1-4　高级 ActionScript 设置

（3）设置完成后，文件命名为 Arrow.fla 保存到与 Arrow.as 相同的文件夹下，按 Ctrl+Enter 组合键测试影片，可以看到绘制的箭头效果。

1.3.3　案例小结

在本案例中，使用独立存在的 Arrow.as 脚本文件，被另一个设计内容空白的 Arrow.fla 设计文件通过文档类的方式调用，集成在一起，形成最后的 Flash 作品。

Arrow.as 文件的类代码可以被重复使用，任何其他影片作品文档都可调用它，原因在于它是独立存在的。

1.4 案例——补间动画播放程序

1.4.1 案例任务描述

本案例首先设计一段常规的形状补间动画，然后添加动画中的交互行为，包括两个可以单击的按钮：一个用于启动动画，另一个用于导航到单独的 URL（某个网站，比如中国水利水电出版社）。交互行为的动作代码为存放在时间轴上的帧代码。运行效果如图 1-5 所示。

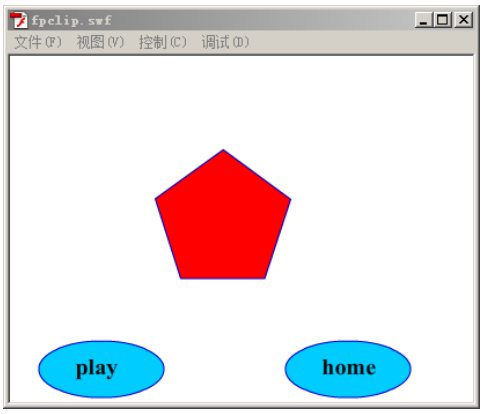

图 1-5 简单补间动画交互程序

1.4.2 操作流程

1. 准备添加交互

向动画添加交互式元素之前，创建一些用于添加新内容的位置将有助于创建 FLA 文件。此任务包括在舞台上创建可放置按钮的实际空间。此外，还包括在 FLA 文件中创建“空间”，以分隔不同的项目。

要创建 FLA 文件以添加交互式元素，请执行下列操作：

（1）创建一个包含简单动画的 FLA 文件，内容为由五边形渐变成圆形的形状补间动画，文件名为 fpclip.fla。

（2）确定两个按钮在屏幕上的显示位置。一个按钮用于启动动画，另一个按钮用于链接到作者的包或主页。如果需要，在舞台上为新内容清除或添加一些空间。如果该动画还没有启动按钮，可以在第一帧上创建一个启动屏幕。这时，可能希望移动动画，使其从第二帧或后面的帧启动播放。

（3）在时间轴中的图层上添加一个新图层，并将其命名为 buttons。这是将要在其中添加按钮的图层。

（4）在 buttons 图层之上创建一个新图层，并将其命名为 actions。这是将要在其中向应用程序添加 ActionScript 代码的图层。最后完成的时间轴，如图 1-6 所示。

2. 创建和添加按钮

接下来，将实际创建和放置构成交互式应用程序中心的按钮。要创建按钮并将其添加到 FLA 文件，请执行下列操作：

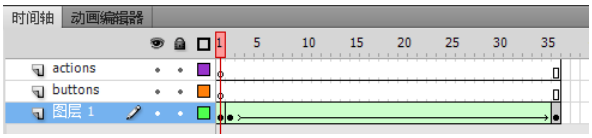

图 1-6　简单补间动画的时间轴

（5）使用绘图工具在 buttons 图层上创建第一个按钮（“play”按钮）的可视外观，例如，绘制一个顶部带文本的水平椭圆。

（6）使用“选择”工具选择单个按钮的所有图形部分。

（7）在主菜单中，选择【修改】→【转换为元件】。

（8）在对话框中，选择“按钮”作为元件类型，将该元件命名为 play，单击“确定”按钮。

（9）在“库”中双击 play 按钮，对 play 按钮的“指针”、“按下”等帧加入关键帧，更换不同的颜色。

（10）返回场景后，选择按钮，在属性检查器中为按钮的实例命名为 playButton。

（11）重复步骤（5）至（10），创建可将查看者链接至某个网站的按钮 home。将该按钮的实例命名为 homeButton。

3. 编写代码

虽然此应用程序的 ActionScript 代码全都是在同一个位置输入的，但是功能可分为三组。代码的这三组功能分别是：

- 一旦 SWF 文件开始加载（当播放头进入第 1 帧时），就停止播放头。
- 侦听一个事件，该事件在用户单击 play 按钮时开始播放 SWF 文件。
- 侦听一个事件，该事件在用户单击 home 按钮时将浏览器定向至相应的 URL。

要创建代码，使得在播放头进入第 1 帧时停止播放头，请执行下列操作：

（12）在 actions 图层的第 1 帧上选择关键帧。

（13）若要打开“动作”面板，请从主菜单中选择【窗口】→【动作】。

（14）在“脚本”窗口中，输入以下代码：

```
stop();
```

接着编写程序代码，使得单击 play 按钮时启动动画播放，请执行下列操作：

（15）在前面步骤中输入的代码的末尾添加两个空行。

（16）在脚本底部输入以下代码：

```
function startMovie(event:MouseEvent):void  {
    this.play();
}
```

此代码定义一个名为 startMovie()的函数，调用该函数会导致主时间轴开始播放。

（17）接下来，输入以下代码行：

```
playButton.addEventListener(MouseEvent.CLICK, startMovie);
```

此代码行将 startMovie() 函数注册为 playButton 的 CLICK 事件的侦听器。也就是说，只要单击名为 playButton 的按钮，就会调用 startMovie() 函数。

接着编写代码，使得单击 home 按钮时将浏览器定向至相应的 URL，请执行下列操作：

（18）在前面步骤中输入的代码的末尾添加两个空行。

（19）在脚本底部输入以下代码：

```
function gotoAuthorPage(event:MouseEvent):void  {
```

```
    var targetURL:URLRequest = new URLRequest("http://www.waterpub.com.cn");
    navigateToURL(targetURL);
}
```

此代码定义一个名为 gotoAuthorPage()的函数。此函数首先创建一个表示 URL http://www.hxedu.com.cn 的 URLRequest 实例，然后将该 URL 传递给 navigateToURL() 函数，使用户的浏览器打开此 URL。

（20）在上一步中添加的代码的下一行中，输入以下代码：

```
homeButton.addEventListener(MouseEvent.CLICK, gotoAuthorPage);
```

此代码将 gotoAuthorPage() 函数注册为 homeButton 的 CLICK 事件的侦听器。也就是说，只要单击名为 homeButton 的按钮，就会调用 gotoAuthorPage() 函数。

（21）随时记得保存文件。至此，再对 fpclip.fla 最终源文档保存。

4. 测试应用程序

应用程序的功能已实现，测试一下应用程序，看是否达到预期效果。测试应用程序：

（22）从主菜单中选择【控制】→【测试影片】。Flash Professional 将创建 SWF 文件，并在 Flash Player 窗口中打开该文件。

（23）试用这两个按钮，确保它们按预期的方式工作。

（24）如果按钮不起作用，请检查下列事项：

- 这两个按钮是否具有不同的实例名？
- addEventListener()方法调用使用的名称是否与按钮的实例名相同？
- addEventListener()方法调用中使用的事件名称是否正确？
- 为各个函数指定的参数是否正确？

上述错误和大多数其他错误都会导致出现错误消息。选择【测试影片】命令时，或测试项目时单击此按钮，都会出现错误消息。在“编译器错误”面板中查看编译器错误（第一次选择【测试影片】时出现的错误）。检查“输出”面板看看是否有在播放内容（如单击按钮）时发生的运行时（runtime）错误。

1.4.3 案例小结

本案例将各段 ActionScript 代码合并后形成一个完整的应用程序，实现如何利用现有线性动画以及添加某些交互式元素。

程序代码全部放置在时间轴的帧上，不存在或没有用到外部 AS 文件中的代码。

1.5 案例——Hello World 程序

1.5.1 案例任务描述

本案例 ActionScript 应用程序是一个标准“Hello World”应用程序，设计要求较为简单：应用程序名为 HelloWorld；将显示一个包含“Hello World!”字样的文本字段；应用程序主要实现一个名为 Greeter 的面向对象的类，存在于外部 AS 文件中。这种设计允许在 Flash Professional 中使用该类。

首先创建该应用程序的基本版本，即标准输出“Hello World!”字符串。然后添加功能，使用

户可以输入用户名，应用程序可以对照已知用户列表检查输入的用户名称。最终实现后的效果如图 1-7 所示。

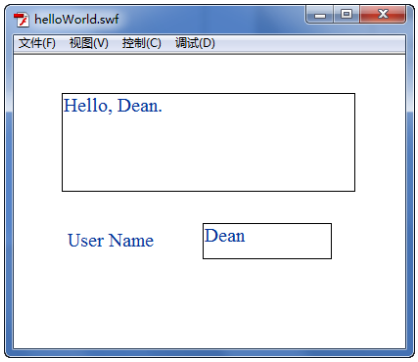

图 1-7　Hello World 程序的最后效果图

1.5.2　操作流程

1. Greeter 类的创建

Hello World 应用程序的设计说明指出该应用程序的代码易于重用。为了达到这个目标，应用程序使用一个名为 Greeter 的面向对象的类，在 Flash Professional 中创建的应用程序将使用该类。

要在 Flash Professional 中创建 Greeter 类，请执行以下操作：

（1）在 Flash Professional 中，从主菜单中选择【文件】→【新建...】。

（2）在“新建文档”对话框中，选择“ActionScript 文件”类型，然后单击“确定”按钮，将显示一个新的 ActionScript 编辑窗口。

（3）选择【文件】→【保存】。选择一个文件夹存放该应用程序，将 ActionScript 文件命名为 Greeter.as，然后单击“确定”按钮。继续完成在 Greeter 类中添加代码。

2. 在 Greeter 类中添加代码

Greeter 类定义一个对象 Greeter，我们将在 HelloWorld 应用程序中使用该对象。要向 Greeter 类中添加代码，请执行以下操作：

（4）在新文件中键入下列代码（有些代码可能已添加）：

```
package {
    public class Greeter {
        public function sayHello():String {
            var greeting:String;
            greeting = "Hello World!";
            return greeting;
        }
    }
}
```

Greeter 类包含一个 sayHello()方法，该方法被调用时返回字符串“Hello World!”。

（5）选择【文件】→【保存】保存此 ActionScript 文件。Greeter 类准备就绪，可在应用程序中使用。

3. 创建使用 ActionScript 代码的应用程序

刚才构建的 Greeter 类定义一组自包含的软件功能，但它不表示完整的应用程序。为使用该类，

创建一个 Flash Professional 文档。该代码需要 Greeter 类的实例。下面说明如何在应用程序中使用 Greeter 类。

要使用 Flash Professional 创建 ActionScript 应用程序，执行以下操作：

（6）从主菜单中选择【文件】→【新建...】。

（7）在“新建文档”对话框中，选择“ActionScript 3.0”（Flash 文件）类型，然后单击“确定”按钮。将显示一个新的 FLA 文档窗口。

（8）选择【文件】→【保存】。选择包含该 Greeter.as 类文件的同一文件夹，将 Flash 文档命名为 HelloWorld.fla，然后单击“确定”按钮。

（9）在 Flash Professional 工具箱中，选择“文本”工具。在舞台上拖动来定义一个大约 300 像素宽、100 像素高的新文本字段。

（10）在“属性”面板中，在仍然选中舞台中的文本字段时，将文本类型设置为“动态文本”。键入 mainText 作为该文本字段的实例名称。

（11）单击主时间轴的第一帧。从主菜单中选择【窗口】→【动作】来打开“动作”面板。

（12）在“动作”面板中键入以下脚本：

```
var myGreeter:Greeter = new Greeter();
mainText.text = myGreeter.sayHello();
```

（13）保存该文件。继续完成发布和测试 ActionScript 应用程序的步骤。

4. *发布和测试 ActionScript 应用程序*

使用 Flash Professional 发布和测试 ActionScript 应用程序，执行以下操作：

（14）发布应用程序并观察编译错误。在 Flash Professional 中，选择【控制】→【测试影片】，编译程序的 ActionScript 代码并运行 HelloWorld 应用程序。

（15）如果测试应用程序时“编译器错误”面板中显示任何错误或警告，则在 HelloWorld.fla 或 HelloWorld.as 文件中修复这些错误。如出现“1046：找不到类型：Greeter”和“1180：调用的方法 Greeter 可能未定义”，则返回场景，从主菜单中选择【文件】→【ActionScript 设置】，打开“高级 ActionScript 3.0 设置”对话框，将 Greeter.as 文件所在的文件夹填加到“源路径”。

（16）如果没有编译错误，则会出现一个显示 Hello World 应用程序的 Flash Player 窗口，如图 1-8 所示。

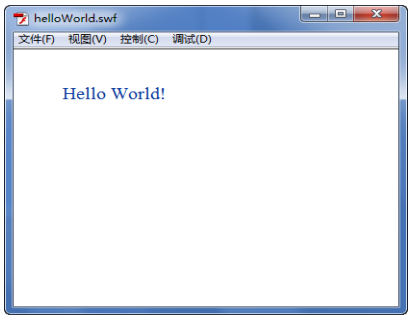

图 1-8 标准 HelloWorld 程序输出

5. *改进 HelloWorld 应用程序*

若要使该应用程序更有趣，可让应用程序要求用户输入用户名并对照预定义的名称列表来验证该用户名。首先，更新 Greeter 类以添加新功能。然后更新应用程序以使用新功能。

更新 Greeter.as 文件，请执行以下操作：

（17）重新打开 Greeter.as 文件。

（18）将文件的内容更改为如下内容：

```
package  {
public class Greeter  {
    public static var validNames:Array = ["Sammy", "Frank", "Dean"];
    public function sayHello(userName:String = ""):String  {
        var greeting:String;
        if (userName == "")  {
        greeting = "Hello. Please type your user name, and then"
            + " press the Enter key.";  }
        else if (validName(userName)) {
            greeting = "Hello, " + userName + ".";  }
        else {
            greeting = "Sorry " + userName + ", you are not on the list.";
        }
    return greeting;
    }
    public static function validName(inputName:String = ""):Boolean {
        if (validNames.indexOf(inputName) > -1) {
            return true; }
        else { return false; }
        }
    }
}
```

现在，Greeter类拥有许多新功能：

- validNames 数组列出了有效的用户名。在加载 Greeter 类时该数组将初始化为包含三个名称的列表。
- sayHello()方法现在接受一个用户名并根据一些条件来更改问候语。如果 userName 是一个空字符串（""），则 greeting 属性将设置为提示用户输入用户名的语句。如果用户名有效，则问候语会是“Hello, username”。最后，如果这两个条件都不满足，则 greeting 变量设置为“Sorry userName, you are not on the list”。
- 如果在 validNames 数组中找到 inputName，则 validName()方法将返回 true；否则，返回 false。语句 validNames.indexOf(inputName)对照 inputName 字符串检查 validNames 数组中的每个字符串。Array.indexOf()方法返回数组中某个对象的第一个实例的索引位置。如果在数组中找不到对象，则返回值-1。

接下来，编辑引用此 ActionScript 类的应用程序文件。使用 Flash Professional 修改应用程序，执行以下操作：

（19）打开 HelloWorld.fla 文件。

（20）修改第一帧中的脚本，将一个空字符串（""）传递到 Greeter 类的 sayHello()方法：

```
var myGreeter:Greeter = new Greeter();
mainText.text = myGreeter.sayHello("");
```

（21）从工具箱中选择“文本”工具，在舞台上新建两个文本字段。将这两个文本字段直接并排放在现有的 mainText 文本字段下面。

（22）第一个新文本字段是标签，在其中键入文本 User Name:。

（23）选择另一个新文本字段，并在属性检查器中选择“输入文本”作为该文本字段的类型，

选择“单行”作为“行类型”，键入 textIn 作为实例名称。

（24）单击主时间轴的第一帧。在“动作”面板中，在现有脚本末尾添加以下行：

```
mainText.border = true;
textIn.border = true;
textIn.addEventListener(KeyboardEvent.KEY_DOWN, keyPressed);
function keyPressed(event:KeyboardEvent):void  {
if (event.keyCode == Keyboard.ENTER)  {
    mainText.text = myGreeter.sayHello(textIn.text);        }
}
```

新代码添加以下功能：

- 前两行只定义两个文本字段的边框。
- 输入文本字段（例如 textIn 字段）具有一组可以调度的事件。使用 addEventListener()方法可以定义一个函数，该函数在发生某一类型的事件时运行。在本例中，该事件指的是按键盘上的某个键。
- keyPressed()自定义函数会检查按下的键是否为 Enter 键。如果是，则该函数调用 myGreeter 对象的 sayHello()方法，同时传递 textIn 文本字段中的文本作为参数。该方法基于传入的值返回字符串“greeting”。返回的字符串随后分配给 mainText 文本字段的 text 属性。

（25）保存该文件，选择【文件】→【保存】。

（26）选择【控制】→【测试影片】，运行此应用程序。

运行该应用程序时，它会提示输入用户名。如果用户名有效（Sammy、Frank 或 Dean），应用程序则会显示如“Hello，Dean”的确认消息。

1.5.3 案例小结

本案例是较为典型的 Flash 动画作品的表现形式，ActionScript 脚本程序和作品设计相结合，有一定的交互性，并体现了一种在 Flash Professional 中使用外部 ActionScript 3.0 类文件的简单模式。

整个程序代码有一部分放置在时间轴的帧上，有一部分以外部独立 AS 文件的形式存在，将二者相结合，实现最后的程序。

拓展训练

编写 AS 3.0 的程序代码有两个情形，即外部文件和作品的时间轴。请思考如何高效地管理程序代码？使代码易于查找、阅读、修改，甚至灵活多用、易于移植。比如时间轴代码管理，封装函数——按不同功能将代码封装成模块；纵向布局——代码分放在“同帧不同层”里；横向延伸——代码分放在“同层不同帧”里；一劳永逸——把优秀时间轴样式另存为模板。