

## AS2Unit

## 介绍

[AS2Unit](#) 是一个 ActionScript 2 的测试框架，允许你创建、重复运行测试你的 AS 类，在开发阶段独立地运行来测试，如同 JUnit 单元测试软件。

## 作用

AS2Unit 促进代码的设计与测试，提高软件可靠性与可维护性。

ActionScript 2 类可以在你设计时进行相应的独立测试，允许的多线程并发进行开发。这样可以提高团队生产力的显着增加和减少在系统整合期间问题的出现。单元测试同时也可以与源码控制软件协作来进行持续集成测试。

## 下载

软件: <http://www.as2unit.org/as2unit.zip>

示例: <http://www.as2unit.org/as2unit-samples.zip>

## 安装

关闭打开的 Flash MX 2004 程序；

解压 as2unit.zip 文件；

把 as2unit.swc 放到：

在 Windows 下，目录是

Documents and Settings\<用户名>\Local Settings\Application Data\Macromedia\Flash MX 2004\<相应的语言>\Configuration\Components\

如 Administrator 用户，目录如下：

X:\Documents and Settings\Administrator\Local Settings\Application Data\Macromedia\Flash MX 2004\zh\_cn\Configuration\Components\

重新启动 Flash MX 2004 程序即可。

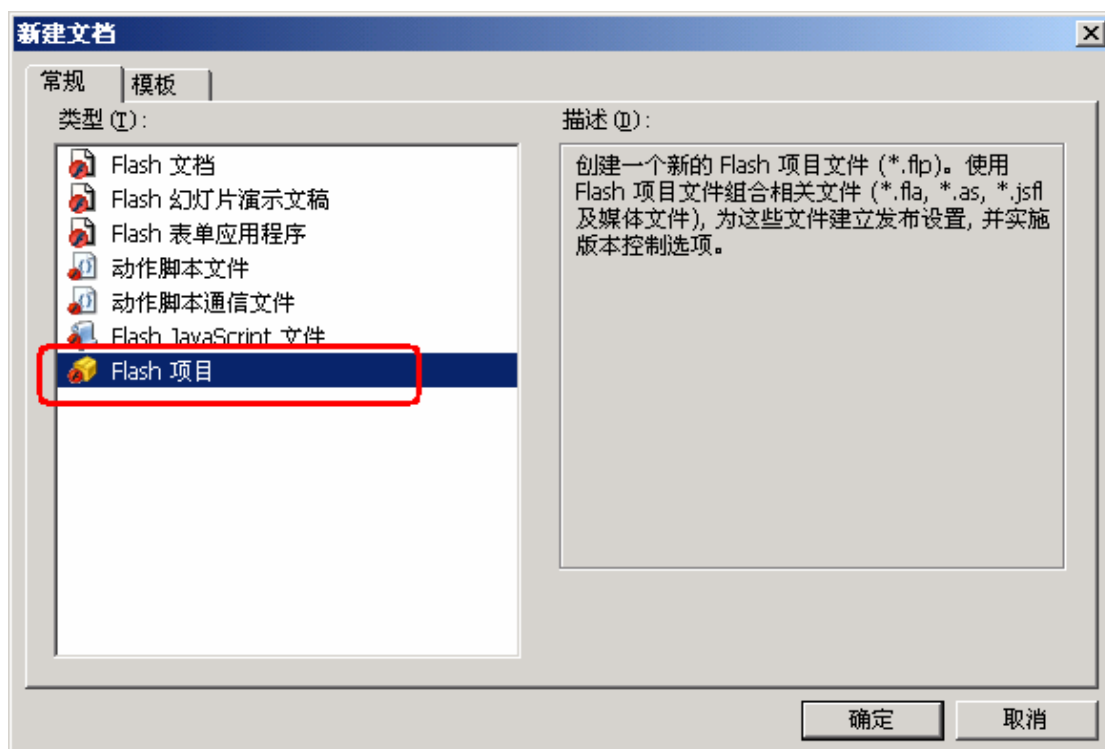
## 使用

打开 Flash MX 2004 后，就可以在“组件”面板里看到如下内容，“标准组件”，AS2Unit

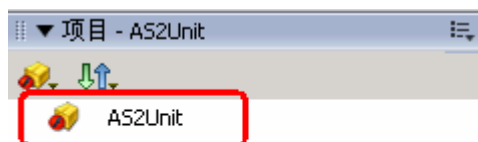


假设你的 Web 应用名称叫做 flex，目录在 C:\flex 目录下，自己根据实际开发情况进行相应的变动，先建立 as 子目录 C:\flex\as，专门用来存放与 ActionScript 相关的内容，以后创建的文件都以此目录为根目录进行操作。

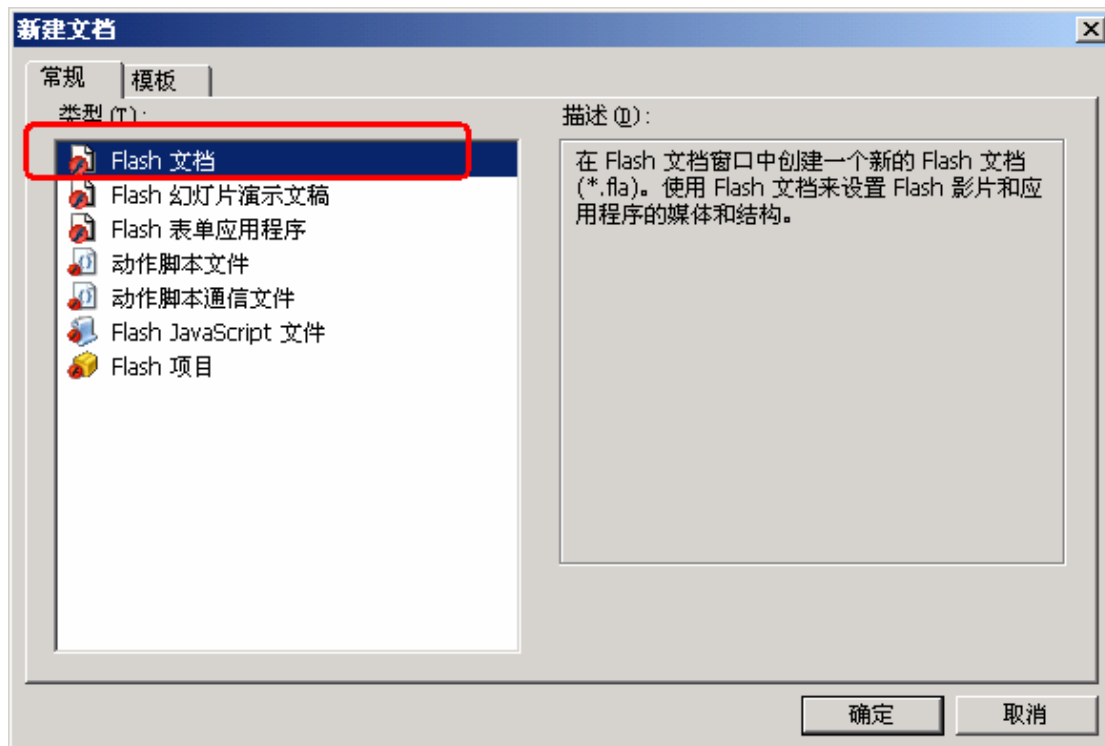
先创建一个 Flash 工程，保存到 C:\flex\as 下取名为 AS2Unit.flp



保存完成后，可以在“项目”面板上可以看到工程信息：



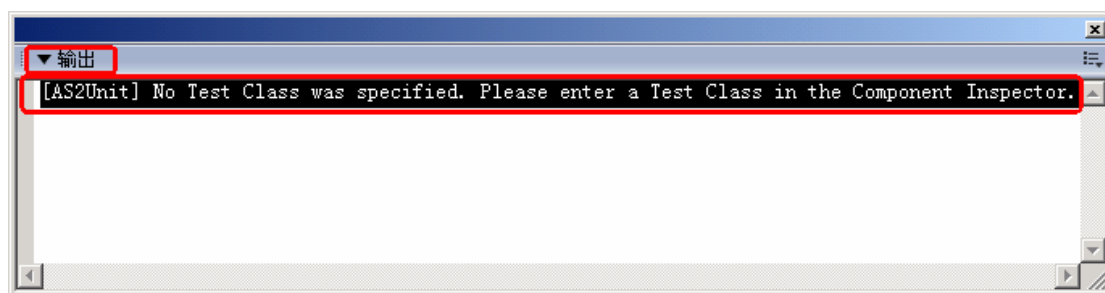
再创建一个空 Flash 文档，保存到 C:\flex\as 下取名 AS2Unit.fla，



然后从“组件”面板里将 AS2Unit 组件拖入场景中。



此时，在输出框里会显示如下信息，提示你测试类没有指定，这个先不用理会它：



在场景中选中它 AS2Unit 组件，可以在“组件检查器”面板里看到如下信息：  
名称“Test Class”对应的“值”，这里面就是要填入你所要测试的类的全称，包括包名：



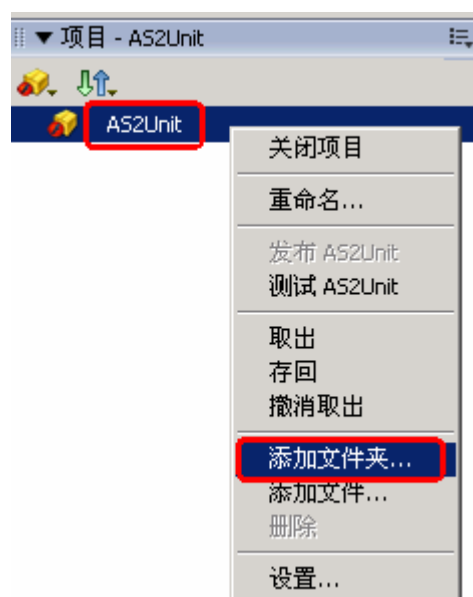
## 创建包与类

现在，你也可以解压 AS2Unit 自带的示例 as2unit-samples.zip 出来，打开 AS2UnitSamples.flp 工程来直接学习如何使用测试即可。

以下的说明也是从这个示例中的类进行的，只是按真正的实际开发过程，把一些相关的包，类等过程说明一下：

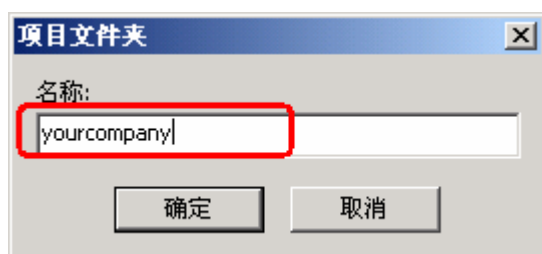
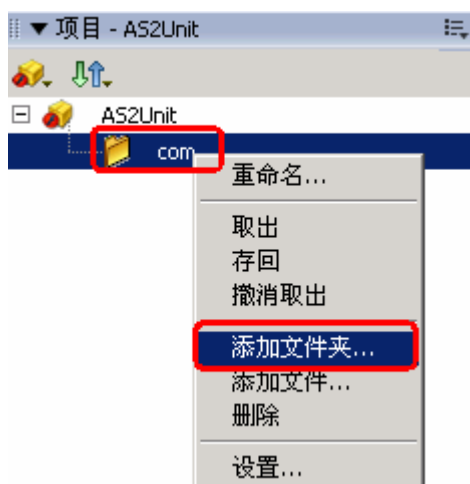
### 创建包

在“工程”面板里，右键工程 AS2Unit，在弹出的窗口中选择“添加文件夹...”，在弹出的窗口中填入你要创建的第一级包名，比如 com，确定完成创建。



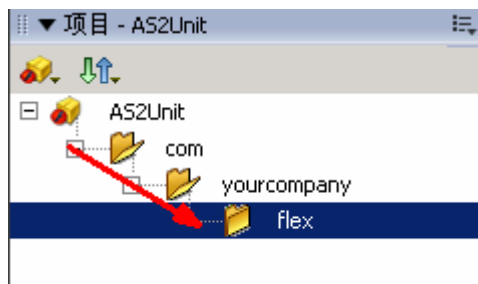
以此类推，再创建二级、三级、N 级包名：

二级包名：在 com 目录上右键，，在弹出的窗口中选择“添加文件夹...”，在弹出的窗口中填入你要创建的第一级包名，比如 yourcompany，确定完成创建。

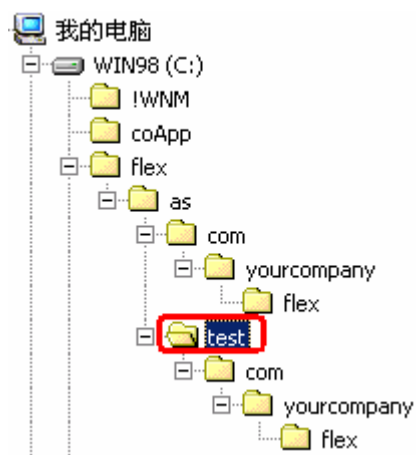


再创建一个三级包名，比如取名为：flex 吧

最后如下所示，这样，在 Flash MX 2004 中的包结构创建完成了

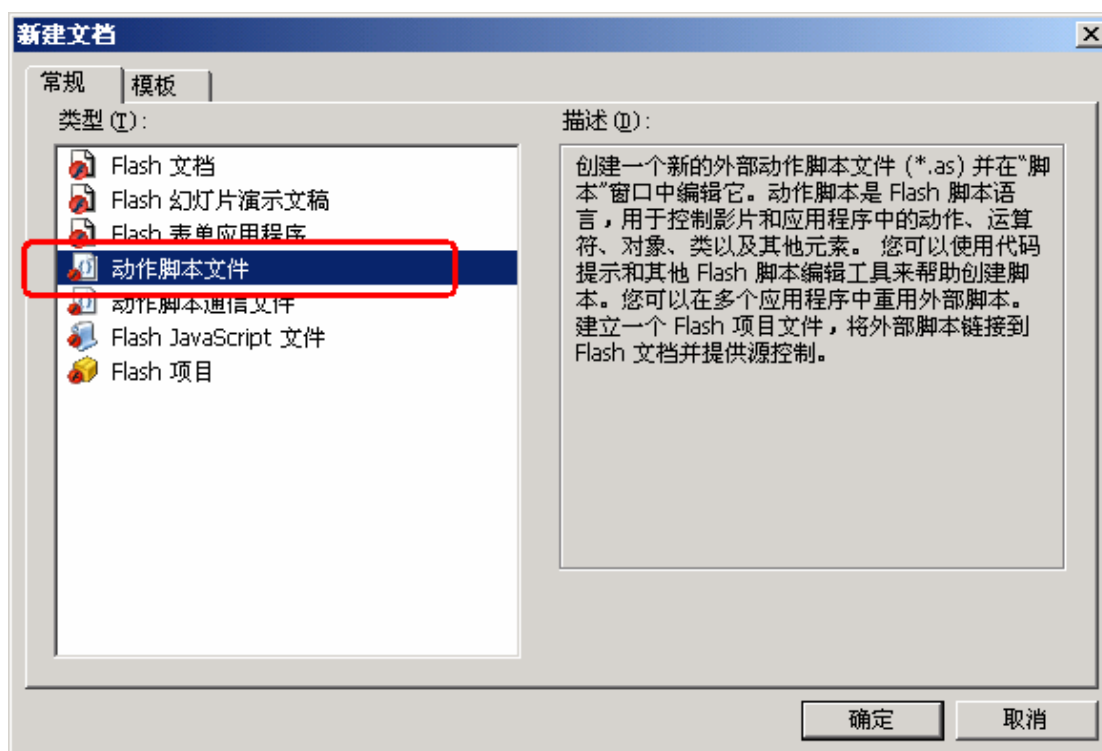


相应地，我们要在资源管理器的应用目录中创建相应的目录层次结构，如下图所示：

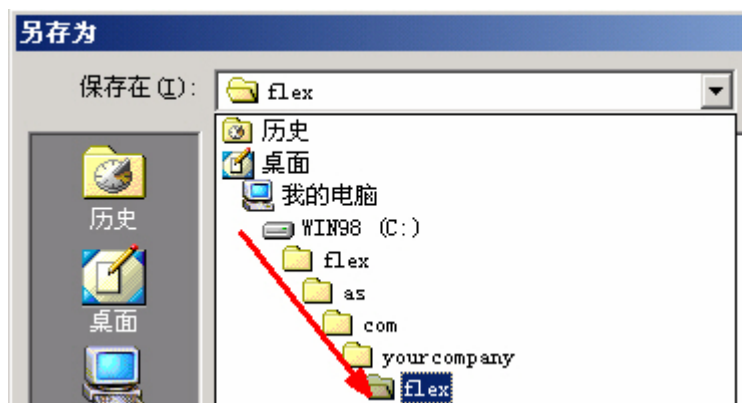


注意这里有个 test 目录，里面的包结构与 as 的下面的 com...结构一样，专门用来存放测试类的。

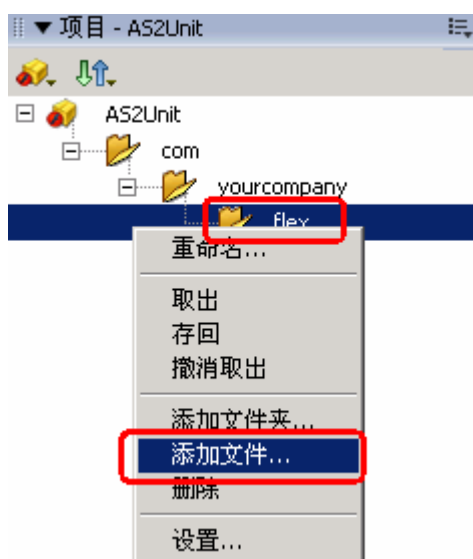
## 创建类



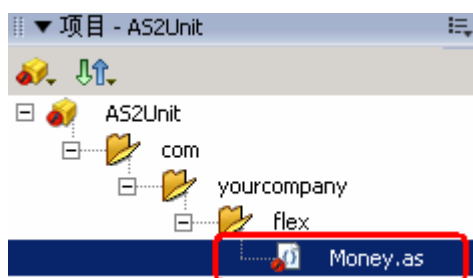
打开示例里面的 Money.as，把里面的内容全部拷贝出来，粘贴到脚本工作区里，  
把 `com.iterationtwo.Money` 改为 `com.yourcompany.flex.Money`  
然后保存到 `com\yourcompany\flex\`目录下，取名为 `Money.as`



同时，在“工程”面板里，右键 flex 目录，在弹出的菜单中选择“添加文件...”。



把刚保存的 Money 文件添加进来，如下所示



## 创建测试类

同样，再创建一个 AS 类，把 TestMoney.as 的内容全部拷贝并粘贴过来；

把 `import com.iterationtwo.Money;` 这一句改为 `import com.yourcompany.flex.Money;`；

把 `test.com.iterationtwo.TestMoney` 改为 `com.yourcompany.flex.TestMoney`

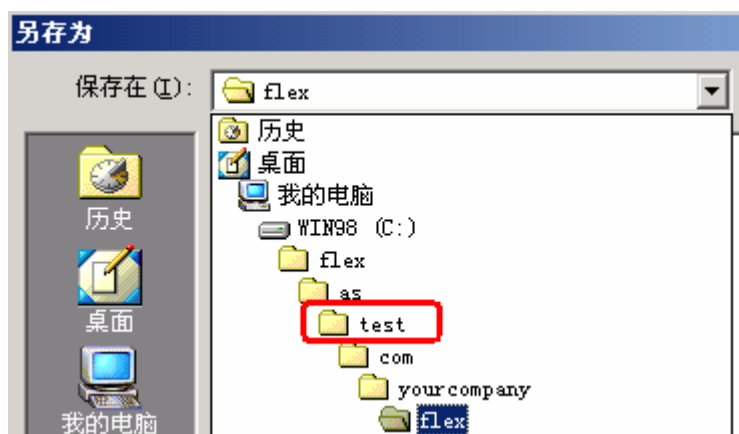
作者：俞黎敏

第 8 页 共 23 页

主页：<http://202.101.111.1/123/>

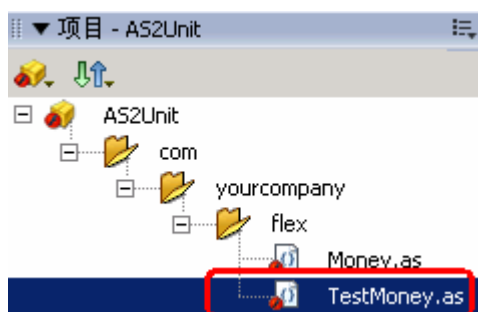
博客：<http://iAMin.BlogDriver.com>





保存到 test\com\yourcompany.flex 目录下，取名 TestMoney.as

相应地在“工程”面板里添加这个文件到 flex 目录下

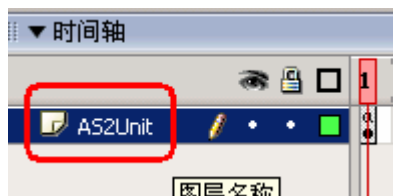


## 测试

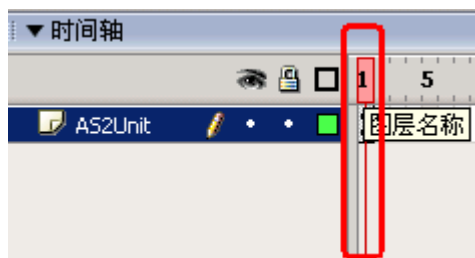
返回到 AS2Unit.fla 文件编辑窗口，在时间轴下，



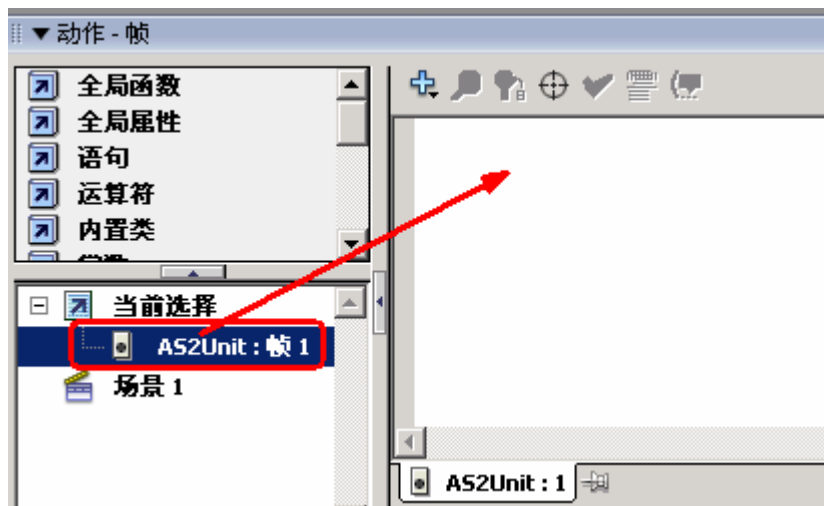
把“图层 1”改为你所想要的内容，如 AS2Unit



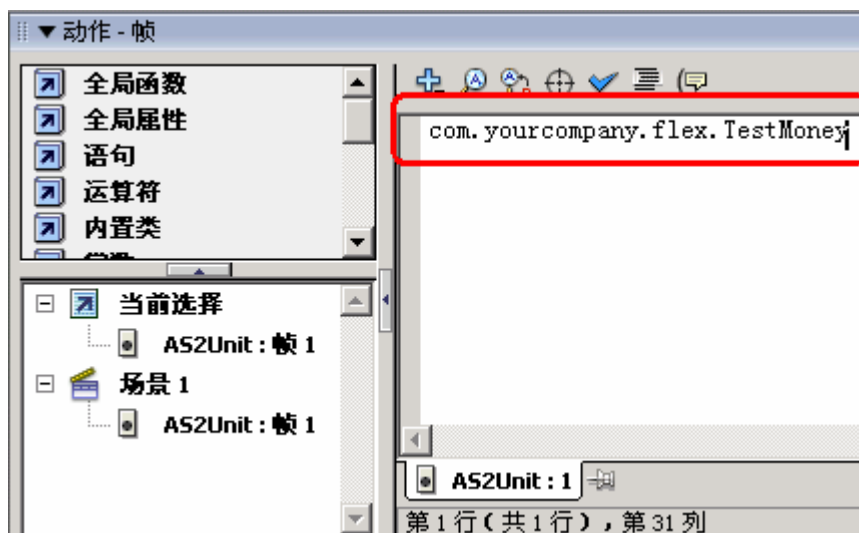
在时间轴里，点中帧 1



在“动作”面板里可以看到



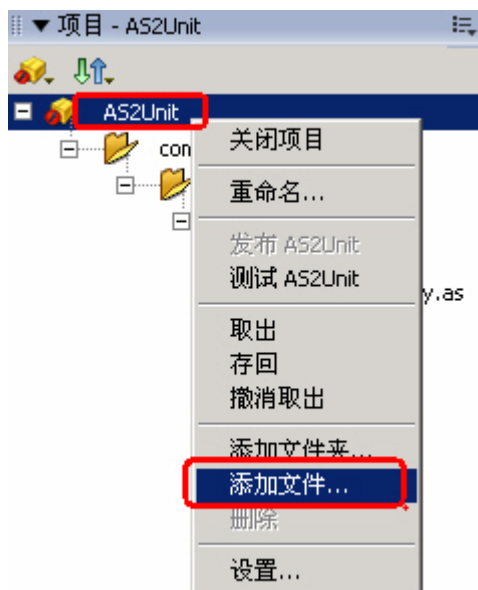
在当前帧里填入 `com.yourcompany.flex.TestMoney` 测试类全名如下，于是会出现场景 1、AS2Unit:帧 1



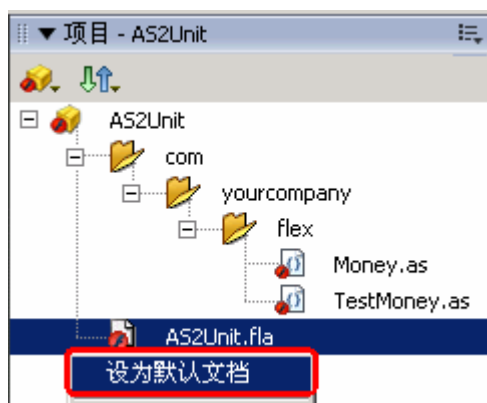
然后，选中 AS2Unit 组件，并在“组件检查器”里的 Test Class 值里填入 `com.yourcompany.flex.TestMoney` 测试类全名，如下：



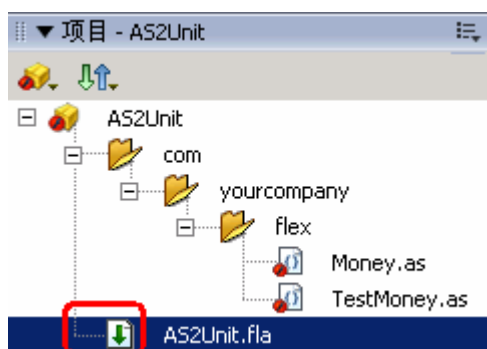
在“项目”面板里，右键工程名称“AS2Unit”，在弹出的菜单中选择“添加文件...”



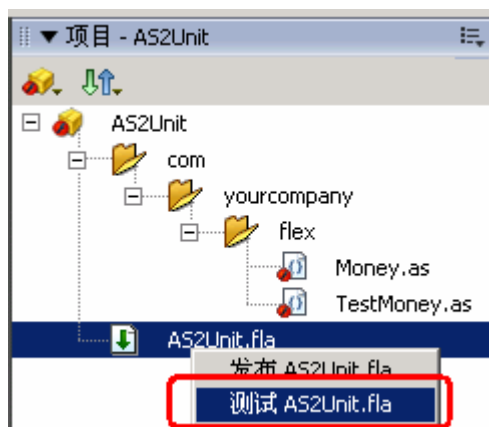
将 AS2Unit.fla 文件添加进来，如下，右键“AS2Unit.flas”文件，可以将它设为默认文档，不设也没有关系。



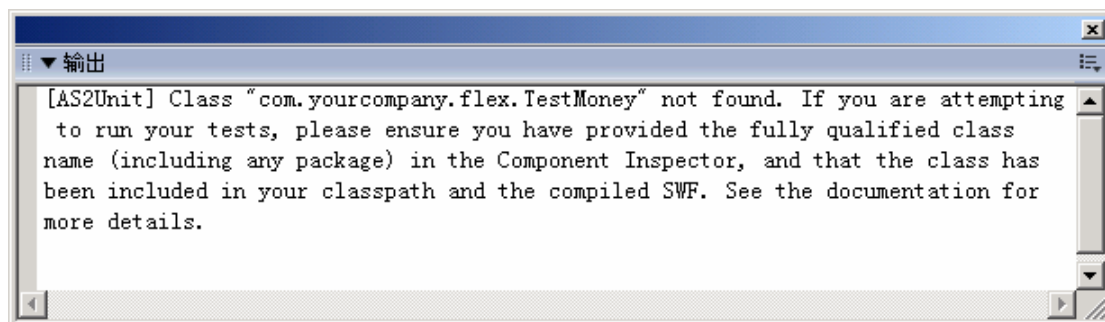
设置为默认文档后，图标变为绿色的向下箭头，如下：



正式运行测试文件，右键“AS2Unit.fla”文件，选择“测试 AS2Unit.fla”进行运行。



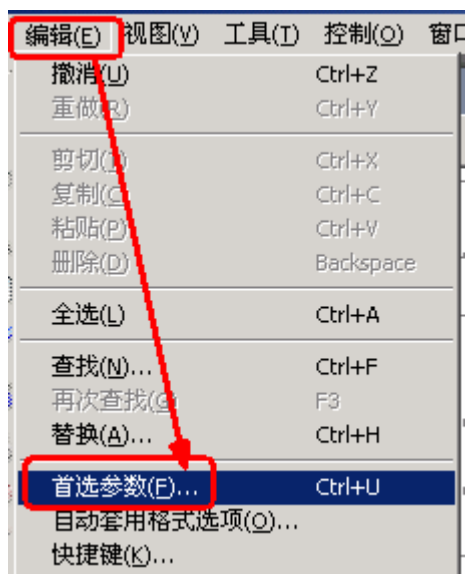
在“输出”面板里出现如下信息：



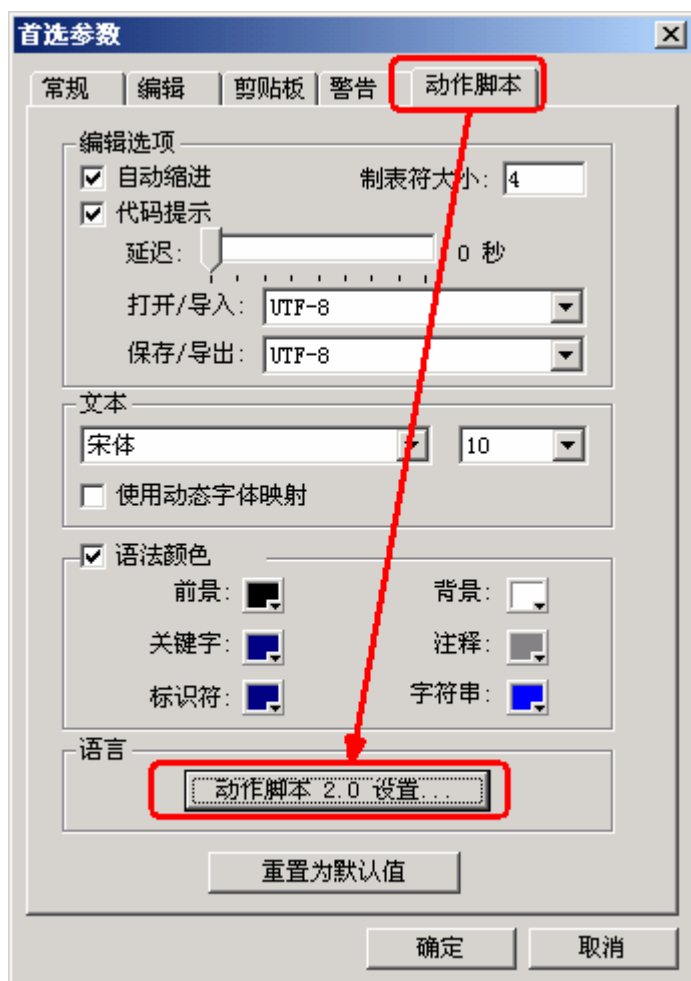
晕，找不到类文件，与 Java 类运行时差不多 classpath 无法找到一样的错误，这里就要说明刚才上面建立一个 test 目录，里面的结构与 as 下的结构一样的作用了，因为类与测试类在不同的目录下面，把以要设置相关的 classpath，当然如果你把类都放在一起就没有关系了，建议分开。

## classpath 设置

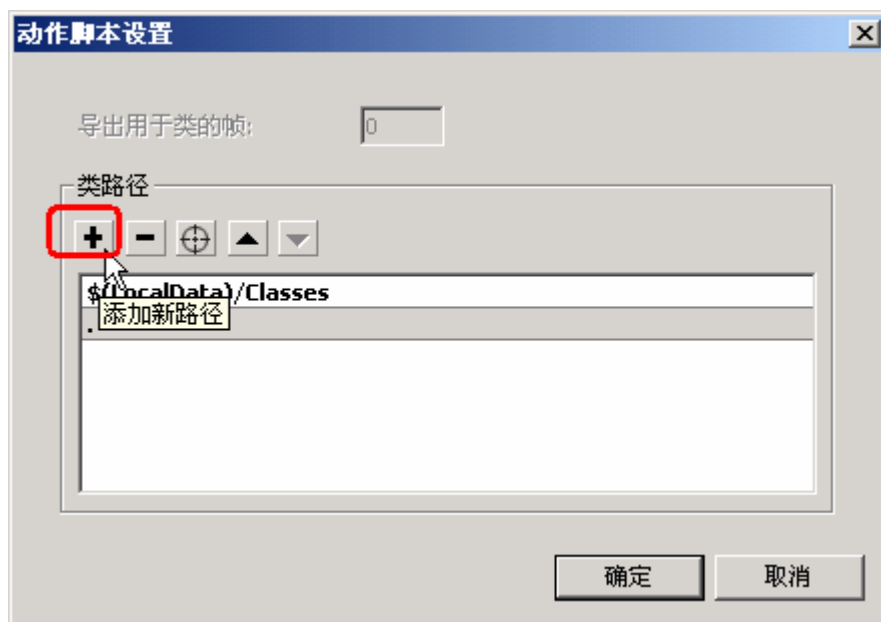
我们做如下操作，在 Flash MX 2004 中，菜单“编辑”，“首选参数(F)...”



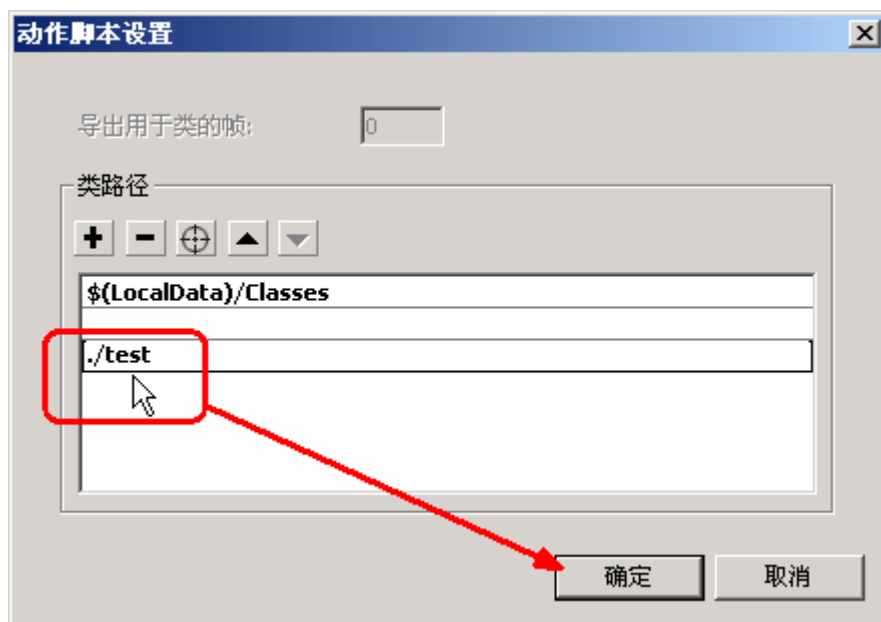
在“首选参数”，选择“动作脚本”，再选择“动作脚本 2.0 设置...”



这里我们可以看到默认的 classpath 设置内容，\$(LocalData)/Classes 和。我们把 test 路径添加进来，如下：



点“+”号，然后填入./test，确定即可



确定之后，我们再运行一下测试文件，这样，结果出来了，如下



还有两个 TestMoneyWithSetUpTearDown.as 与 AllTests.as 文件，重复以上的方法进行测试即可。



# AS2Unit 的使用

## 参数说明

AS2Unit 组件有三个参数

### Test Class

测试类

这个参数是你要告诉 AS2Unit 进行测试的测试类的名字，这个名字必须是全称，即包含包名与类名。

### Run Tests

运行测试

这个参数是开关参数，可以设置 true 或 false，告诉 AS2Unit 是否要运行这个测试。默认的初始值是 true，设置为 false 表示你通知 AS2Unit 不要在 Flash 运行时运行这个测试。

### Display Results

显示结果

这个参数表示测试的结果将显示于哪里，目前，这个选项只能选 Output Panel

## 创建类

根据你的业务需求，创建相关的类

## 创建测试类

以下相关内容与原来进行简单的对比说明，中文翻译时略有精简。

在测试类里，要引入 as2unit.framework.TestCase

```
import as2unit.framework.TestCase;
```

作者：俞黎敏

第 17 页 共 23 页

主页：<http://202.101.111.1/123/>

博客：<http://iAMin.BlogDriver.com>

//同时引入你的业务类

import 业务类;

```
class <测试类全称> extends TestCase//测试类必须继承 as2unit.framework.TestCase 类
{
    //必须有一个构造函数，而且要有一个类型为 String 的参数，通过 super()传递给 TestCase
    public function 测试类( methodName:String )
    {
        super( methodName );
    }
}
```

一个类里可以包含多个单独的测试方法，测试方法必须以 **test** 开头，而且是 **public function** 的（与 JUnit 一样的），AS2Unit 是大小不写敏感的。但是建议测试方法的名称以小写的字母开始，然后。。。

A single test class contains many individual tests. Individual tests are identifiable as being public functions where the function name starts with the word "test".AS2Unit is case insensitive in this regard, but we recommend the naming convention for functions of starting the function with a lower case letter, then capitalising each subsequent word within the function.

现在增加第一个测试方法 `testCreateMoney` 到类里，它会被 AS2Unit 所识别。

We will now add our first test to the test class, naming the function `testCreateMoney()` so that it will be indentified as a test by the AS2Unit component:

```
public function testCreateMoney()
{
    var pounds:Number = 10;
    var pence:Number = 0;

    var money:Money = new Money( pounds, pence );

    assertNotNull( money );
    assertNotUndefined( money );

    assertEquals( 10, money.pounds );
    assertEquals( 0, money.pence );
}
```

The `testCreateMoney` test instantiates an instance of the `Money` class using local variables defined within the test function. Though these local variables are not necessary, and their values could just as easily have been hard-coded within the `Money` class constructor call, extracting their definitions to local variables clarifies their use to readers of the test class. This pre-empts the refactoring known as Replace Magic Number with Symbolic Constant.

Once the class has been instantiated, we want to test that the object is created correctly. This is where we first introduce assertions.

### Assertions

Assertions are the fundamental manner in which you tell AS2Unit how to determine whether the class is acting as you expect it to. In our test methods, we assert that objects and attributes should be in a specified state. If that is not the case, the test fails and the AS2Unit framework tracks the failure for later display.

**assertNotNull、assertNotUndefined:** 用来判定对象是否为空、是否未定义  
就一个参数，即要进行判定的对象  
为 **null**、未定义则测试失败

The basic `assertNotNull` and `assertNotUndefined` statements take a single parameter, that being the object being asserted. If the object has the value being asserted against, in this case, either `null` or `undefined` respectively, then the test fails.

**assertEquals:** 判定两个对象是否相等  
要有两个参数，第一个是期望的结果，第二个是实际上返回的结果  
不相等则返回失败

The `assertEquals` statement take two parameters, the first being the expected result, the second the actual result. Again, AS2Unit will mark the test as having failed if the pounds and pence attributes of the money instance are not as we expect.

We will see later how you can supply an additional user message to assert statements, to provide additional information to you if the test fails.

Assuming a valid `com.iterationtwo.Money` class is in place, running the above test class via AS2Unit will results in the following

For each test run via AS2Unit, the component displays a single period. If the test is successful, AS2Unit progresses to the next test. If the test fails on one of your assert statements, AS2Unit will display an F after the period. If the test produces an error, as will happen if an exception is thrown, AS2Unit will display an E. AS2Unit keeps tracks of all tests and at the end of the test run, displays the total number of tests along with the number of failures, the number of errors and the time take for the tests to run.

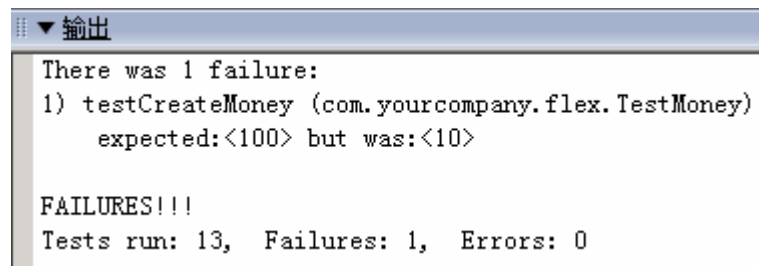
尝试测试失败的结果，如下调整一下，再运行就可以获悉

To show how AS2Unit presents failing tests to you, we can change the `testCreateMoney` function. If we change the following line

```
assertEquals( 10, money.pounds );  
to  
assertEquals( 100, money.pounds );
```

and rerun the tests via AS2Unit, the following error is produced:

失败的结果显示



```
▼ 输出
There was 1 failure:
1) testCreateMoney (com.yourcompany.flex.TestMoney)
   expected:<100> but was:<10>

FAILURES!!!
Tests run: 13,  Failures: 1,  Errors: 0
```

We told AS2Unit that we expected the value of money.pounds to be 100, but it was in fact 10. AS2Unit recognised this and informed us via the failed test.

With all failing tests, we are provided with the name of the test function and the name of the test class, so we can easily identify which test failed.

An individual test function will stop processing on reaching the first assertion failure within that function. So, in the example above, the assertion on the pence attribute of the money instance was never run because the test failed when asserting the value of the pounds attribute.

再增加一些测试内容进行测试

We will now add another test to the TestMoney class, to test the addition of monetary values via the Money class:

```
public function testAddMoney()
{
    var pounds1:Number = 3;
    var pence1:Number = 50;
    var money1:Money = new Money( pounds1, pence1 );

    var pounds2:Number = 3;
    var pence2:Number = 20;
    var money2:Money = new Money( pounds2, pence2 );

    var money3:Money = money1.addMoney( money2 );

    assertNotNull( &money was null&, money3 );
    assertNotUndefined( &money was undefined&, money3 );

    assertEquals( "Pounds should be 6", 6, money3.pounds );
    assertEquals( "Pence should be 70", 70, money3.pence );
}
```

This test instantiates two individual Money instances money1 and money2, and calls the

作者: 俞黎敏

第 20 页 共 23 页

主页: <http://202.101.111.1/123/>

博客: <http://iAMin.BlogDriver.com>

addMoney() function to add them together, with the result going into money3. We then assert that the money3 instance is not null or undefined and that the pounds and pence attributes of money3 are as expected.

You will notice that the assert statements are slightly different in this test function. All assert statements within AS2Unit can accept an additional first parameter, which is a user error message. If a test fails, AS2Unit will show that message in addition to the message shown by default. If we change the assert within the above test to

```
assertEquals( "Pounds should be 666", 666, money3.pounds );
```

to force a failure, the following is produced by AS2Unit:

As you can see, AS2Unit has shown the message defined within the test, prefixed to the message it would normally show.

也提供对 **boolean** 的判定，用 **assertTrue** 或 **assertFalse** 方法

AS2Unit also provides boolean assertions, as shown in the following test:

```
public function testEquals()
{
    var pounds1:Number = 3;
    var pence1:Number = 20;
    var money1:Money = new Money( pounds1, pence1 );

    var pounds2:Number = 3;
    var pence2:Number = 20;
    var money2:Money = new Money( pounds2, pence2 );

    assertTrue( "&monies should be equal&", money2.equals( money1 ) );
}
```

The assertTrue() statement will cause the test to fail if the result of the money2.equals( money1 ) is not true. AS2Unit also provides an assertFalse() statement.

### 异常测试

#### Testing For Exceptions

There is one slightly different statement also available to authors of test classes, that being the fail() statement. Authors of test classes can call fail() when they want to force a test to fail. fail() is often used when checking whether an expected exception is thrown within the class being tested, as in the following example:

```
public function testCreateMoneyBadConstruction()
```

作者：俞黎敏

第 21 页 共 23 页

主页：<http://202.101.111.1/123/>

博客：<http://iAMin.BlogDriver.com>

```
{
    var e:Error;
    var exceptionThrown:Boolean = false;
    try
    {
        var money:Money = new Money();
    }
    catch ( e:Error )
    {
        exceptionThrown = true;
    }
    if ( !exceptionThrown )
        fail( &Exception should have been thrown& )
}
```

当测试失败后，就会捕捉到异常，接着进行异常信息的显示

When this test fails, the following result is produced:

Below is the complete list of statements available to test class authors using AS2Unit:

```
assertEquals()
assertTrue()
assertFalse()
assertNotNull()
assertNull()
assertUndefined()
assertNotUndefined()
fail( userMessage:String )
```

Most of these have been covered above; the usage of the others is obvious from their names.

If you take a look at the complete source for the TestMoney class, you will see that we have a total of 13 tests. Running this complete test class via AS2Unit results in the following test output:

Our TestMoney class is now complete, until, that is, we want to give additional functionality to our Money class. When this occurs, we will write more tests and can be confident that any changes we make to the Money class will be protected by the existing tests.

#### 相关链接

JUnit:<http://www.junit.org>

AS2Unit:<http://www.as2unit.org>

Using AS2Unit:<http://www.as2unit.org/usingas2unit.html>

Unit Testing ActionScript 2.0 Using AS2Unit:<http://www.flashmagazine.com/html/863.htm>

Flash MX 2004: <http://www.macromedia.com/software/flash/>

Flash MX 2004 CN:<http://www.macromedia.com/cn/software/flash/>