

本ドキュメントは、アメリカ macromedia 社サイトの  
<http://www.macromedia.com/devnet/mx/flash/articles/jsapi.html>  
をヒム・カンパニーの永井勝則が独自に訳したものです。

## Flash Articles

### JSFL File API のご紹介



**Guy Watson**

Macromedia Flash MX 2004 の最新のアップデートで、JavaScript API(JSAPI)に新たなファイル・インプット/アウトプットの機能が加わりました。Flash 2004 7.2 アップデータをインストールした共有ライブラリには、JavaScript Flash(JSFL)を可能にする新しい Flfile オブジェクトが生まれました。このオブジェクトは、14の新しい関数を有し、それによって Flash デベロッパーは、ローカルのファイル・システムにあるファイルやフォルダにアクセスし、修正し消去できる Flash の拡張機能を作成できるようになります。

この記事では、File API の新たな機能をご紹介し、JSFL スクリプトを使っての実行方法をご覧にいたします。

Note : この記事は、Flash オーサリング環境での機能の拡張を論じたものであり、Macromedia Flash Player 適用できません。また SWF ファイルにファイル・インプット/アウトプット機能を持たせることはできません。

#### **File API 共有ライブラリ**

JSFL File API は、JSAPI を拡張した形で提供されます。この拡張は共有ライブラリと呼ばれ、Flash Configuration フォルダにある External Libraries サブフォルダにある、ファイル名 Flfile.dll です。Flash Configuration フォルダの場所は、コンピュータにインストールされている OS によって異なります。以下のリストは共有ライブラリ・ファイルの場所を示したものです。

## Windows 2000/XP

C:\Documents and Settings\\Local Settings\Application Data\Macromedia\Flash MX  
2004<language>\Configuration\External Libraries\FLfile.dll

## Windows98

C:\Windows\Application Data\Macromedia\Flash MX  
2004<language>\Configuration\ExternalLibraries\FLfile.dll

## Mac OS X

Hard Drive/Users/<username>/Library/Application Support/Macromedia/Flash MX  
2004/<language>/Configuration/ External Libraries\FLfile.dll

File API を使って JSFL スクリプトを書くには、JSFL スクリプトを実行するマシンに Flash MX 2004 7.2 がインストールされている必要があります。

Note : Flash ドキュメント内でシンボルを含む共有ライブラリと、JSAPI 共有ライブラリを混同しないでください。それらは全く異なるものです。

## File API を使用する

File API は 14 の関数からなり、JSFL スクリプト内のどこでも使用できます。それぞれの関数には Flfile オブジェクトの静的なメソッドとしてアクセスします。

そのメソッドにアクセスするのに Flfile オブジェクトのインスタンスを生成する必要はありません。Flfile オブジェクトを通して直接関数にアクセスします。

例えば、Flfile オブジェクトの関数の 1 つは createFolder と呼ばれ、その名前でメソッドが想像できるように、ローカルのファイル・システムにフォルダを作るものです。JSFL スクリプトでこのメソッドを使うには、以下のシンタックスを使用します。

```
FLfile.createFolder("file:///C:/myFolder");
```

## File URI での作業

File API の全ての関数は、ファイル・システム上のファイルやフォルダで作業するものです。ですから、File API の関数は、作用するファイルやフォルダの場所を関数に伝える引

数を取ります。ファイルやフォルダの場所は、ウェブサイトの URL と非常に似た形でストリングで表されます。これは URI(Uniform Resource Identifier)と呼ばれ、以下のように記します。

```
file:///<drive>/folder/anotherFolder/
```

例えば、Cドライブに config というフォルダを作りたいなら、File オブジェクトのメソッド createFolder に以下の URI を渡します。

```
file:///C:/config/
```

### File API 関数を使う

JSFL スクリプトによって、様々なファイル・システム操作ができます。いくつかの関数では、Mac OS X のコンピュータでの結果とは異なるものもあります。表 1 と表 2 は、File API でできることの説明です。見やすくするため、作業と実際の関数を相互参照の形にしています。

表 1 . ファイル操作

作業	使用する関数
ファイル作成	FLfile.write()
ファイル書き込み	FLfile.write()
ファイル上書き	FLfile.write()
ファイルの内容を読む	FLfile.read()
ファイル消去	FLfile.remove()
ファイルコピー	FLfile.copy()
ファイルが存在するかどうか	FLfile.exists()
ファイルサイズの取得	FLfile.getSize()
ファイルが作成された日付の取得	FLfile.getCreationDateObj() FLfile.getCreationDate()
ファイルが修正された日付の取得	FLfile.getModificationDateObj() FLfile.getModificationDate()
ファイルが読み取りのみであるかどうか調べる	FLfile.getAttributes()
ファイルが隠しファイルであるかどうか調べる	FLfile.getAttributes()
ファイルがシステム・ファイルであるかどうか調べる	FLfile.getAttributes()
ファイルを書き込み可能にする	FLfile.setAttributes()
ファイルを隠しファイルにする	FLfile.setAttributes()
ファイルを可視ファイルにする	FLfile.setAttributes()
ファイルを読み取りのみにする	FLfile.setAttributes()

表2 . フォルダ操作

作業	使用する関数
フォルダ作成	FLfile.createFolder()
フォルダの内容をリスト表示する	FLfile.listFolder()
フォルダ消去	FLfile.remove()
フォルダが存在するかどうか	FLfile.exists()
フォルダが作成された日付の取得	FLfile.getCreationDateObj() FLfile.getCreationDate()
フォルダが修正された日付の取得	FLfile.getModificationDateObj() FLfile.getModificationDate()
フォルダが読み取りのみであるかどうか調べる	FLfile.getAttributes()
フォルダが隠しフォルダであるかどうか調べる	FLfile.getAttributes()
フォルダを隠しフォルダにする	FLfile.setAttributes()
フォルダを可視フォルダにする	FLfile.setAttributes()
フォルダを読み取りのみにする	FLfile.setAttributes()
フォルダを書き込み可能にする	FLfile.setAttributes()

Note : File API 関数は、Flash MX 2004 7.2 がインストールされているコンピュータで実行される JSFL スクリプト内で使用できます。Flash MX 2004 の初期リリース版の実行環境下では、残念ながら今のところ File API メソッドは機能しません。

## File API 関数リファレンス

このセクションは様々な File API 関数の詳細について説明します。

### FLfile.copy(fileURI,copyURI)

copy メソッドを使って、ファイルを別の場所にコピーできます。この関数は2つの引数を取ります。初めの引数は、コピーしたいファイルの場所を示す URL で、2つめの引数は、ファイルをコピーしたい場所を示す URI です。

以下の例では、config.ini という名前のファイルのバックアップ・コピーを作り、新しい名前で同じディレクトリに保存します。

```
originalFileURI="file:///C:/Program Files/MyApp/config.ini";
newFileURI="file:///C:/Program Files/MyApp/config_backup.ini"
```

```
Flfile.copy(originalFileURI,newFileURI);
```

copy メソッドはファイルのコピーが成功すると true を返し、失敗すると false を返します。またこのメソッドはただファイルの名前を変えるためにも使用できます。コピーされたファイルに新しい名前をつけ、元のファイルは消去します。

### **FLfile.read(fileURI)**

Read メソッドは、ファイル内容をストリングで取得するために使用します。この関数は 1 つの引数、内容を読みみたいファイルのある場所を指定する URI を取ります。

以下の例は、クラス・ファイルから ActionScript コードを読み取ります。

```
classFileURI="file:///C:/MyApplication/TextCarousel.as";  
code=Flfile.read(classFileURI);
```

read メソッドは、成功したなら指定されたファイル内容をストリングで返し、失敗したら null を返します。

### **FLfile.write(fileURI,outputStr)+**

write メソッドはファイルにストリングを書き込むために使用します。この関数は、3 つの引数を取ります。最初の引数は、書き込みたいファイルの場所を指定する URI です。2 つめの引数は、そのファイルに書き込みたいストリングを指定します。3 つめの引数は、指定したファイルが既に存在する場合、どうするかを関数に伝えます。デフォルトでは、ファイルは上書きされ、ストリングは新しいファイルに書き込まれます。3 つめの引数は必ずしも必要ではありません。

以下は、ActionScript クラスの基本的な短いコードを新しいクラス・ファイルに書き込む例です。

```
classFileURI="file:///C:/MyApplication/TextCarousel.as";  
code="import mx.core.UIObject¥n"  
code+="class TextCarousel extends UIObject¥n{¥n"  
Flfile.write(classFileURI,code)
```

以下は、上の例のクラス・ファイルに ActionScript のコードをさらに加える例です。

```
classFileURI="file:///C:/MyApplication/TextCarousel.as";
moreCode="%tfunction TextCarousel()¥n{¥n"
moreCode+="%t¥t this.init()¥n";
moreCode+="%t}¥n"
moreCode+="%}"
Flfile.write(classFileURI,moreCode,"append")
```

write メソッドはストリングの書き込みが成功すると true を返し、失敗すると false を返します。

### **FLfile.getSize(fileURI)**

getSize メソッドは、ファイルのサイズをバイトで返すために使用されます。この関数は、1つの引数、調べたいファイルの場所を指定する URI を取ります。

以下の例では、あるアプリケーションの実行ファイルが、そのファイル・サイズが予想されるサイズかどうか調べることによって、修正されていないかをチェックします。

```
mainAppLocation="file:///C:/Program Files/Macromedia/Flash MX 2004/flash.exe";
var expectedSize="13205504" //12.5 mb
if(Flfile.getSize(mainAppLocation) == expectedSize)
{
    alert("This program is in its original state")
}
else
{
    alert("Modifications have been made to this application")
}
```

getSize メソッドは、指定されたファイル・サイズをバイト単位の数値で返します。ファイルが見つからない場合は、0 を返します。

バイトをキロバイトに変換するには、その値を 1024 で割ります。

```
alert((FLfile.getSize("file:///C:/WINNT/explorer.exe")/1024)+"kb")
```

### **FLfile.exists(fileURI)**

exists メソッドは、ファイルやフォルダの存在を調べるために使用できます。この関数は 1 つの引数、探しているファイルの場所を指定する URI を取ります。

以下は、与えられたコンフィグレーション・ファイルが存在するかどうかを調べる例です。もし存在しなければ、それを作成します。

```
configFileLocation="file:///C:/MyApplication/config.ini";
if(!FLfile.exists(configFileLocation))
{
    FLfile.write(configFileLocation,"")
}
```

exists メソッドは指定したファイルがファイル・システムに存在していれば、true を返し、そうでなければ false を返します。

#### **FLfile.createFolder(fileURI)**

CreateFolder メソッドはフォルダやフォルダ階層を作成するために使用できます。この関数は、1 つの引数、作りたいフォルダや一連のフォルダの場所を示す URI をとります。

以下は、コンフィグレーション・フォルダを作成する例です。その中でアプリケーションはユーザーのコンフィグレーション設定を保存するファイルを作成します。

```
applicationFolder="file:///C:/Program Files/Auto Save/"
FLfile.createFolder(applicationFolder+"/Configuration/")
```

以下は、ActionScript クラス・パスのミラーとなる一連のフォルダを作る例です。

```
classPath="controls.cellRenderers"
var folders=classPath.split(".")
var folderPath=folders.join("/")
var projectFolder="file:///C:/MyApplication"
FLfile.createFolder(projectFolder+"/"+folderPath+"/")
```

指定したフォルダの場所が存在しない場合は、作成されます。

createFolder メソッドはフォルダや一連のフォルダの作成に成功すると true を返し、失敗すると false を返します。

### **FLfile.listFolder(fileURI[+fileMask],constraint)**

listFolder メソッドは、指定したフォルダ内にあるファイルやフォルダの名前を持つ配列を検索するために使用できます。listFolder メソッドは2つの引数を取ります。1つめの引数は、内容のリストを検索したいフォルダの場所を指定する URI です。また、名前に特定の文字を持つファイルやフォルダだけを返すことができる、ワイルドカード・ファイル・マスクを指定することもできます。2つめはオプションで、関数に、指定したフォルダ内にあるファイルの名前だけを返させるか、または指定したフォルダ内にあるフォルダの名前だけを返させるかどうかを指定します。2種の値を持つことができます。

- ・ file-ファイル名だけを返します。
- ・ directories-フォルダ名だけを返します。

2つめの引数で値を指定しない場合は、関数は、指定したフォルダにあるファイルとフォルダ両方の名前を返します。

以下は、Windows アプリケーション・フォルダにある全フォルダの名前の配列を返し、それらをアラート・ウィンドウに出力する例です。

```
folderNames=FLfile.listFolder("file:///C:/WINNT","directories")
alert(folderNames.join("\n"))
```

Windows アプリケーション・フォルダにあるフォルダの名前だけを返すには、2つめの引数の値としてストリング “ directories ” を渡します。

以下は、Windows アプリケーション・フォルダにある全ファイルの名前の配列を返し、それらをアラート・ウィンドウに出力する例です。

```
filenames=FLfile.listFolder("file:///C:/WINNT/","files")
alert(filenames.join("\n"))
```

Windows アプリケーション・フォルダにあるファイルの名前だけを返すには、2つめの引数の値としてストリング “ files ” を渡します。

以下は、Windows アプリケーション・フォルダにある全ファイルと全フォルダの名前の配列を返し、それらをアラート・ウィンドウに出力する例です。

```
contents=FLfile.listFolder("file:///C:/WINNT/")
alert(contents.join("\n"))
```

Windows アプリケーション・フォルダにあるファイルとフォルダ両方の名前を返すには、2 つめの引数に値は渡しません。

以下は、指定した URI にファイル・マスクを使って、Windows アプリケーション・フォルダにある全実行ファイルの名前を返す例です。

```
executables=FLfile.listFolder("file:///C:/WINNT/*.exe","files")
alert(executables.join("\n"))
```

\*のワイルド・カードの印は、1 文字以上のファイル名にマッチします。

### **FLfile.remove(fileURI)**

remove メソッドは、ローカルのマシンから読み取りのみでないファイルやフォルダを消去するために使用できます。ファイルやフォルダを含むフォルダを消去すると、それらもなくなります。そのファイルやフォルダのいくつかを読み取りのみであった場合には、それらは消去されません。それらを含むフォルダも消去されません。この関数は、1 つの引数、消去したいファイルやフォルダの場所を示す URI をとります。

以下は、アプリケーションによって作られたコンフィグレーション・ファイルを消去する例です。

```
if(FLfile.remove("file:///C:/MyApplication/config.ini"))
{
    alert("Configuration file deleted")
}
```

以下は、Configuration フォルダとその中身を消去する例です。

```
FLfile.remove("file:///C:/MyApplication/Configuration/")
```

remove メソッドは、ファイルやフォルダの消去が成功すると true を返し、失敗すると false を返します。

### FLfile.getAttributes(fileURI)

getAttributes メソッドは、ファイルやフォルダが読み取りのみか、隠しかどうか、またシステムに属するものかどうかを調べるために使用できます。またこのメソッドを使って、指定した場所がフォルダかどうかを調べることもできます。この関数は、1つの引数、属性を調べたいファイルやフォルダの場所を指定する URI をとります。

以下の例は、フォルダの内容を調べ、そのフォルダの中身が読み込みのみかどうかを見る例です。そうでなければそのフォルダは安全に消去されます。

```
applicationFolder="file:///C:/MyApplication/"
function anyContentsReadOnly(folder)
{
    var contents=FLfile.listFolder(folder)
    var numContents=contents.length
    var attributes
    for(var f=0;f<numContents;++f)
    {
        attributes=FLfile.getAttributes(folder+contents[f])
        if(attributes.indexOf("R") != -1) return true
    }
    return false
}
if(!anyContentsReadOnly(applicationFolder))
{
    FLfile.remove(applicationFolder)
}
```

getAttributes メソッドは 0 か以下の文字を含む文字列を返します。

- R-読み取りのみ
- D-ディレクトリ
- H-隠し(Windows のみ)
- S-システムに属する(Windows のみ)
- A-アーカイブ用(Windows のみ)

返された文字列は、上記にリストした属性の順に作られます。

読み取りのみのシステム・ファイルであるファイルやフォルダに `getAttributes` メソッドを使うと、`RS` が戻ります。同じように、読み取りのみで隠しであるファイルやフォルダに `getAttributes` メソッドを使うと、`RH` が戻ります。

指定したファイルやフォルダが存在しない場合は、`null` が戻ります。指定したファイルやフォルダに属性がない場合は、空の文字列が戻ります。

#### **FLfile.setAttributes(fileURI,strAttrs)**

`setAttributes` メソッドは、ファイルやフォルダを読み取りのみ、書き込み可能、隠し、可視にするために使用できます。この関数は、2つの引数をとります。1つめの引数は、指定した属性に設定したいファイルやフォルダの場所を示す URI です。2つめの引数は、8種の値を持つ文字列です。

- ・ R-指定したファイルやフォルダを読み取りのみにします
- ・ W-指定したファイルやフォルダを書き込み可能にします
- ・ H-指定したファイルやフォルダを隠しにします(Windows のみ)
- ・ V-指定したファイルやフォルダを可視にします(Windows のみ)
- ・ RH-指定したファイルやフォルダを読み取りのみ、隠しにします
- ・ RV-指定したファイルやフォルダを読み取りのみ、可視にします
- ・ WH-指定したファイルやフォルダを書き込み可、隠しにします
- ・ WV-指定したファイルやフォルダを書き込み可、可視にします

ファイルやフォルダを同時に読み取りができ書き込み可能にすることはできません。また同時にファイルやフォルダを隠しで可視にすることもできません。指定する属性だけが指定したファイルやフォルダで更新され、ファイルやフォルダの他の属性は同じままです。

以下は、フォルダの中身を取得し、読み取りのみかどうかを調べる例です。そうであった場合、書き込み可にしフォルダを消去します。

```
applicationFolder="file:///C:/MyApplication/"
function makeFolderContentsWritable(folder)
{
    var contents=FLfile.listFolder(folder)
    var numContents=contents.length
    var attributes
    for(var f=0;f<numContents;++f)
    {
```

```

        attributes=FLfile.getAttributes(folder+contents[f])
        if(attributes.indexOf("R") != -1)
        {
            FLfile.setAttributes(folder+contents[f],"R")
        }
    }

}

makeFolderContentsWritable(applicationFolder)
FLfile.remove(applicationFolder)

```

getAttributes メソッドは、指定したファイルやフォルダの属性設定に成功すると true を返します。失敗すると false を返します。

#### **FLfile.getCreationDate(fileURI)**

getCreationDate メソッドは、ファイルやフォルダが作られた時間とベース・タイムを比べ経過したユニット数値を調べます。ベース・タイムはプラットフォームによって異なります。Windows ではベース・タイムは、1600年1月1日で、ユニットは100ナノセカンドです。

以下はファイルが作成されてから修正されたかどうか調べる例です。

```

fileURI="file:///C:/MyApplication/StateEngine.asp";
var creationTime=FLfile.getCreationDate(fileURI)
var modificationTime=FLfile.getModificationDate(fileURI)
if(modificationTime > creationTime)
{
    alert("The file has been modified since it was created")
}
else
{
    alert("The file is in the state it was when it was created")
}

```

getCreationDate メソッドは、指定したファイルかフォルダが存在すると16進数を含むストリングを返します。そうでなければ false を返します。

### **FLfile.getModificationDate(fileURI)**

getModificationDate メソッドは、ファイルやフォルダが最後に修正された時間とベース・タイムを比較し、経過したユニット数値を調べます。ベース・タイムはプラットフォームによって異なります。Windows ではベース・タイムは、1600年1月1日で、ユニットは100ナノセカンドです。

以下は2つのファイルの修正日を比較し、2つのうちの最後に修正された方を決定します。

```
file1="file:///C:/MyApplication/StateEngine.as"
file2="file:///C:/MyApplication/StateController.as"
modificationTime1=FLfile.getModificationDate(file1)
modificationTime2=FLfile.getModificationDate(file2)

if(modificationTime1 > modificationTime2)
{
    alert("File 2 is older than File 1")
}
else if(modificationTime1 < modificationTime2)
{
    alert("File 1 is older than File 2")
}
else
{
    alert("File 1 and File 2 were saved at the same time")
}
```

getModificationDate メソッドは、指定したファイルかフォルダが存在すると16進数を含むストリングを返します。そうでなければ false を返します。

### **FLfile.getModificationDateObj(fileURI)**

getModificationDateObj メソッドは、getModificationDate メソッドと同じように機能します。例外は、16進数でなく JSFL Date オブジェクトを返す点です。

### **FLfile.getCreationDateObj(fileURI)**

getCreationDateObj メソッドは、getCreationDate メソッドと同じように機能します。例外は、

16進数でなく JSFL Date オブジェクトを返す点です。

### **著者について**

ガイ・ワトソンは、イギリスのロンドンをベースに置く Flash 開発専門会社、FlashGuru 社のマネージング・ディレクター。世界中の様々な Flash 産業イベントにプレゼンテーションを提供し、その作品で多くの産業賞を獲得しています。人気の高い Flash リソース web サイトである FlashGuru Knowledge Base を運営しています。