

# 2.0 がリリースされたので NativeProcess と戯れる

:: Spark Project 勉強会 SP3

:: station5 at 表参道

:: Yukiya Okuda a.k.a alumican.net

# AIR 1.0～1.5時代の個人的な感想

---

- AIRアプリからexeを起動できない
  - 他言語のライブラリを取り込めない
  - ライトなガジェットにはいいんだけど
- ・・・などの絶妙に漂う微妙感により

# AIR 1.0～1.5時代の個人的な感想

---

- AIRアプリからexeを起動できない
  - 他言語のライブラリを取り込めない
  - ライトなガジェットにはいいんだけど
- ・・・などの絶妙に漂う微妙感により

**AIR先生の次回作にご期待ください**



# AIR 2.0 の新機能

---

- ネイティブプロセスAPI
- マルチタッチとジェスチャー
- グローバルエラーハンドラー
- 既定のアプリケーションによるファイルオープン
- ローカルマイクAPI
- UDPネットワークのサポート
- WebKitでのHTML5/CSS3サポート
- スクリーンリーダーのサポート
- ソケットサーバーとP2Pアプリケーション
- TLS/SSLソケット
- etc, etc . . .

# AIR 2.0 の新機能

---

- **ネイティブプロセスAPI**
- マルチタッチとジェスチャー
- グローバルエラーハンドラー
- 既定のアプリケーションによるファイルオープン
- ローカルマイクAPI
- UDPネットワークのサポート
- WebKitでのHTML5/CSS3サポート
- スクリーンリーダーのサポート
- ソケットサーバーとP2Pアプリケーション
- TLS/SSLソケット
- etc, etc . . .

**よし！**

# ネイティブプロセスAPI

---

- ネイティブアプリケーションとは
  - プロセッサが直接処理できるプログラム（exeとか）
  - 非ネイティブに比べて非常に高速なことが多い
- ネイティブプロセスAPIでできること
  - ネイティブアプリケーションの起動
  - ネイティブアプリケーションの終了
  - ネイティブアプリケーションとの相互通信

# ネイティブプロセスAPI

---

- ネイティブアプリケーションとは
  - プロセッサが直接処理できるプログラム (exeとか)
  - 非ネイティブに比べて非常に高速なことが多い
- ネイティブプロセスAPIでできること
  - ネイティブアプリケーションの起動
  - ネイティブアプリケーションの終了
  - ネイティブアプリケーションとの相互通信

**要は、exeをいい感じに  
操作できるようになりました**

# ネイティブアプリケーションの起動

---

//アプリケーション情報を生成する

```
var processInfo:NativeProcessStartupInfo = new NativeProcessStartupInfo();
```

//起動するアプリケーションのパス（例：メモ帳）

```
processInfo.executable = new File("C:/Windows/notepad.exe");
```

//起動時に渡すコマンドライン引数（例：hoge.txtを開く）

```
processInfo.arguments = Vector.<String>(["hoge.txt"]);
```

//アプリケーションの作業ディレクトリ

```
processInfo.workingDirectory = processInfo.executable.parent;
```

//アプリケーションを起動する

```
var process:NativeProcess = new NativeProcess();
```

```
process.start(processInfo);
```

# ネイティブアプリケーションの終了

---

```
//メソッド一発で終了を『試みる』  
process.exit();
```

# ネイティブアプリケーションとの相互通信

---

```
//AIR → ネイティブアプリケーション
//ネイティブアプリケーションの標準入力へ書き込む
process.standardInput.writeUTF("Hello AIR\n");

//-----
//ネイティブアプリケーション → AIR
//ネイティブアプリケーションの標準出力から受け取る
process.addListener(
    ProgressEvent.STANDARD_OUTPUT_DATA, outputHandler);

function outputHandler(e:ProgressEvent):void
{
    //ByteArrayを文字列に変換
    var message:String = process.standardOutput.readUTFBytes(
        process.standardOutput.bytesAvailable);
}
```

# ネイティブアプリケーションとの相互通信

---

//AIR → ネイティブアプリケーション

//標準入力を読み込む

- `char input[256]; gets(input); //C (確認していないけど)`
- `string input; std::cin >> input; //C++`
- `string input = System.Console.ReadLine(); //C#`

//-----

//ネイティブアプリケーション → AIR

//標準出力へ書き出す

- `printf("Hello AIR¥n"); //C`
- `std::cout << "Hello AIR" << std::endl; //C++`
- `System.Console.WriteLine("Hello AIR"); //C#`

# Demo (AIR + OpenCV)

---

- OpenCVの顔認識ライブラリ
  - AS3に移植して実行すると結構重い
  - C++でOpenCVを走らせて、検出結果をFlashで頂く
- 手順
  - AIRからexeを起動する
  - exeが検出した顔領域（複数可）を標準出力する
    - $x_1, y_1, w_1, h_1; x_2, y_2, w_2, h_2; x_3, y_3, w_3, h_3; \dots$
    - カメラはC++アプリが取得し、AIRは透明ウィンドウで上に重ねる（妥協）
  - 標準出力を受け取ってAIRを更新する

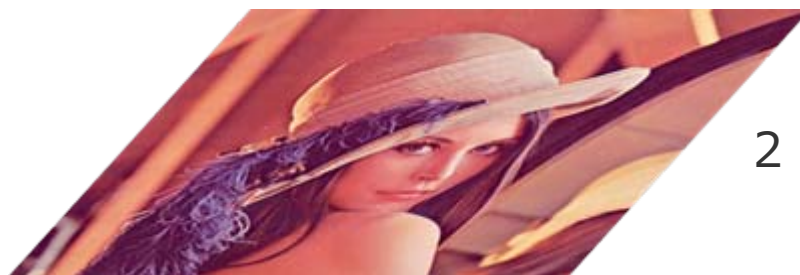
# Demo (AIR + OpenCV)

AIR (透過ウィンドウ)



4 検出結果を表示

ネイティブ  
アプリケーション



3 標準出力で検出結果を送信

2 OpenCVによる顔検出

Webカメラ



1 Webカメラからの映像取得

# 高まる期待

---

- ランチャーとして

- 任意のネイティブアプリケーションを起動させるランチャーアプリケーションの開発

- インターフェースとして

- 重たい処理や、他言語で既実装されている処理を裏側で高速に実行する

ご清聴ありがとうございました