UNIX 実験

-Adobe Flex による作品作り-

琉球大学工学部情報工学科 3 年次
e055746 中山翔
e055751 林哲史
e055755 真玉橋朝明
e055758 宮城良典
e055766 與久田龍一
e055714　亀沢哲郎

提出日

目 次

1 LEVEL 0

LEVEL0 では、Flex についてやインストールの方法についてまとめる。

1.1 Adobe Flex とは

Flex は、リッチインターネットアプリケーションの作成ツールです。Flex で作成したアプリケーションは、Flash Player 9 をインストールしてあるパソコンのブラウザー上で動かす事ができます。Adobe Flash の開発環境ではタイムラインなどの要素があり、プログラマはその独特の開発手法を覚える必要があったが、Adobe Flex ではコーディングがメインになるため従来のプログラミング手法を生かせるため Flex はプログラマー向けだと言われている。

1.2 インストール

Adobe のサイトで無料ユーザ登録を行ない、３０日間の体験版を利用できるようになる。Flex Builder　と SDK があったので以下ではそれについてまとめる。

1.2.1 Builder

Flex Builder は、Eclipse をベースとした IDE（Integrated Development Environment:統合開発環境）です。Eclipse のようにエラーの場所を表示したりして、インターフェース自体も Eclipse に似ています。Builder は 30 日の体験晩です。

1.2.2 SDK

SDK(Software Development Kit) とは、ソフトウェアを作成する際に必要なツールのセットの事です。emacs などの CUI でプログラムを作成する事ができます。こちらは、無料で利用できます。

コンパイル時のコマンド：mxmlc hogehoge.mxml

2　LEVEL1　Flex による簡単なプログラム

2.1　おみくじプログラム

Flex の練習として簡単なプログラムを作成した。Flex は、デフォルトで関数が数多く定義されているのでその関数などを用いた。

2.2　ソースコード

～

```
    <mx:Script>
        <![CDATA[
        import mx.controls.Alert;
        private function onClick():void {
            var message:String;
            var num:int = Math.floor( Math.random() * 4 ) + 1;
            if (num == 1){
                message = "大吉";
            }
            else if (num == 2) {
                message = "中吉";
            }
            else if (num == 3) {
                message = "小吉";
            }
            else {
                message = "凶";
            }
            Alert.show( message );
        }
        ]]>
    </mx:Script>
```

～

2.3 実行結果

図 1: おみくじの実行結果

2.4 説明

コンポーネントの名前	動作
Math.random	0 以上 1 以下の小数の乱数を生成
Math.floor	小数点以下を切り捨てる

表 1: コンポーネントの動作

Math.random 関数で 0 以上 1 以下の乱数を作成しています。それを 4 倍して 1 を加算します。その後、Math.floor で小数を切り捨てています。
これにより、変数 num に入る値は,1,2,3,4 のどれかで、これを else if 文で判別して結果を表示させています。

3　LEVEL2 イベントモデルとマウス操作

3.1　イベントモデル

アプリケーションでユーザがマウスの移動やボタンをクリックなどのユーザ操作など行われるときに「イベント」が生成される。イベントに応じた関数のことを、「イベントハンドラ」または「イベントリスナー」と呼ぶ。以下に主要なマウスイベントを示す。

MXML での指定	MouseEvent クラスの定義	発生
mouseMove	MouseEvent.MOUSE_MOVE	マウスポインタが移動したとき
mouseDown	MouseEvent.MOUSE_DOWN	マウスボタンを押したとき
mouseup	MouseEvent.MOUSE_UP	マウスボタンを離したとき
click	MouseEvent.CLICK	マウスをクリックしたとき
mouseWheel	MouseEvent.MOUSE_WHEEL	マウスホイールを回したとき
doubleClick	MouseEvent.DOUBLE_CLICK	ダブルクリックしたとき

表 2: 主要なマウスイベント一覧表

3.2　お絵描きソフト

マウスイベントを用いた簡単なお絵描きソフトを作成する。マウスボタンを押したまま、画面上でマウスを移動するとマウスポインタの下に円を描く。

3.3　MXML ファイル

ソースコード

～

```
    <mx:Image id="a_img"
        initialize="onInit()"
        mouseMove="onMouseMove(event)"
    />
```

～

説明

デザインを定義した MXML のソースである。画面に円を描くために、Image コンポーネントを利用している。また、画面を初期化するための initialize イベントと、マウスを動かしたときのイベント mouseMove を設定している。

3.4 Action Script ファイル

ソースコード

```
import flash.events.MouseEvent;
private function onInit():void {
    a_img.graphics.beginFill(0xFFFFFF);
    a_img.graphics.drawRect(0,0,this.width,this.height);
    a_img.graphics.endFill();
}
private function onMouseMove(event:MouseEvent):void
{
    if (!event.buttonDown) return;
    var x:Number = a_img.mouseX;
    var y:Number = a_img.mouseY;
    a_img.graphics.beginFill(0xFF0000);
    a_img.graphics.drawCircle(x, y, 4);
    a_img.graphics.endFill();
}
```

説明

oninit() の中では、背景画面のために、画面のサイズに合わせて白色に塗りつぶしている。そして、マウスの移動 mouseMove のイベントハンドラ onMouseMove の中では、マウス座標に円を描いている。onMouseMove では、event.buttonDown でボタンが押されているかを調べている。

3.5 実行結果

図 2: ボタンを押しながらマウスを移動

4　LEVEL3 コンポーネントによる画面の設計

4.1　コンポーネント

Flex は豊富なコンポーネントをもっている。ボタン、エディタ、リストなど最低限のコンポーネントから、グリッド、カレンダー、タブページ、ツリーなどの Web2.0 的なリッチなインターフェースをもつコンポーネントなども用意されている。

4.2　電卓アプリケーション

ボタンコンポーネントは、コンポーネントの基本である。複数のボタンを扱う電卓を作成する。Flex Builder デザインビューで直接コンポーネントパネルからボタンを配置した。

4.3　MXML ファイル

ソースコード

～

```
<mx:TextInput id="disp_txt" x="10" y="10" width="263"/>
<mx:Button x="10" y="40" label="7" width="42" height="42" click="onClick(event)"/>
<mx:Button x="60" y="40" label="8" width="42" height="42" click="onClick(event)"/>
<mx:Button x="110" y="40" label="9" width="42" height="42" click="onClick(event)"/>
<mx:Button x="10" y="90" label="4" width="42" height="42" click="onClick(event)"/>
<mx:Button x="60" y="90" label="5" width="42" height="42" click="onClick(event)"/>
<mx:Button x="110" y="90" label="6" width="42" height="42" click="onClick(event)"/>
<mx:Button x="10" y="140" label="1" width="42" height="42" click="onClick(event)"/>
<mx:Button x="60" y="140" label="2" width="42" height="42" click="onClick(event)"/>
<mx:Button x="110" y="140" label="3" width="42" height="42" click="onClick(event)"/>
<mx:Button x="10" y="190" label="0" width="42" height="42" click="onClick(event)"/>
```

～

説明

デザインビューによりボタンを配置した場合、ソースでは以下のように座標で記述される。ボタンは座標とラベルが違うだけで、後はほとんど同じである。

4.4 Action Script ファイル

ソースコード

～

```
private function onClick(event:MouseEvent):void {
    var btn:Button = event.target as Button;
    var str:String = btn.label;
    // クリアボタンなら初期化
    if (str == "C") {
        clear();
        return;
    }
    // 数字ボタンなら、ディスプレイに追加
    if (str.match(/[0-9\.]/)) {
        addNum(str);
        return;
    }
    // 演算子ボタンが押されたら演算
    eval(str);
```

～

説明

プログラム中のボタンがクリックされたときの動作を記述している。イベントハンドラの引数 (MouseEvent 型の)event には、ボタンをクリックしたコンポーネントを表す target という値が設定されている。ここで、event.target を「as Button」とキャストしている。target 自身は Object 型なので Button 型として得るには、キャストが必要となる。電卓では、ボタンの label の文字によって処理を分岐するので label の値を参照している。

4.5　作成したアプリケーション

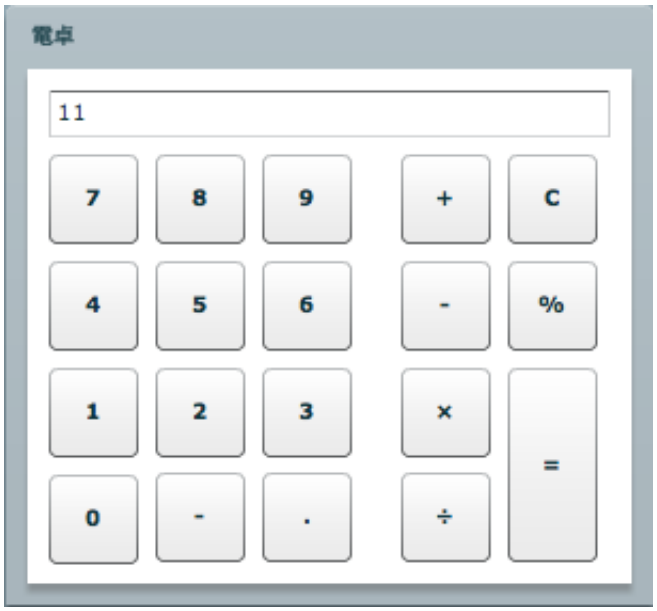

図 3: 5+6 を演算した結果

実際に作成したアプリケーションで演算を実行してみた。図は、作成したアプリケーションで 5 + 6 の演算をしたものである。ボタン操作は、5、+、6、= と入力すると、答えである 11 を表示した。

5　LEVEL4 天気予報リーダー

5.1　天気予報リーダー

Flex のコンポーネントを用いて天気予報の RSS リーダーを作成する。

6　実行結果

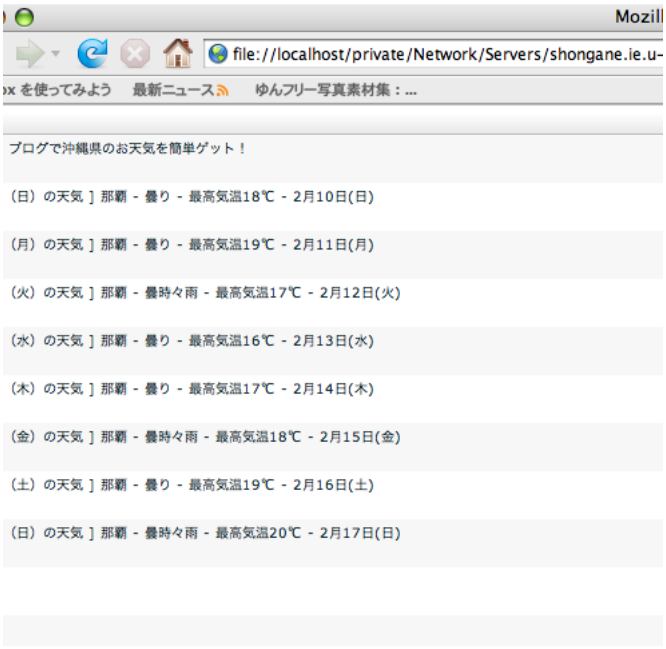

図 4: 実行画面

6.1　見た目の設定

～

```
<mx:DataGrid width="100%" height="100%"
    dataProvider="{a_service.lastResult.channel.item}">
    <mx:columns>
        <mx:DataGridColumn headerText="天気" dataField="title"/>
        <mx:DataGridColumn headerText="アイコン" width="100">
            <mx:itemRenderer>
                <mx:Component>
                    <mx:HBox height="32"
                        horizontalAlign="center">
```

```
                                <mx:Image source="{data.image.url}"/>
```

～

セルの大きさの設定や、入寮区するデータのタグ、イメージの貼付けをここで定義している。

6.2 サーバからのデータの読み込み

```
    <mx:HTTPService id="a_service"
        url="feed://weather.livedoor.com/forecast/rss/47/136.xml"
        useProxy="false"
        resultFormat="e4x"/>
</mx:Application>
```

HTTPService タグを使ってサーバーと通信を行なっている。
プロキシ機能や結果のフォーマットの設定などもここで行なっている。

7　LEVEL5 TODO リスト

7.1　To Do list

Flex を用いて TODO リストを作成する。ここでの TODO リストは、やるべきことをメモする TODO を管理するツールのことである。

7.2　作成したアプリケーション

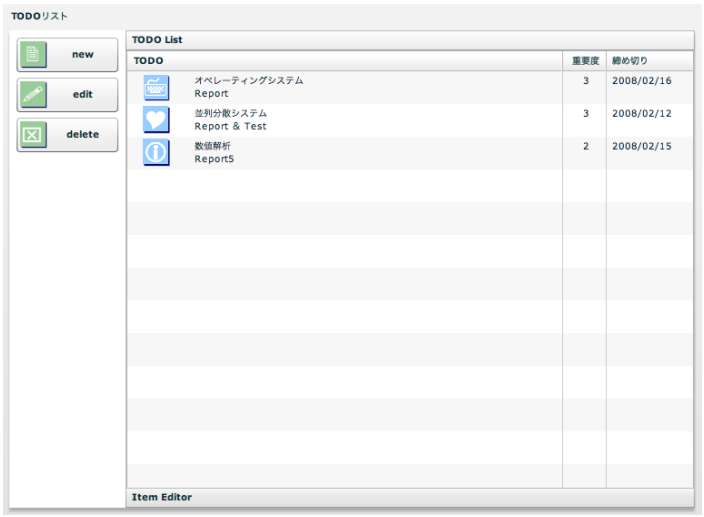

図 5: 画面構成

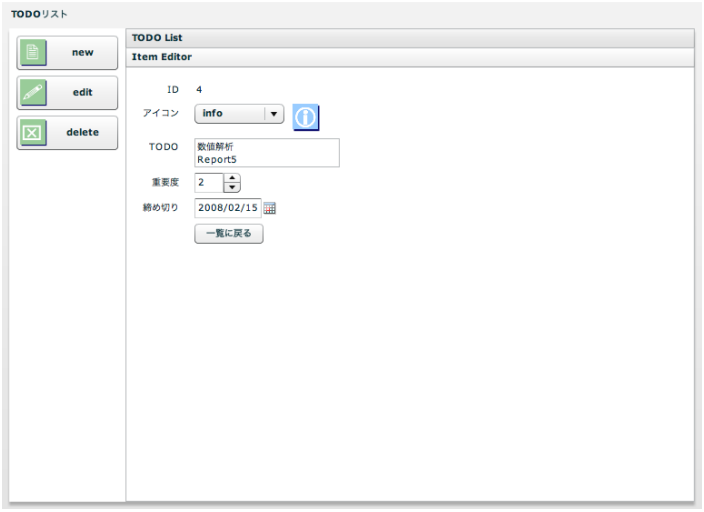

図 6: 編集画面

7.3　データグリッドのアイテムレンダラー

グリッド中の任意のアイコンを表示する。グリッドの TODO 列にアイコンと、TODO テキストを表示するようにする。そこで、グリッドの TODO 列に、アイテムレンダラーを設定し、Image と Text の 2 つのコンポーネントを水平方向に配置するようにする。

```
-----AppTodoList.mxml-----
〜〜
    <!-- アイテムレンダラー -->
    <mx:Component id="todo_grid_renderer">
        <mx:HBox height="32">
            <mx:Image source="{data.icon}.png"
                verticalAlign="middle"
                horizontalAlign="center"
                height="32"/>
            <mx:Text width="100%" height="100%"
                text="{data.text}" selectable="false"/>
        </mx:HBox>
    </mx:Component>
〜〜
```

7.4　ドラッグ＆ドロップ

データグリッドにあるアイテムをゴミ箱の上までドラッグ＆ドロップしたときに、そのアイテムを削除するようにする。これを実装するため、ドラッグ＆ドロップを制御しているオブジェクト DragManager を直接操作する。以下のように、Button の上にアイテムがドラッグされたときに、DragManager.acceptDragDrop() メソッドを呼んで、明示的にドラッグ中のアイテムを受け入れる処理を行う。

```
-----AppTodoListAction.as-----

private function acceptDrop(event:DragEvent):void {
    DragManager.acceptDragDrop(event.currentTarget as UIComponent);
}
```

この関数を呼ぶタイミングは、ドラッグしているオブジェクトがコンポーネントに重なったタイミングで起きる dragEnter イベント中である。

```
-----AppTodoList.mxml-----

<mx:Button width="100%" height="38" label="edit"
    click="editItem()"
    dragEnter="acceptDrop(event)"
    dragDrop="editItem()" icon="@Embed(source='edit.png')"/>
```

あとは、dragDrop イベントで、アイテムを削除する処理を行う。

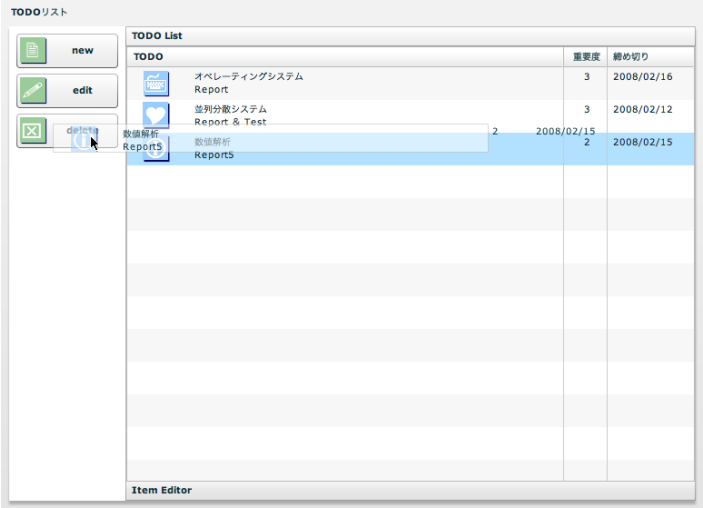

図 7: ドラッグ＆ドロップの図

7.5 データの管理

データの管理には、ArrayCollection 型のオブジェクトを利用する。ローカル領域の保存には、SharedObject オブジェクトを利用する。アプリケーションの初期化イベントで、以下のようにローカルオブジェクトを取得しておく。

```
-----AppTodoListAction.as-----

    save_so = SharedObject.getLocal("todo.data");
```

そして保存されたデータがあれば、TODO データを管理している ArrayCollection 型の変数 todo_items に読み込む。

```
-----AppTodoListAction.as-----

   if (save_so.data.items) {
        todo_items = save_so.data.items;
    }
```

保存も読み込みと同様に行う。

```
-----AppTodoListAction.as-----

private function saveToStrage():void {
    save_so.data.items = todo_items;
}
```

8　Level6 素材検索エンジン

8.1　フォト蔵を用いた画像検索エンジン

　写真共有サイトの「フォト蔵」が公開している API をもちいて、写真を検索する。検索エンジンを作成する事にした。

API の利用方法：http://api.photozou.jp/rest/メソッド名

8.2　作成したアプリケーション

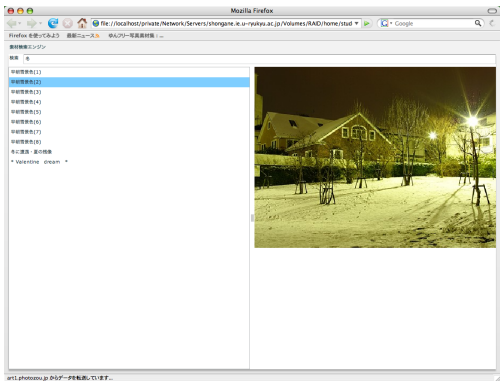

図 8: 冬を検索

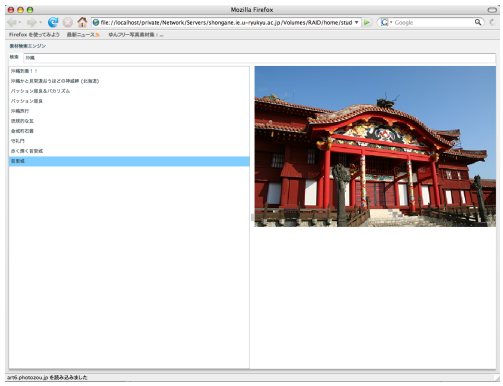

図 9: 沖縄を検索

8.3　説明

～

/*画面の大きさなどの宣言を行なっている。*/

```
    <mx:Panel title="素材検索エンジン" width="100%" height="100%">
    <mx:VBox width="100%" height="100%">
        <mx:HBox width="100%">
            <mx:Label text="検索"/>

/*検索バーを作成している。入力したテキスト（デフォルトでは犬）をkeyword_txt
に格納している。*/

            <mx:TextInput text="犬" id="keyword_txt"
                width="100%"
                keyUp="if(event.keyCode==13) a_service.send()"/>
        </mx:HBox>

/*検索結果を表示する枠の定義*/
        <mx:HDividedBox width="100%" height="100%">
            <mx:List width="100%" height="100%"
                id="a_list"
/*ラベルやクリック時にイメージのURLを読み込み、表示させている。*/
                dataProvider="{a_service.lastResult.info.photo}"
                labelField="photo_title"
                click="a_img.source= a_list.selectedItem.image_url;"/>
            <mx:Image id="a_img" width="100%" height="100%"
                click="navigateToURL(new URLRequest(a_list.selectedItem.url))"/>
        </mx:HDividedBox>
    </mx:VBox>
    </mx:Panel>
～

    <mx:HTTPService id="a_service" resultFormat="e4x"
/*サーバーのURL*/
        url='http://api.photozou.jp/rest/search_public'>
        <mx:request>
/*検索結果の表示の上限*/
            <limit>10</limit>
            <keyword>{keyword_txt.text}</keyword>
/*著作権の項目にcreativecommonsの入っている画像を画像のみを検索*/
            <copyright>creativecommons</copyright>
        </mx:request>
    </mx:HTTPService>
```

9　セキュリテー規約

Flash Player にはセキュリティーの制約があり、セキュリティー制約を回避するには以下の種類のものがある。

・クライアント側で、ユーザーが許可するタイプのもの
・サーバー側で、Web ページの管理人がアクセスを許可するもの

デフォルトでは Flash Player は、その実行された SWF のあるサイト以下のファイルにのみアクセスする事ができる。
SWF のあるサイト以外の Web サイトにある、画像ファイルなどのファイルにアクセスする場合には、そのサイトに、クロスドメインポリシーファイル「crossdomain.xml」が設置されている必要がある。自分の管理する Web サイトを読みたい場合は、自分でファイルを設置するだけですむが、他人の管理する Web サイトのファイルを読むためには、そのサイトの管理人と連絡をして、crossdomain.xml ファイルを設置してもらう必要がある。

9.1　crossdomain.xml

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="~hogeIP アドレス~" />
</cross-domain-policy>
```

参考文献

[1] Adobe Flex2 プロフェッショナルガイド
ISBN978-4-8399-2589-5

[2] ActionScript3.0　ゲームプログラミングブック
ISBN4-8399-2193-8

[3] Adobe - Flex 2 - Web アプリケーション開発ソフトウェア
http://www.adobe.com/jp/products/flex/